Junction Jewelers Project Documentation

Authors:
Kendra Bach, Mark Ferrall, Cooper Stansbury
Nolan Gage, Nicholas Crooker

Document Organization

There are two documents related to this project: (1) Implementation specific information that covers requirements and test plans for the creation of Junction Jewelers website. (2) The data dictionary provides descriptions of entities, business logic, and data types. In general this document starts from a general description and proceeds to more granular technical detail.

Each document has a separate table of contents. Each document restarts page numbering at their respective page ‘1’. They are briefly summarized below:

Implementation Contents

Data Dictionary Contents

Junction Jewelers Implementation Documentation

Authors:
Kendra Bach, Mark Ferrall, Cooper Stansbury
Nolan Gage, Nicholas Crooker

Implementation Documentation

Implementation documentation is project specific details. This section contains requirements, testing, and status of the current implementation. This document is included in order to provide clear reasoning for decisions made for the initial website launch.

Implementation Contents

Affiliation

University of Michigan Dearborn
College of Engineering and Computer Science
Department of Computer and Information Science
4901 Evergreen Rd, Dearborn, MI 48128

Development Team Skills Overview

The skills of each team member contributed to the design choices that were made throughout the project. Development time was reduced by by choosing technologies that team members were already comfortable with. This required integrating several technology stacks into the final project, dependent on each team member's area of responsibility.

- Kendra Bach: Background in software engineering and web application development. Built Java Spring MVC web application to integrate with team members' database.
- Nick Crooker: Background in software engineering and embedded systems. Handled emailing receipts and the newsletter.
- Mark Ferrall (Team Lead): Experienced with Microsoft Access, limited SQL, and documentation. Project management experience and comfort managing teams informed his role as Team Lead.
- Cooper Stansbury: Background as ETL developer. Built database and loaded initial state.
- Nolan Gage: Background in software development including developing SQL queries and front-end code for corporate finance reporting. Wrote SQL queries and report pages to run our reports and render results.

Development Team Contact Information

- Mark Ferrall: mdferral@umich.edu
- Cooper Stansbury: cstansbu@umich.edu
- Kendra Bach: kmbach@umich.edu
- Nicholas Crooker: ncrooker@umich.edu
- Nolan Gage: nolang@umich.edu

Development Team Photo

Business Mission Statement

Customer experience is our North Star and our first priority. We engage our customers by creating a simple, efficient, and no-nonsense shopping experience. This principal guides our choices from inventory, to web-design, to our hiring processes.

Implementation Statement

Our small business requires a website to facilitate e-commerce retail. Our website must handle both the front end customer interface and backend inventory processes related to transactions, including reporting. The front end will handle customer accounts, sales orders and shipping cost calculation. The backend must handle all relevant inventory and reporting functionality. Core customer types must be used to validate the design.

Additional Functionality

The assignment requires the project team to identify two additional features beyond the project requirements. The project team chose **emailed receipts** to the user and **product pagination** of the products on the product display webpage.

Email receipts

Our product supports direct customer communication via newsletters and emails of completed transactions. The SMTP scripts are written in Python and managed by the Java application.

Pagination

Pagination, in our context, refers to separating products into a number of discrete pages in order to make a cleaner 'look and feel' for customers when shopping. This was handled by the business layer of the application, but depends on the results of HQL queries executed against the MySQL instance.

Source Code Repository

The application source code, supporting applications, database structure, and initial database load can be found at the following public GitHub repository:

- https://github.com/KendraMBach/CIS556_Project

Notes on Development Strategy

- We use Git as our version control system for this implementation. The latest release and developer documentation can be found here: https://github.com/KendraMBach/CIS556_Project
- To track in-development issues and feature requests we use the GitHub Issue tracker, which can be found here: https://github.com/KendraMBach/CIS556_Project/issues

Fundamental Implementation Assumptions

Below is a list of fundamental assumptions made by the project team while designing and implementing the Junction Jewelers website solution.

- All non-ecommerce related processes are functioning, and covered elsewhere.
- There is a window of time after testing to migrate local processes to production servers.
- Production servers have all required software.
- Only one customer shops at any given time.
- Junction Jewelers conducts business only in the United States of America, including Hawaii and Alaska. In the future we plan to expand business globally.
- Junction Jewers accepts only the US Dollar (not bitcoin). As such, not transaction details are necessary through the website. (We do not store credit cards).
- All transactions are tendered with the same payment type.
- Timestamps for orders are always specified in EST, regardless of where order is placed.
- Charms and birthstones have no limit to their supply.
- Charms and birthstones have have no cost to Junction Jewelers, there inventory is not important to track.
- Junction Jewelers has domain that can be used instead of localhost.

Project Status

The project addresses all functional and non-functional requirements in the Business Specifications section of this document. Several issues have been identified and are not yet addressed. Those include:

- Duplicate items appearing in the Popular Items category
- Search does not include Color or Material
- The same items with different customizations cannot be added to the cart
- No concurrency control
- No security
- No maintenance measures in place

All issues are documented on the source code repository:

- https://github.com/KendraMBach/CIS556_Project/issues

Business Specifications

This section details the project requirements established to ensure the implementation performs correctly. Project requirements are broken into (1) functional requirements and (2) non-functional requirements.

Functional Requirements

The Functional Requirements were developed upon review of the needs provided by Junction Jewelers and review of the products carried by the store. This review revealed several logical groupings of requirements: Website, Product Pages, Shopping Cart, Database, Reports, and User Account. The project team must document addressing all functional requirements and their acceptance criteria for the project to be complete.

Requirement	Acceptance Criteria
Website	1. The website must include: 1.1. An initial homepage, including names and images of the team members 1.2. The ability to create a new customer account 1.3. The ability to log in to an existing customer account 1.4. A newsletter signup, which will send the user an immediate confirmation/welcome email 1.5. A page listing all products carried by the store 1.5.1. Views of products must be paginated, with no more than 6 products per page 1.6. A list of the most popular products carried by the store 1.7. A product page for all products (see product page requirements) 1.8. The ability to place an order 1.9. Ability to place multiple orders 1.10. Ability to select customizations for some products 1.11. Ability to order more than 1 of an item 1.12. Ability to search for items by item name 1.13. Ability to view products by their category 1.14. An interface for generating reports 1.15. The ability to add multiple items to the cart 1.16. The ability to log out of an account
Products Pages	2. The Products Pages Must 2.1. Provide the ability to select distinct sizes, color, or material for products with those options 2.2. Provide the ability to add a product to the cart 2.3. Display the description and image of each product 2.4. Provide the ability to specify engravings, birthstones, or charms for products with those options
Shopping Cart	3. The Shopping Cart must 3.1. Provide the ability to add/remove items from the cart 3.2. Provide the ability to change the quantity of an item in the cart

	3.3. Provide the ability to checkout, adding the cost of shipping to the final total 3.4. Prevent checkout with an empty cart, instead redirecting the customer to the products page 3.5. Prevent checkout without a valid user account 3.6. Send users an emailed receipt upon checkout 3.7. Prevent purchase of an out of stock item 3.8. Prevent purchasing more of an item than there are items in stock
Database	4. The database must include 4.1. Table to support all products carried by store 4.2. Table to support the cart 4.3. Table to support all customers 4.4. Table to support orders 4.5. Table to support birthstone options 4.6. Table to support charm options 4.7. Table to support shipping locations and costs 4.8. Support for customizations of some products (engravings, birthstones, and charms) 4.9. Support reporting of sales, inventory, and customer information (see report requirements) 4.10. Constraints to prevent 4.10.1. Improperly specified new product listings 4.10.2. Improperly specified orders 4.10.3. Improperly specified user accounts
Reports	5. The Reports must: 5.1. Generate a Monthly/Yearly Sales Report which includes 5.1.1. The ability to specify the start/end date of the reporting period 5.1.2. The ability to prevent invalid reporting period dates 5.1.3. The total dollar sales for the specified period 5.1.4. The total dollar profit for the specified period 5.1.5. The total number of items sold by category 5.2. Generate a report of Inventory Levels and Costs which includes 5.2.1. The names and amounts of items currently in inventory 5.2.2. The total cost of items currently in inventory 5.3. Generate a report of Customers which includes 5.3.1. The names, addresses, email, and phone number of all customers 5.4. Generate mailing labels for a customer
User Account	6. The User Account Must 6.1. Allow customers to create password protected accounts to

	order from the store 6.2. Allow customers to store their name and shipping information 6.3. Constrain improperly specified user account information

Non-Functional Requirements

Review of the business needs of Junction Jewelers revealed several types of non-functional requirements: Aesthetic, User Stories, and Content. Developing User Personas, sample target customers of Junction Jewelers, will help clarify development of features and testing strategies around customer needs.

Aesthetic	● Website should look and feel modern ● Reports should be polished (but simple)
Support User Stories:	● Persona 1: New Male customer using site to find a specific product ● Persona 2: New female customer browsing, no endpoint ● Persona 3: Returning Male customer using site to find a specific product
Content	● Must include information about company ○ Company name ○ Location ○ Contact information

Implementation Test Plan

The persona test paths test the application from the perspective of three target customers of the business. By focusing on these users, the project team better targeted the development and testing of the application with the real-world demands placed on it. Three personas are described below.

Persona 1

New male customer using site to find a specific product.

- He is getting married soon and needs to pick out a wedding band.
- He knows he needs to purchase a wedding ring, but isn't sure which one he wants, or what his ring size is.

Persona 2

New female customer expecting a baby in April, just browsing for items

- She's looking for a series of bracelets for her, her mother, and the baby
- She'd like a customized engraving of the baby's name
- She wants her own bracelet to feel unique
- She might want to look at some other items for herself while she's shopping

Persona 3

Returning male customer purchasing groomsman gifts

- He is purchasing some matching black stud earrings for his groomsmen
- He previously used the site to purchase his wedding band

Constraint Tests Specification

The constraint tests demonstrate the functionality of the limitations placed on the ordering process, and the protection of the database. These limits prevent ordering quantities of products greater than the store inventory. The test cases will show the user messages upon entering invalid information.

Constraints Tested

- Prevent Adding Items to Cart without Required Options (Size)
- Prevent Registration without Required Information
- Non-Matching Search
- Prevent Registration with Invalid Phone Number
- Prevent Registration with Invalid Email
- Order Quantity Exceeding Inventory
- Prevent Ordering Items Out of Stock
- Prevent Ordering Without an Account

Reporting Tests Description

The constraint tests demonstrate the functionality of the reporting features built into the application. Test cases will show the anticipated and actual outputs using test data, and the changes to those reports after completing the Persona Test Paths.

Test Plan Validation

The following subsections walk through synthetic digital paths for each of the personas described. This section is intended to show how core requirements are met, and handled by the customer front-end in terms of our customers.

For dynamic demonstrations links to short videos have been provided:

- Persona 1 Video: https://youtu.be/JTOYhzku-S0
- Persona 2 Video: https://youtu.be/M0GgMxB-N8o
- Persona 3 Video: https://youtu.be/N8nQjVFqV_g

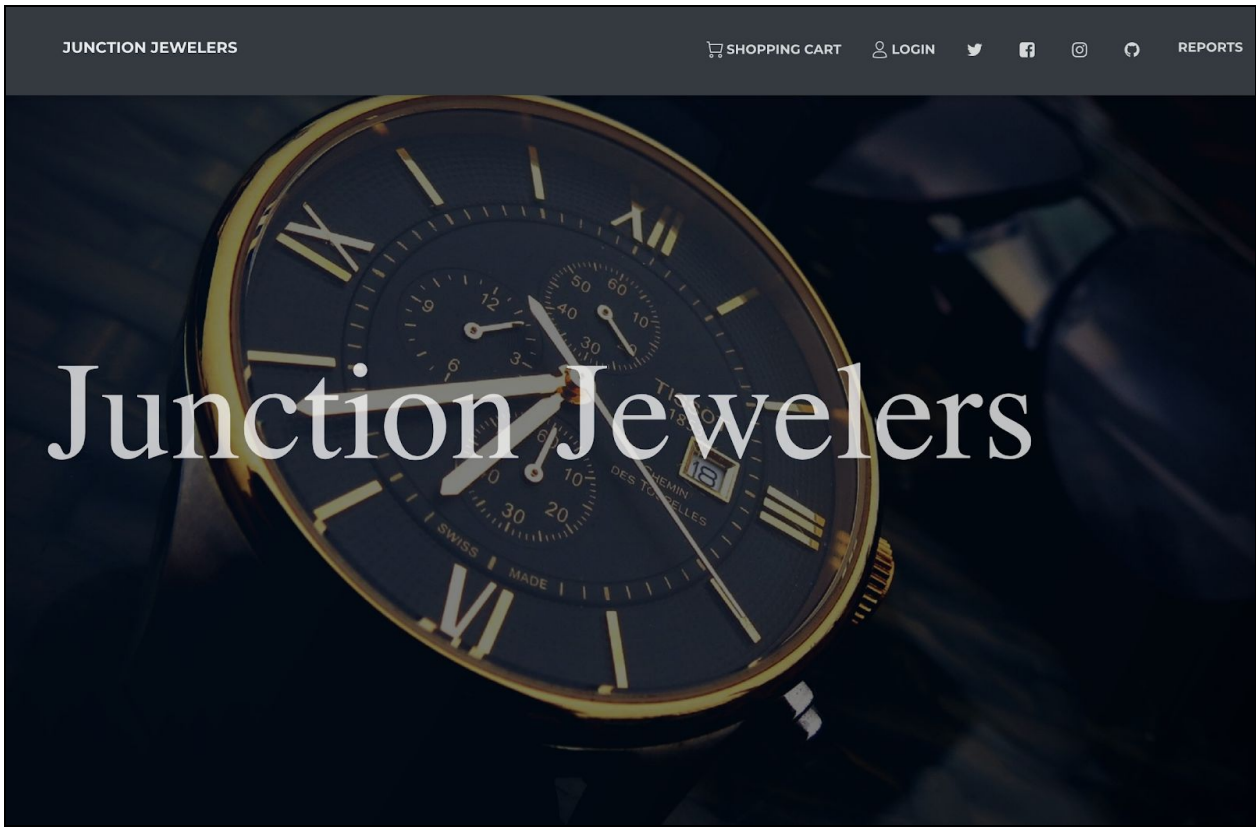
Persona 1 Test Path Validation

Action 1: Customer Arrives at Store Website, Navigates to Product Page

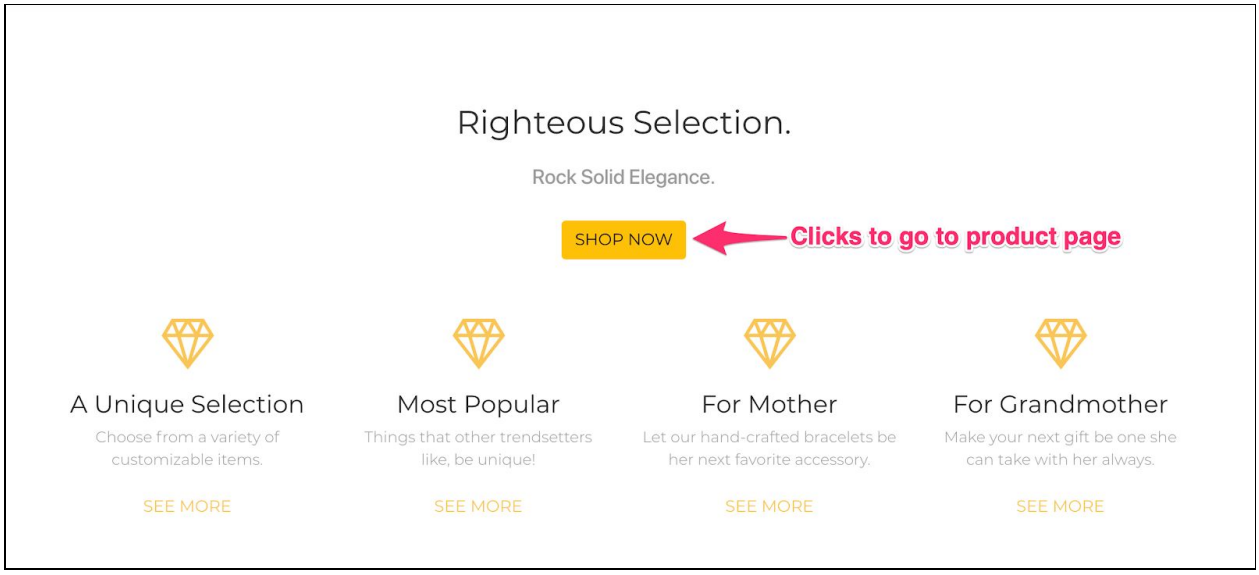
Requirements Addressed

1.1 - An initial homepage, including images of the team members

Arrives at Store Landing Page



Scrolls Down to Click Product Link



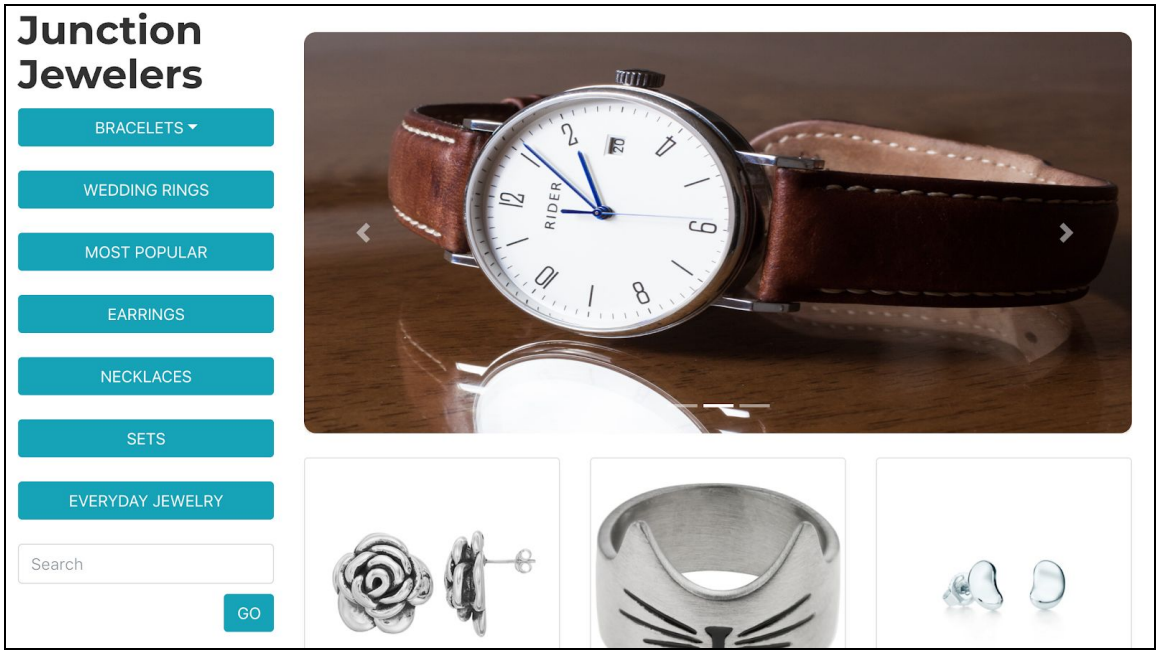
Action: Goes to Product Page, Navigates to Gold Wedding Band

Requirements Addressed:

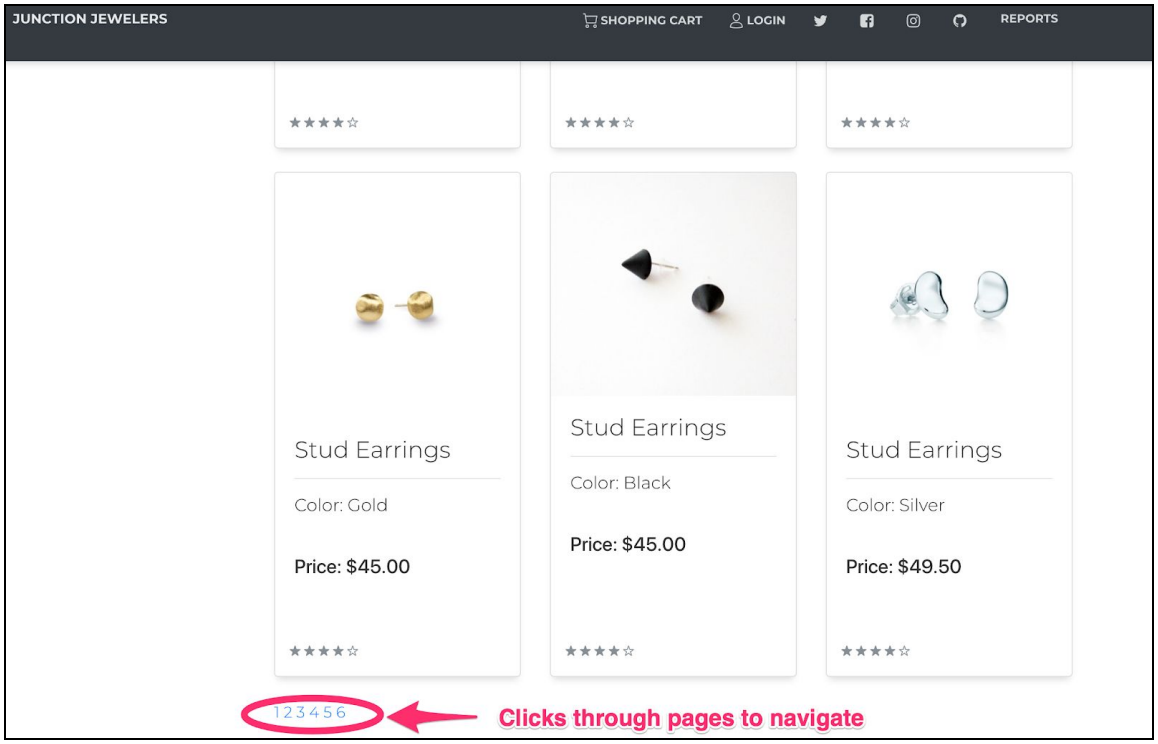
1.5 - A page listing all products carried by the store

