	REQ 1	REQ 2	REQ 3	REQ 4	REQ 5	REQ 6	REQ 7	REQ 8	REQ 9	REQ 10	REQ 11	REQ 12	REQ 13	REQ 14	REQ 15	REQ 16	REQ 17
UC - 1 Make Purchase	✓		✓			✓	✓			✓							
UC - 2 Add Item to Cart	✓		✓	✓	✓		✓			✓				✓		✓	
UC - 3 Add Item to Wishlist	✓		✓	✓			✓			✓				✓	✓	✓	✓
UC - 4 Restock Item	✓		✓	✓			✓		✓								
UC - 5 Return Item	✓		✓			✓	✓				✓	✓	✓				
UC - 6 Make Account								✓									
UC - 7 Ship/Send Item	✓		✓				✓			✓	✓						
UC - 8 Add New Product	✓	✓	✓							✓							
UC - 9 Delete Product	✓	✓	✓														
UC - 10 View Product Avalibility	✓		✓	✓					✓	✓	✓			✓		✓	✓
UC - 11 Generate Report						✓	✓		✓							✓	
UC - 12 Remove Item from Cart	✓		✓		✓		✓										

Use Case Name: Add Item to Cart

Scope: Store

Level: user goal

Primary Actor: customer

Stakeholders and Interests:

- Customer: wants to be able to purchase multiple different items
- Company: wants to be able to keep track of the items the customer purchases so that they can get their items in a timely manner

Preconditions: Customer wants to add an item to their cart

Success Guarantee (or Postconditions):

The item is added to the cart, and the subtotal is updated

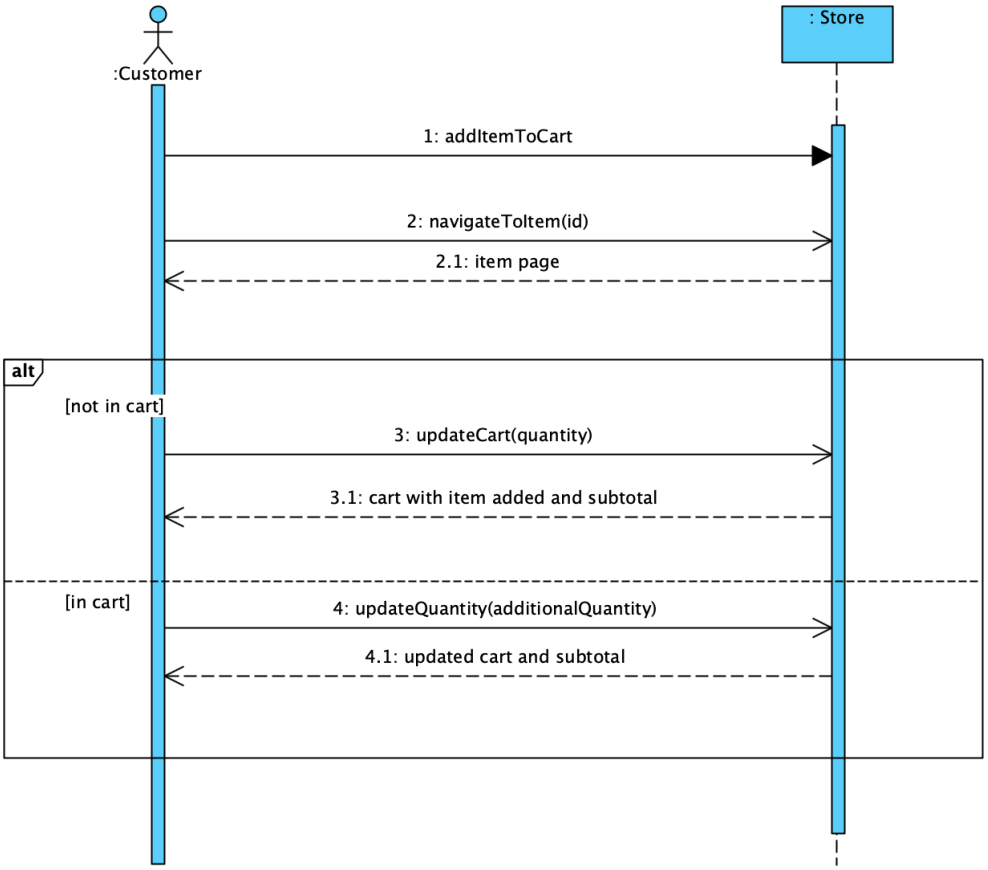
Main Success Scenario (or Basic Flow):

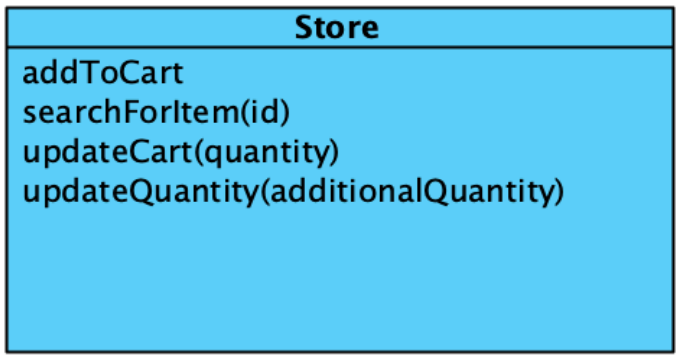
1. Customer navigates to the item
2. Customer clicks add to cart button
3. Customer enters the quantity they wish to purchase
4. Items are added to cart
5. Subtotal is updated
6. The customer is informed that the items have been successfully added to the cart

Extensions (or Alternative Flows):

2a. Item is already in cart:

1. Customer selects how many they wish to add to the cart
2. The customer's selection is added to their previous selection


:Customer
: Store
1: addItemToCart
2: navigateToItem(id)
2.1: item page
alt
[not in cart]
3: updateCart(quantity)
3.1: cart with item added and subtotal
[in cart]
4: updateQuantity(additionalQuantity)
4.1: updated cart and subtotal



Operation: addToCart
Cross References: Add Item to Cart
Preconditions: There is an instance of a cart created
Postconditions: The cart has an item added to it

Operation: searchForItem(id)
Cross References: Add Item to Cart
Preconditions: The customer knows what item they are searching for
Postconditions: The customer has navigated to the item page

Operation: updateCart(quantity)
Cross References: Add Item to Cart
Preconditions: The customer has navigated to the item page, and the item isn't in the cart
Postconditions: The item and its desired quantity is added to the cart

Operation: updateQuantity(additionalQuantity)
Cross References: Add Item to Cart
Preconditions: The customer has navigated to the item page, and the item is already in the cart
Postconditions: The quantity of the item in the cart is increased by additionalQuantity

Use Case Name: Remove Item from Cart

Scope: Store

Level: user goal

Primary Actor: customer

Stakeholders and Interests:

- Customer: wants to be able to keep track of the items to be purchased, and make adjustments to them (remove, alter number of) if needed.
- Company: wants to be able to properly track an order, including when items are removed, so that purchases are accurate.

Preconditions: Customer wants to remove an item from their cart

Success Guarantee (or Postconditions): The item the customer wanted removed from the cart will be removed and the subtotal calculated.

Main Success Scenario (or Basic Flow):

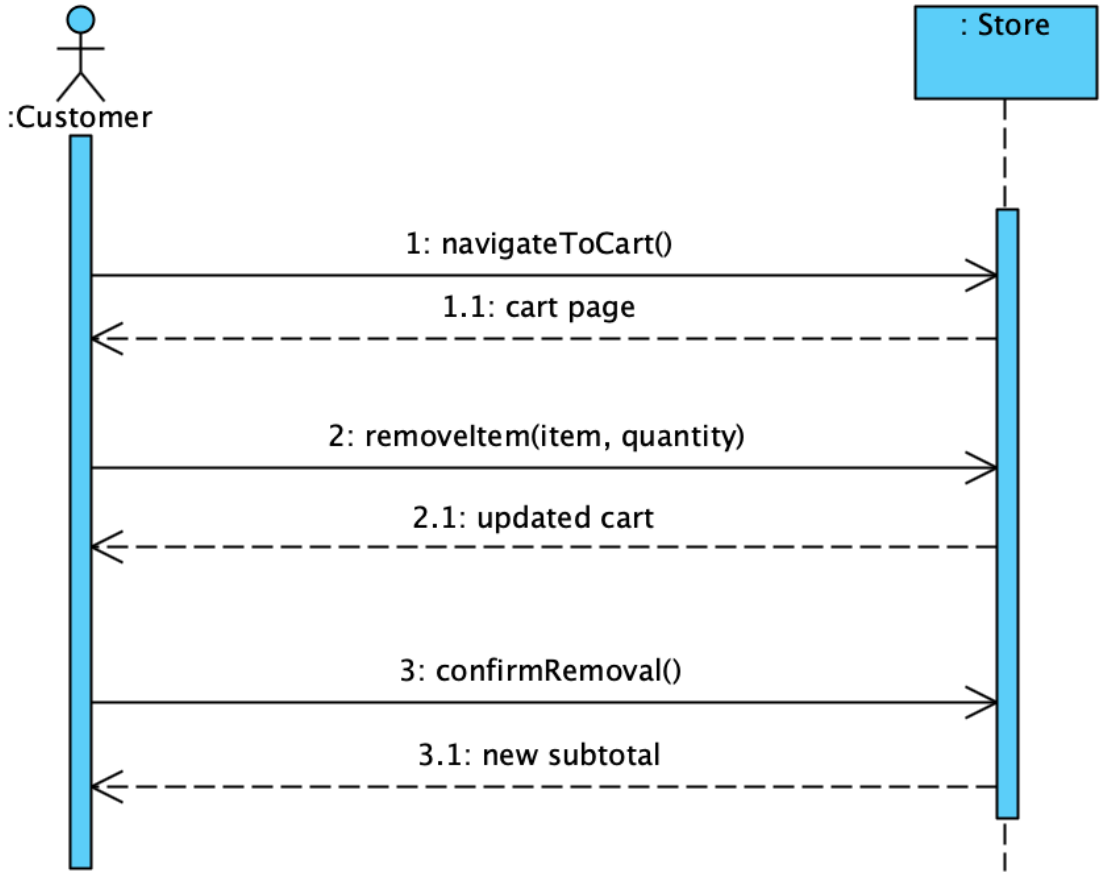
1. Customer navigates to their current shopping cart instance
2. The customer clicks on the item to be removed from the cart
3. The customer clicks on the UPDATE ORDER button
4. The customer clicks remove item
5. The machine shows a verification that the user wants to remove the item
6. The customer selects yes
7. The subtotal is updated

