

Runtime Terror - Nozama

Project Vision:

This software will benefit the user because it will allow for easy management of an online store's inventory, allowing the store to add/remove items, keep track of the total inventory, keep track of orders. The system will manage the store's inventory allowing the employees to focus on other tasks. The main stakeholders for this project are the customers of the store and the store's employees. The customers will only be able to purchase the items if they are in stock, and the employees will need to know if stock is getting low so that they can order more. The system will have a GUI that allows for the user to look up an item by an ID number and see its stock, a general description, and make a purchase. The software will also need to keep track of the items that the customer has added to their cart, to enable the purchase of multiple items.

Website Link w/ Link to GitHub and Issue Tracking:

<https://runtimeerrorbu.github.io/Nozama/>

Group Members w/ Timecard:

Ashley Bickham:

- 25 hours worked
- 23 total commits

Joshua Hunter (Project Leader):

- 28 hours worked
- 30 total commits

Austin Lehman:

- 31 hours worked
- 35 total commits

Tyler Ross:

- 21 hours worked
- 11 total commits

Total Commits: 127

**All commits at: <https://github.com/RuntimeTerrorBU/Nozama/commits/gh-pages>*

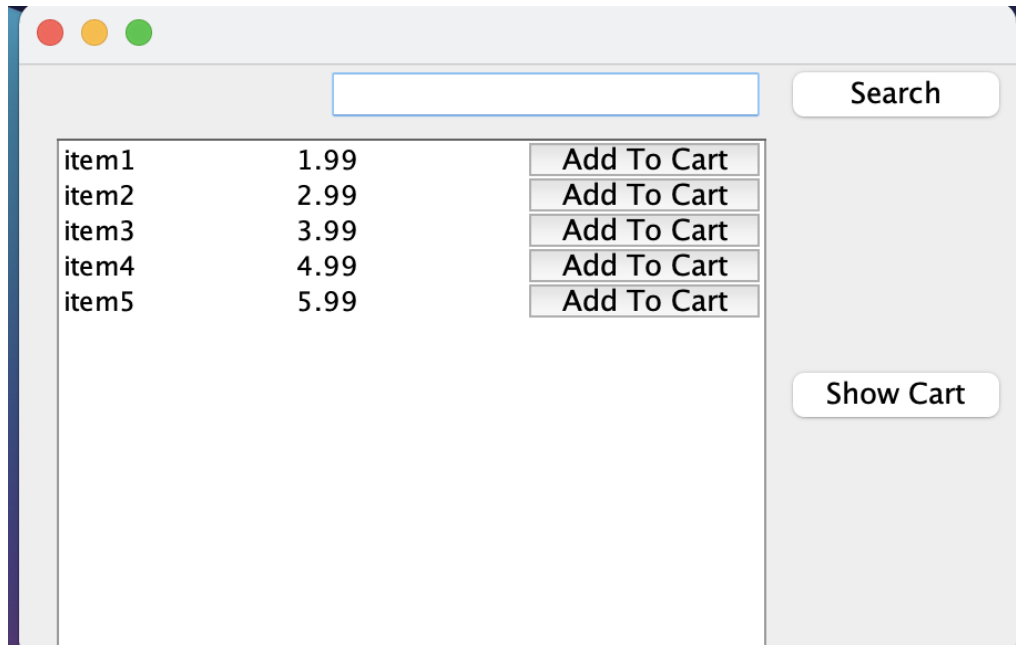
Revised Analysis:

This software that we are creating will be beneficial to those who use it because it allows for easy management of an online store's inventory, permits the store to add or remove items, keep track of the total inventory that the store has, as well as monitor orders placed through the store.

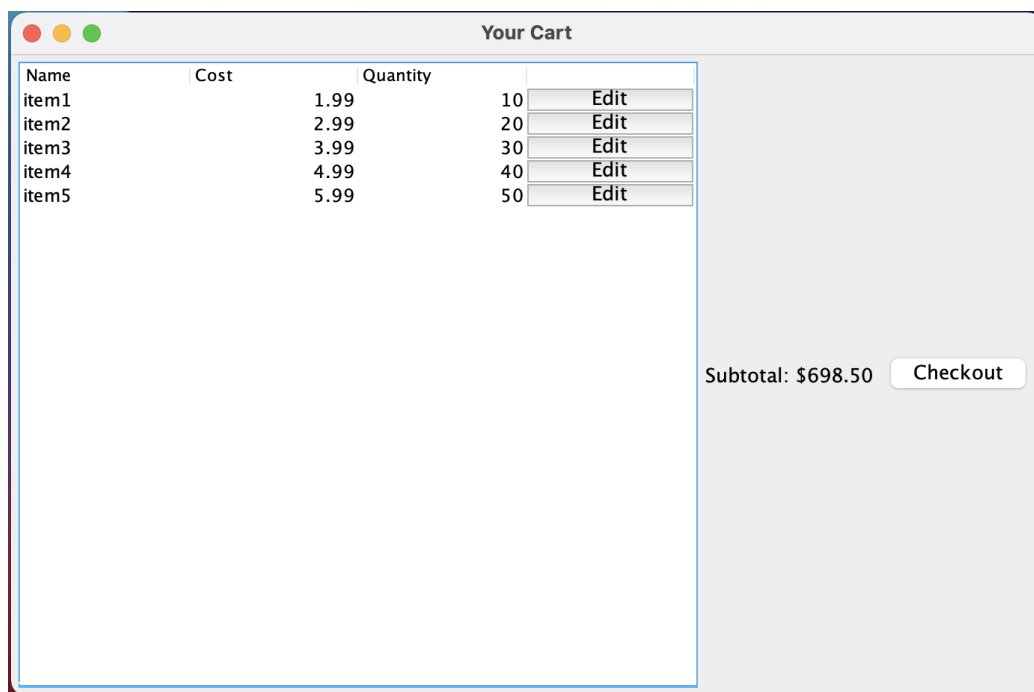
The system we create will supervise the store's inventory, which will make it easier for employees to focus on other tasks in order to make the company as efficient as possible, since one of the main stakeholders of this project is the company, as well as its customers. The software should permit the customers to purchase items that are in stock, keep track of these items, and for the employees to restock items. The GUI included in the system lets a customer look up an item on the store by its ID number and will provide in return the stock of the item, a general description of the item, and the option for the user to purchase the item. This iteration of the project, we focused more on code and program development, as well as monitoring these changes through GitHub and documentation of our work. Through GitHub, we have all been able to work on similar files that are able to be changed collectively each time someone pushes a change to the website.