1.5.1 - Views of products must be paginated, with no more than 6 products per page

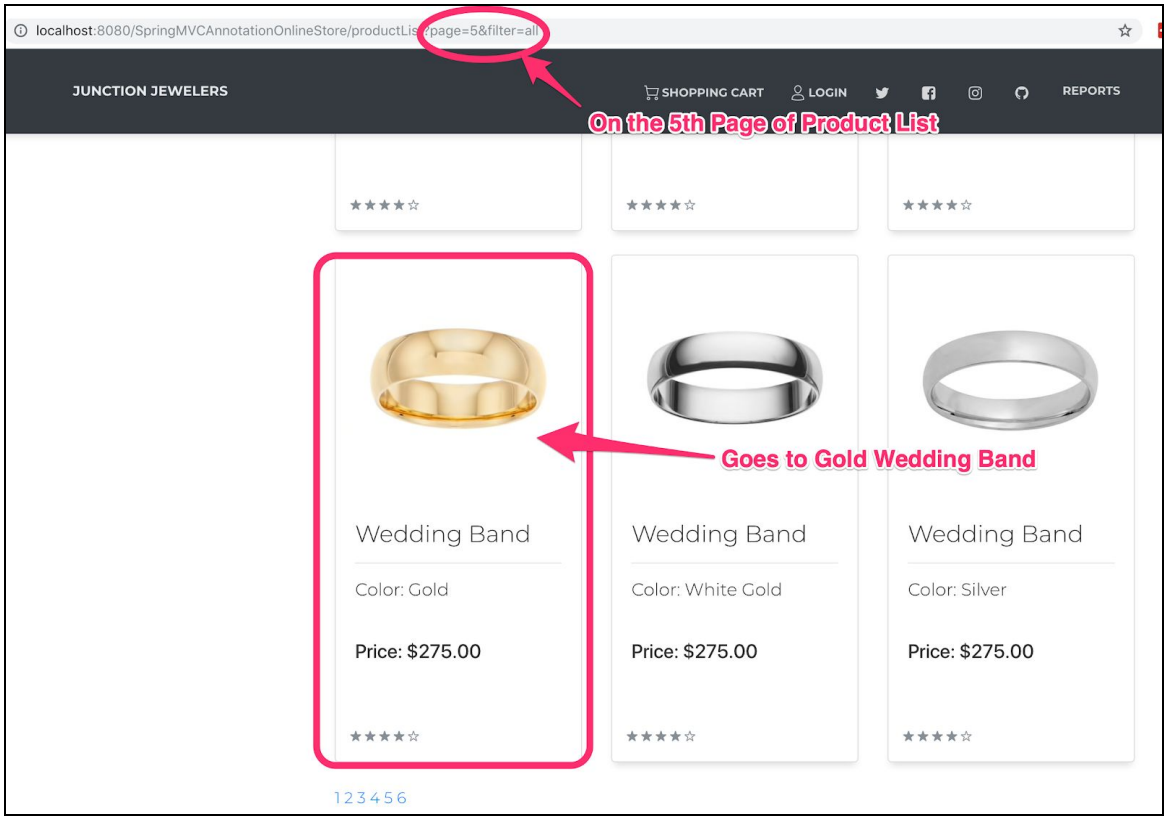
Initial arrival to product page



Scrolls Down, clicks page navigation until arrives at gold wedding band



Finds gold wedding band, clicks to view

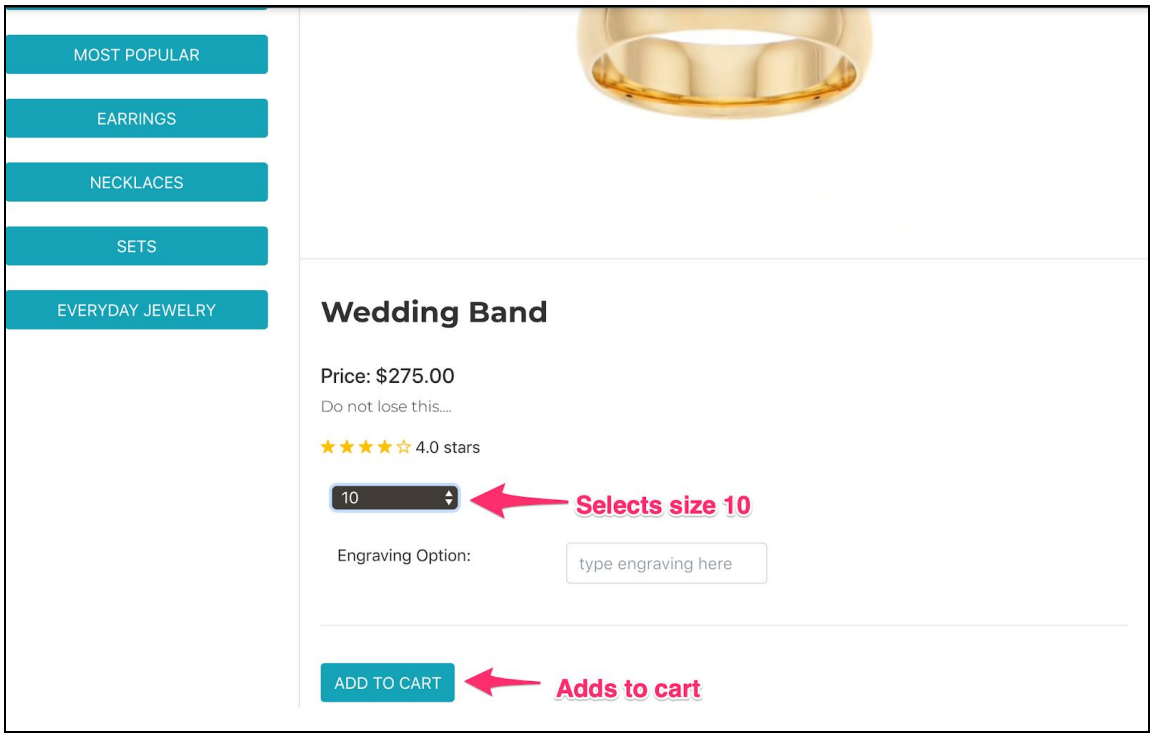


Action: Adds Size 10 Gold Wedding Band to Cart

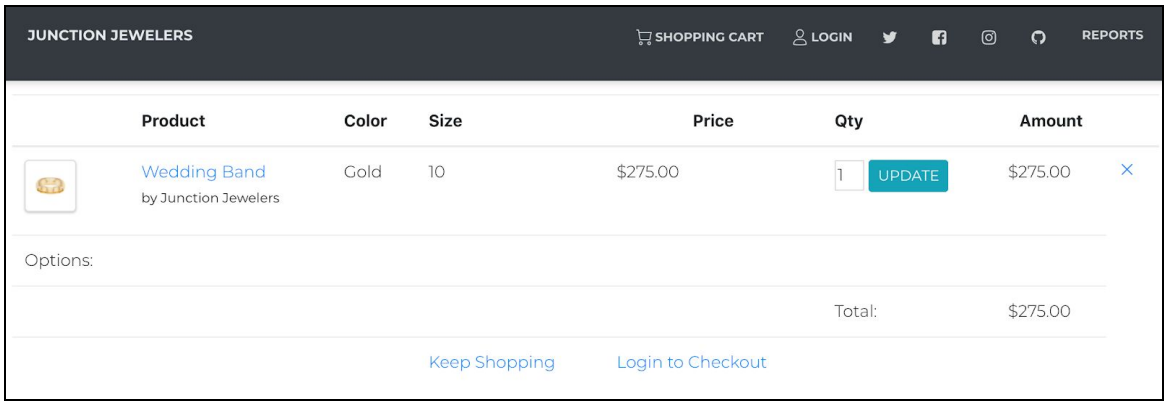
Requirements Addressed:

1.7 - A product page for all products (see product page requirements)
2.1 - Provide the ability to select distinct sizes, color, or material for products with those options
2.2 - Provide the ability to add a product to the cart
2.3 - Display the description and image of each product
2.4 - Provide the ability to specify engravings, birthstones, or charms for products with those options
3.1 - Provide the ability to add/remove items from the cart

Selects Size, adds to cart



Visits Cart



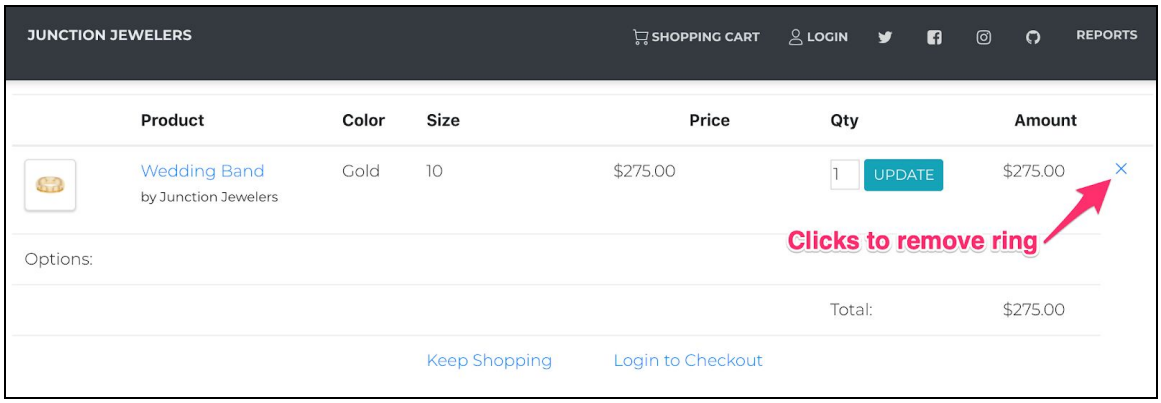
Action: Changes Mind on Color Ring, Removes Product from Cart

Requirements Addressed

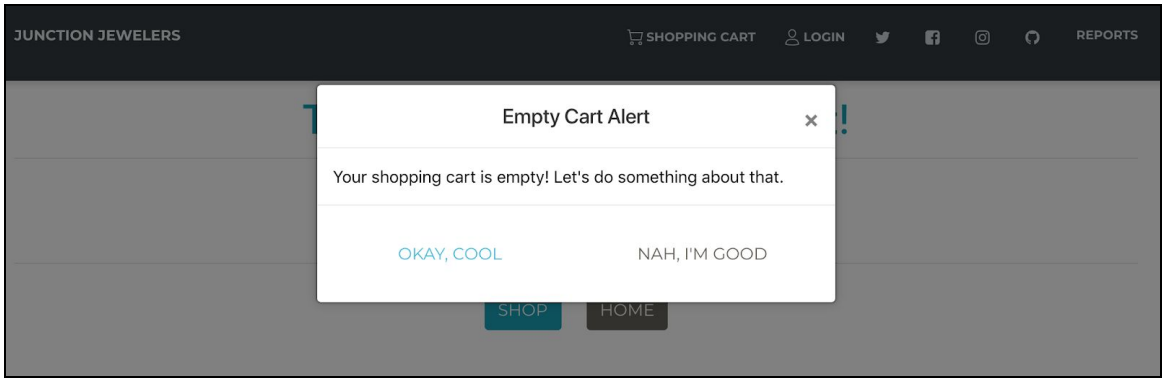
3.1 - Provide the ability to add/remove items from the cart

3.4 - Prevent checkout with an empty cart, instead redirecting the customer to the products page

Clicks 'X' to remove ring from cart



View of Empty Cart Confirmation

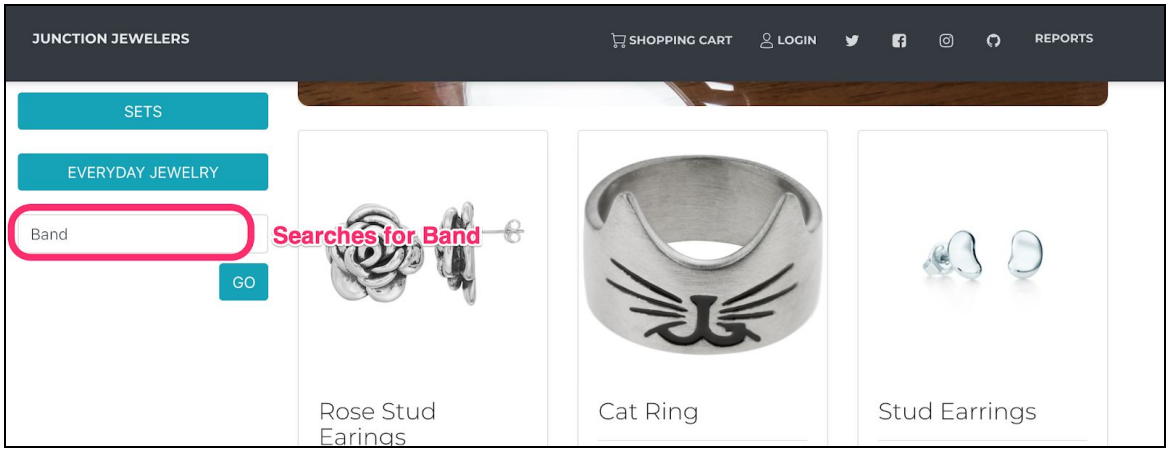


Action: Goes to Product Page, Searches for Band

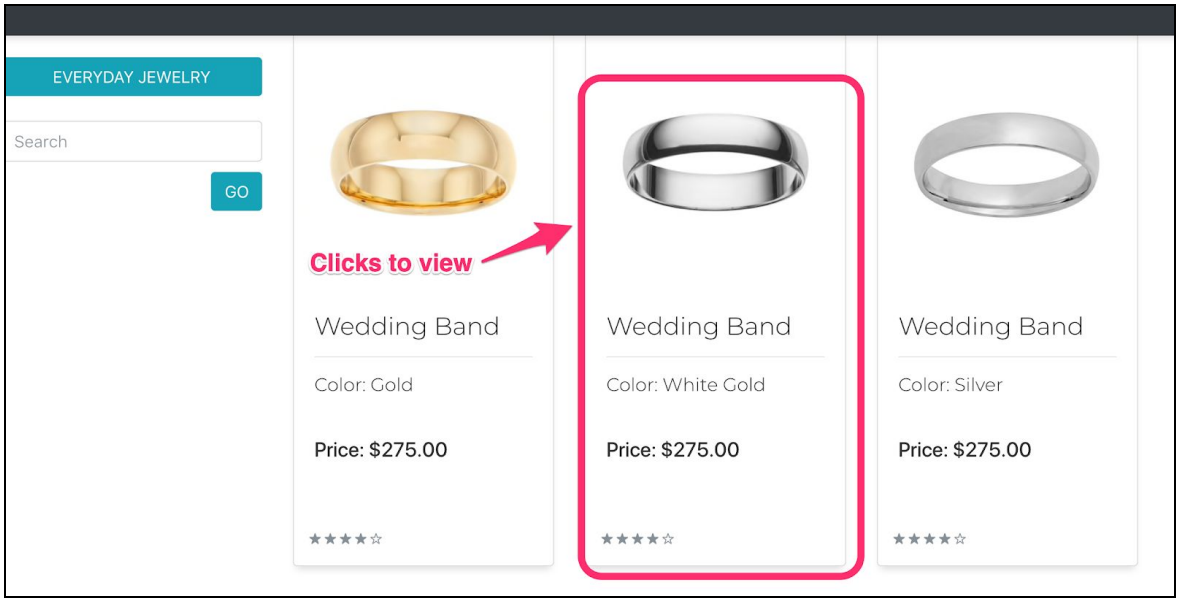
Requirements Addressed

1.12 - Ability to search for items by item name

Enters Band in the Search Box, Clicks 'GO'



Band Results, Clicks White Gold Band



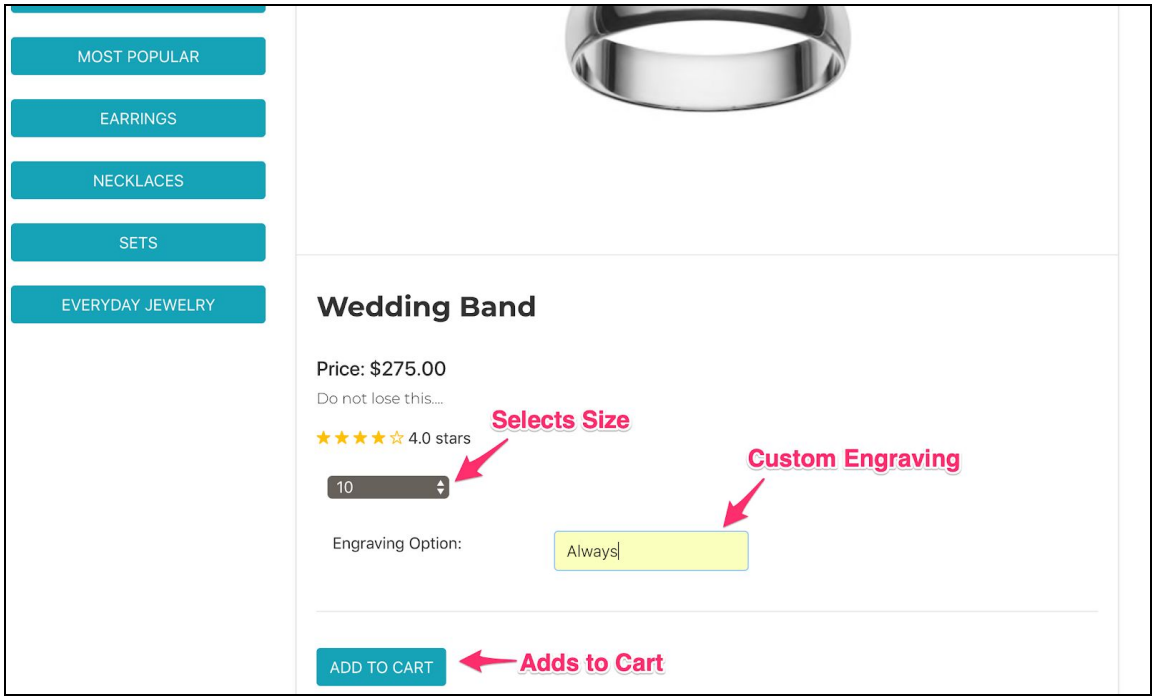
<u>Action: Adds Customized Size 10 White Gold Band to Cart</u>

Requirements Addressed

1.10 - Ability to select customizations for some products

2.4 - Provide the ability to specify engravings, birthstones, or charms for products with those options

3.1 - Provide the ability to add/remove items from the cart

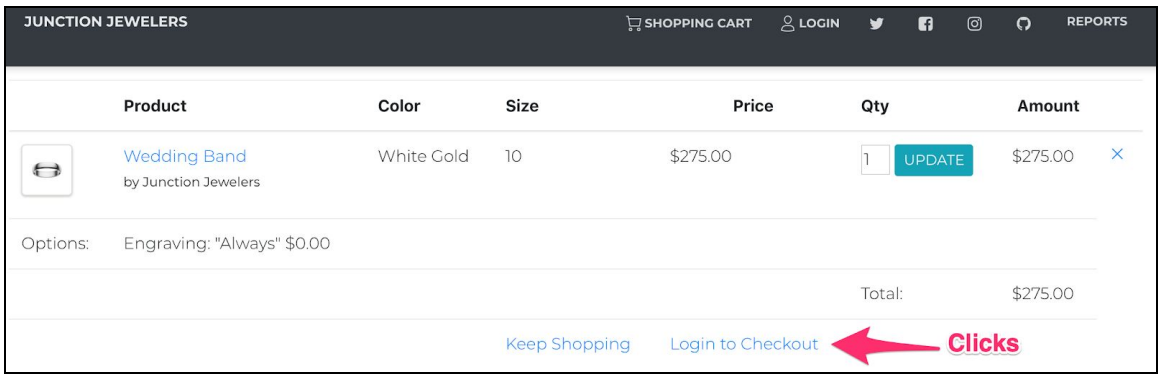


Action: Creates new Account to Checkout

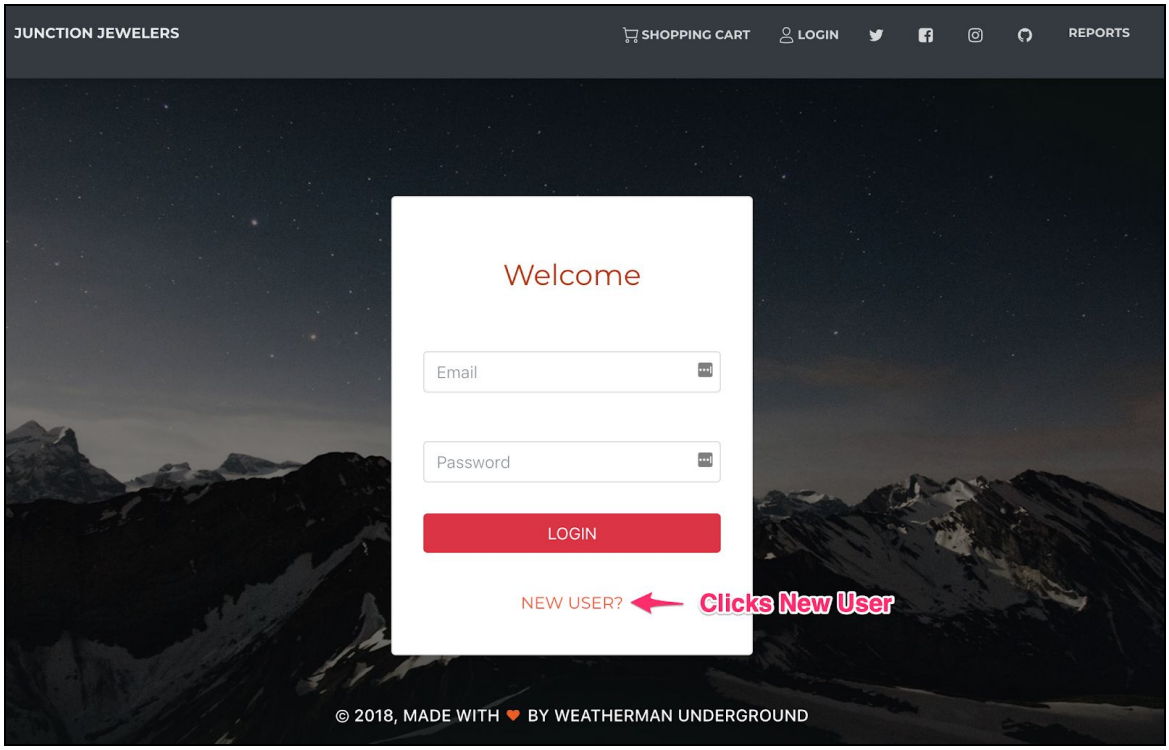
Requirements Addressed

1.2 - The ability to create a new customer account
3.5 - Prevent checkout without a valid user account
6.1 - Allow customers to create password protected accounts to order from the store
6.2 - Allow customers to store their name and shipping information