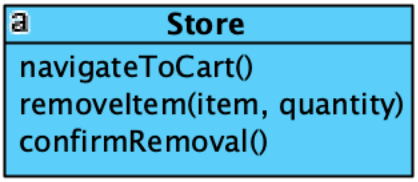
Extensions (or Alternative Flows):

4a. There are multiple of the same item in the cart
    1. Customer selects how many of the item they would like to remove
    2. Customer confirms the amount to remove

6a. The customer doesn't want to remove the item from cart
    1. The customer changes their mind and selects no

7a. There is no resulting change in the order
    1. The subtotal is not changed and thus not updated


:Customer
: Store
1: navigateToCart()
1.1: cart page
2: removeItem(item, quantity)
2.1: updated cart
3: confirmRemoval()
3.1: new subtotal



Operation: navigateToCart()
Cross References: Remove Item from Cart
Preconditions: The cart exists
Postconditions: The cart is displayed to the user

Operation: removeItem(item, quantity)
Cross References: Remove Item from Cart
Preconditions: there are at least “quantity” instances of “item” in the carty
Postconditions: the item is

Operation: confirmRemoval()
Cross References: Remove Item from Cart
Preconditions: The customer has selected to remove an item from their cart
Postconditions: If confirmed, the removal is finalized
Else, the removal is cancelled

Use Case Name: Make Account

Scope: Store

Level: user-goal

Primary Actor: Customer

Stakeholders and Interests:

- Customer: wants to store their information so that it's saved for next time
- Company: has to keep track of the customer's login information

Preconditions: Customer is asked if they want to create an account

Success Guarantee (or Postconditions): The customer's login information is saved and their cart is tied to their account

Main Success Scenario (or Basic Flow):

1. Customer selects they want to create an account
2. The customer is prompted for a username
3. The customer is prompted for a password

Repeat steps 2 and 3 until the login is validated

4. The user is asked to re-enter their username and password for validation
5. The customer is asked to confirm the account creation

Extensions (or Alternative Flows):

2a. The username is invalid

1. The customer is informed why the username is invalid (either it is taken or does not meet requirements)
2. The customer is asked to enter another username

3a. The password is invalid

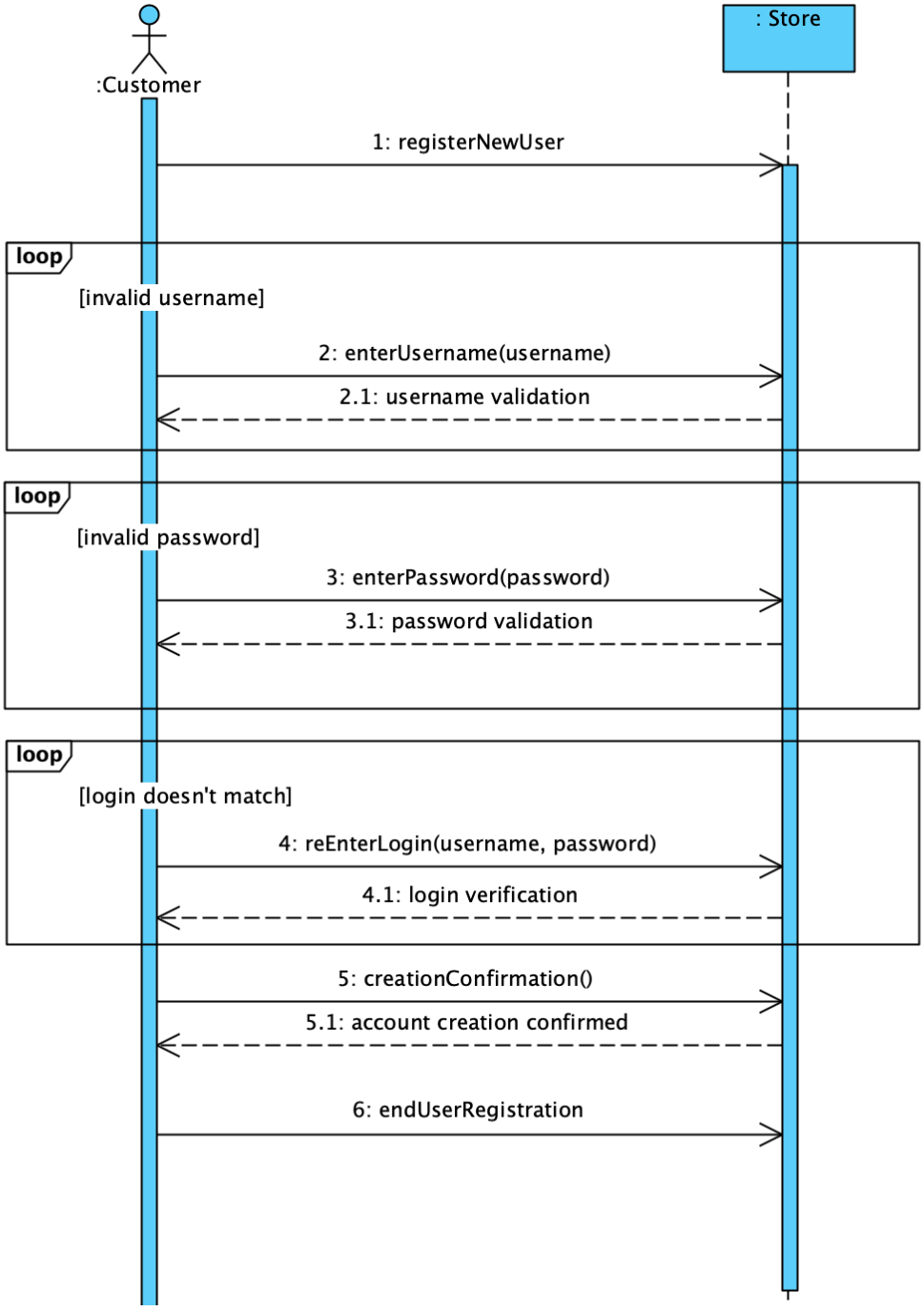
1. The customer is informed why the password is invalid (i.e. the requirements it violates)
2. The customer is asked to enter another password

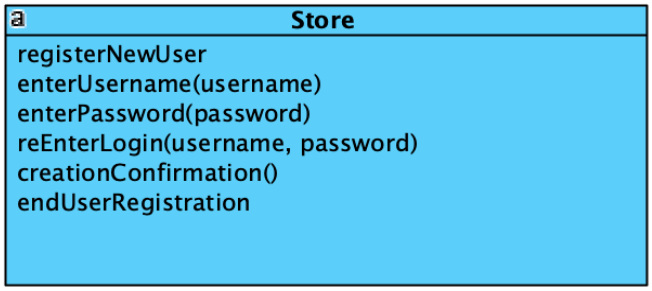
4a. The customer's username does not match

1. The customer is informed that their usernames don't match
2. The customer is prompted to try again

4b. The customer's password doesn't match

1. The customer is informed that their passwords don't match
2. The customer is prompted to try again


:Customer
: Store
1: registerNewUser
loop
[invalid username]
2: enterUsername(username)
2.1: username validation
loop
[invalid password]
3: enterPassword(password)
3.1: password validation
loop
[login doesn't match]
4: reEnterLogin(username, password)
4.1: login verification
5: creationConfirmation()
5.1: account creation confirmed
6: endUserRegistration



Operation: registerNewUser
Cross References: Make Account
Preconditions: The customer has elected to create an account.
Postconditions: A new account is registered.

Operation: enterUsername(username)
Cross References: Make Account
Preconditions: The customer is prompted to enter a username.
Postconditions: The username is validated.