Prototypes of Interface using Maven:

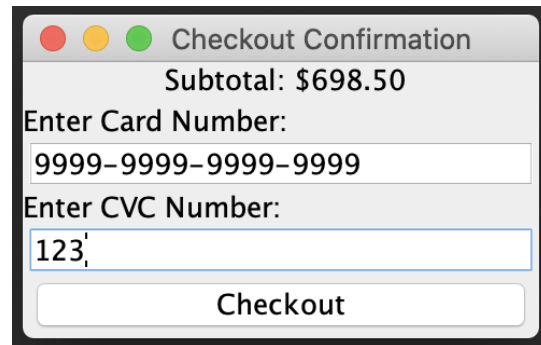
Initial Screen of Interface:



Current Cart Screen:



Checkout Confirmation Screen:



A macOS-style dialog box titled "Checkout Confirmation". It displays a subtotal of \$698.50. Below this, it prompts for a card number, which is entered as "9999-9999-9999-9999". It then prompts for a CVC number, which is entered as "123". At the bottom is a "Checkout" button.

Checkout Confirmation

Subtotal: \$698.50

Enter Card Number:

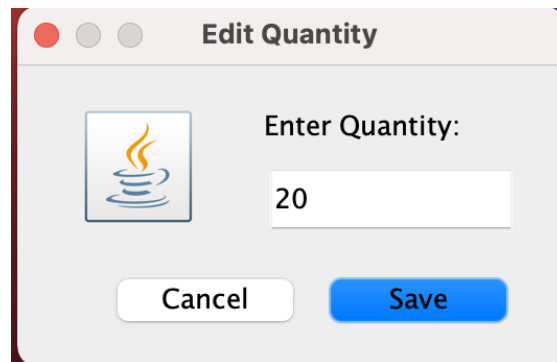
9999-9999-9999-9999

Enter CVC Number:

123

Checkout

Edit Quantity within Cart Screen:



A macOS-style dialog box titled "Edit Quantity". It features a coffee cup icon with steam. To the right, it says "Enter Quantity:" followed by a text input field containing the number "20". At the bottom are "Cancel" and "Save" buttons.

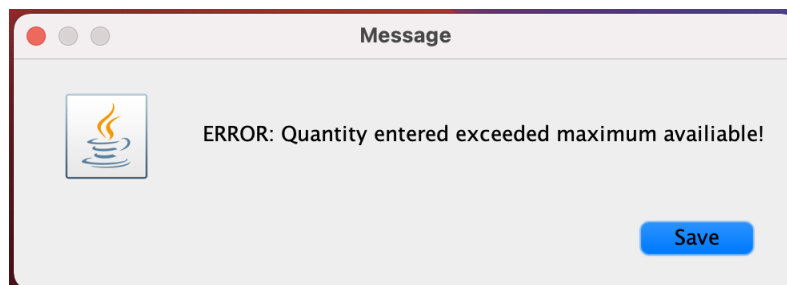
Edit Quantity

Enter Quantity:

20

Cancel Save

Sample Error Screens:

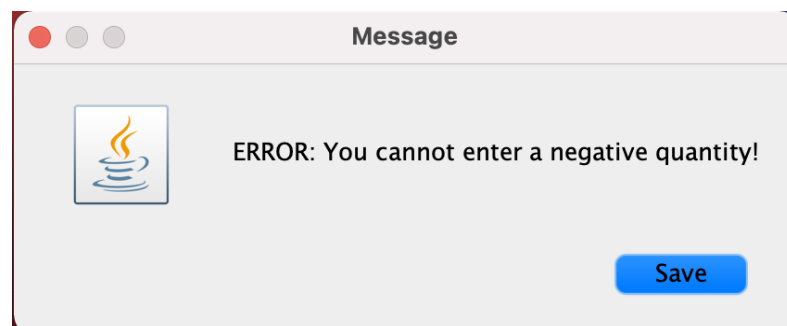


A macOS-style message dialog box titled "Message". It features a coffee cup icon. The text reads "ERROR: Quantity entered exceeded maximum available!". There is a "Save" button at the bottom right.

Message

ERROR: Quantity entered exceeded maximum available!

Save



A macOS-style message dialog box titled "Message". It features a coffee cup icon. The text reads "ERROR: You cannot enter a negative quantity!". There is a "Save" button at the bottom right.

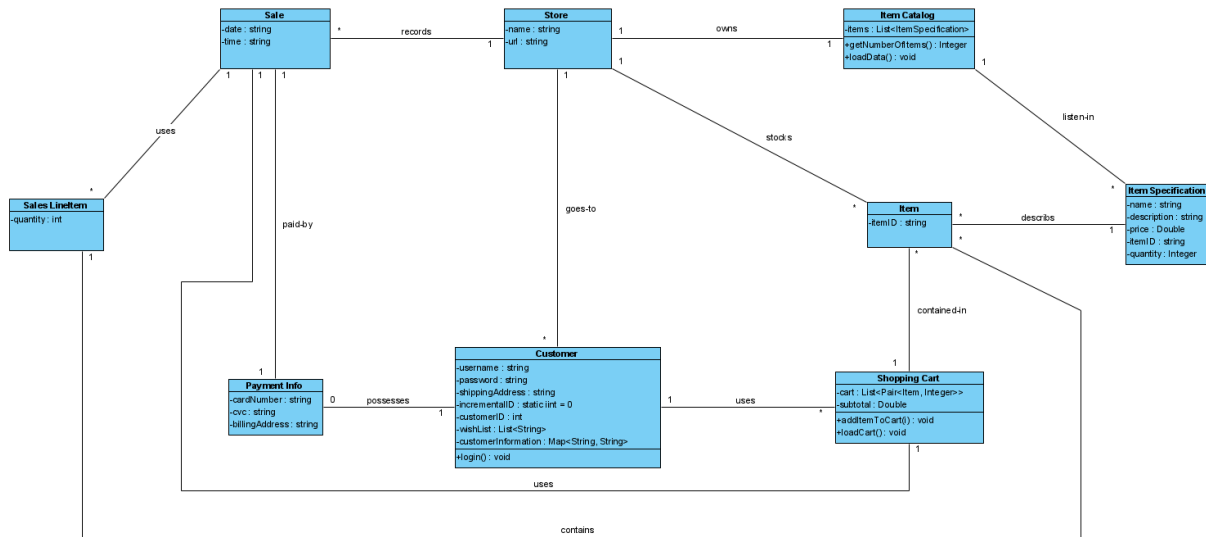
Message

ERROR: You cannot enter a negative quantity!