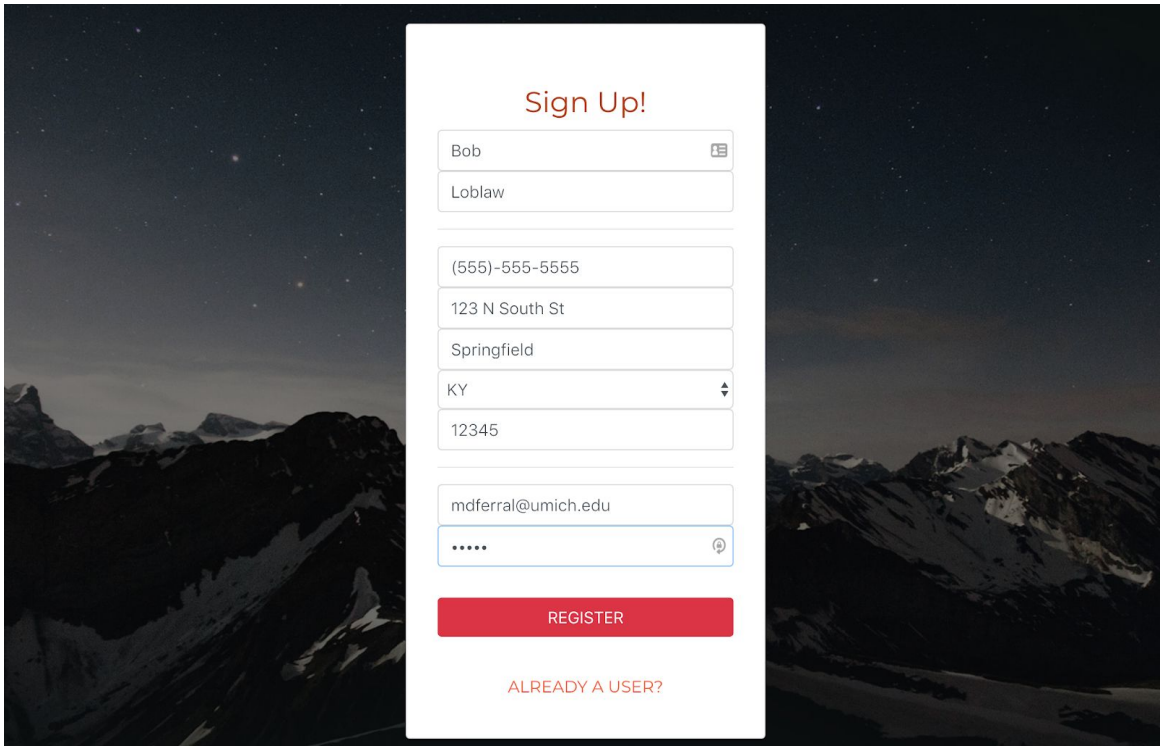
Attempts to check out, but doesn't have account yet



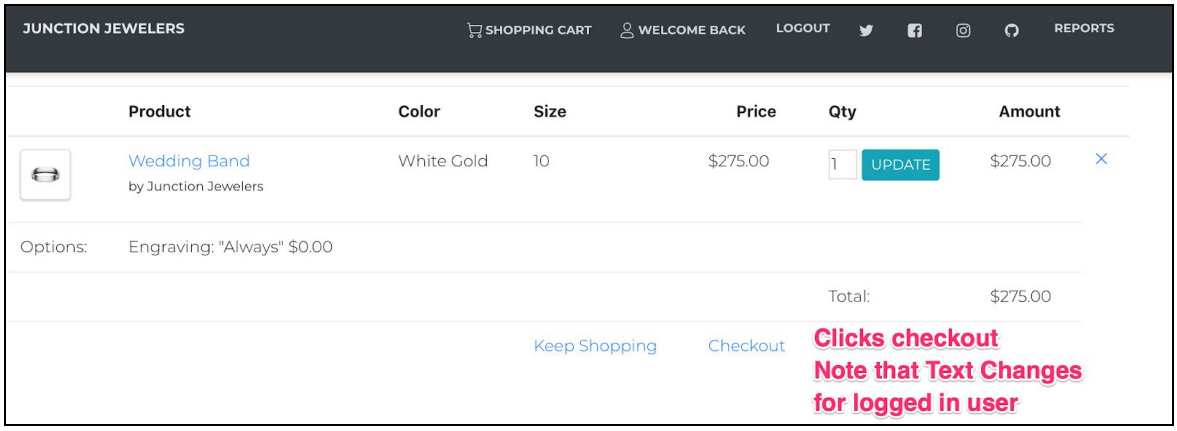
Taken to Login Page, Clicks New User



Creates New Account



Goes Back to Cart, Clicks Checkout



Action: Checks Out to Purchase Item

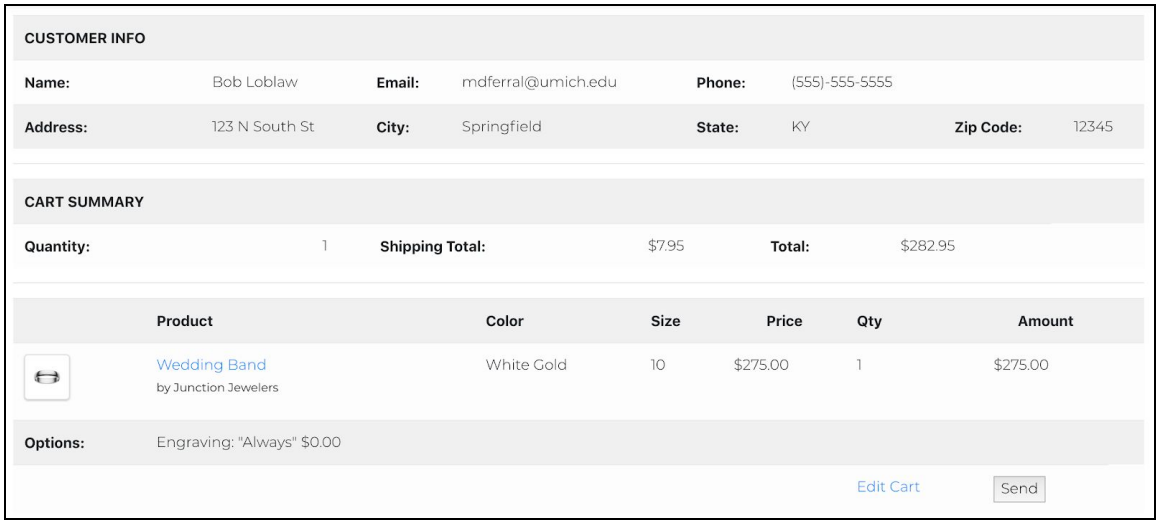
Requirements Addressed

1.8 - The ability to place an order

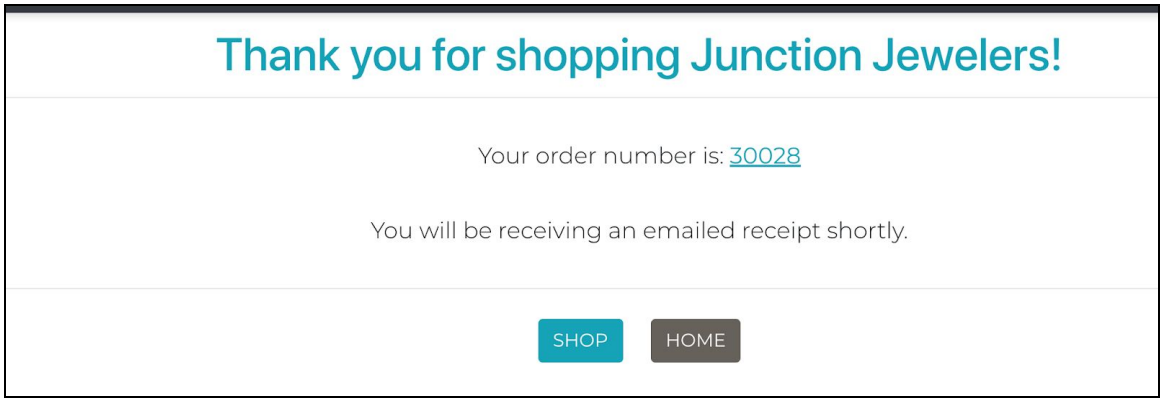
3.3 - Provide the ability to checkout, adding the cost of shipping to the final total

3.6 - Send users an emailed receipt upon checkout

Pre-Checkout Confirmation



Order Confirmation



Emailed Receipt

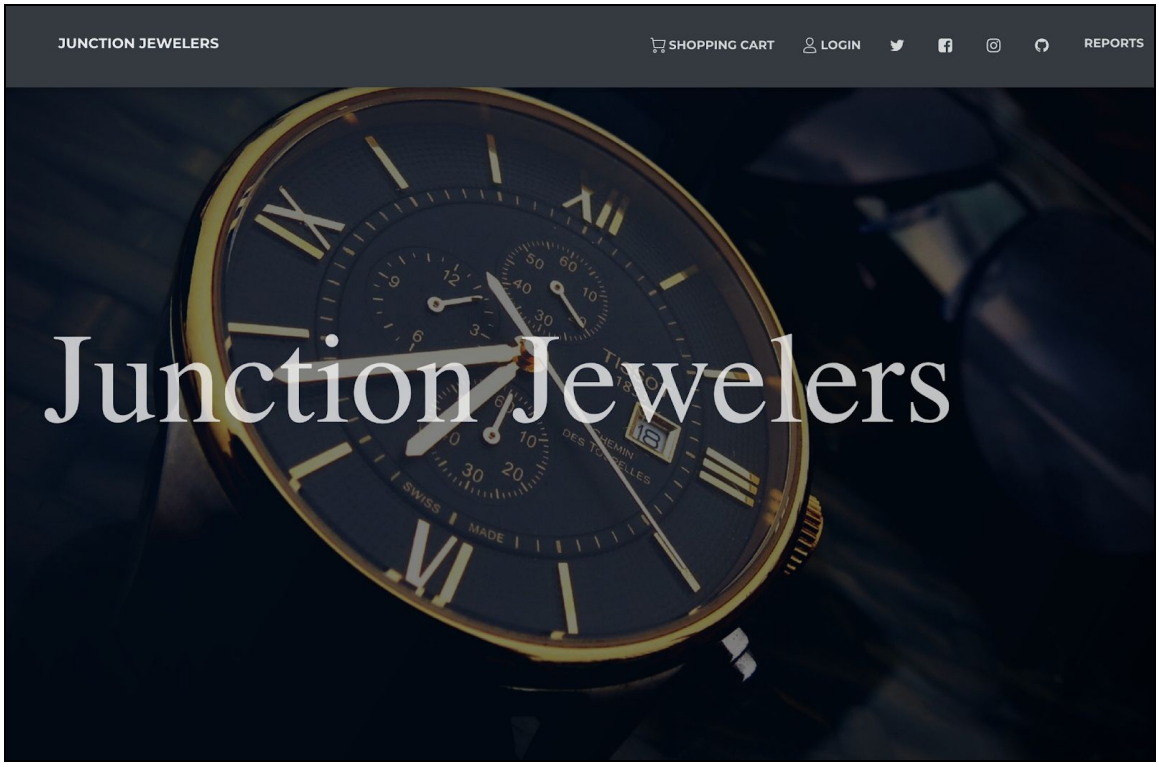
Persona 2 Test Path Validation

Action: Arrives at Store, Signs Up for Newsletter

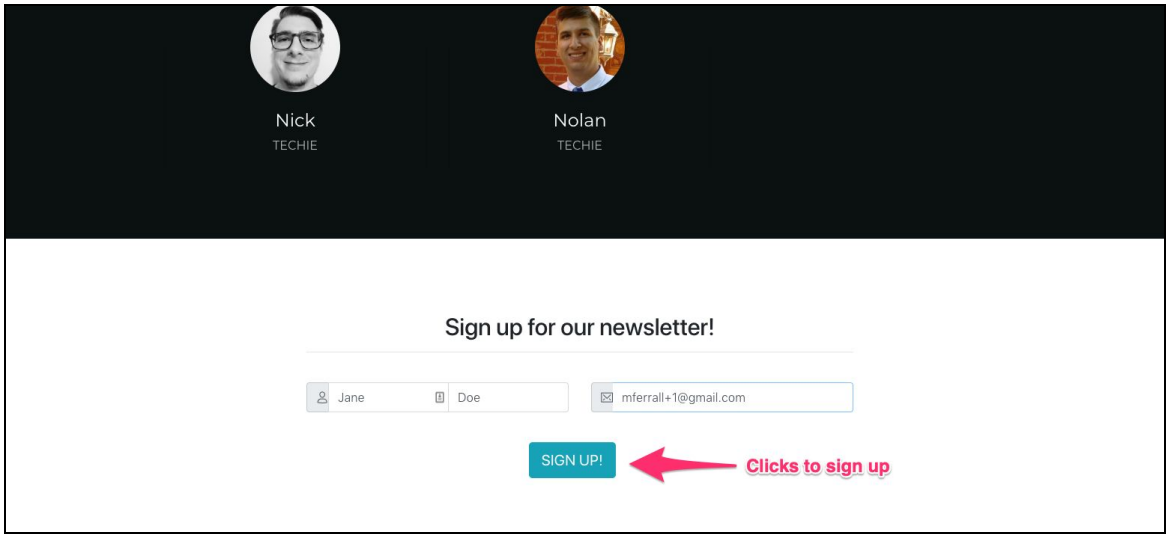
Requirements Addressed:

1.4 - A newsletter signup, which will send the user an immediate confirmation/welcome email

Arrives at Store Landing Page



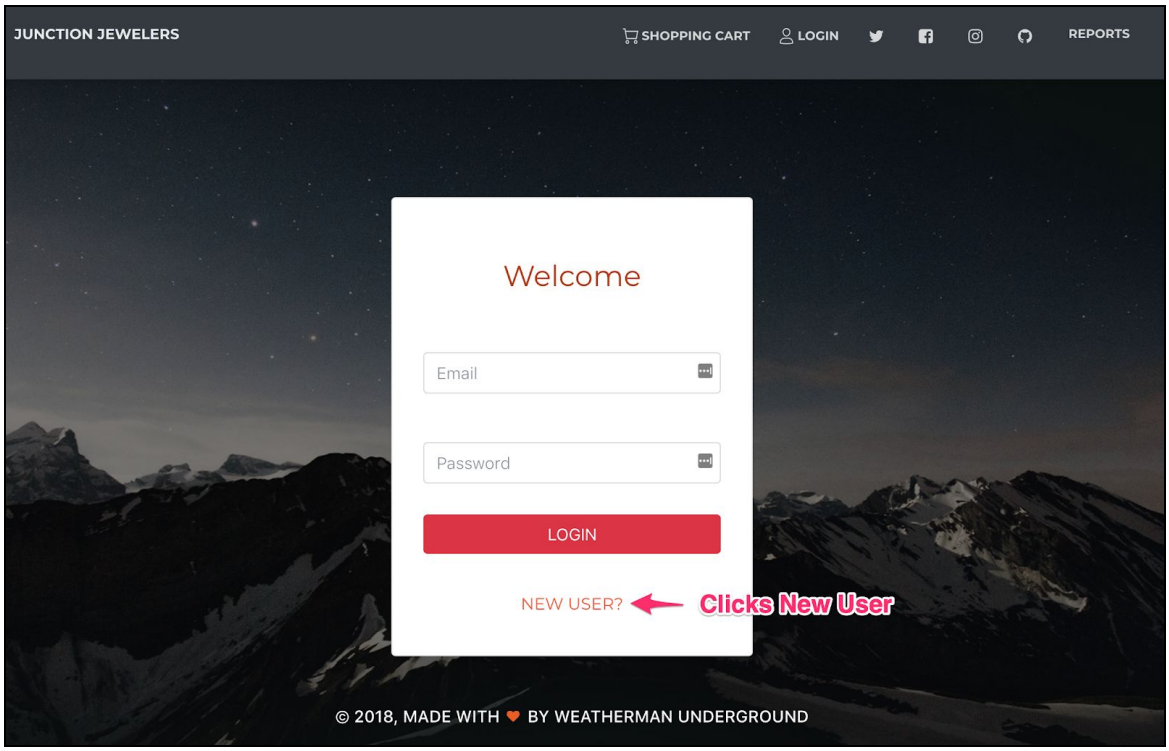
Scrolls down, signs up for newsletter



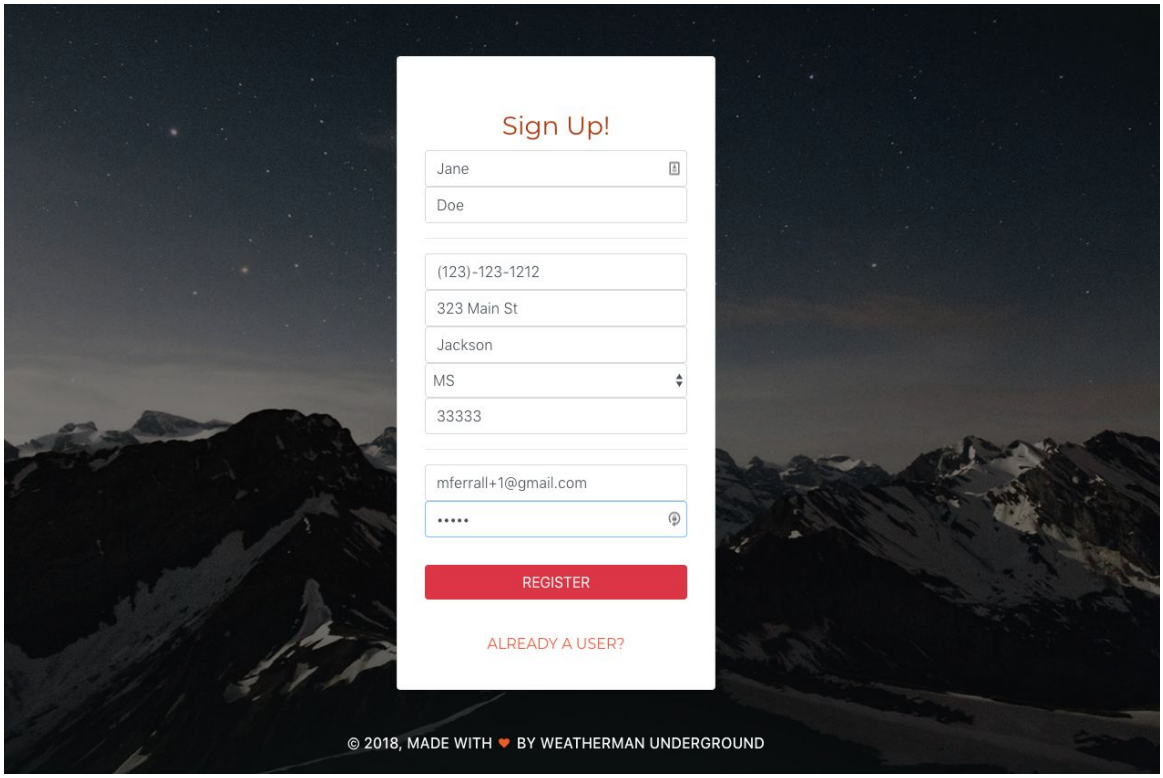
Newsletter email

Action: Creates New Account

Taken to Login Page, Clicks New User



Creates New Account

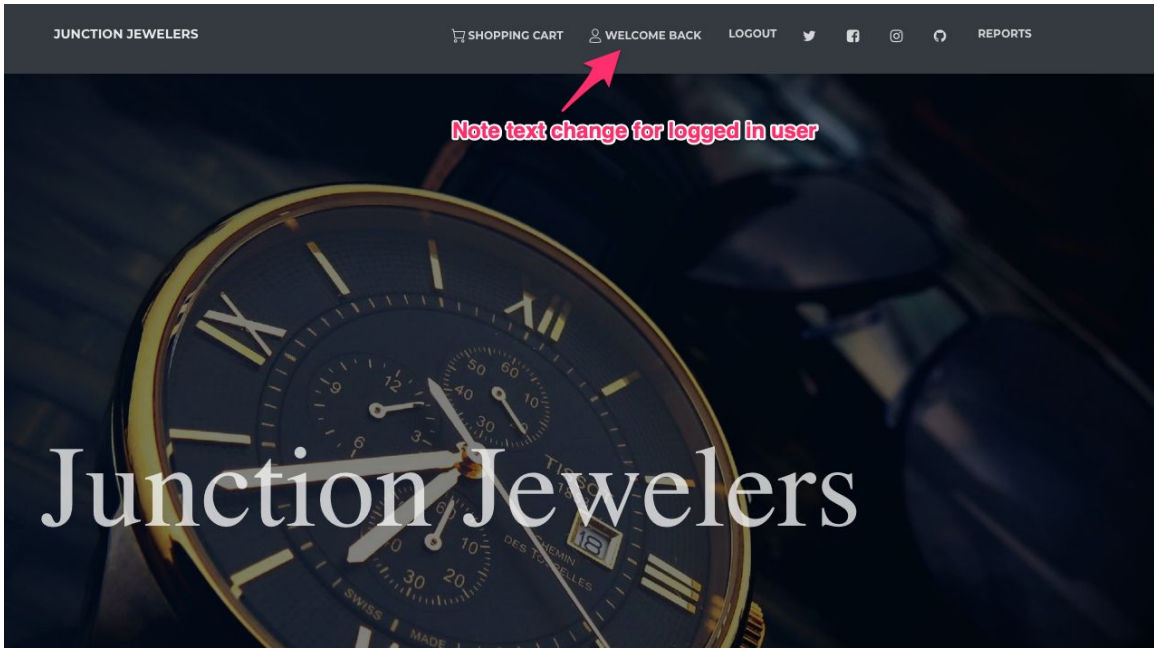


Action: Adds Most Popular Item to Cart

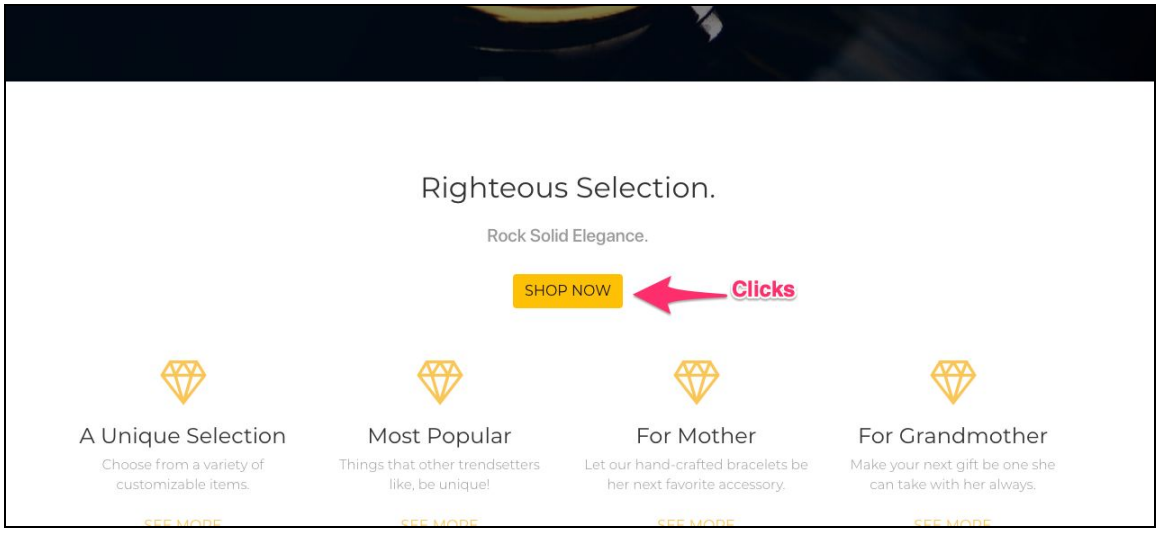
Requirements Addressed:

1.6 - A list of the most popular products in inventory

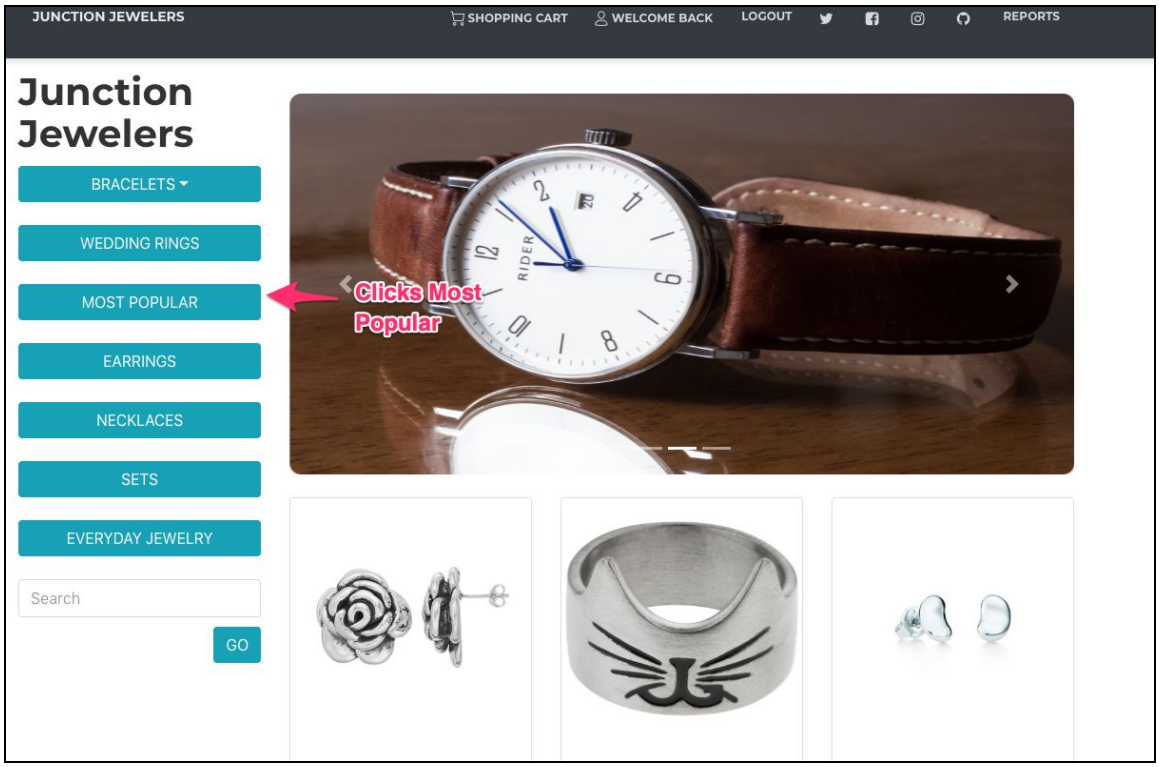
Goes back to homepage



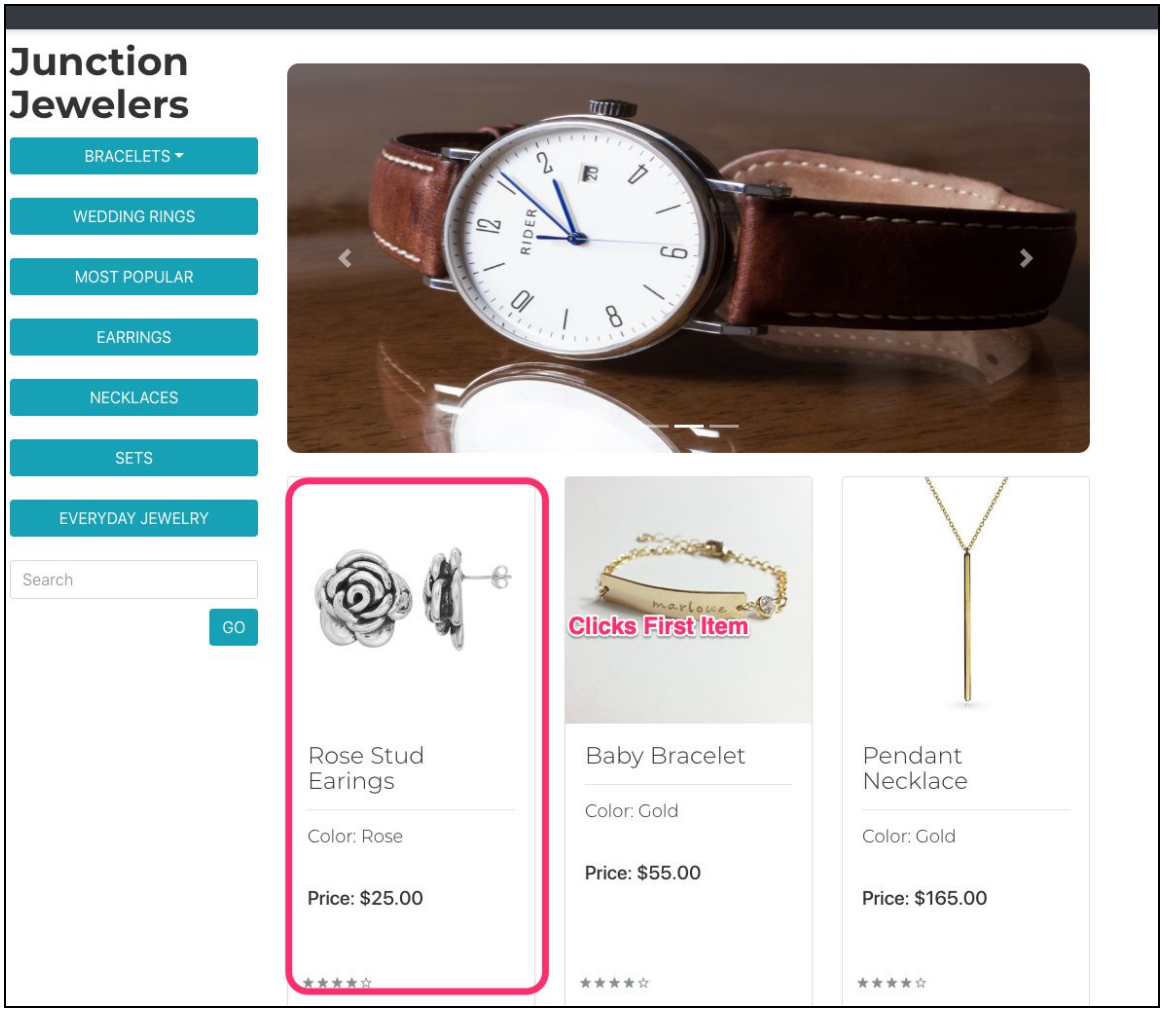
Clicks Shop Now to go to store



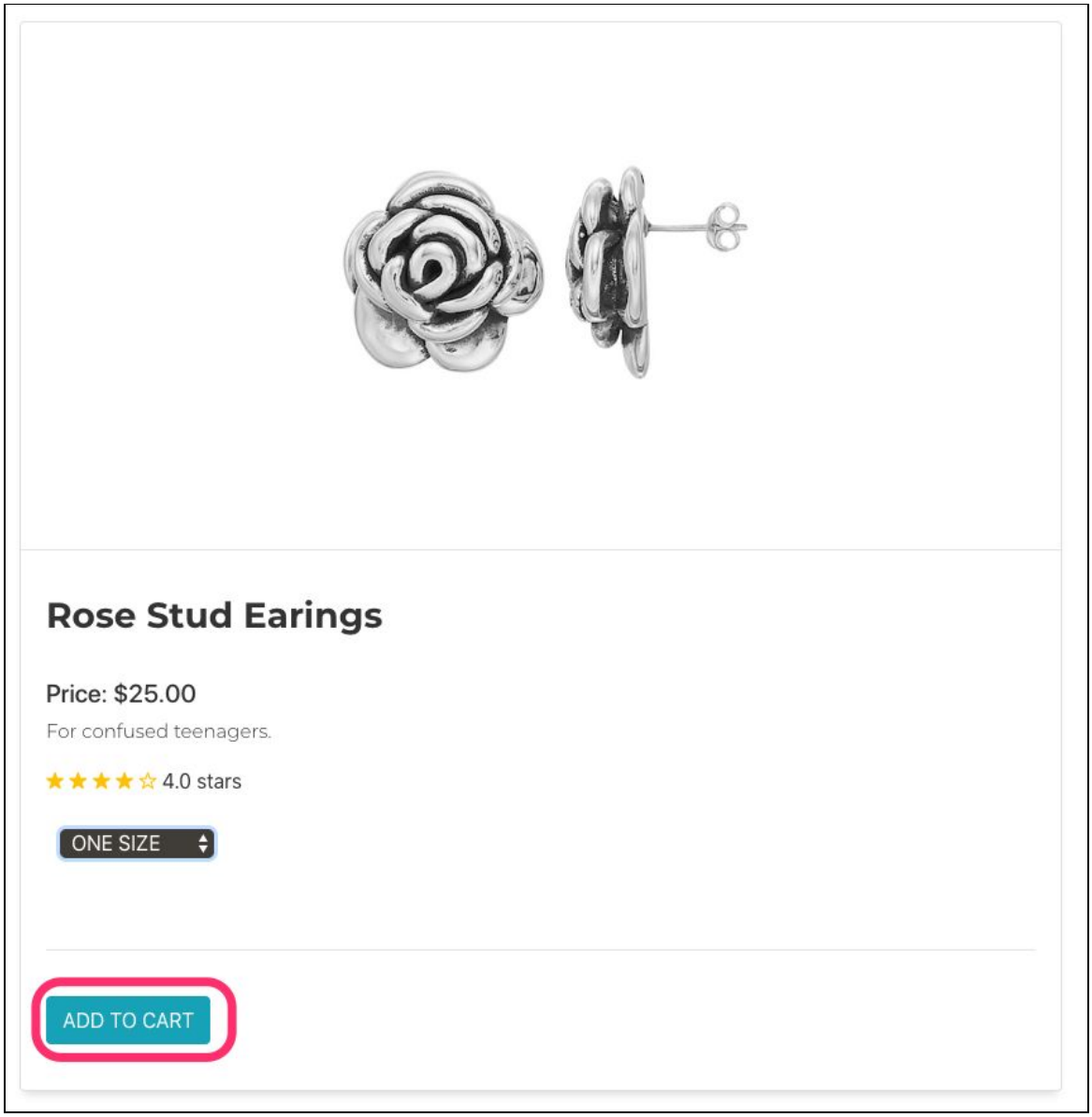
Clicks most popular items



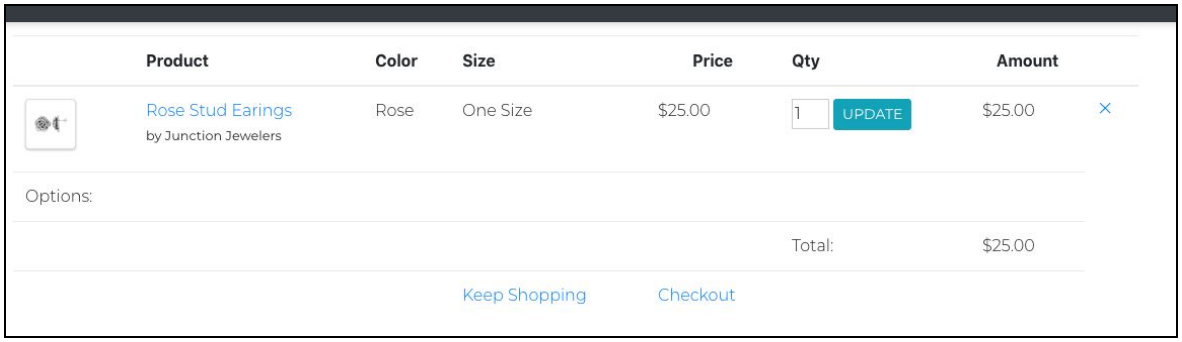
Clicks first item in Most Popular



Adds item to cart



Cart is updated with product



Action: Adds Customized Mothers Bracelet to Cart

Requirements Addressed

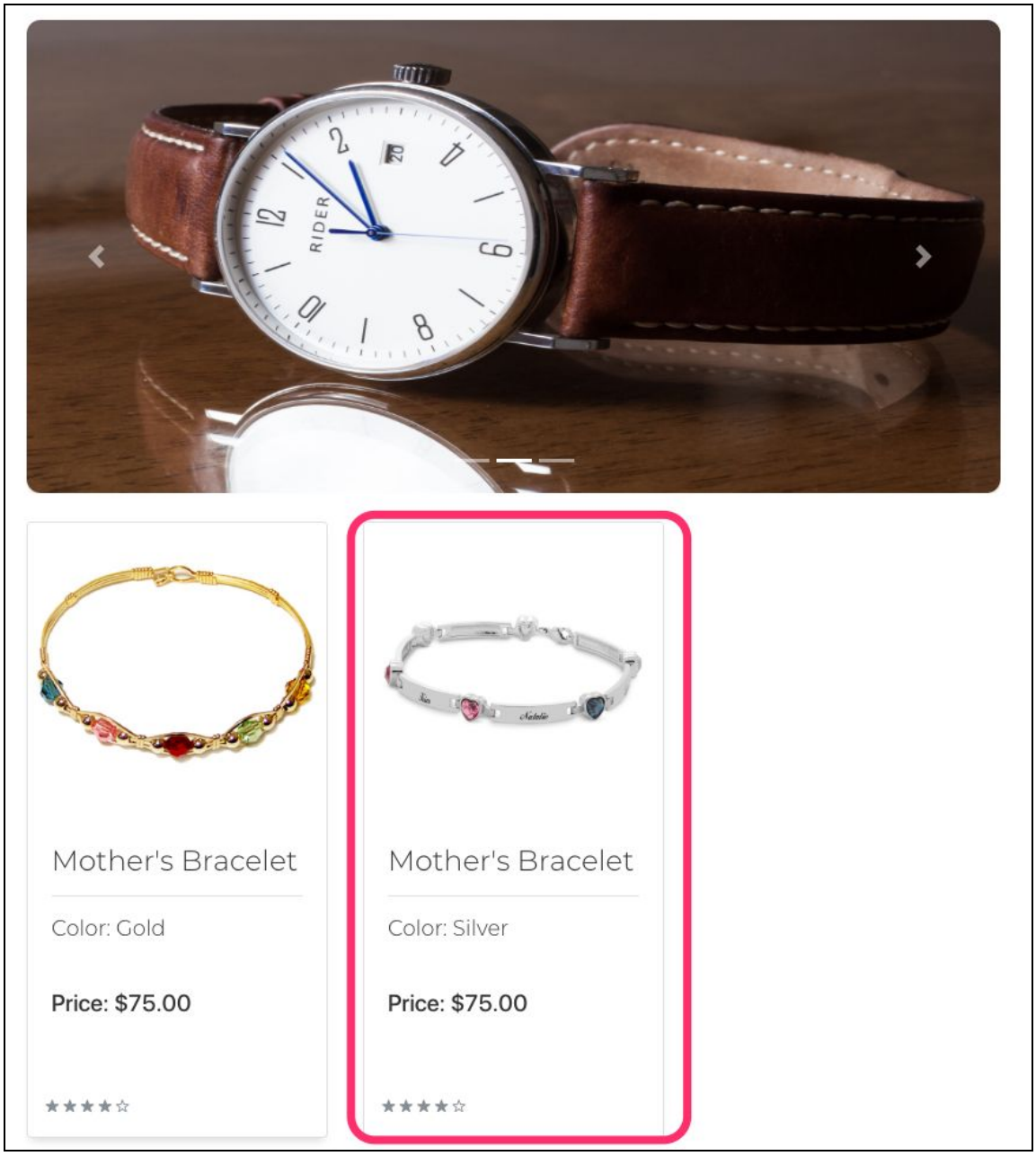
1.10 - Ability to select customizations for some products

1.13 - Ability to view items by their category

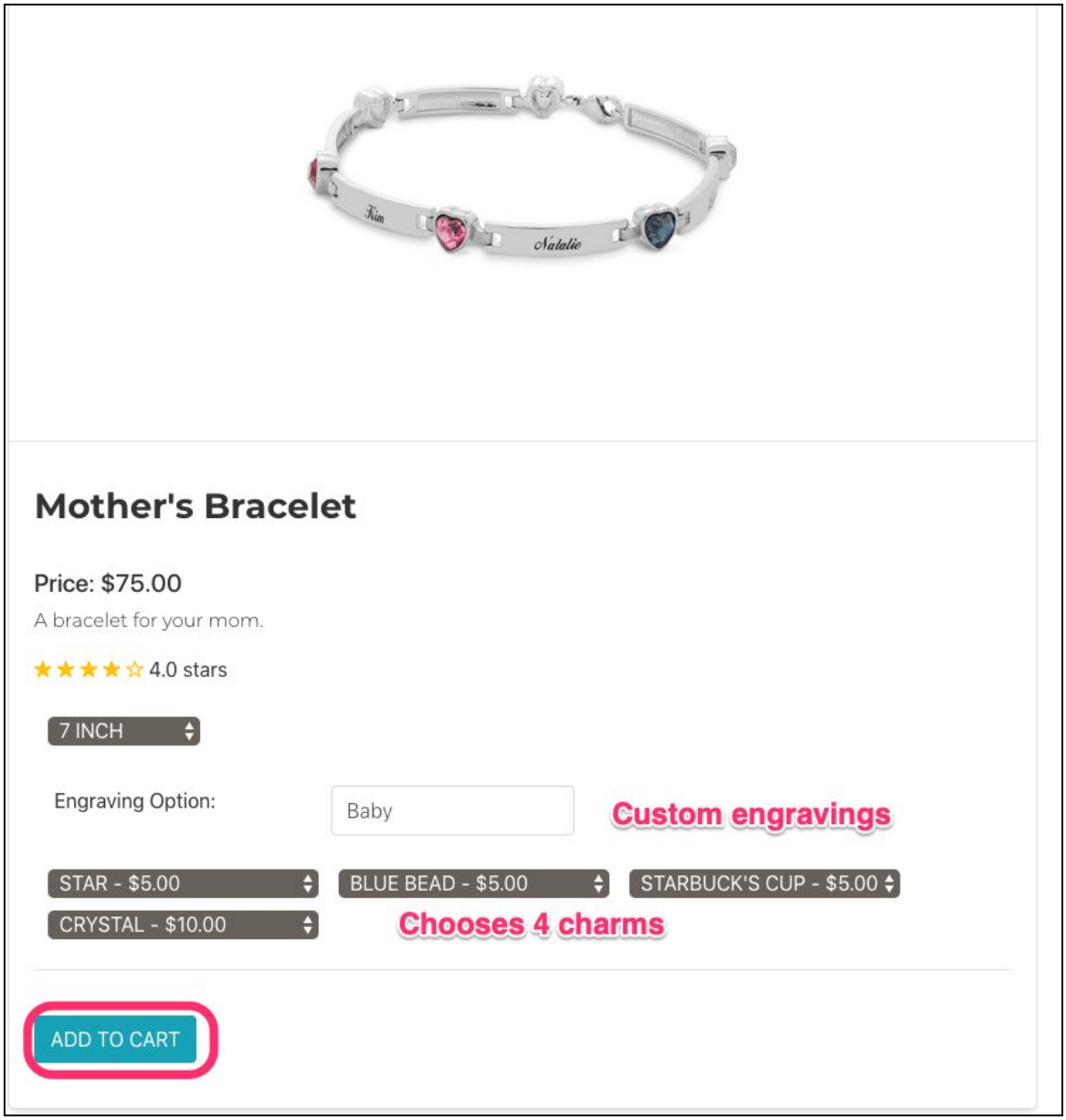
1.15 - The ability to add multiple items to the cart