Operation: enterPassword(password)
Cross References: Make Account
Preconditions: The customer is prompted to enter a password.
Postconditions: The password is validated.

Operation: reEnterLogin(username, password)
Cross References: Make Account
Preconditions: The customer is prompted to re-enter their valid username and password
Postconditions: The two logins are compared, then validated if they match

Operation: creationConfirmation()
Cross References: Make Account
Preconditions: The two logins matched
Postconditions: If the user confirms, the account is created
Else, account creation is canceled

Operation: endUserRegistration
Cross References: Make Account
Preconditions: The registration was sucessful
Postconditions: The process of creating an account ends

Use Case Name: View Product Availability

Scope: Store

Level: Subfunction

Primary Actor: Company

Stakeholders and Interests:

- Customer: Wants to have the option to pick from a variety of items and have what is needed available.
- Company: Wants to be able to properly track the inventory of all of the products available.

Preconditions: The website is up and running and the product being looked for is valid and have an unique id and product page

Success Guarantee (or Postconditions): The availability of a product is checked and updated if needed

Main Success Scenario (or Basic Flow):

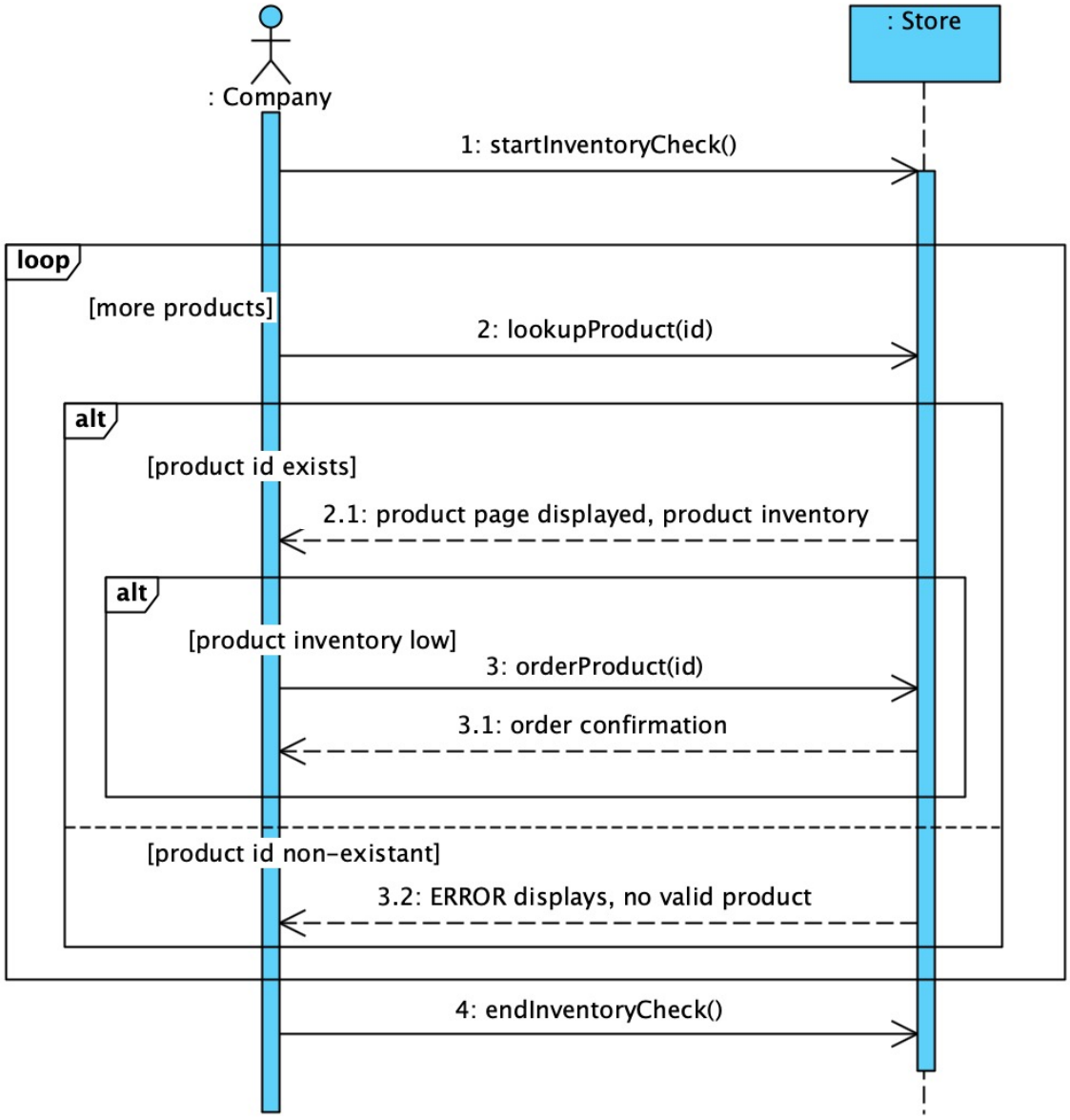
1. A product inventory is needed to be checked
2. The product is looked up based on its unique id
3. The inventory available will be displayed on the product page
4. The company will request more inventory to be shipped to the containment center if product is getting low.

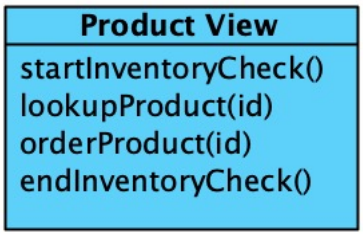
Steps 2-4 can be repeated if multiple products are having their inventory checked

Extensions (or Alternative Flows):

2a. The unique id does not exist

1. The product being looked up doesn't exist and results an ITEM NOT FOUND message


: Company
: Store
1: startInventoryCheck()
loop
[more products]
2: lookupProduct(id)
alt
[product id exists]
2.1: product page displayed, product inventory
alt
[product inventory low]
3: orderProduct(id)
3.1: order confirmation
[product id non-existant]
3.2: ERROR displays, no valid product
4: endInventoryCheck()



Operation: startInventoryCheck()
Cross References: View Product availability
Preconditions: Website is up and running and has products with different product ids
Postconditions: The inventory checking has been started

Operation: lookupProduct(id)
Cross References: View Product availability
Preconditions: Product exists in the database and is coupled with a unique id
Postconditions: Product page is displayed from database

Operation: orderProduct(id)
Cross References: View Product availability
Preconditions: Product exists in the database and is coupled with a unique id
Postconditions: Product is ordered to wearhouse

Operation: endInventoryCheck()
Cross References: View Product availability
Preconditions: All products need to be inventory checked have been
Postconditions: The inventory checking has been finished

Use Case Name: Delete a Product

Scope: Store

Level: Subfunction

Primary Actor: Company

Stakeholders and Interests:

- Customer: Wants to have the option to pick from a variety of items and have what is needed available.
- Company: Wants to remove product from the store to generate more revenue and make more space for more popular items

Preconditions: There is a product the company wants to remove

Success Guarantee (or Postconditions): The product is removed from the site

Main Success Scenario (or Basic Flow):

1. A product is deemed needed to be removed from the store
2. The product is deleted from the sites database based on its unique id
3. The products id is freed to be used by another future product
4. The products page is removed as result of a freed unique id