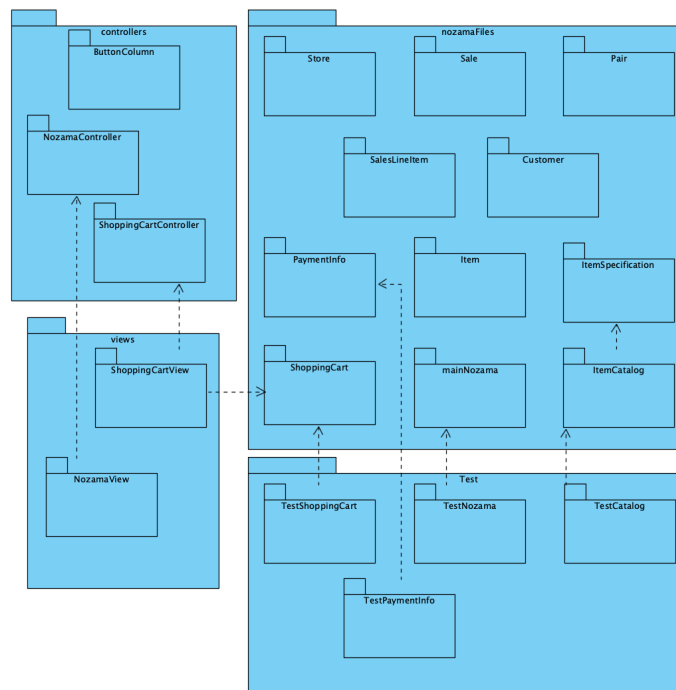
Save

Design diagrams

Design Class diagram:

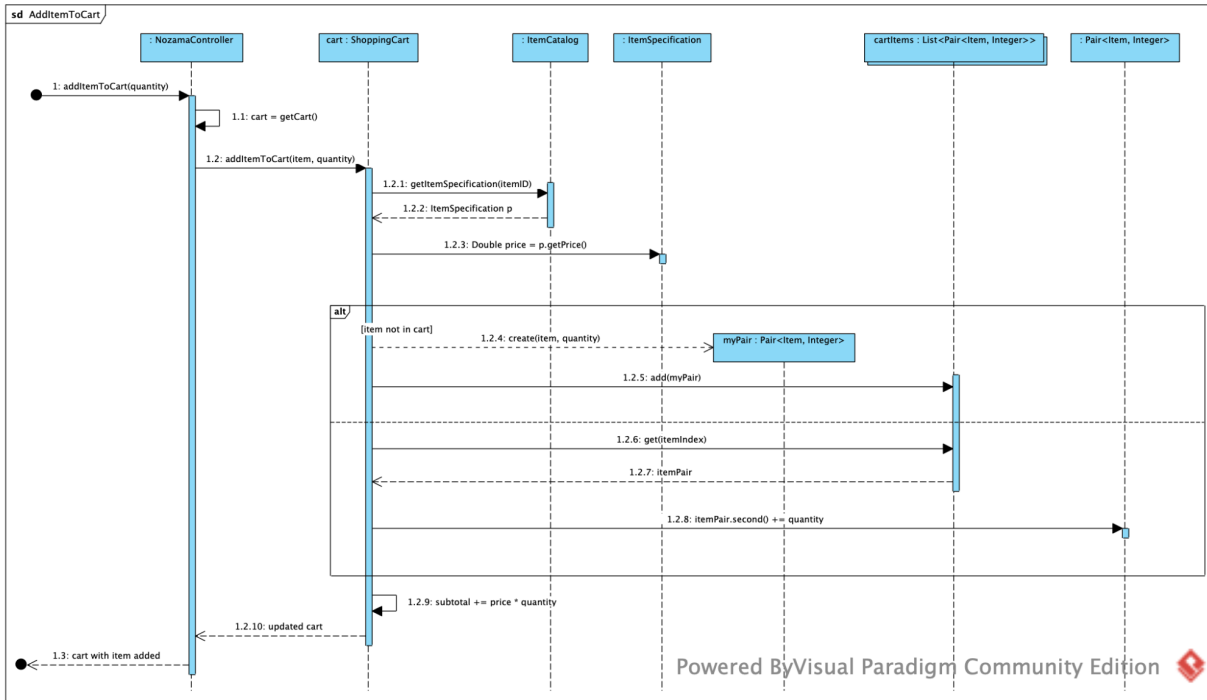


Package Diagram/Architecture:

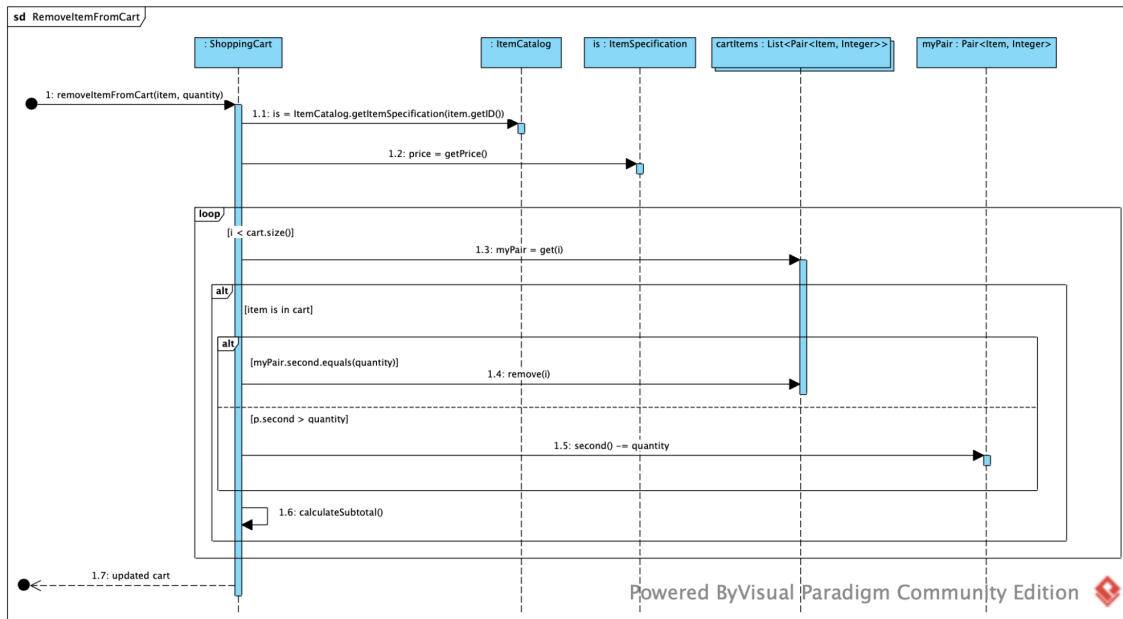


Sequence Diagrams:

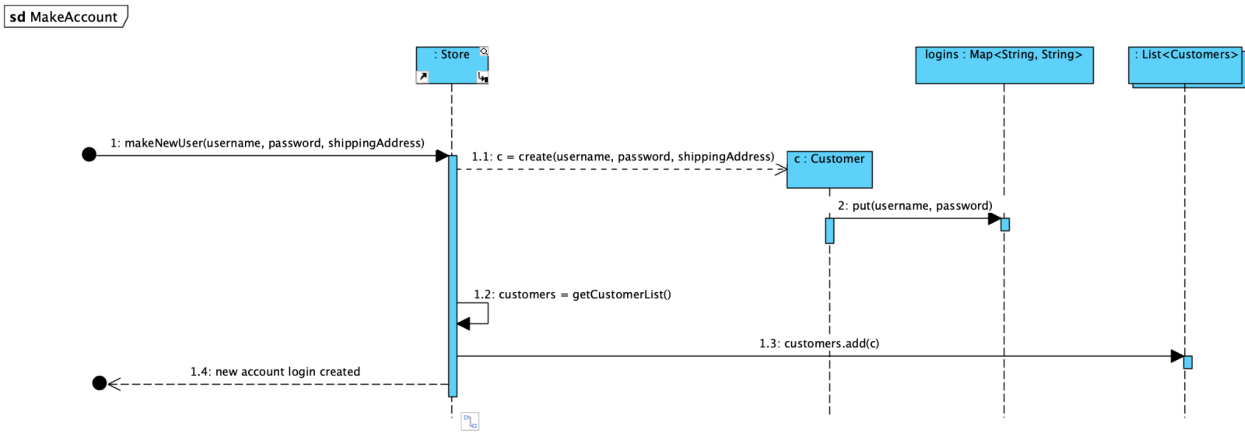
Add Item to Cart:



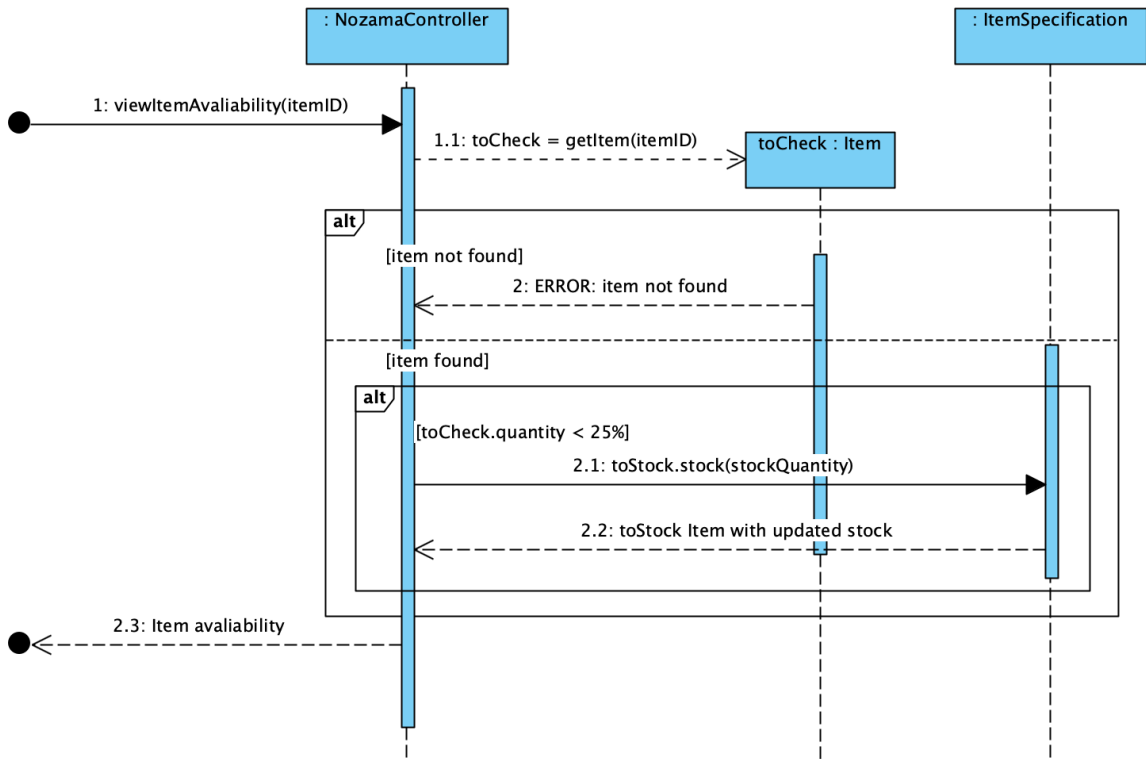
Remove Item from Cart



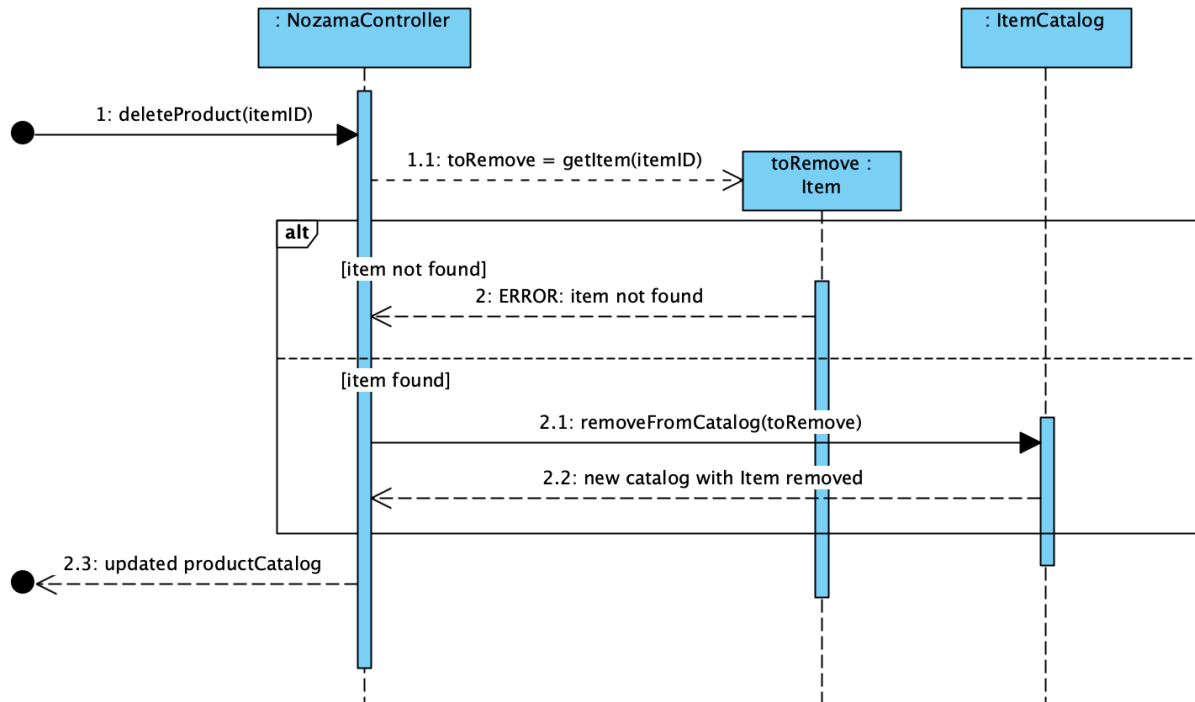
Make Account



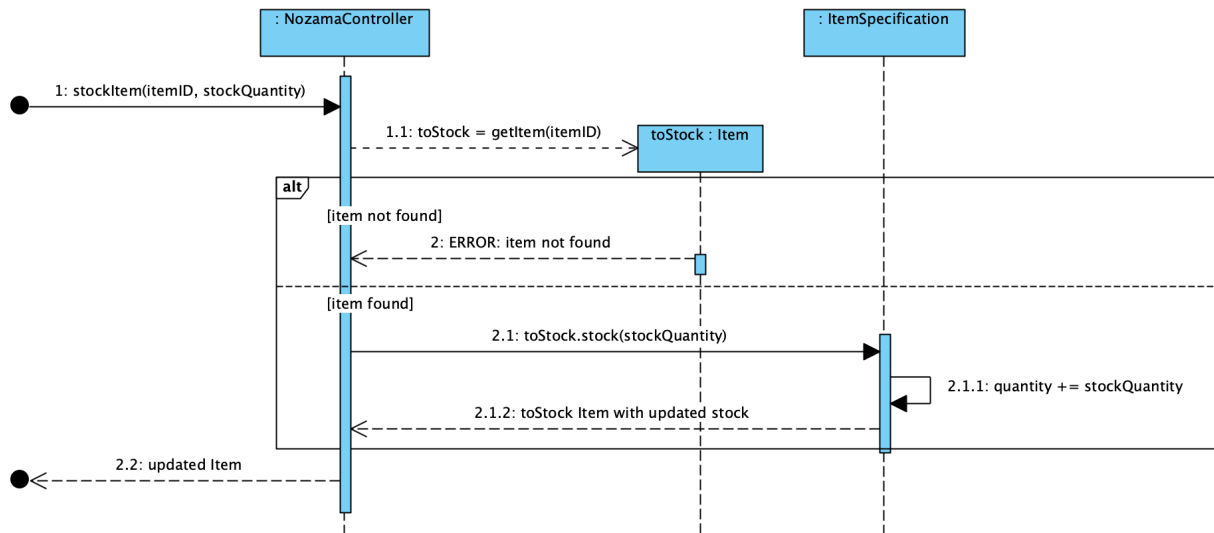
View Product Availability



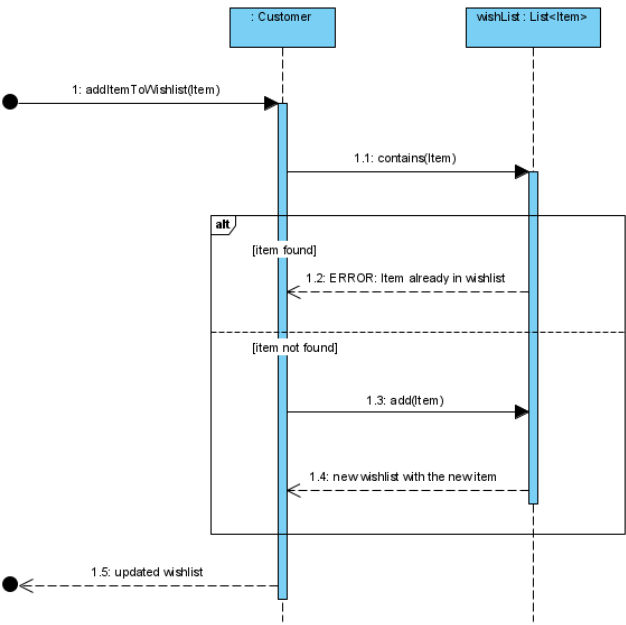
Delete Product



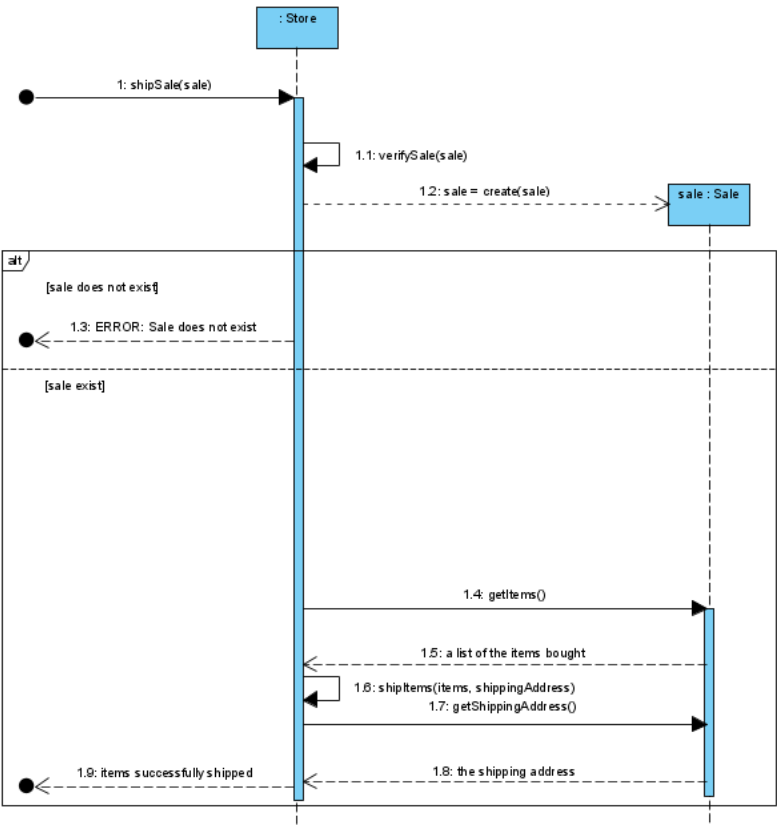
Stock Item



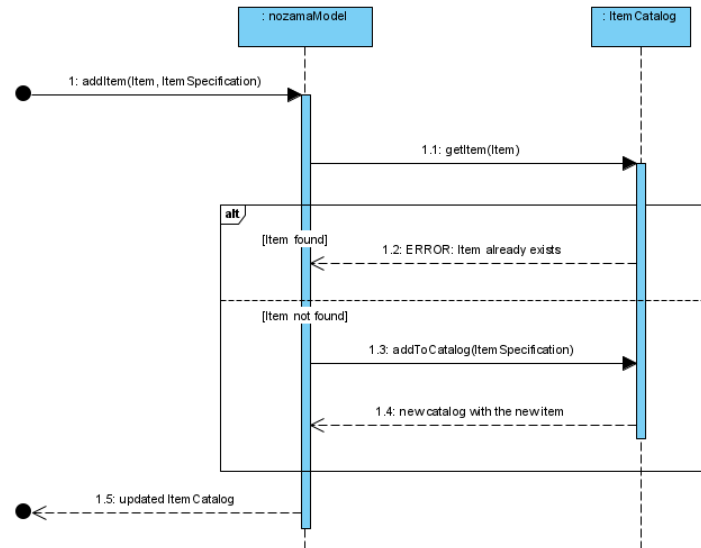
Add Item to Wishlist



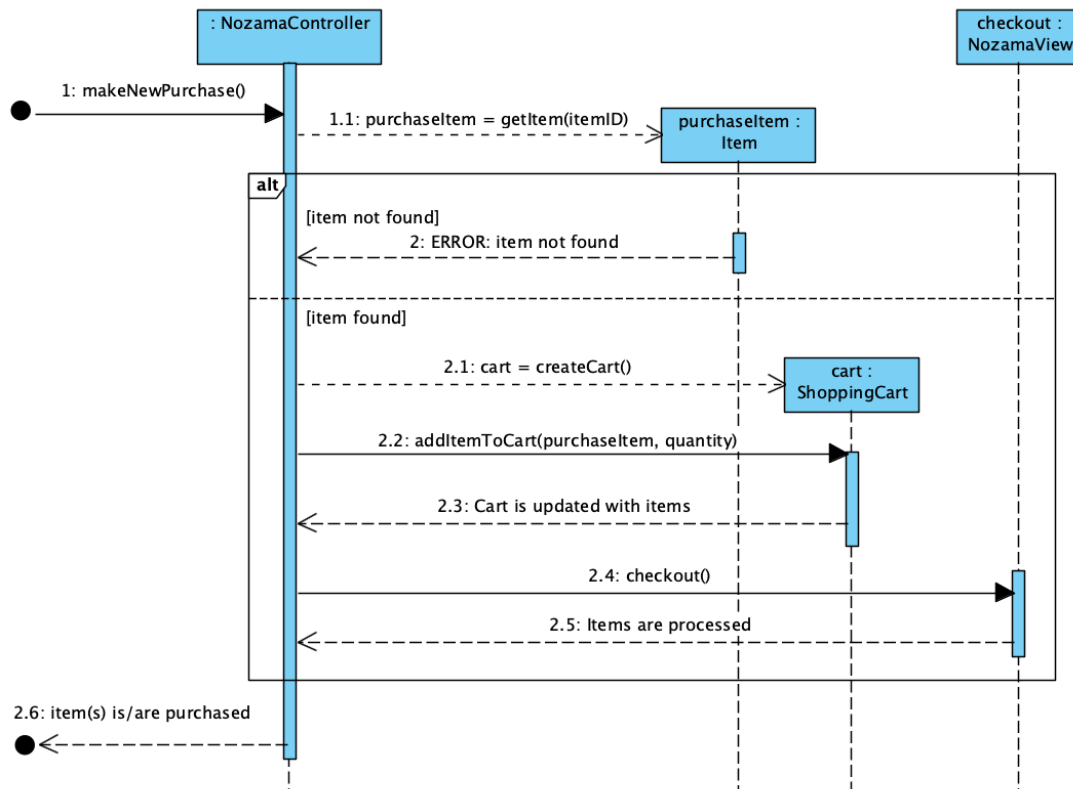
Ship/Send Item



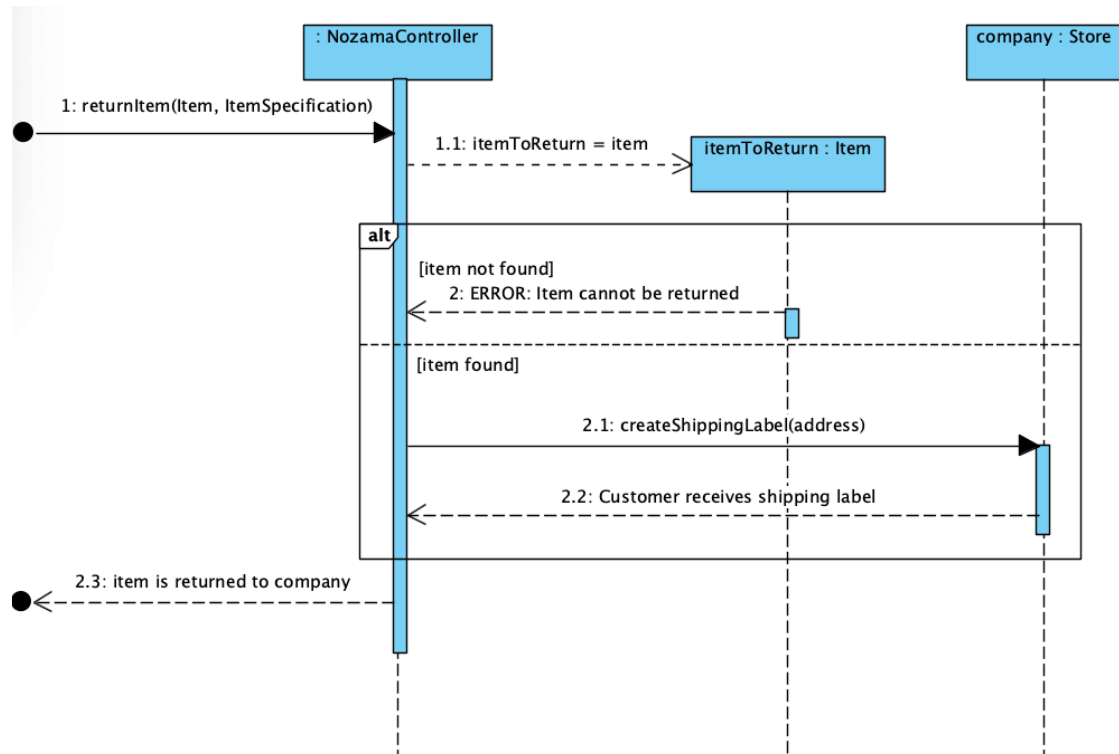
Add New Product



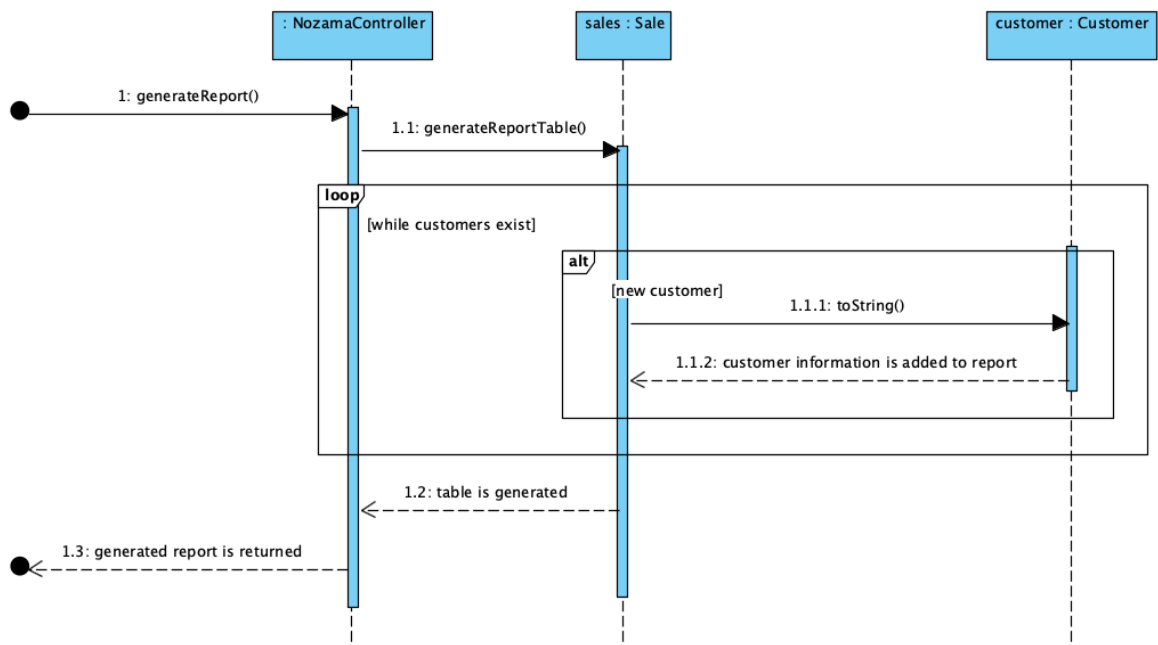
Make Purchase



Return Item



Generate Report



GRASP Patterns:

Controllers:

ButtonColumn is a controller that creates a JButton in the columns of a JTable. When clicked the button executes some code when clicked. We primarily used it to edit or remove values in our JTables.

ButtonColumn is useful because it allows us to easily change the data in a JTable.