Navigates to Mothers Bracelets

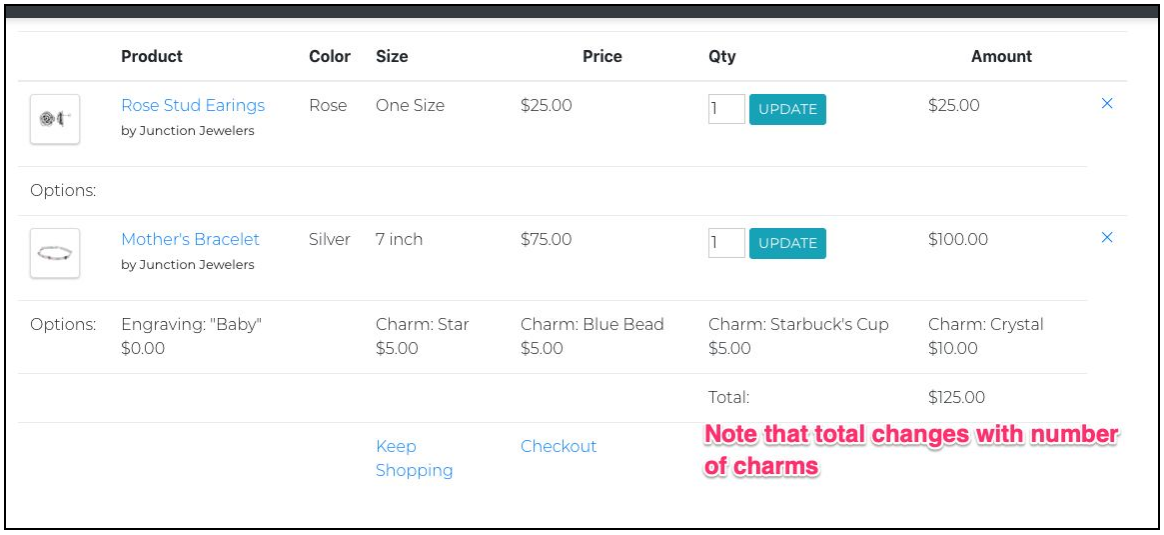
Chooses Silver Bracelet



Customizes 7 inch bracelet with “Baby” engraving, and 4 charms: a blue bead, starbucks cup, star and crystal



Mothers Bracelet is added to Cart



Action: Adds Customized Baby Bracelet to Cart

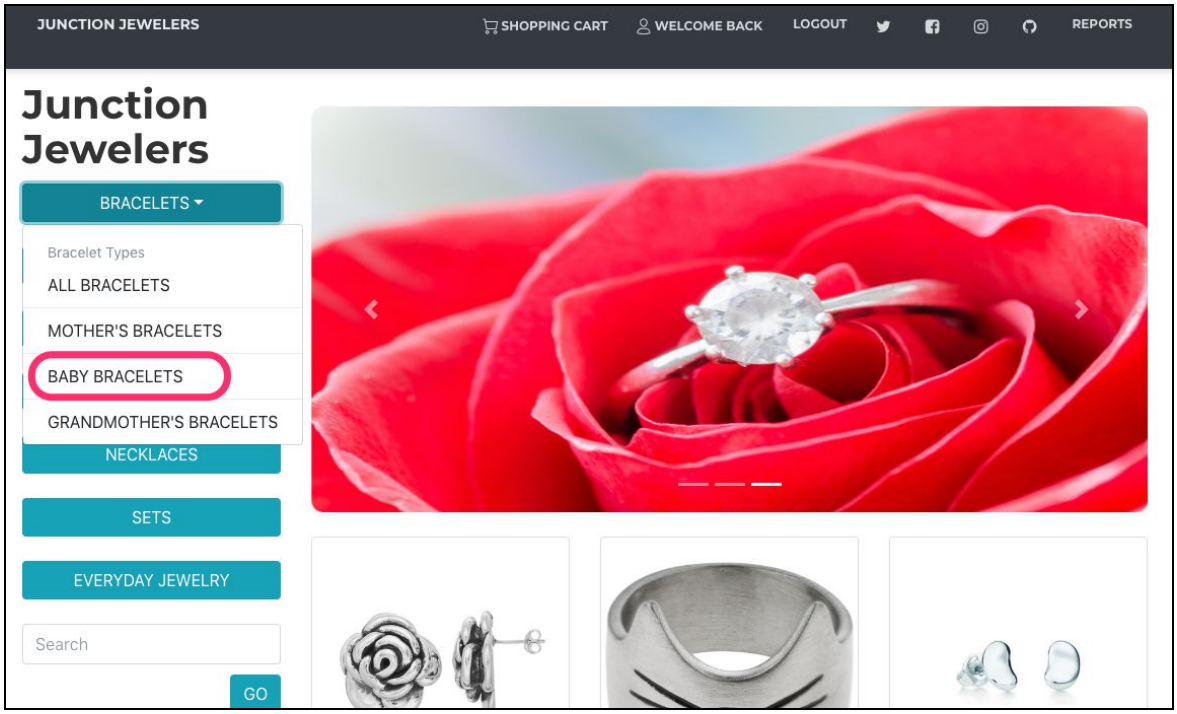
Requirements Addressed

1.10 - Ability to select customizations for some products

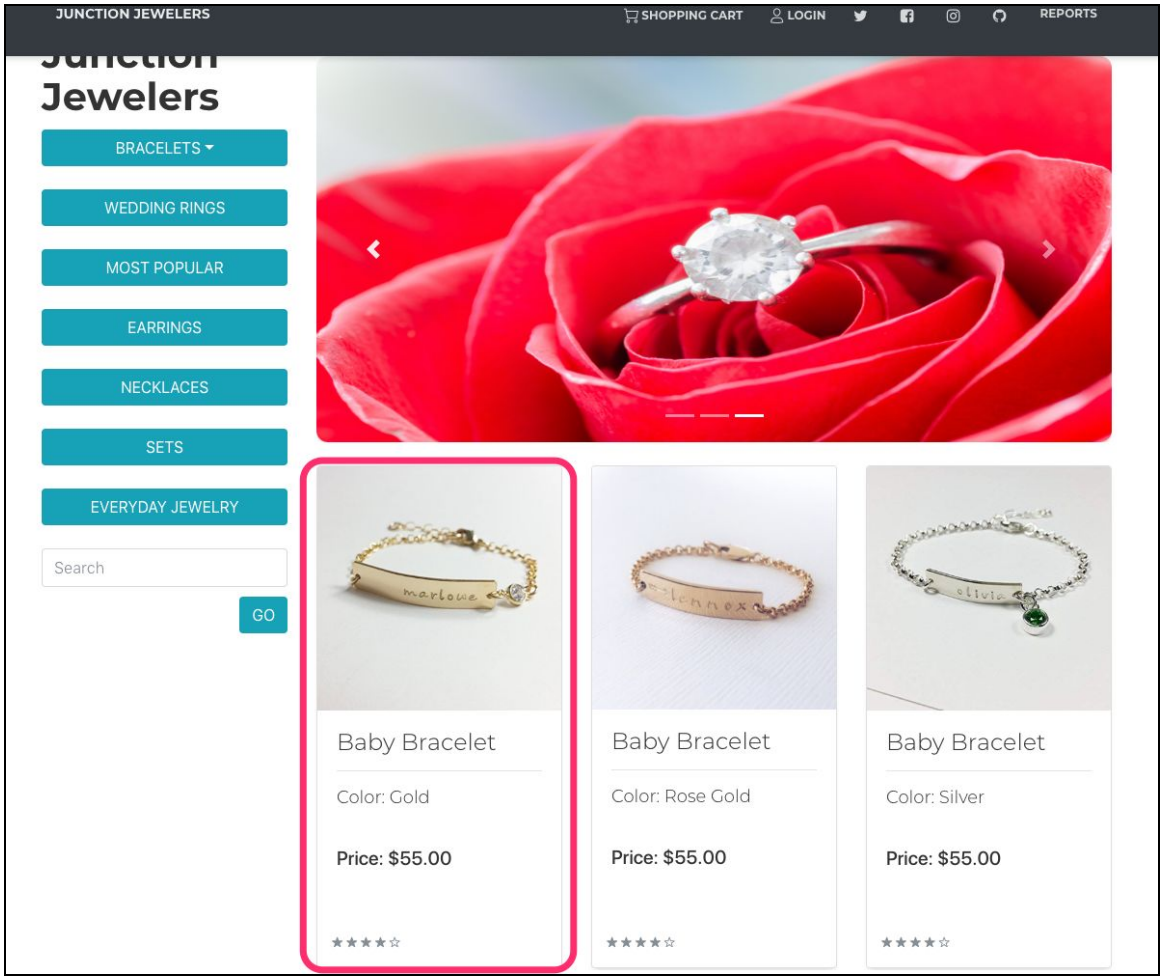
1.13 - Ability to view items by their category

1.15 - The ability to add multiple items to the cart

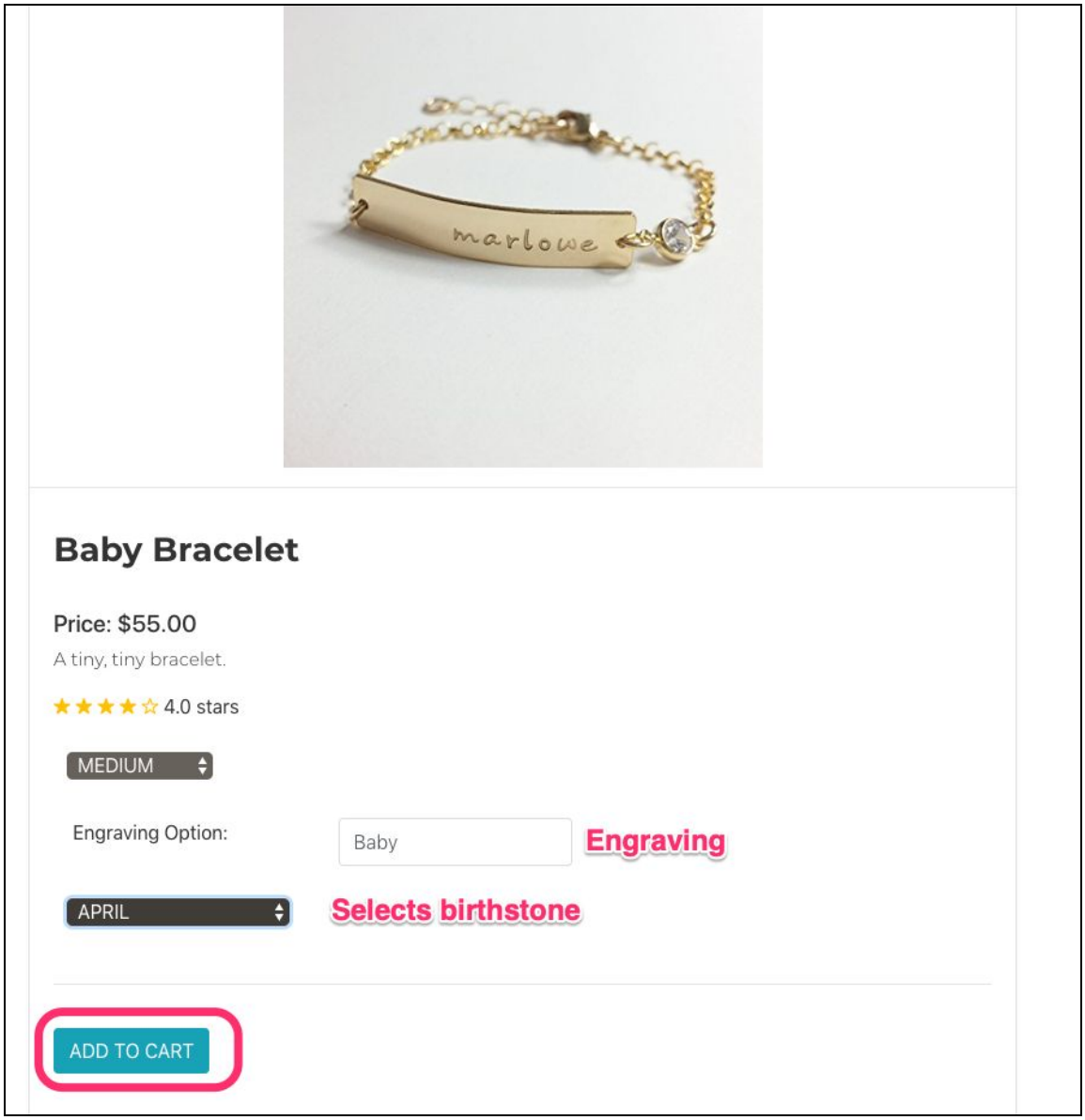
Navigates to Baby Bracelets



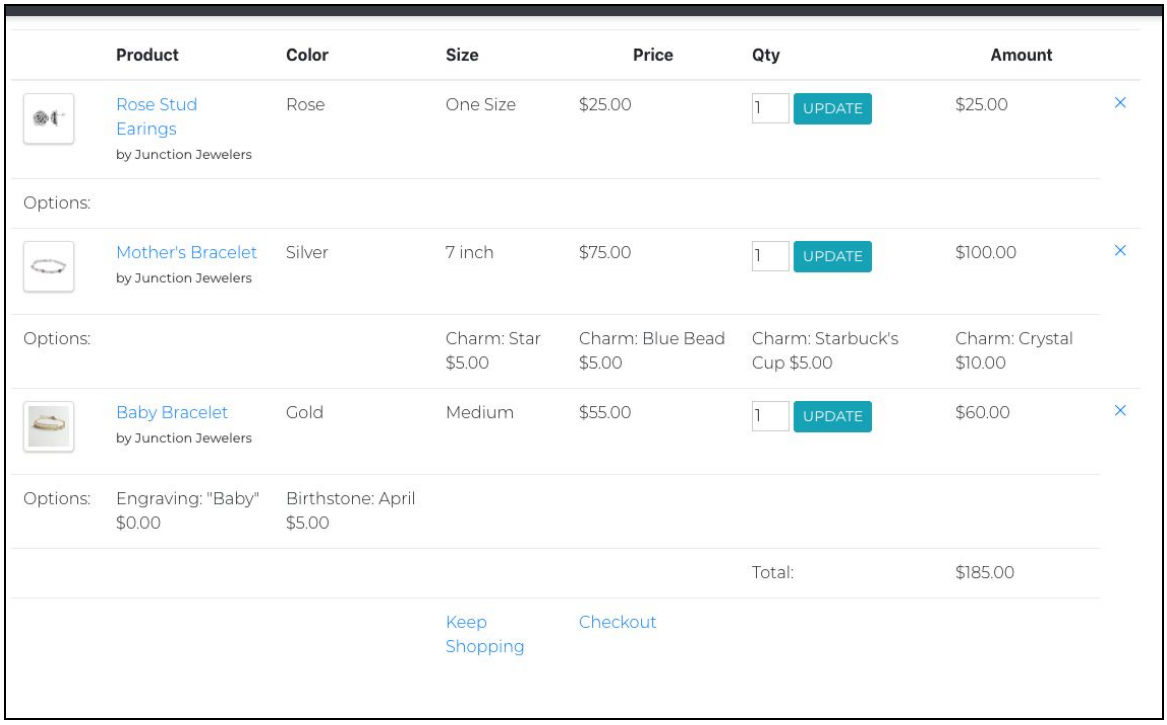
Selects Gold Baby Bracelet



Adds to Cart in Medium Size, with April Birthstone



View of Updated Cart



	Product	Color	Size	Price	Qty	Amount	
	Rose Stud Earings by Junction Jewelers	Rose	One Size	$25.00	1 UPDATE	$25.00	×
Options:							
	Mother's Bracelet by Junction Jewelers	Silver	7 inch	$75.00	1 UPDATE	$100.00	×
Options:			Charm: Star $5.00	Charm: Blue Bead $5.00	Charm: Starbuck's Cup $5.00	Charm: Crystal $10.00	
	Baby Bracelet by Junction Jewelers	Gold	Medium	$55.00	1 UPDATE	$60.00	×
Options:	Engraving: "Baby" $0.00	Birthstone: April $5.00					
					Total:	$185.00	
			Keep Shopping	Checkout			

Adds Engraved Grandmothers Bracelet to Cart

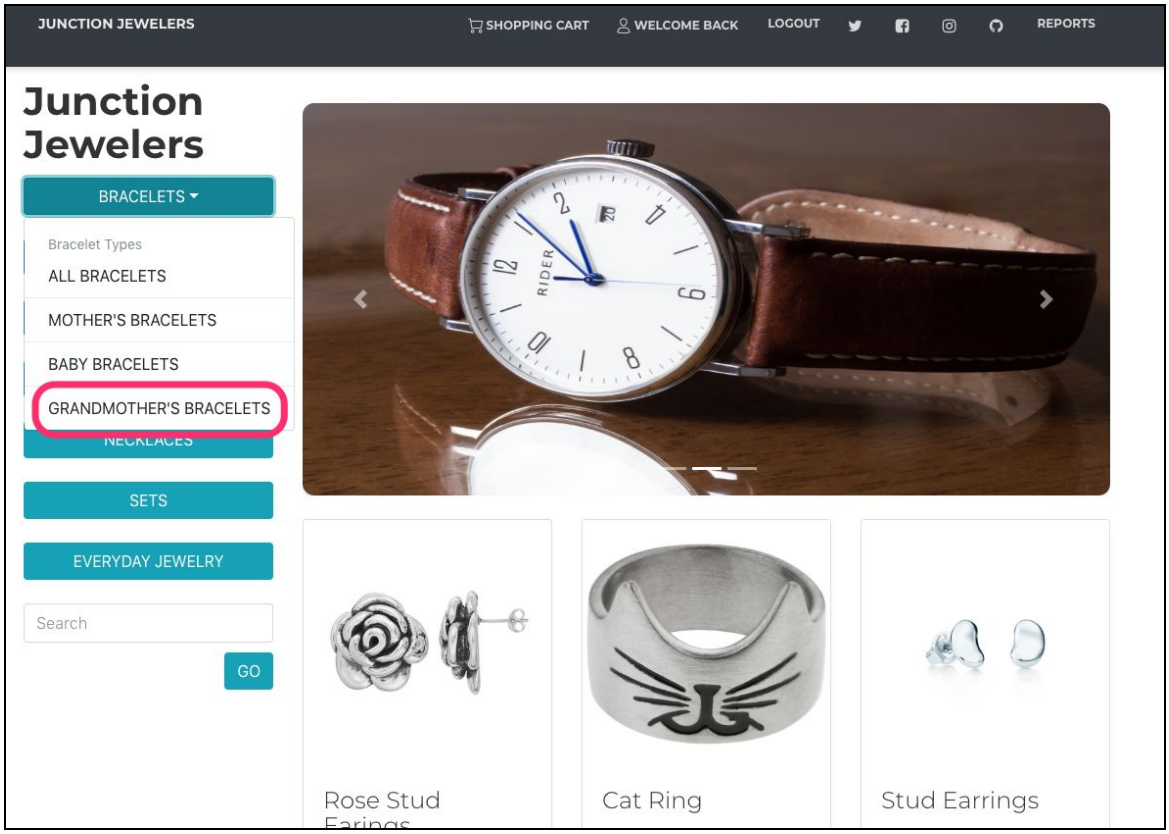
Requirements Addressed

1.10 - Ability to select customizations for some products

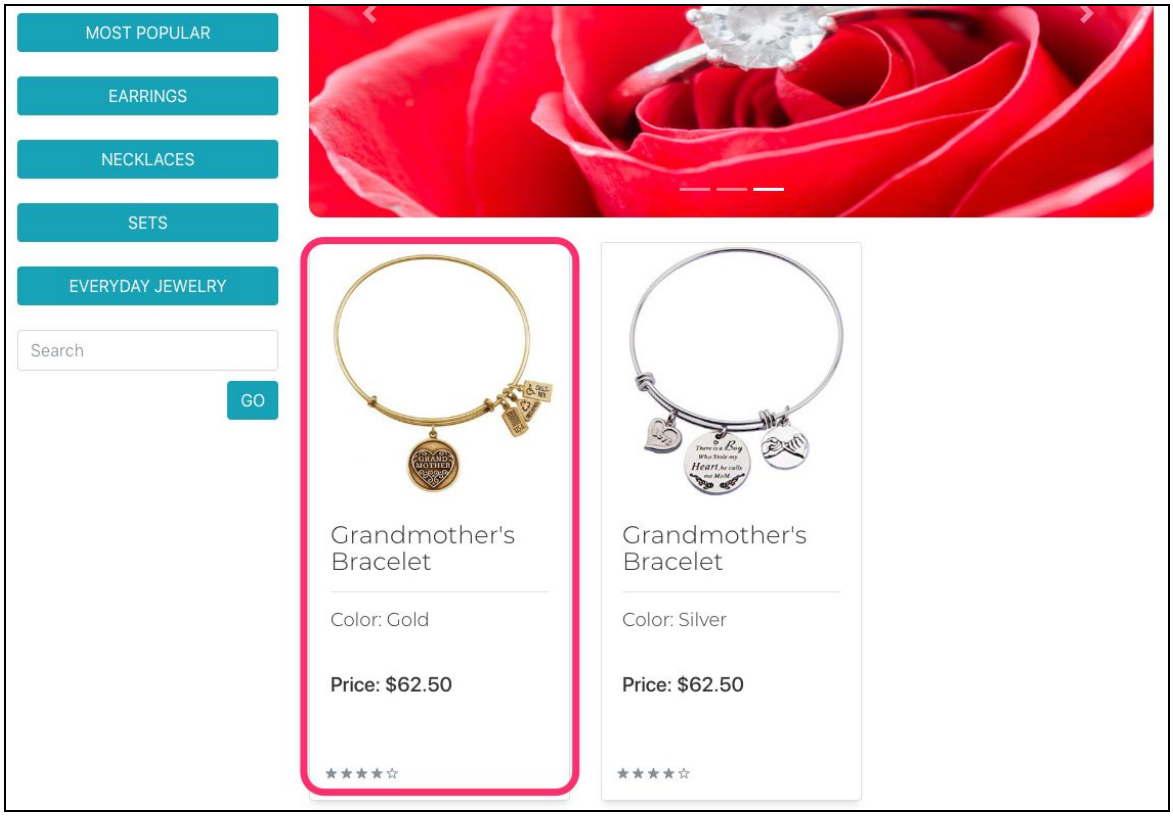
1.13 - Ability to view items by their category