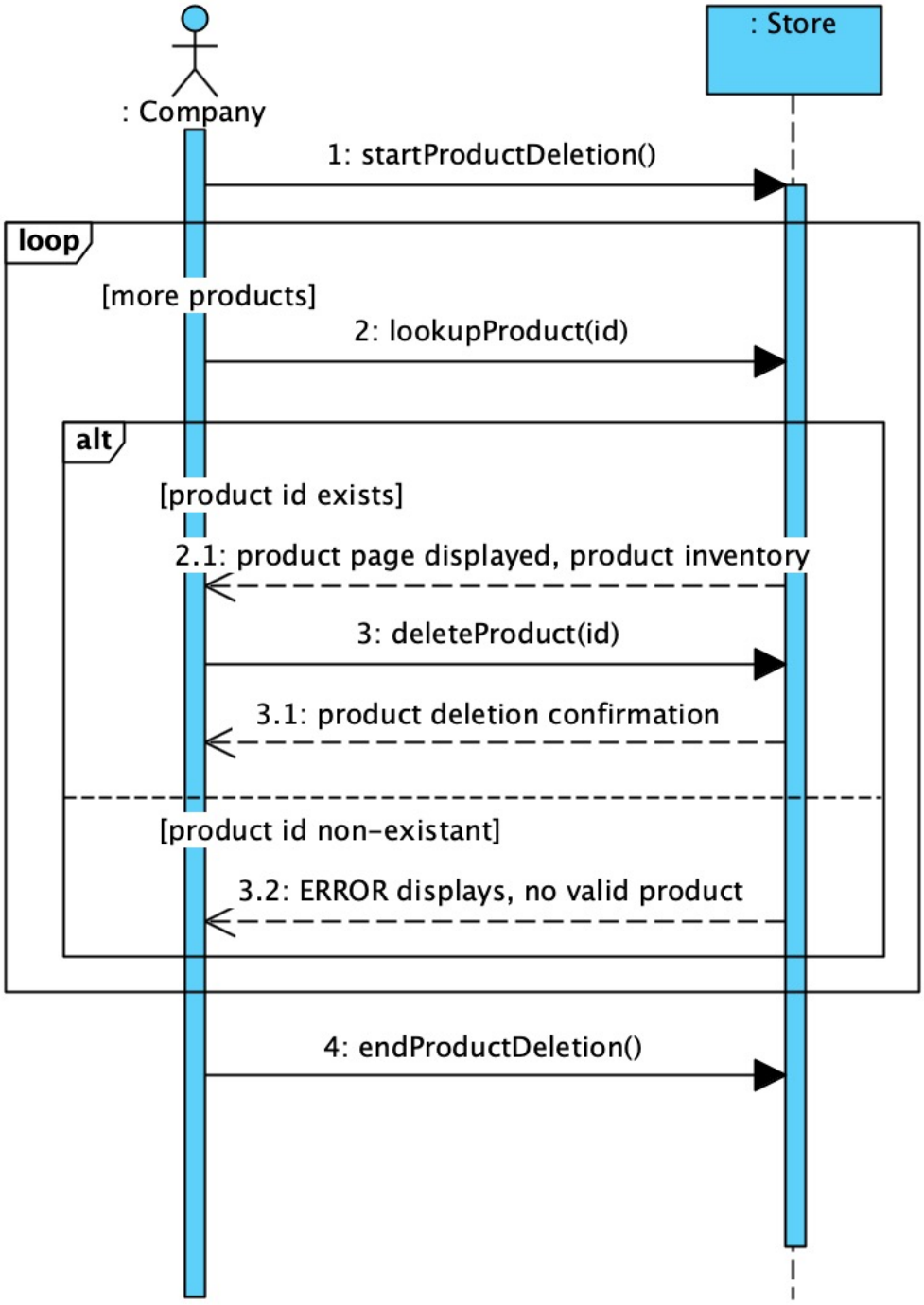
Steps 2-4 can be repeated if multiple products are being deleted

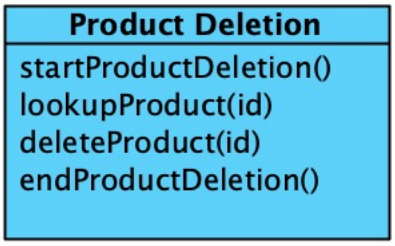
Extensions (or Alternative Flows):

*a. At any point the products removal process can stop, resulting in leaving the product on the store

2a. The unique id does not exist

1. The product being looked up doesn't exist and results an ITEM NOT FOUND message


: Company
: Store
1: startProductDeletion()
loop
[more products]
2: lookupProduct(id)
alt
[product id exists]
2.1: product page displayed, product inventory
3: deleteProduct(id)
3.1: product deletion confirmation
[product id non-existant]
3.2: ERROR displays, no valid product
4: endProductDeletion()



Operation: startProductDeletion()
Cross References: Product Deletion
Preconditions: Website is up and running and has products with different product ids
Postconditions: The product deletion has been started

Operation: lookupProduct(id)
Cross References: Product Deletion
Preconditions: Product exists in the database and is coupled with a unique id
Postconditions: Product page is displayed from database

Operation: deleteProduct(id)
Cross References: Product Deletion
Preconditions: Product exists in the database and is coupled with a unique id
Postconditions: Product is deleted from database

Operation: endProductDeletion()
Cross References: Product Deletion
Preconditions: All products needed to be deleted have been
Postconditions: The inventory deletion has been finished

Use Case Name: Stock Item

Scope: Store

Level: subfunction

Primary Actor: Company

Stakeholders and Interests:

- Company: wants to be able to add their item to the shop so it can be sold
- Customer: wants to be able to purchase the item once it is stocked

Preconditions: The item is out of stock

Success Guarantee (or Postconditions): The item is stocked to the appropriate level

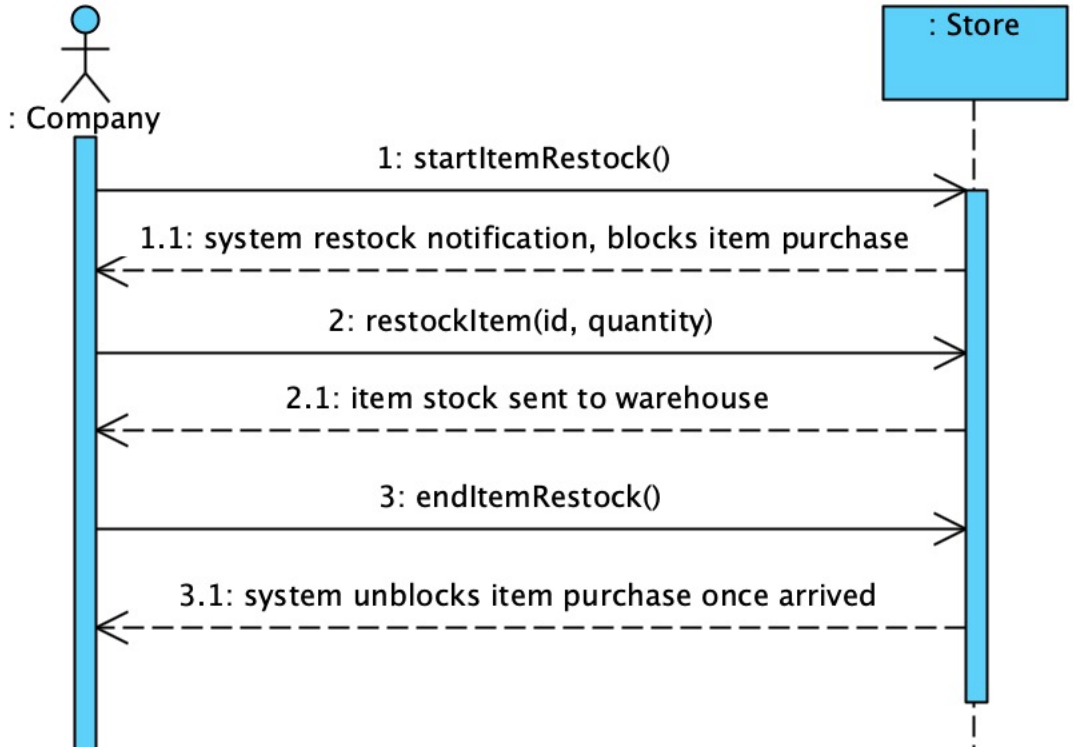
Main Success Scenario (or Basic Flow):

1. The system notifies the company that the item is out of stock
2. The system marks the item as unavailable to purchase
3. The company gets more of the item
4. The company sends the ordered items to the store's warehouse
5. Company updates the amount available in the system
6. System refreshes the item's stock
7. The item is made available for sale again

Extensions (or Alternative Flows):

3a. Company decides not to add more of the item

    1. The system confirms the choice, and resends the message in a week


: Company
: Store
1: startItemRestock()
1.1: system restock notification, blocks item purchase
2: restockItem(id, quantity)
2.1: item stock sent to warehouse
3: endItemRestock()
3.1: system unblocks item purchase once arrived

Operation: startRestockItem()
Cross References: Restock Item
Preconditions: Website is up and running and has products with different product ids
Postconditions: Restock has been started

Operation: itemRestock(id, quantity)
Cross References: Restock Item
Preconditions: Product exists in the database and is coupled with a unique id
Postconditions: Product is ordered to wearhouse in the specific quantity mentioned

Operation: endRestockItem()
Cross References: Restock Item
Preconditions: All products needed to be restocked have been
Postconditions: Restock has been finished

Use Case Name: Add Item to Wishlist

Scope: Customer

Level: user-goal

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants to add the product to their wishlist.

Preconditions: The product must be in the store.

Success Guarantee (or Postconditions): The product will be added to the user's wishlist.