NozamaController is a controller that handles the operations for the JTable in the NozamaView class.

NozamaController also handles the management of the ShoppingCart and wishlist of the user who is currently logged in.

ShoppingCartController is a controller for the JTable in the ShoppingCartView class.

ShoppingCartController handles the changes to the shopping cart, as well as writing to the customer's cart file. ShoppingCartController calculates the cart's subtotal, and converts the JTable to an instance of a ShoppingCart so that it can perform operations on the cart, then convert it back into the data of the JTable.

WishlistController is similar to the ShoppingCart. It handles adding to and removing from the user's individual wishlist, and handles the storing of their data. WishlistController is also responsible for displaying the user's wishlist and keeps track of the user's additions.

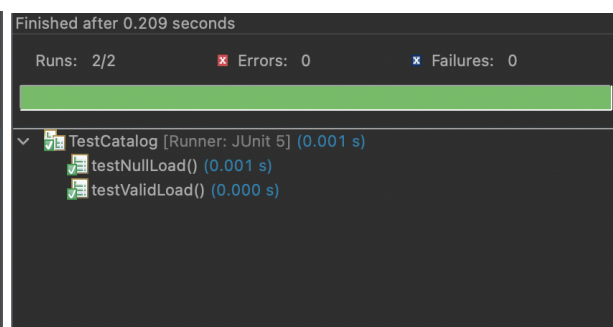
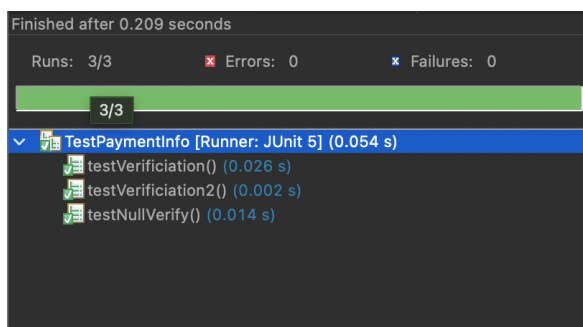
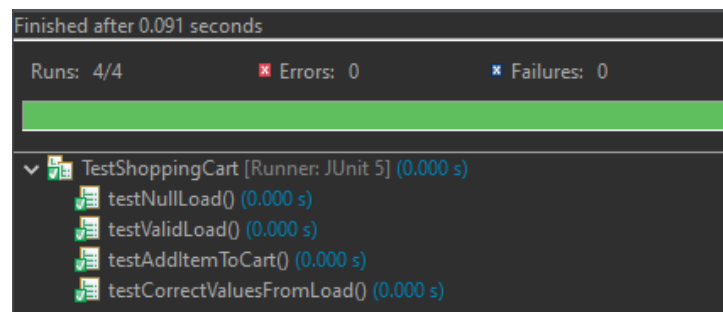
Indirection:

Item uses ItemCatalog to indirectly access the ItemSpecification, this way deleting an instance of the item does not delete the ItemSpecification. This is beneficial because the user's cart keeps track of the items (and any additions or deletions), but if the user deletes an item from their cart we don't want them to be able to delete the ItemSpecification from our system. This also lowers our coupling, as we often don't use the aspects of the ItemSpecification when working with just the Items, and when working with the ItemSpecification we never worry about the individual items. This pattern allows us to separate our data so that we can access and modify parts of it without causing damage or problems elsewhere in the system.