1.15 - The ability to add multiple items to the cart

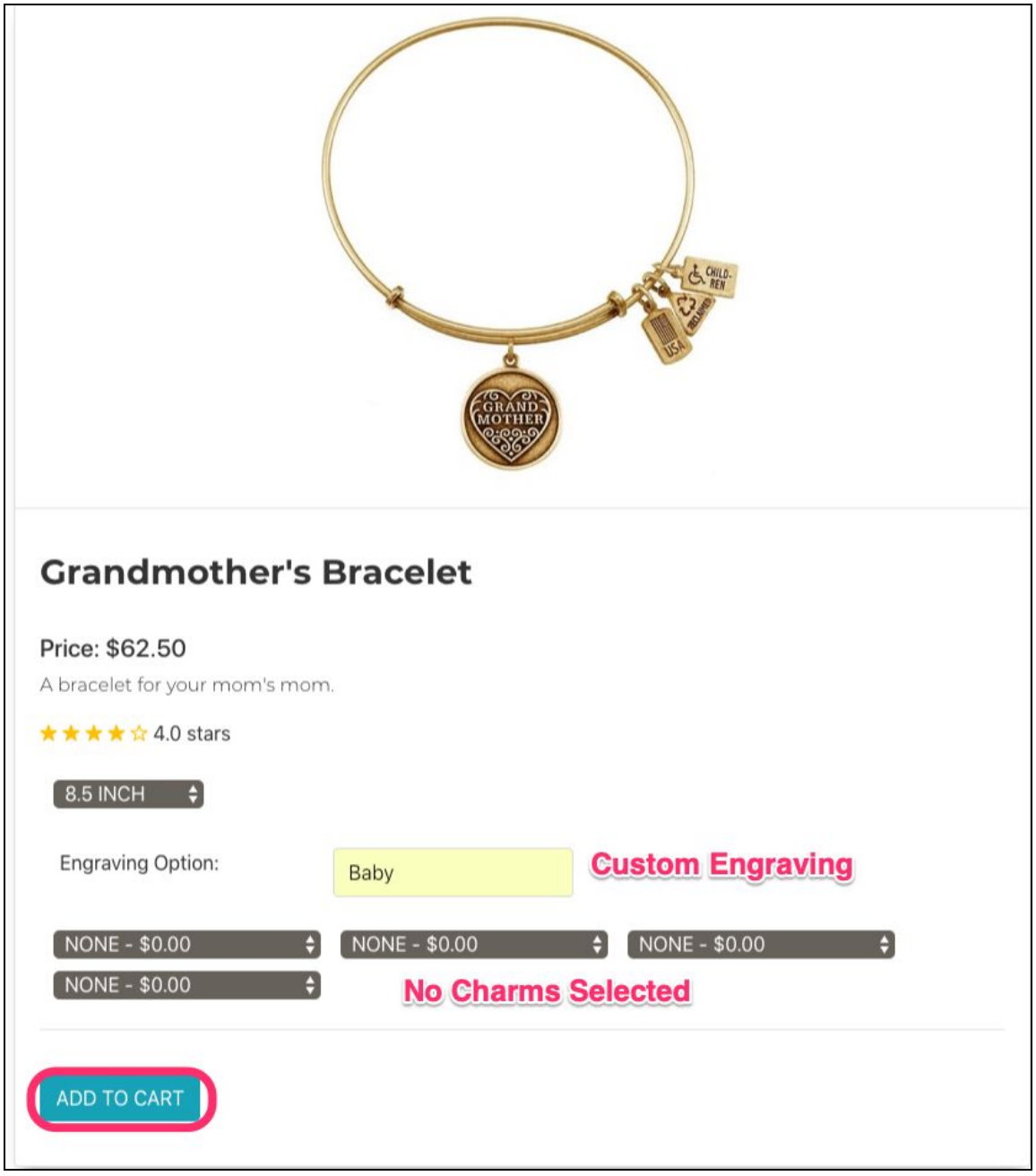
Navigates to Grandmothers Bracelets



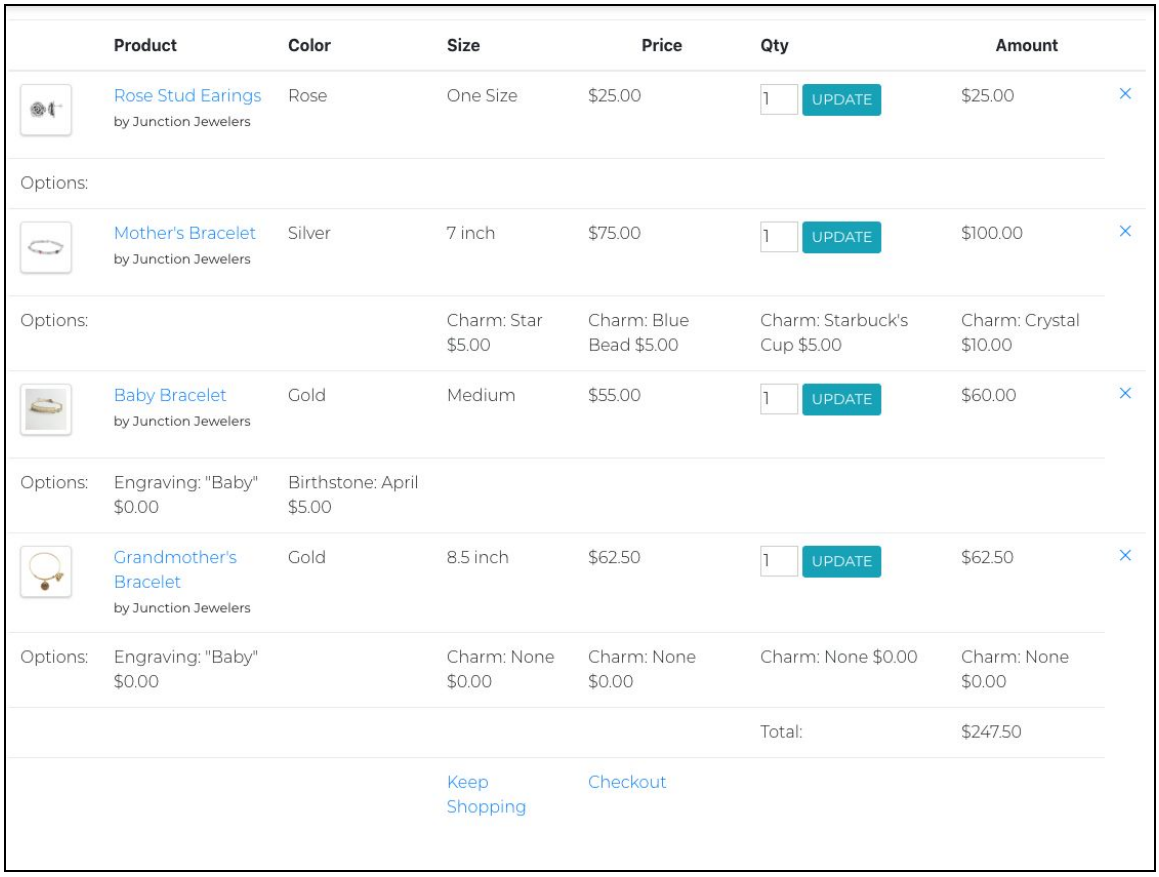
Selects Gold Grandmothers Bracelet



Adds to Cart in 8.5 in length, with engraving



View of Updated Cart



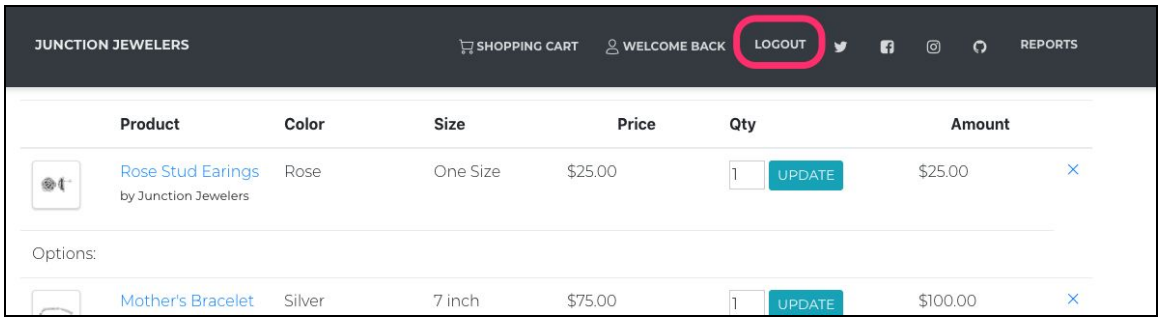
	Product	Color	Size	Price	Qty	Amount	
	Rose Stud Earings by Junction Jewelers	Rose	One Size	$25.00	1 UPDATE	$25.00	×
Options:							
	Mother's Bracelet by Junction Jewelers	Silver	7 inch	$75.00	1 UPDATE	$100.00	×
Options:			Charm: Star $5.00	Charm: Blue Bead $5.00	Charm: Starbuck's Cup $5.00	Charm: Crystal $10.00	
	Baby Bracelet by Junction Jewelers	Gold	Medium	$55.00	1 UPDATE	$60.00	×
Options:	Engraving: "Baby" $0.00	Birthstone: April $5.00					
	Grandmother's Bracelet by Junction Jewelers	Gold	8.5 inch	$62.50	1 UPDATE	$62.50	×
Options:	Engraving: "Baby" $0.00		Charm: None $0.00	Charm: None $0.00	Charm: None $0.00	Charm: None $0.00	
					Total:	$247.50	
			Keep Shopping	Checkout			

<u>Action: Changes Mind and Logs Out of Account</u>

Requirements Addressed:

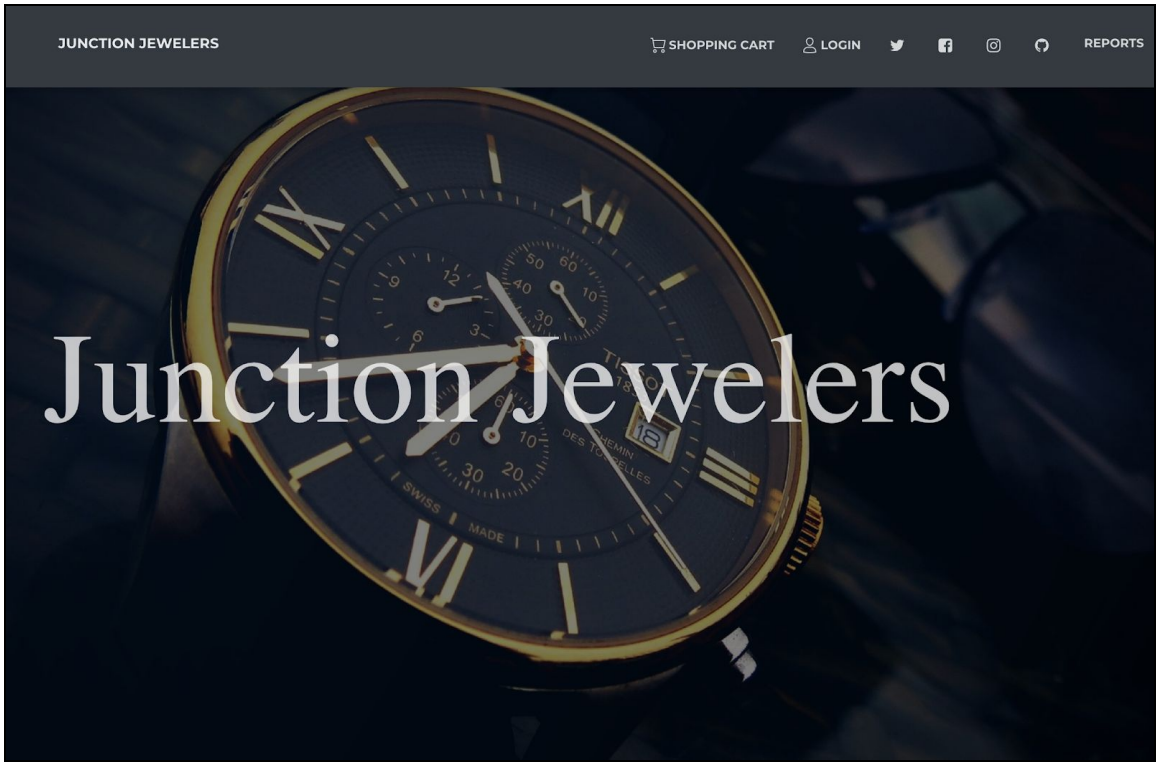
1.16 - The ability to log out of an account

Clicks Logout



	Product	Color	Size	Price	Qty	Amount	
	Rose Stud Earings by Junction Jewelers	Rose	One Size	$25.00	1 UPDATE	$25.00	×
Options:							
	Mother's Bracelet	Silver	7 inch	$75.00	1 UPDATE	$100.00	×

Returns to Homepage



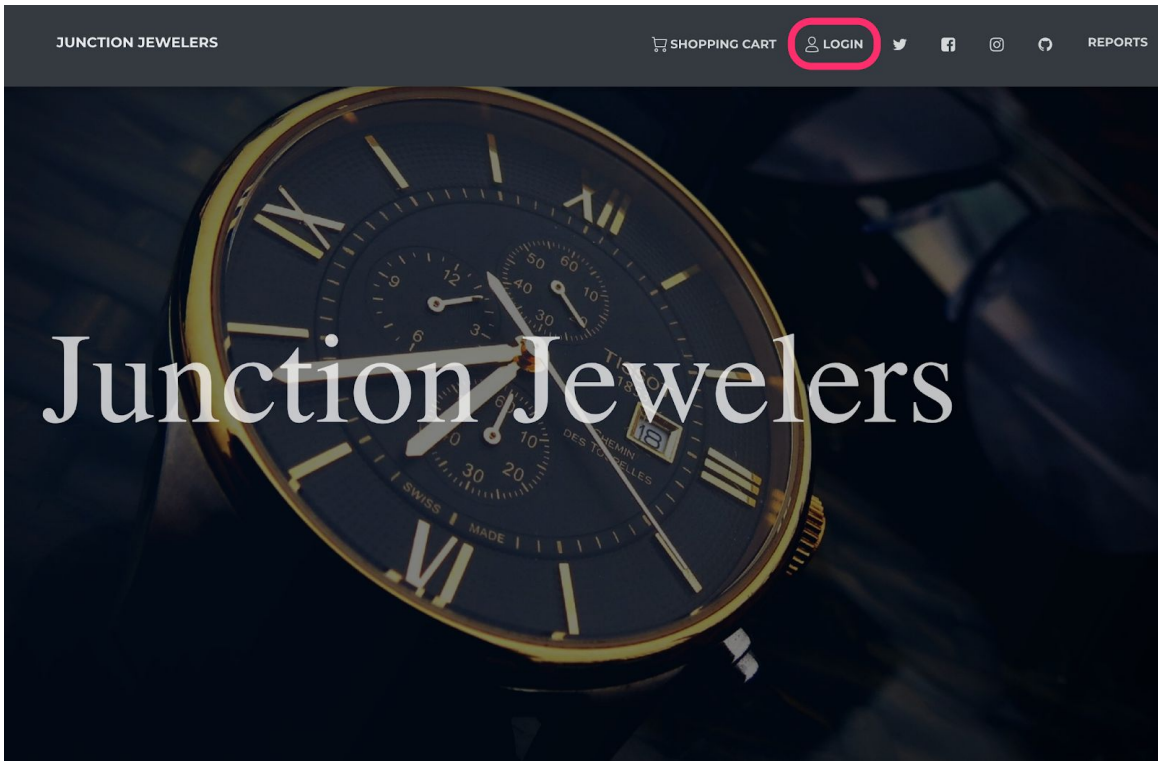
Persona 3 Test Path Validation

Action: Arrive to Store Homepage and Login

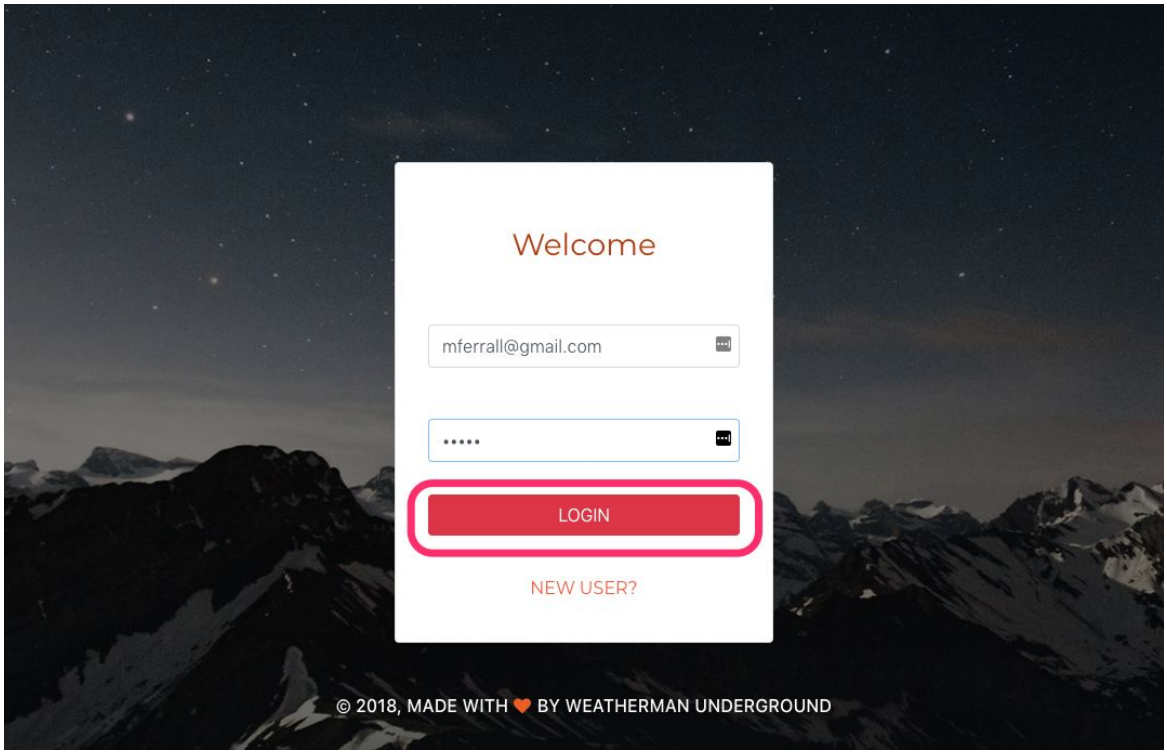
Requirements Addressed:

1.3 - The ability to log in to an existing customer account

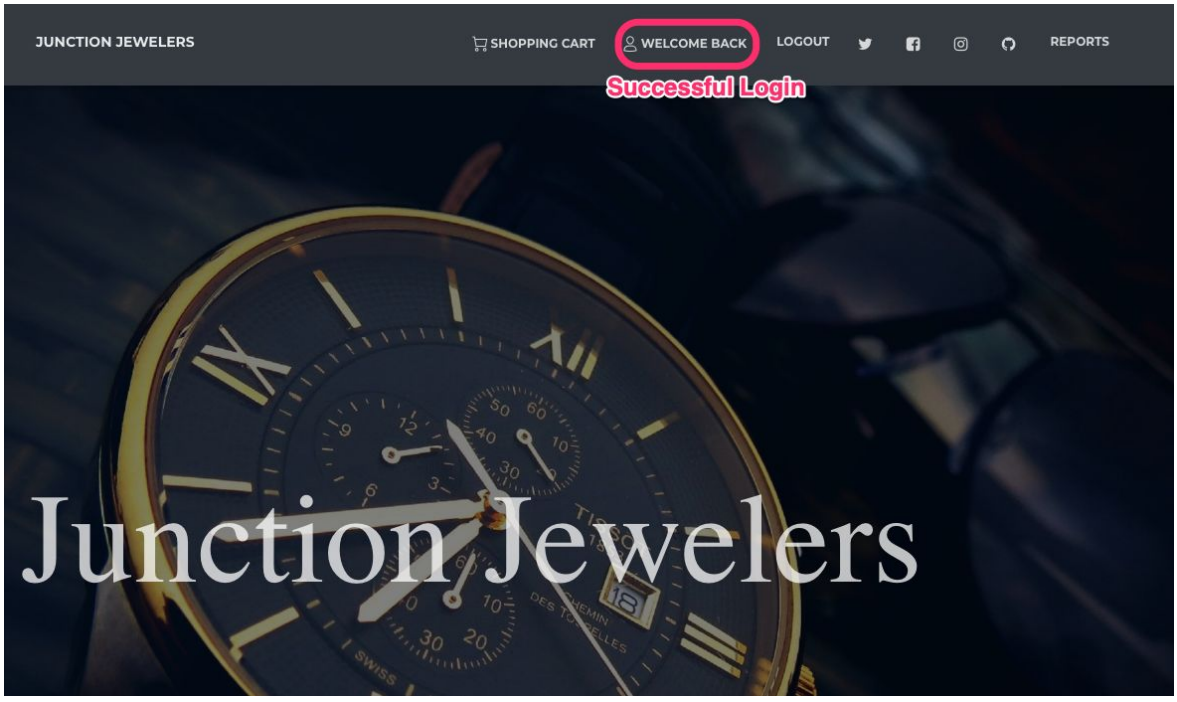
Arrive at Store and Click Login



Enters Username and Password

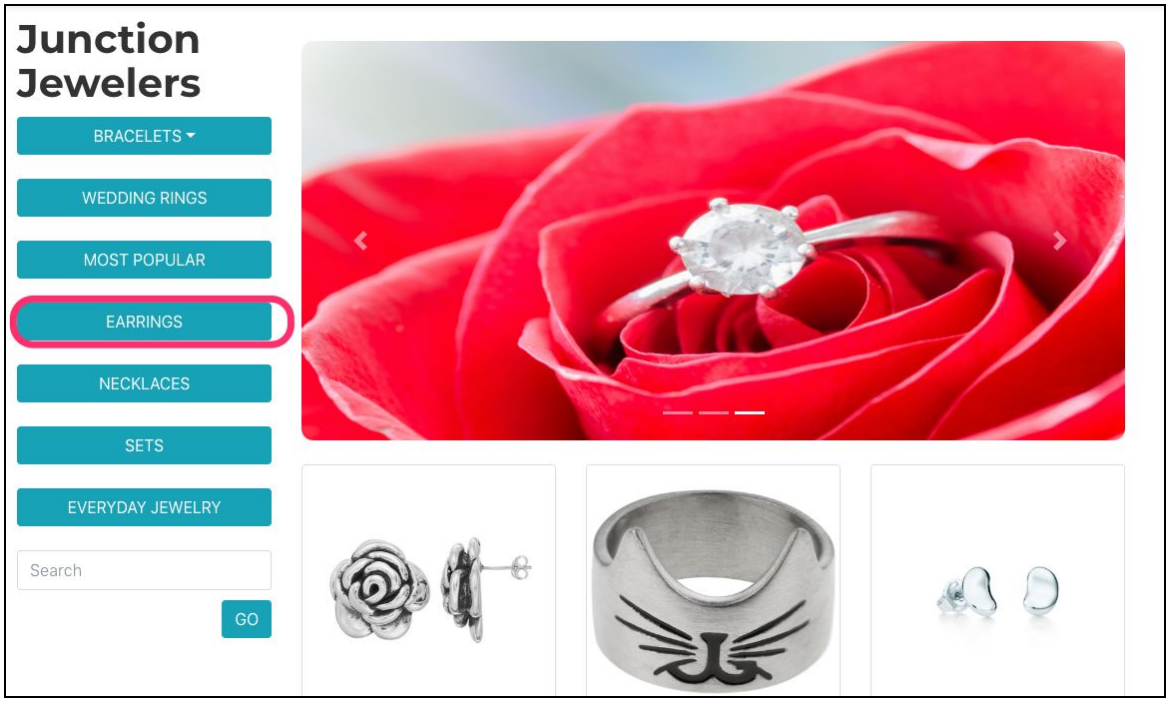


Returns to Home Page

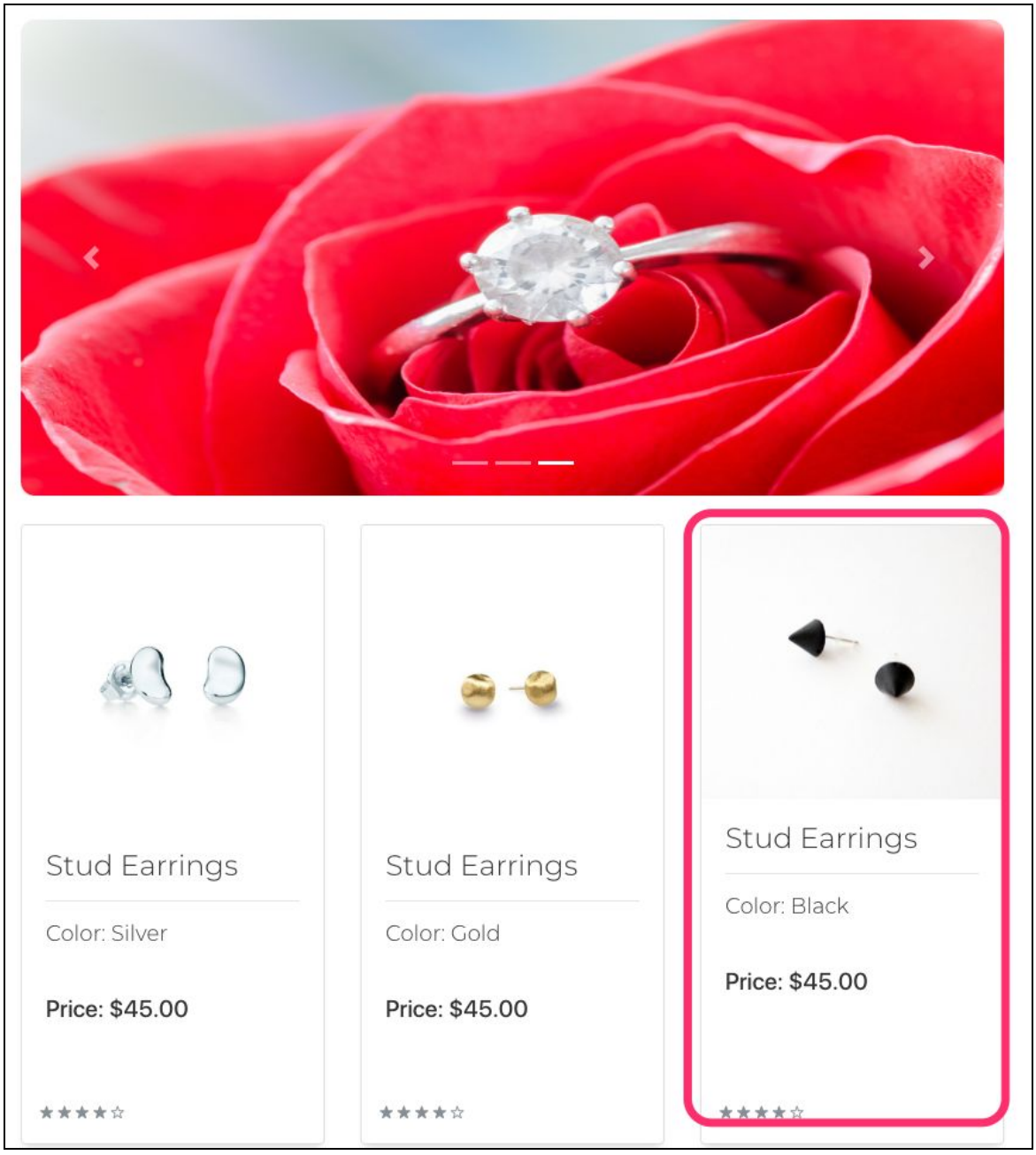


Action: Adds Black Stud Earrings to Cart

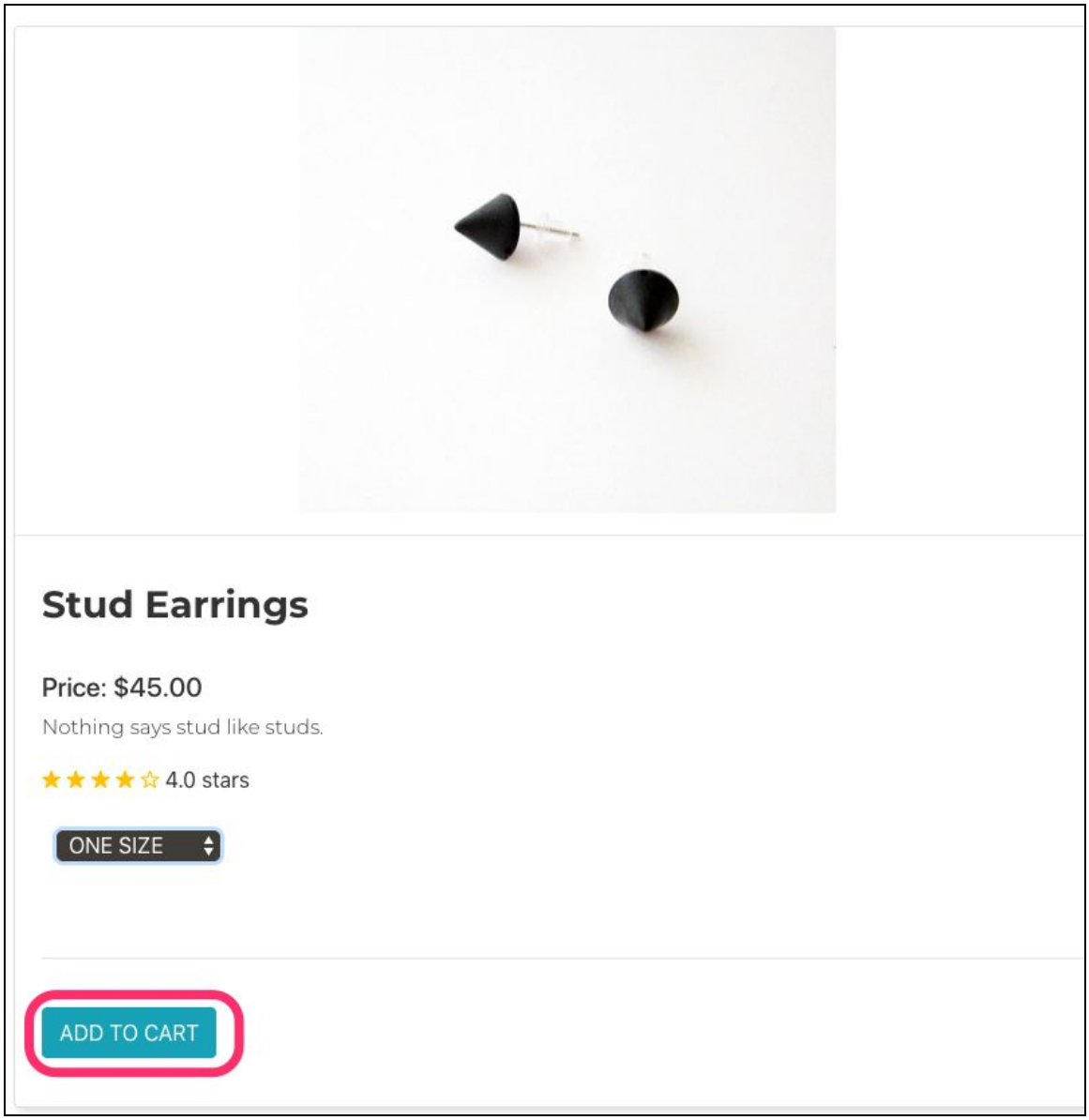
Navigates to Earring Category



Finds Black Stud Earrings



Adds to Cart

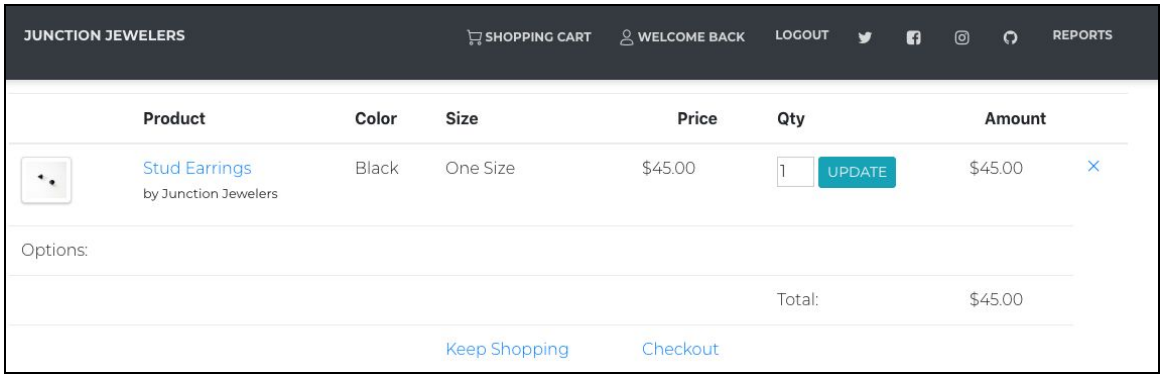

Stud Earrings
Price: $45.00
Nothing says stud like studs.
4.0 stars
ONE SIZE
ADD TO CART

Action: Increases Quantity

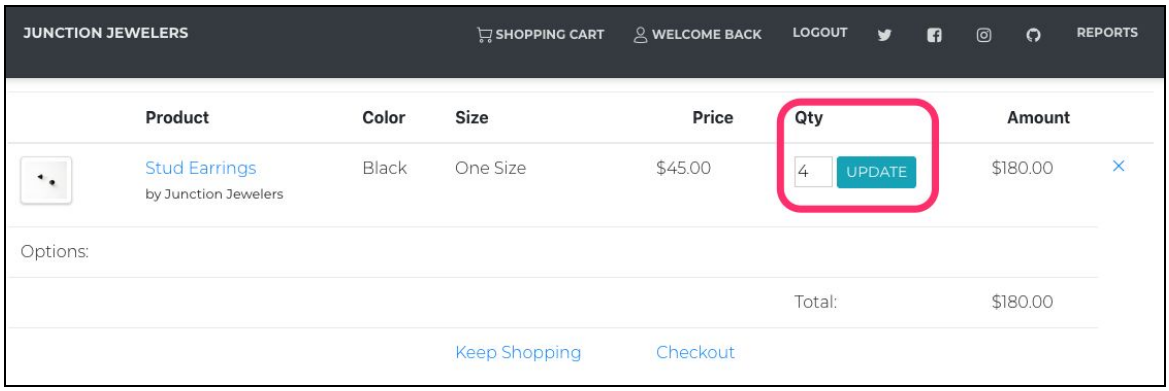
Requirements Addressed

3.2 - Provide the ability to change the quantity of an item in the cart

Initial View of Cart



Updates Quantity to Four



Action: Checks Out

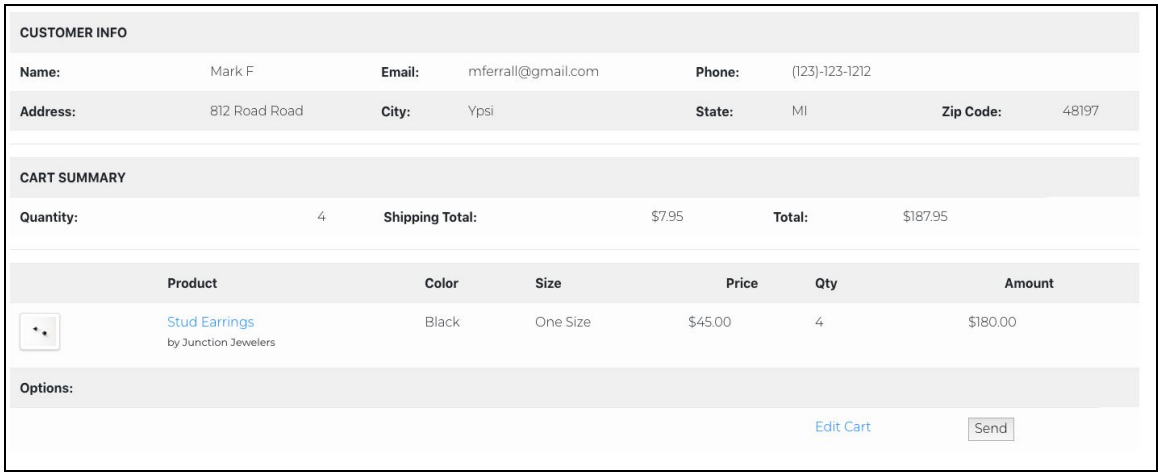
Requirements Addressed

1.9 - Ability to place multiple orders

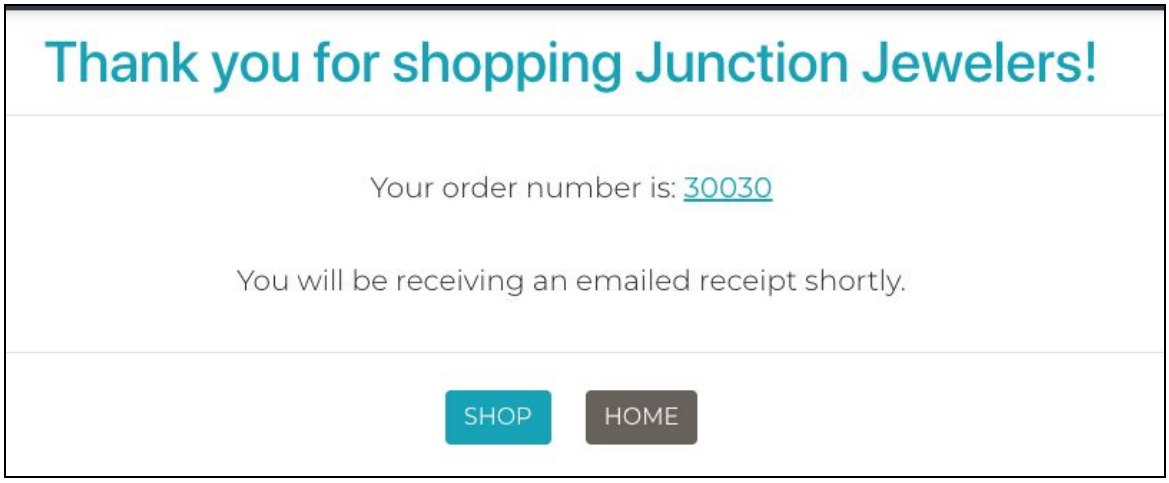
1.11 - Ability to order more than 1 of an item

3.6 - Send users an emailed receipt upon checkout

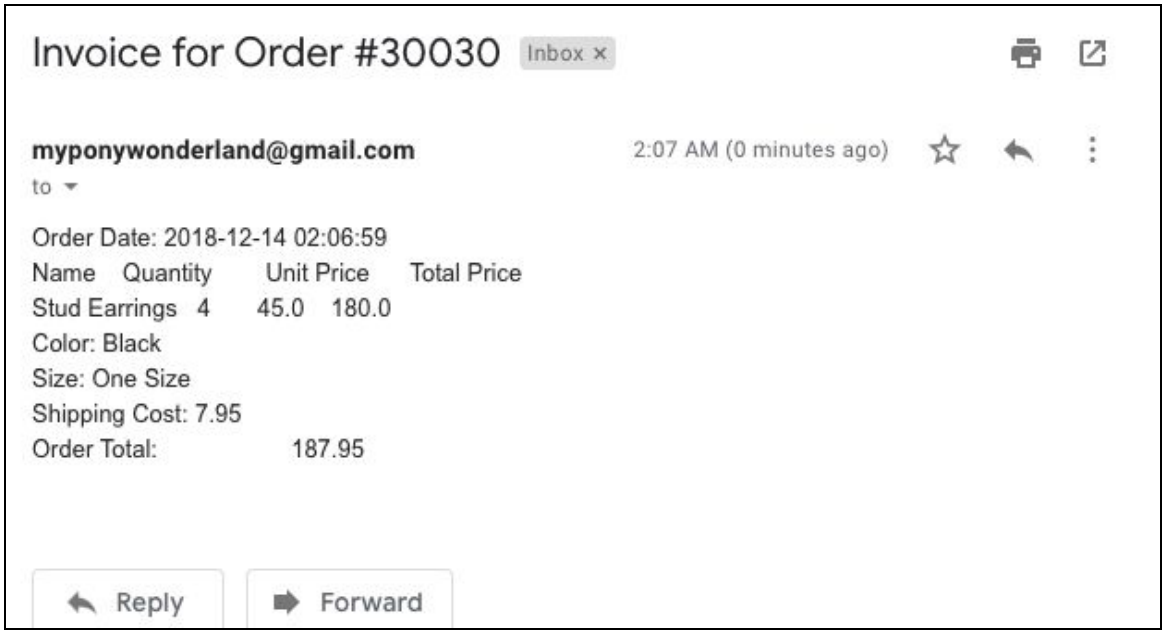
Pre Completion Confirmation Page



Order Confirmation Page

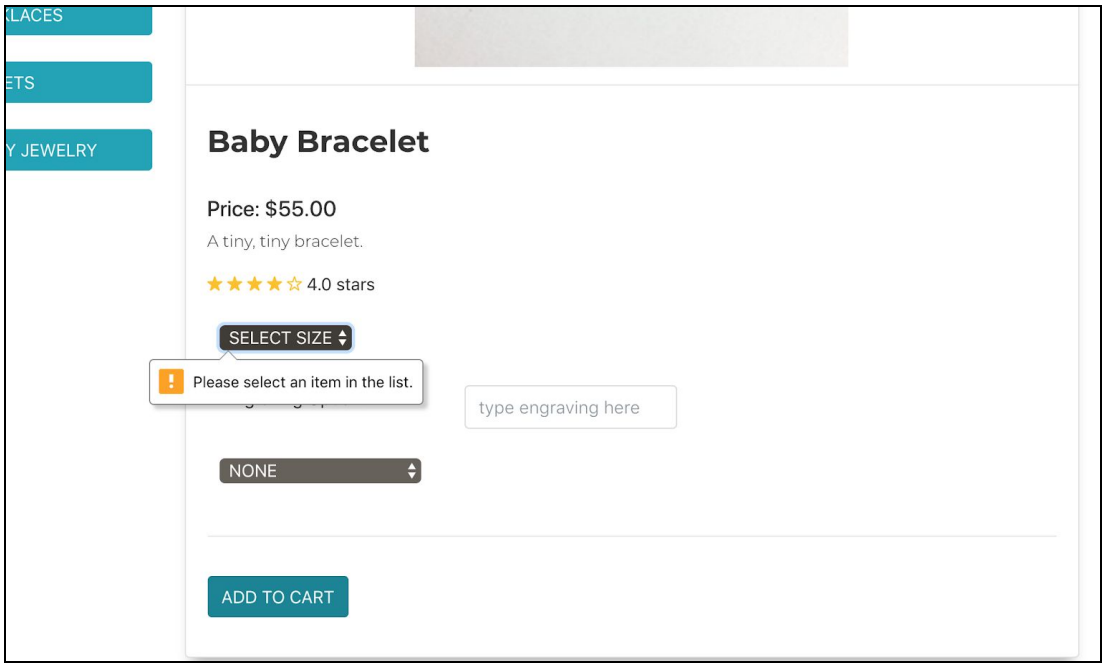


Confirmation Email



Constraint Tests Validation

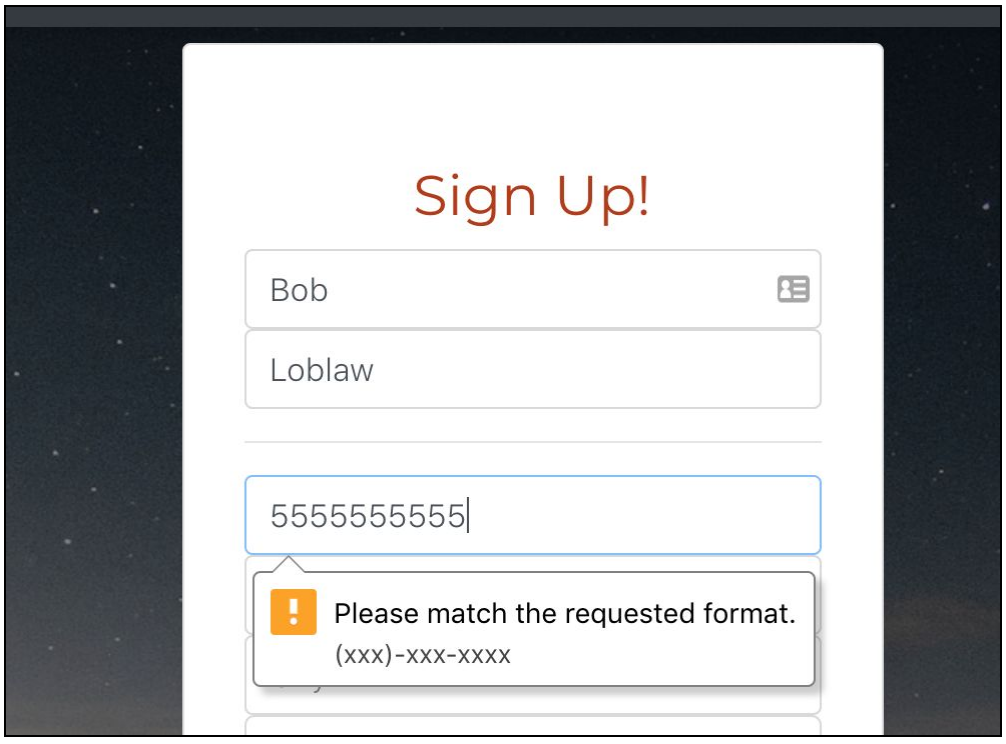
Prevent Adding Items to Cart without Required Options (Size)