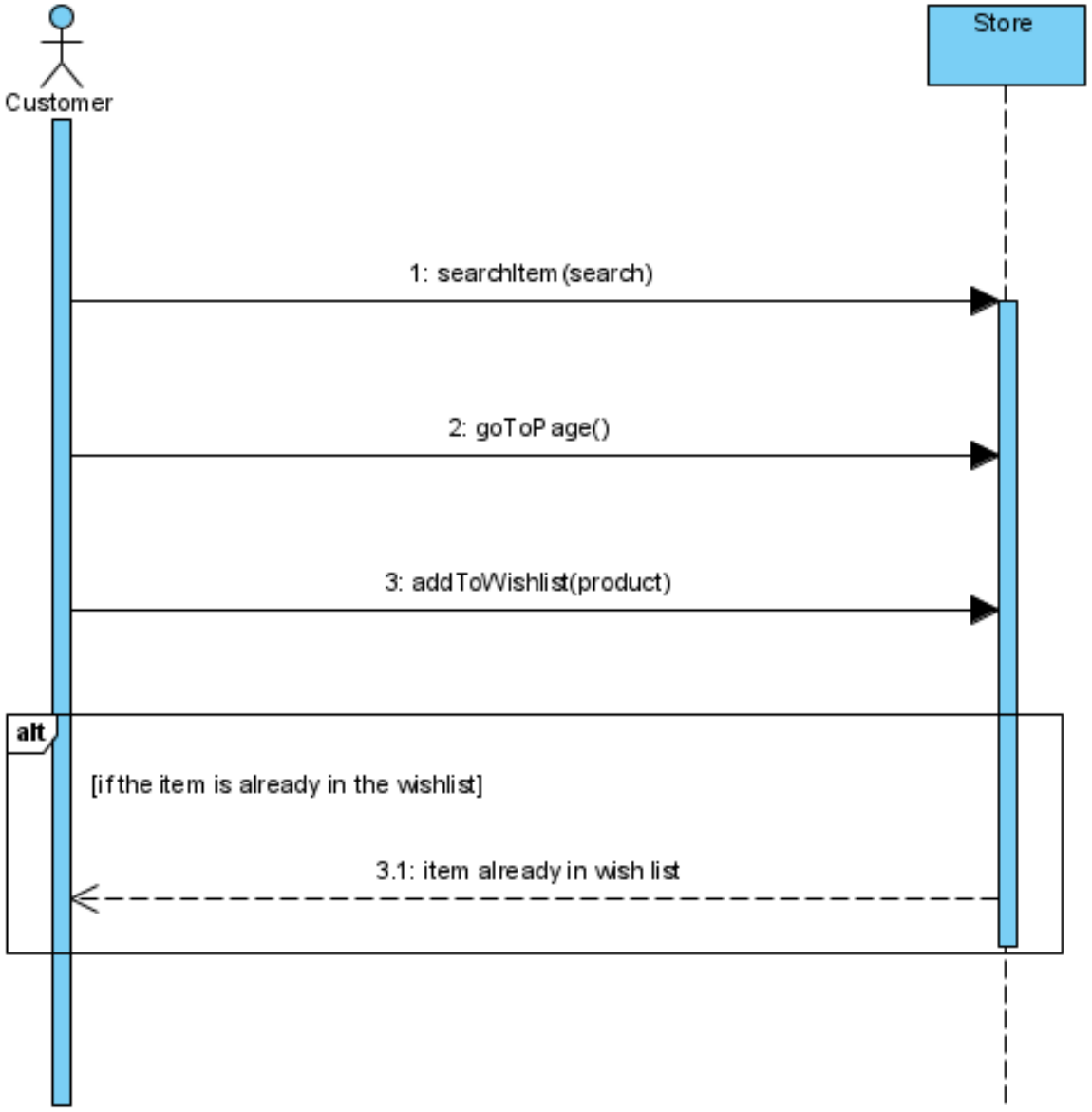
Main Success Scenario (or Basic Flow):

1. User searches for the product.
2. User clicks on the product page.
3. User clicks the "add to wishlist" button,
4. Product is added to the user's wishlist.

Extensions (or Alternative Flows):

4a. The product is already in the user's wishlist.

1. The site tells the user that the item is already in their wishlist.


Customer
Store
1: searchItem(search)
2: goToPage()
3: addToWishlist(product)
alt
[if the item is already in the wishlist]
3.1: item already in wish list

Operation: searchItem
Cross References: Add Item To Wishlist
Preconditions: The customer wants to search for an item
Postconditions: The customer searched for their item

Operation: goToPage
Cross References: Add Item To WishList
Preconditions: The item page exists
Postconditions: The customer has the item page on their screen

Operation: addToWishList
Cross References: Add Item To Wishlist
Preconditions: The customer has an account
Postconditions: The item is in the customer's wishlist

Use Case Name: Ship/Send Item

Scope: Store

Level: Subfunction

Primary Actor: Customer

Stakeholders and Interests:

- Company: The company makes profit if the item is successfully delivered and not returned.
- Customer: The customer is the one who ordered the item.

Preconditions: An item is ordered by the customer.

Success Guarantee (or Postconditions): The item is delivered to the customer.

Main Success Scenario (or Basic Flow):

1. The customer orders a product.
2. The customer inputs their shipping address.
3. The item is paid for.
4. The item is sent to the shipper with the correct address.

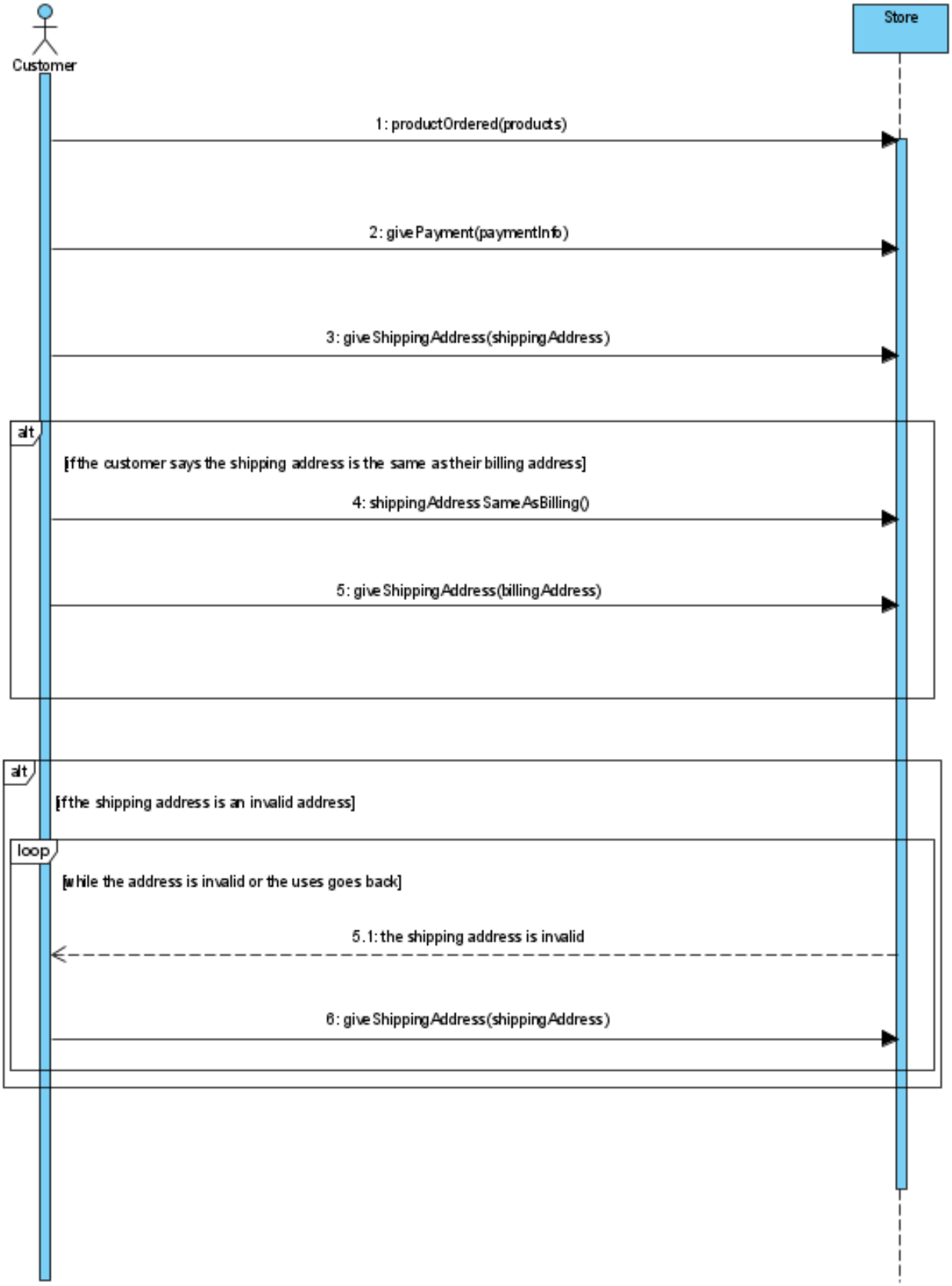
Extensions (or Alternative Flows):

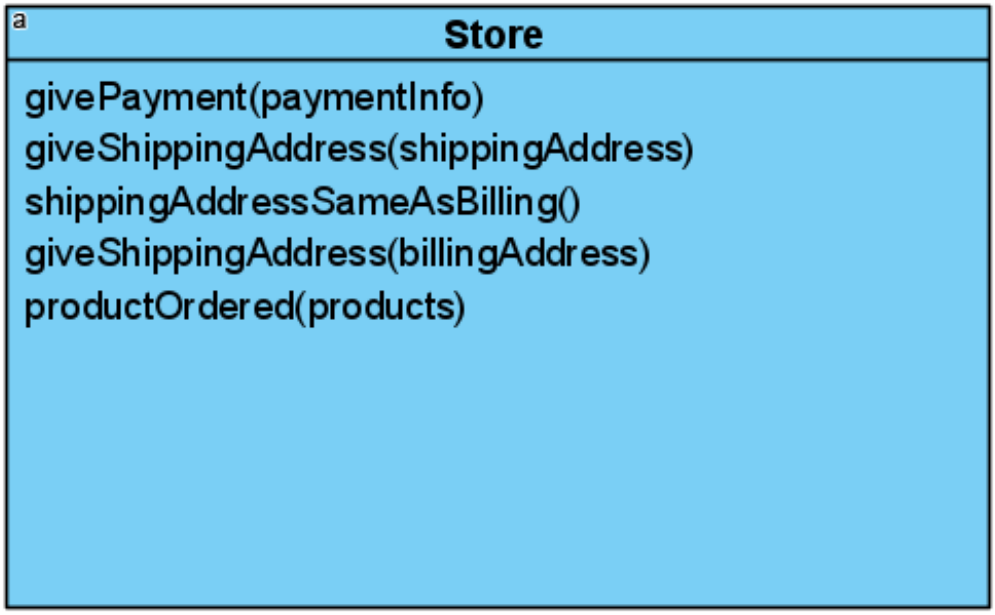
2a. The customer selects that their shipping address is the same as their billing address.