Test coverage plan:

Our project is tested using JUnit 5, which is beneficial because it allows us to see when a test fails, and why. As of right now we have tests implemented for our ItemCatalog, PaymentInfo, and ShoppingCart classes. We validate that our objects work as intended by first testing that our methods throw an instance of NullPointerException when the arguments are null, then we test that the complex methods work as intended. ShoppingCart and ItemCatalog are tested using files testCatalog.csv and testCart.csv, these files populate the ShoppingCarts and ItemCatalogs with data to verify that they work as intended upon creation. We plan on implementing more tests for more classes as our system becomes more complex to make sure that we are able to identify as many errors and bugs as possible.

Example Tests:



Updated Gantt Chart for Iteration 2:

Runtime Terror Gantt Chart

CSI 3471

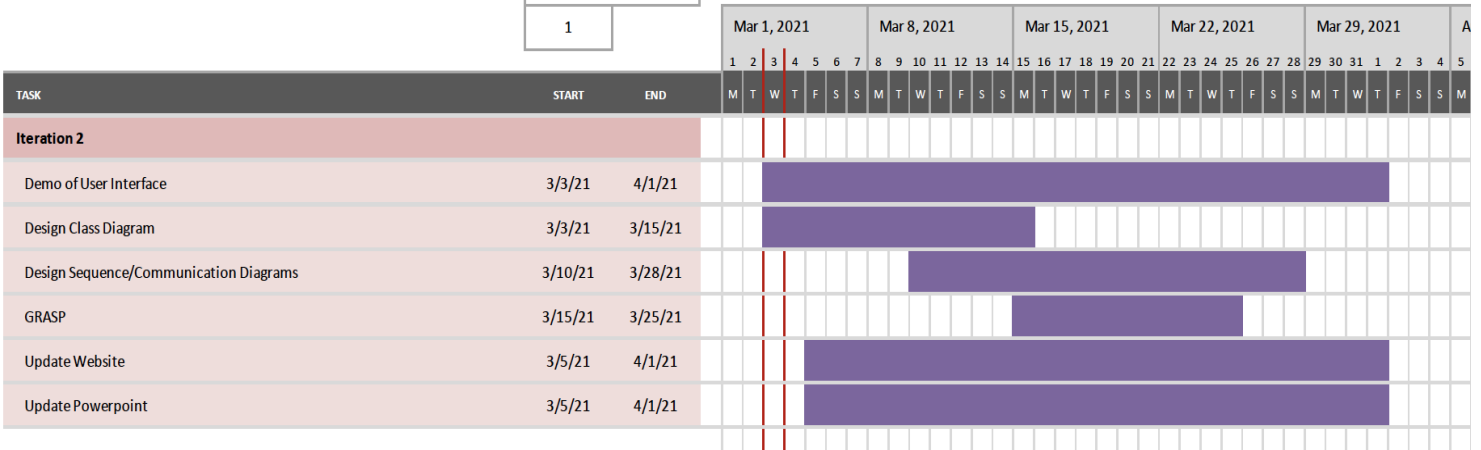
Ashley Bickham, Joshuas Hunter, Austin Lehman, Tyler Ross

Wed, 3/3/2021

1

SIMPLE GANTT CHART by Vertex42.com

<https://www.vertex42.com/ExcelTemplates/simple-gantt-chart.html>



Linked Issue Tracking System/Github

<https://github.com/RuntimeTerrorBU/Nozama/>

Sample Open and Closed Issues - Mid Project (Figure 1 & Figure 2)








<input type="checkbox"/>	🔔 9 Open ✓ 9 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	🔔 Presentation Completion documentation todo #21 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 Package Diagram documentation help wanted todo #20 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 Sequence Diagrams - 3 Each Person documentation todo #19 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 Design Class Diagram documentation help wanted todo #18 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 Implement GRASP patterns help wanted todo #17 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 GUI implementation create model todo #16 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 JUnit testing implement class todo #15 opened 1 hour ago by JoshuaHunter16						
<input type="checkbox"/>	🔔 ProductCatalog implement class todo #3 opened 3 days ago by Alehman74						
<input type="checkbox"/>	🔔 ProductSpecification implement class todo #2 opened 3 days ago by Alehman74						

Figure 1










<input type="checkbox"/>	🔔 10 Open ✓ 9 Closed	Author ▾	Label ▾	Projects ▾	Milestones ▾	Assignee ▾	Sort ▾
<input type="checkbox"/>	🔔 Maven make and configure todo #14 by JoshuaHunter16 was closed 1 hour ago						
<input type="checkbox"/>	🔔 Add a getItemSpecification method to ItemCatalog add method #12 by Alehman74 was closed yesterday						
<input type="checkbox"/>	🔔 Store implement class todo #9 by Alehman74 was closed 1 hour ago						
<input type="checkbox"/>	🔔 Customer implement class todo #8 by Alehman74 was closed 1 hour ago						
<input type="checkbox"/>	🔔 Sale implement class todo #7 by Alehman74 was closed 1 hour ago						
<input type="checkbox"/>	🔔 SalesLineItem implement class todo #6 by Alehman74 was closed 3 hours ago						
<input type="checkbox"/>	🔔 PaymentInfo implement class todo #5 by Alehman74 was closed 3 hours ago						
<input type="checkbox"/>	🔔 ShoppingCart implement class todo #4 by Alehman74 was closed yesterday						
<input type="checkbox"/>	🔔 Item implement class todo #1 by Alehman74 was closed yesterday						

Figure 2

Revised Iteration 1:

Available for download at <https://runtimeerrorbu.github.io/Nozama/>

**Also included in Canvas submission*

Original Point Distribution: 3.3 / 4

- From the original Iteration 1, the major mistake that cost us points was forgetting to make a Use Case Diagram, as well as minor details in the System Sequence Diagrams that cost us points. We have fixed everything that was mentioned in the comments on our original submission, therefore we believe the suggested point redistribution should be full credit.

Suggested Point Redistribution: 4 / 4

Changes that were made:

- Added Use Case Diagram
- Changed System Sequence Diagrams “Add a Product” & “Delete a Product” to allow manager to delete items rather than the company
- Added reply messages from the system on the System Sequence Diagrams of “Ship/Send Item” use case