1. The store will fill in the shipping address with the billing address.

2b. The address is not valid.

1. The site will tell the customer that the address is not valid.
2. The site will not let them proceed till a valid address is input.


Customer
Store
1: productOrdered(products)
2: givePayment(paymentInfo)
3: giveShippingAddress(shippingAddress)
alt
[if the customer says the shipping address is the same as their billing address]
4: shippingAddressSameAsBilling()
5: giveShippingAddress(billingAddress)
alt
[if the shipping address is an invalid address]
loop
[while the address is invalid or the uses goes back]
5.1: the shipping address is invalid
6: giveShippingAddress(shippingAddress)



Operation: givePayment
Cross References: Ship/Send Item
Preconditions: The customer is buy an item
Postconditions: The store has the customer's payment info

Operation: giveShippingAddress
Cross References: Ship/Send Item
Preconditions: The customer bought an item
Postconditions: The store has the customer's shipping address

Operation: shippingAddressSameAsBilling
Cross References: Ship/Send Item
Preconditions: The customer is buying an item
Postconditions: The store has the customer's shipping address

Operation: productOrdered
Cross References: Ship/Send Item
Preconditions: The customer is buying an item
Postconditions: The item is purchased

Use Case Name: Add New Product

Scope: Store

Level: Subfunction

Primary Actor: Company

Stakeholders and Interests:

- Company: The one adding the product to the store.
- Customer: The one that will buy the product.
- Shipper: The place that ships an item to the customer.

Preconditions: There is a product the company wants to add.

Success Guarantee (or Postconditions): The product is added to the site.

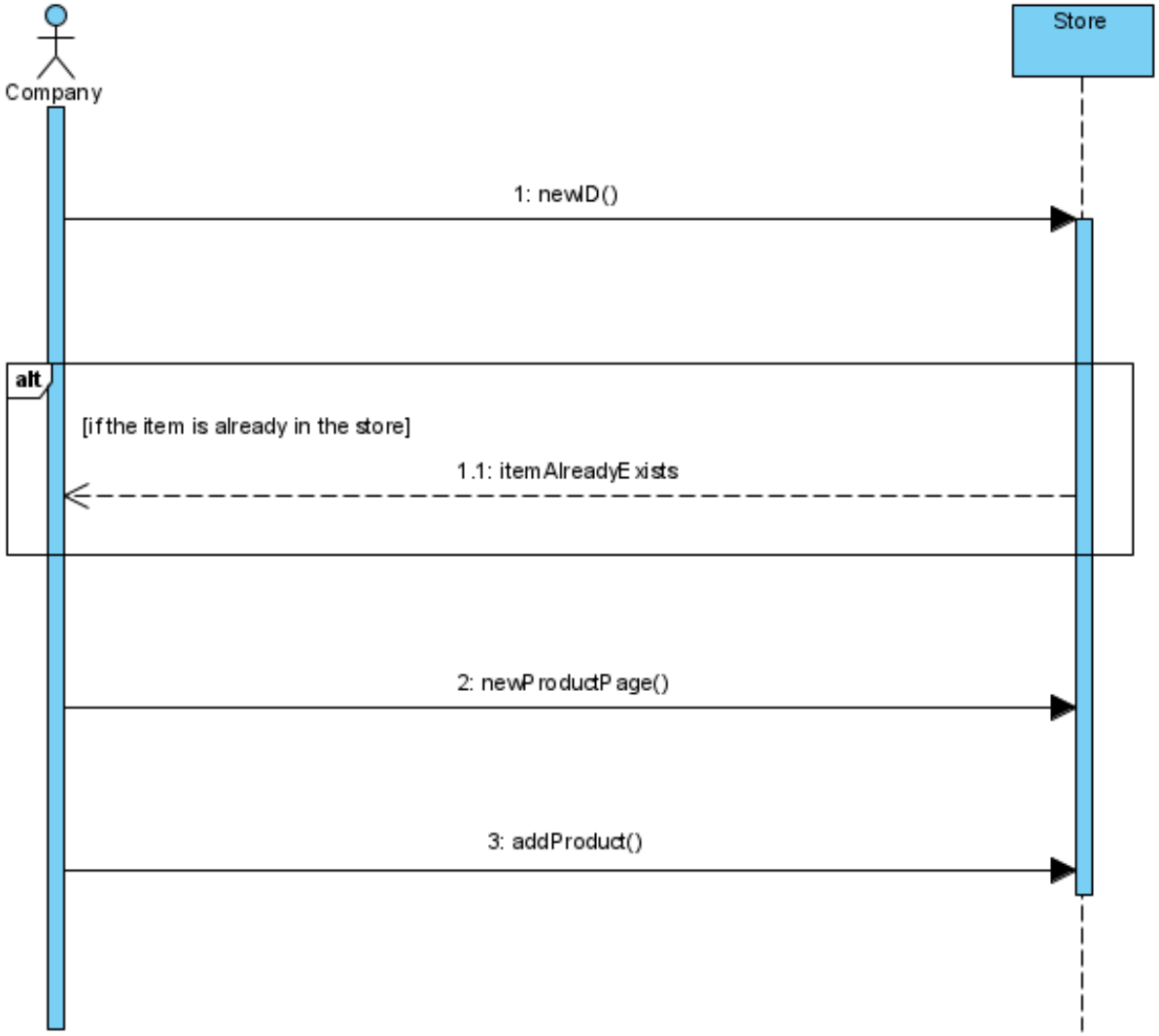
Main Success Scenario (or Basic Flow):

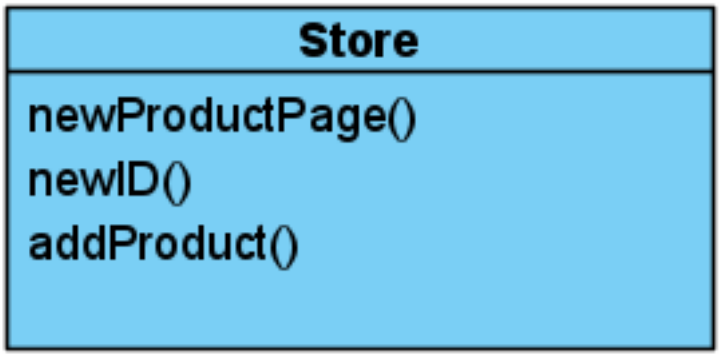
1. A product is chosen to be added to the store.
2. The product is given a unique id.
3. The product is given a product page based off of a template.
4. The product is added to the store.

Extensions (or Alternative Flows):

1a. The product is already in the store.

1. The system will tell the company that the product already exists.


Company
Store
1: newID()
alt
[if the item is already in the store]
1.1: itemAlreadyExists
2: newProductPage()
3: addProduct()



Operation: newProductPage
Cross References: Add New Product
Preconditions: There is a new product to add
Postconditions: A new product page is created

Operation: newID
Cross References: Add New Product
Preconditions: There is a new product
Postconditions: The new product has its own unique ID

Operation: addProduct
Cross References: Add New Product
Preconditions: There is a fully ready new product to add
Postconditions: The new product is added to the store

Use Case Name: Make Purchase

Scope: Store

Level: user goal

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Wants to buy product
- Company: When an item is purchased, revenue goes back to the owner

Preconditions: The item wanted by the customer must be in stock

Success Guarantee (or Postconditions):

The customer is successfully able to purchase the product they wanted

Main Success Scenario (or Basic Flow):

1. Customer searches for an item from the store
2. Customer picks out that item from the store
3. Item goes into the Customers' cart
4. Customer goes to their cart
5. Customer checks out and item is purchased

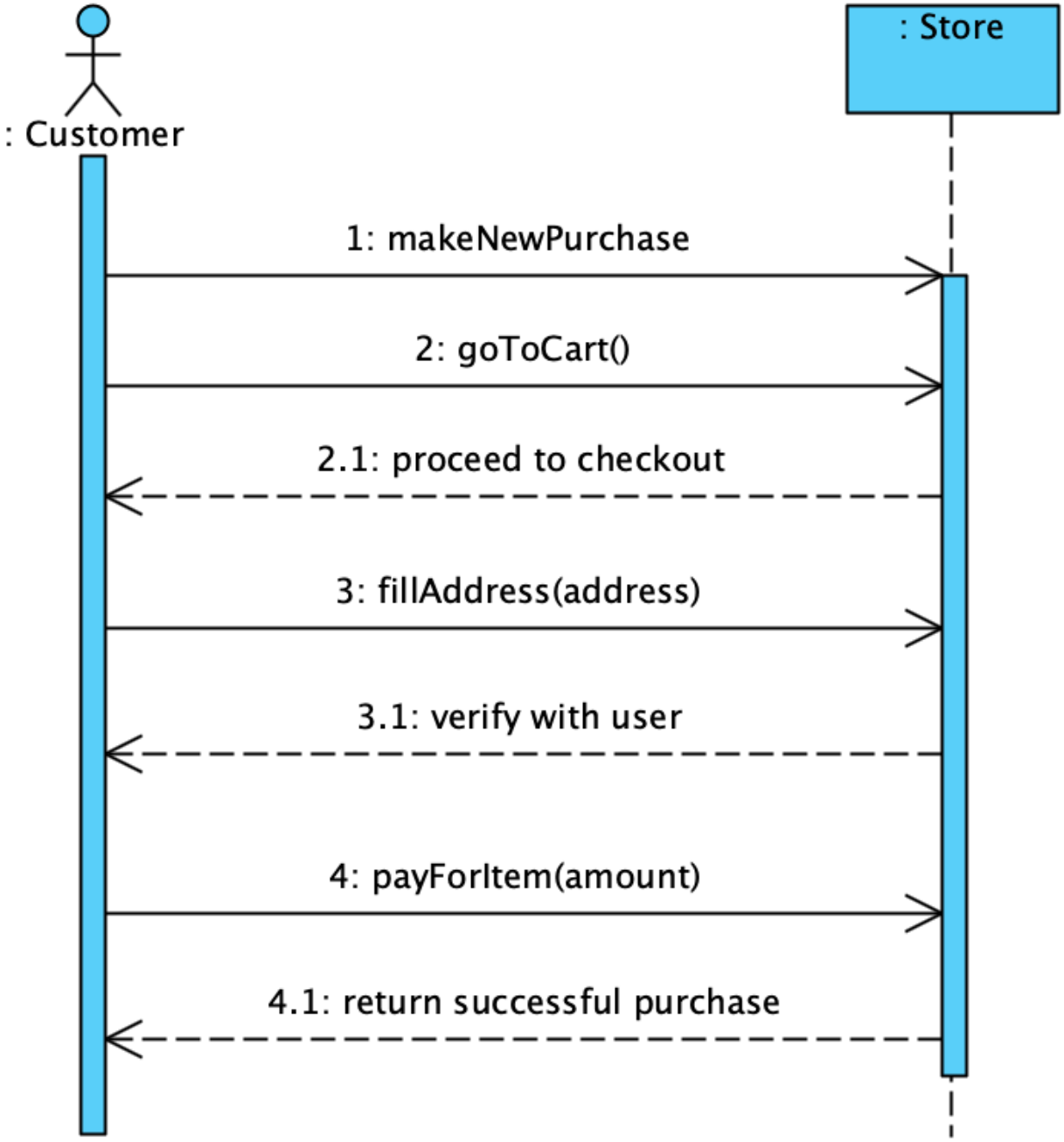
Extensions (or Alternative Flows):

1a. The item isn't sold from the store

1. Customer must request item for the future
2. Customer must find either an off-brand replacement or another store that sells the product

2a. The item is out of stock

1. Customer can request to be notified when the item is back in stock


: Customer
: Store
1: makeNewPurchase
2: goToCart()
2.1: proceed to checkout
3: fillAddress(address)
3.1: verify with user
4: payForItem(amount)
4.1: return successful purchase

Operation: makeNewPurchase
Cross References: Make New Purchase
Preconditions: There is an instance of a purchase created
Postconditions: The item/items are purchased

Operation: goToCart
Cross References: Make New Purchase
Preconditions: The cart exists
Postconditions: The customer is able to open the cart and see the items inside

Operation: fillAddress(address)
Cross References: Make New Purchase
Preconditions: The customer has an address in which they live at
Postconditions: The customer has an address on file to ship to in the future and for the current purchase

Operation: payForItem(amount)
Cross References: Make New Purchase
Preconditions: Customer has items to pay for
Postconditions: The customer pays for their items and the purchase is complete

Use Case Name: Return Item

Scope: Store

Level: user-goal

Primary Actor: Customer

Stakeholders and Interests:

- Customer: Is able to return their product worry free
- Company: Receives product back from customer

Preconditions: Item has been received by the customer in order to return it

Success Guarantee (or Postconditions): The item is successfully returned to the company

Main Success Scenario (or Basic Flow):

1. The customer wishes to return an order
2. The customer requests a return
3. The customer receives a shipment label to use to return the item
4. The customer uses the shipment label to ship the item through a separate shipping company (FedEx, UPS, USPS)
5. The company receives the item returned

Steps 1-5 can be repeated due to how many items they wish to return

Extensions (or Alternative Flows):

2a. The customer is not qualified to return the item

1. Customer must keep the item, donate the item, or throw it away

3a. The shipment label is wrong

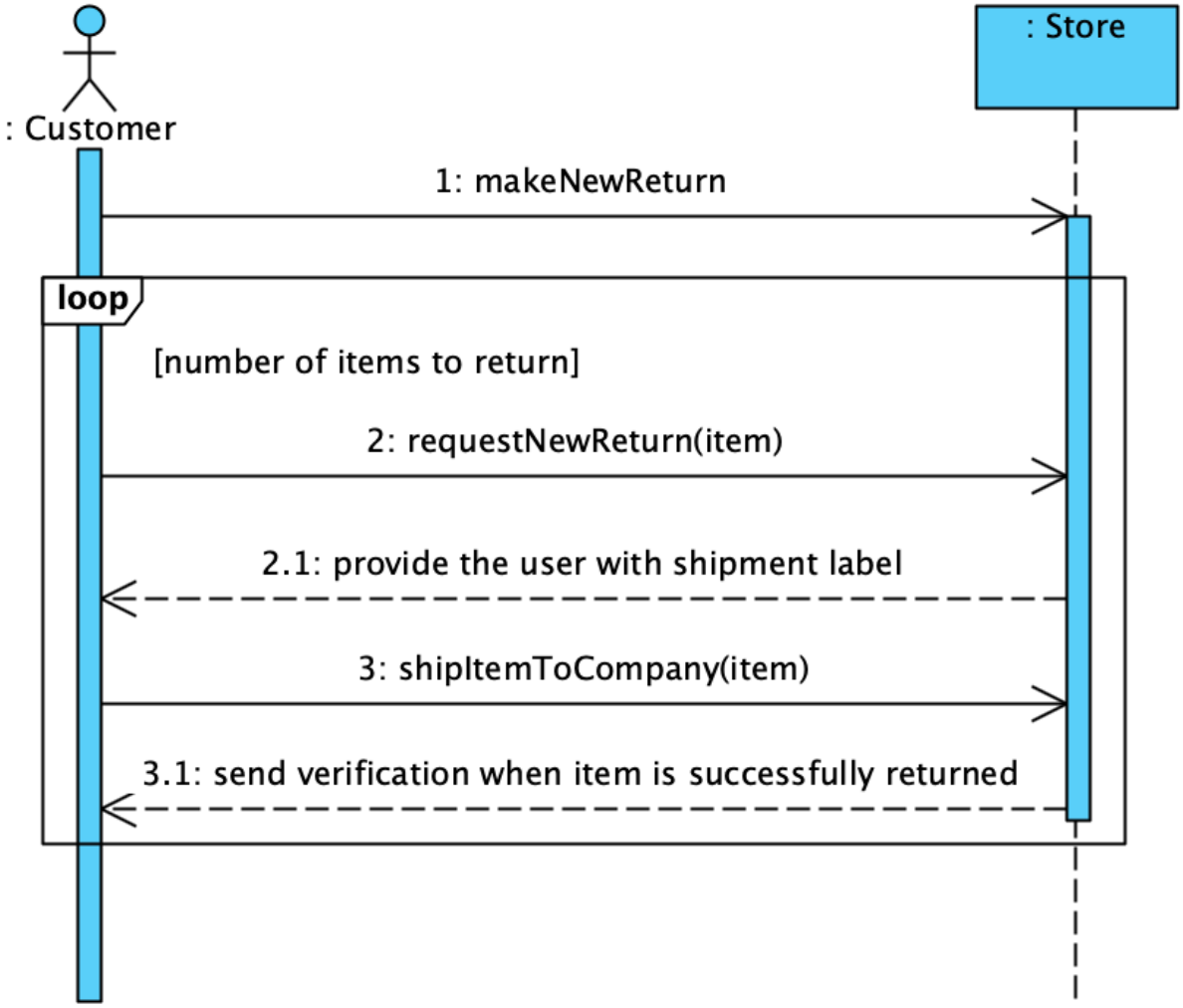
1. Customer must request a new one

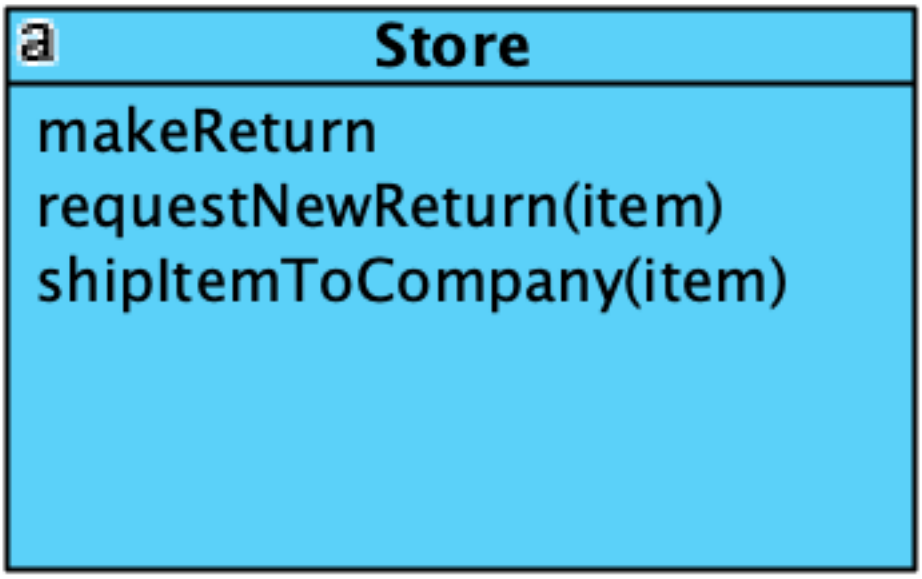
4a. The shipment label is expired

1. The customer must keep the item, donate the item, or throw it away

5a. The item never returns to the company

1. The company must cut their losses due to lost package


: Customer
: Store
1: makeNewReturn
loop
[number of items to return]
2: requestNewReturn(item)
2.1: provide the user with shipment label
3: shipItemToCompany(item)
3.1: send verification when item is successfully returned



Operation: makeReturn
Cross References: Make New Return
Preconditions: An item needs to be returned
Postconditions: The item is successfully returned

Operation: requestNewReturn(item)
Cross References: Make New Return
Preconditions: The customer needs to return an item
Postconditions: The company approves the return request

Operation: shipItemToCompany(item)
Cross References: Make New Return
Preconditions: The customer has a ready shipment label to return the item
Postconditions: The company receives the returned item

Use Case Name: Generate Report

Scope: Store

Level: user goal

Primary Actor: Company

Stakeholders and Interests:

- Company: Wants to visualize their product sales and customer satisfaction/customer usage rate

Preconditions: The website is up and running

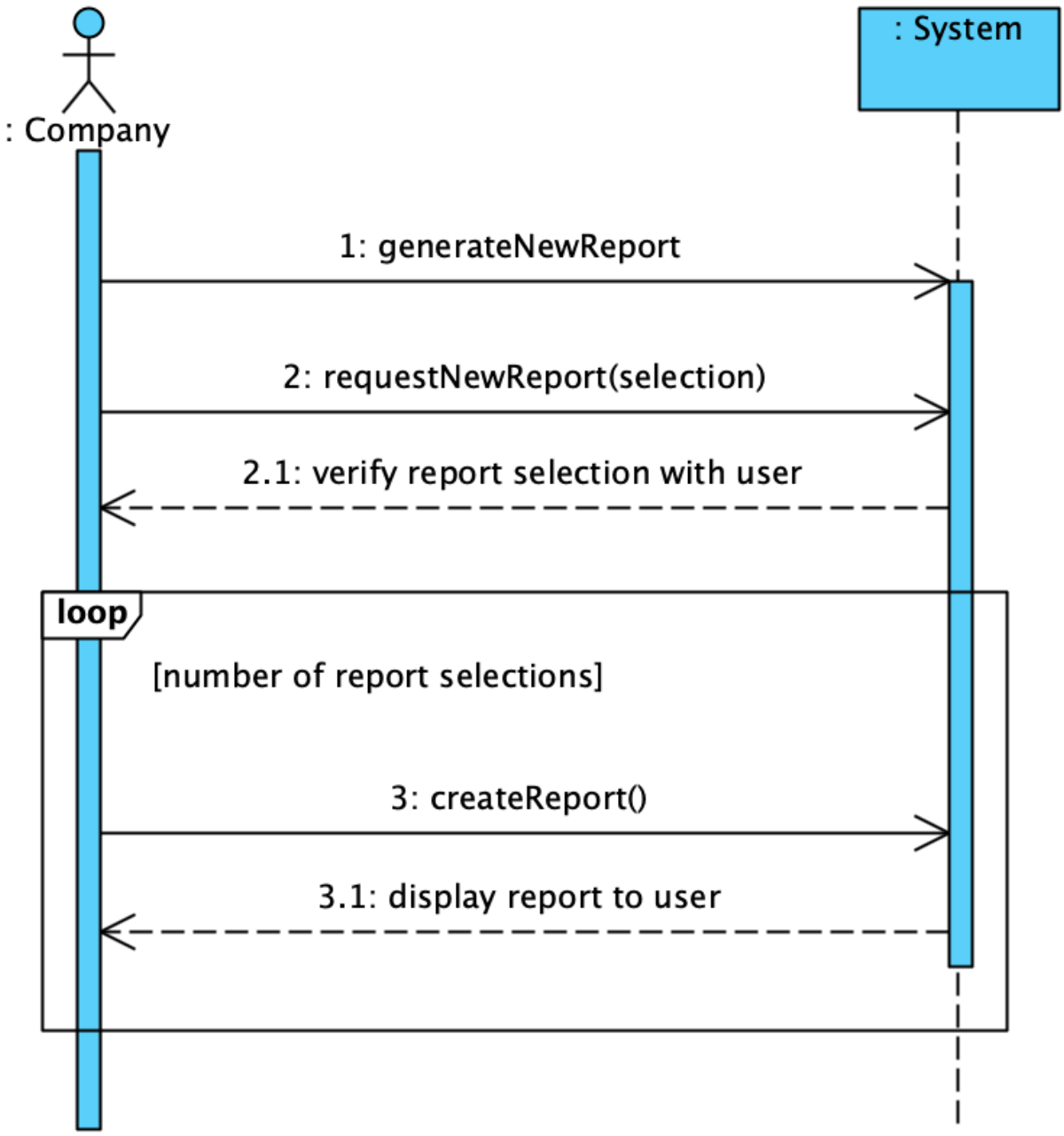
Success Guarantee (or Postconditions): The website produces a report that displays the companys' weekly sales, new customer accounts, and ratings

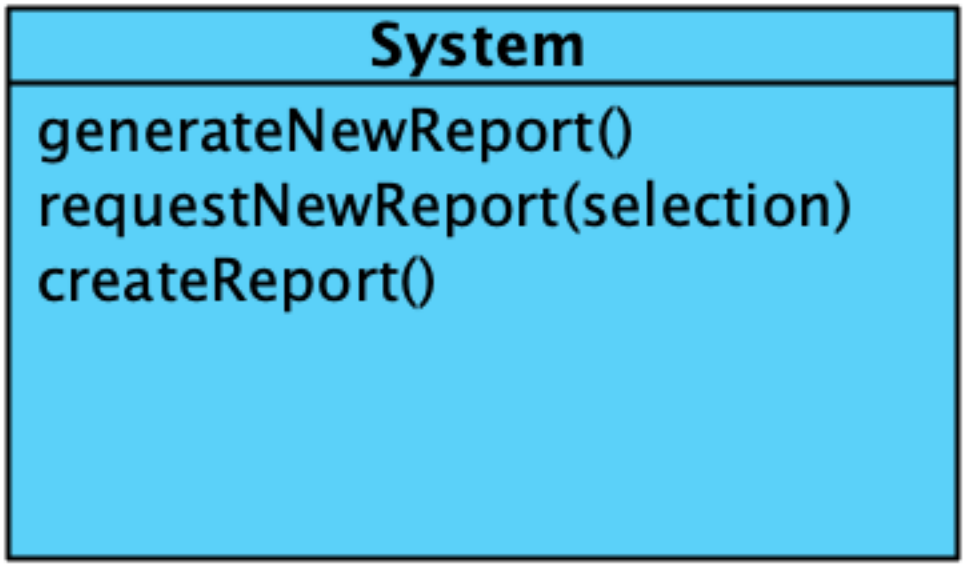
Main Success Scenario (or Basic Flow):

1. The company requests report
    a. The company selects certain aspects to receive a report of
2. The system confirms the selection
3. The report is generated through the system
4. The report is sent to the company

Extensions (or Alternative Flows):

- The system does not respond
    - The system must be restarted manually in order to generate report


: Company
: System
1: generateNewReport
2: requestNewReport(selection)
2.1: verify report selection with user
loop
[number of report selections]
3: createReport()
3.1: display report to user



Operation: generateNewReport()
Cross References: Generate Report
Preconditions: The report does not exist
Postconditions: The report is created

Operation: requestNewReport(selection)
Cross References: Generate Report
Preconditions: The report selection hasn't been made
Postconditions: The report selection is chosen and verified by the system with the user

Operation: createReport()
Cross References: Generate Report
Preconditions: The report has yet to be made
Postconditions: The report is created and sent to the company