Prevent Registration with Invalid Phone Number

Requirements Addressed

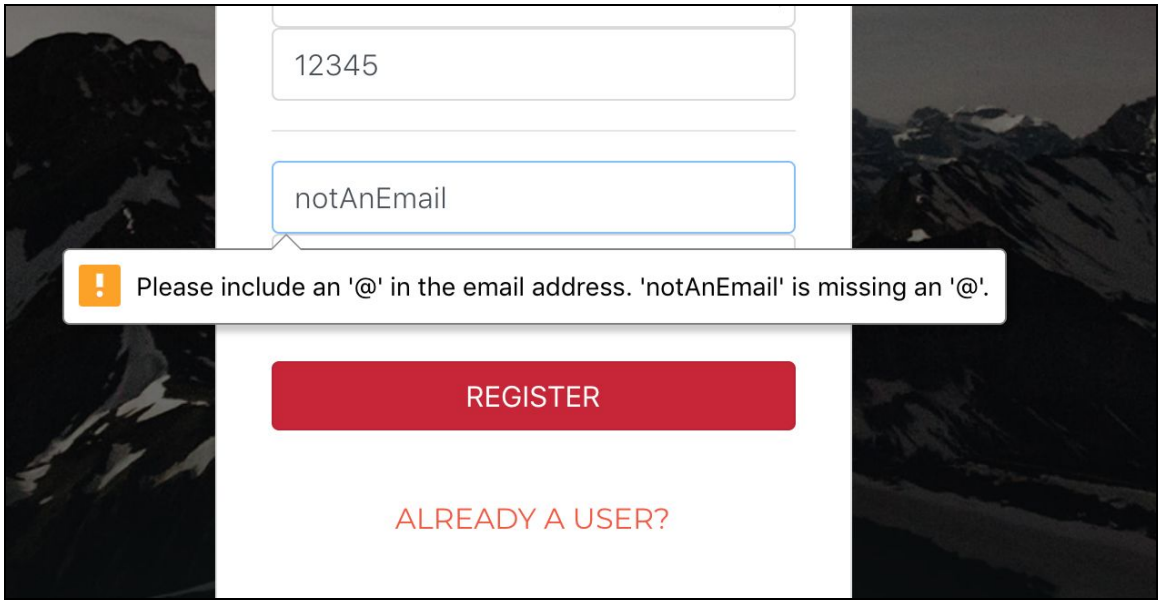
6.3 - Constrain improperly specified user account information



Prevent Registration with Invalid Email

Requirement Addressed

6.3 - Constrain improperly specified user account information

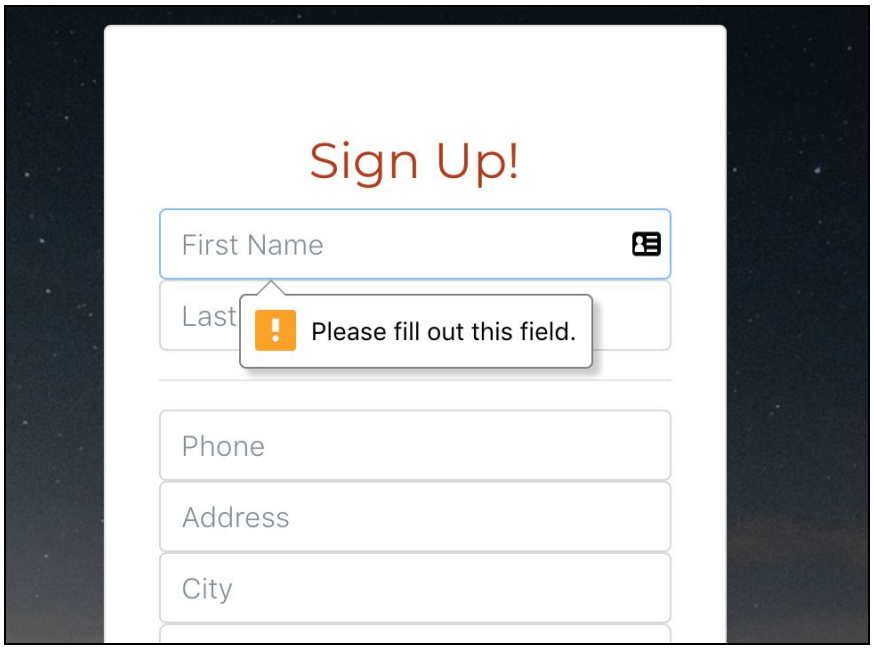


Prevent Registration without Required Information

Requirements Addressed

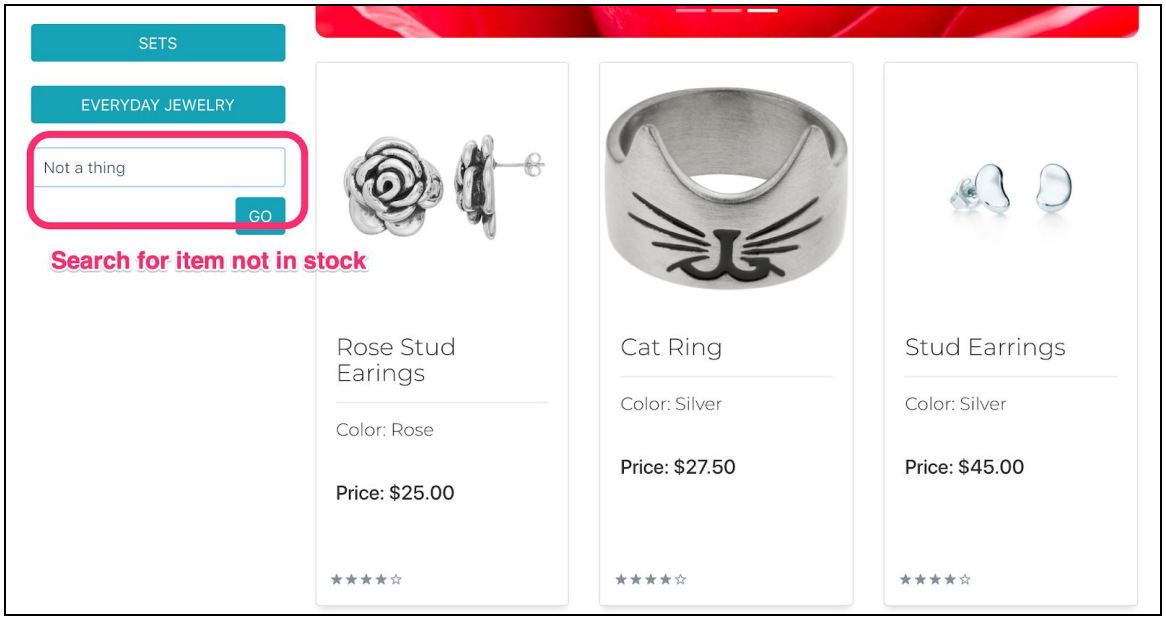
6.3 - Constrain improperly specified user account information

Attempt to Register without Entering Name

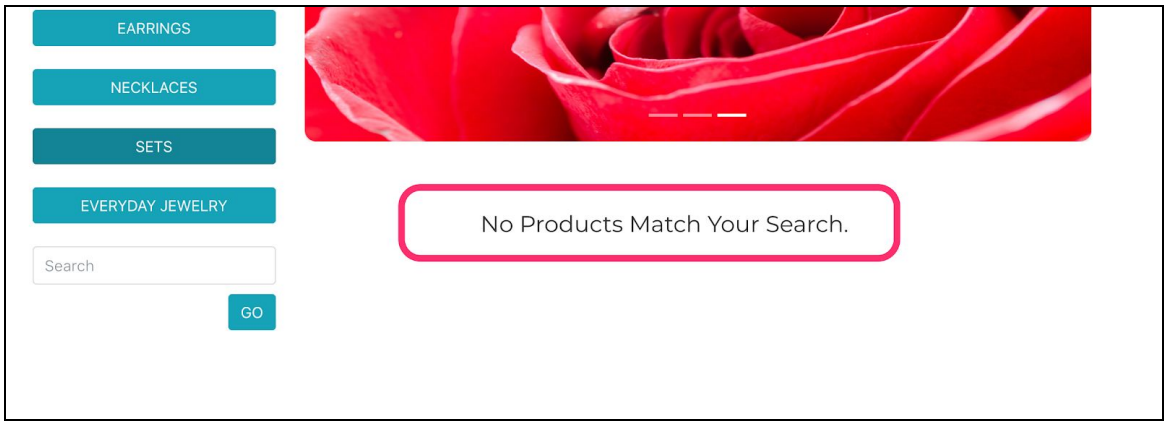


Non-Matching Search

Entry of Non-Matching Search



Result of Search

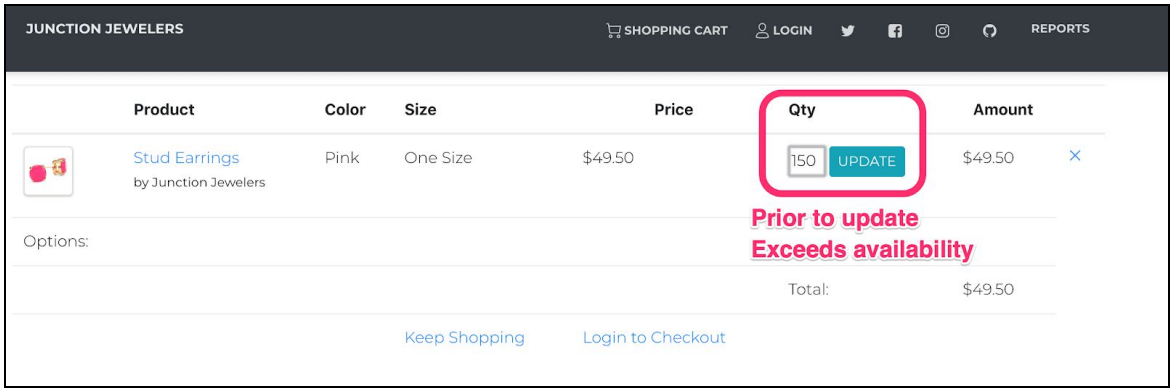


Order Quantity Exceeding Inventory

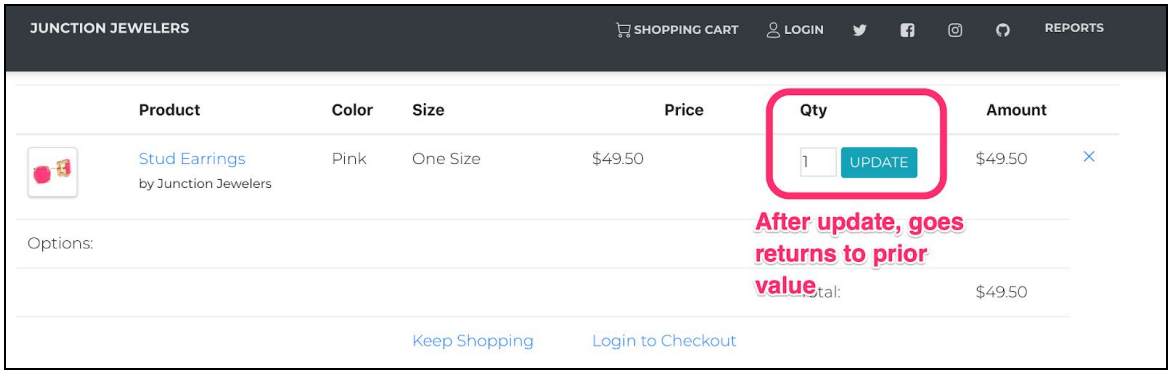
Requirement Addressed

3.8 - Prevent purchasing more of an item than there are items in stock

Prior to Update of Cart Quantity



On Update, if Quantity Exceeds Amount in Stock, Returns to Previous Value‘

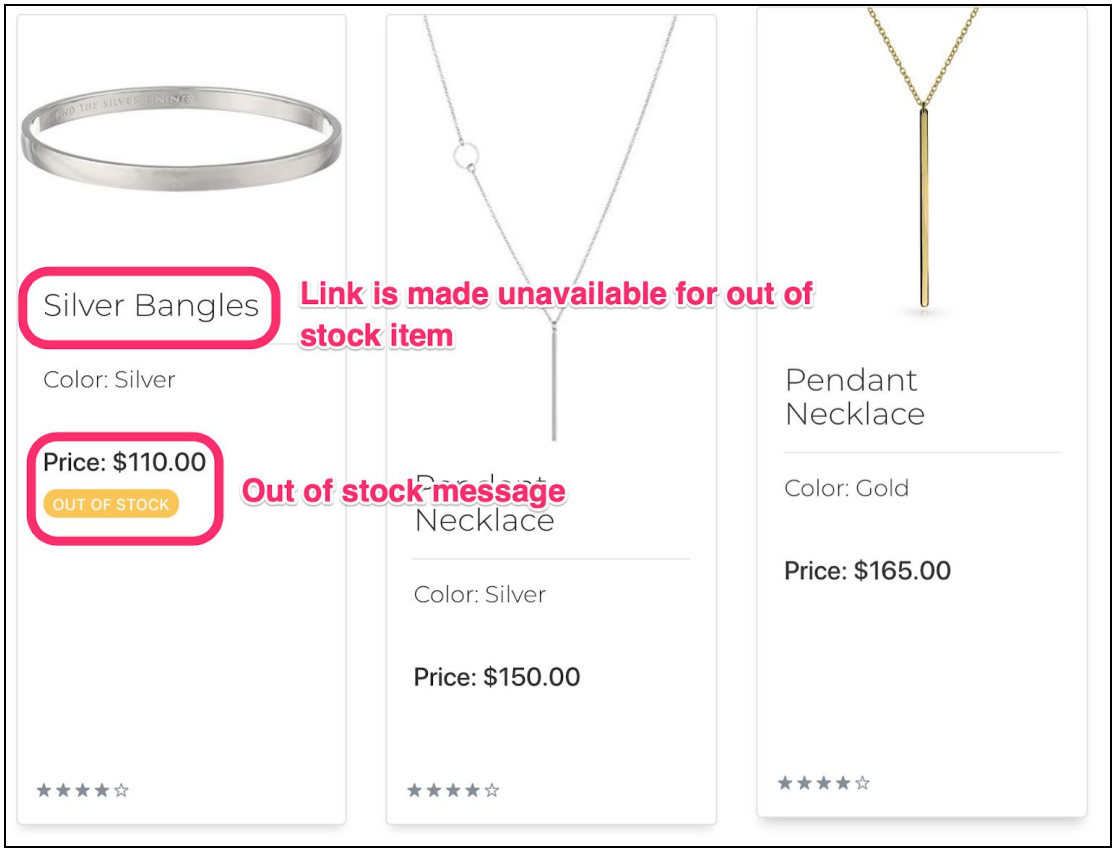


Item Out of Stock

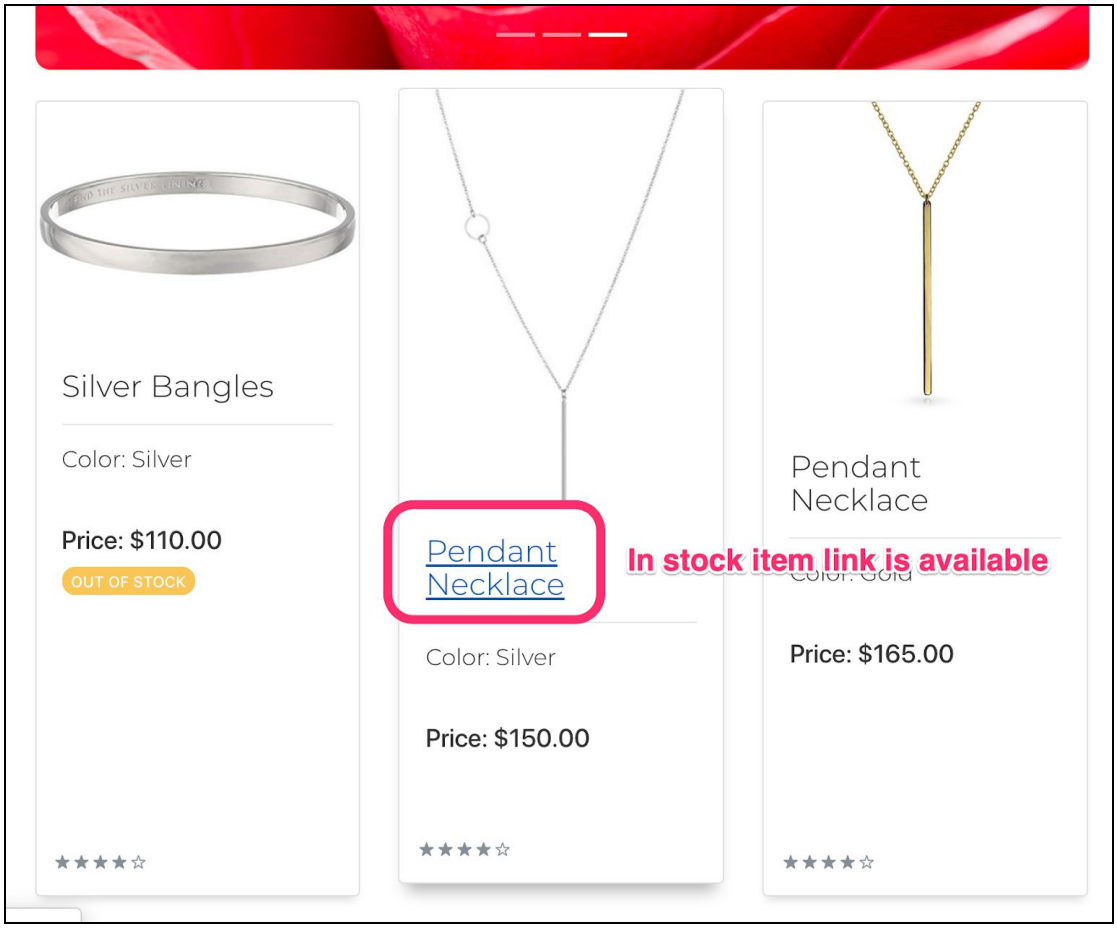
Requirements Addressed

3.7 - Prevent purchase of an out of stock item

Product Page View for Out-of-Stock Item



Contrast with Link Availability for In-Stock Product



Reporting Tests Validation

The database was loaded with a set of sample data to validate the reporting tests features. That test data is available at:
https://github.com/KendraMBach/CIS556_Project/tree/master/database/import_data

Report Initial State and Expected Values Tests

These tests demonstrate both the successful completion of orders in the database and the report functionality. Using the initial state as the base case, the team calculated what values would be for target values after the completion of the three test personas. The products that the test personas added to their cart and purchased are listed below.

Products Added to Cart or Purchased by Test Personas

	Number Added to Cart	Number Purchased
Black Stud Earring Quantity	4	4
Grand Mothers Bracelets - 8.5 Inch Gold	1	0
Mothers Bracelets - 7 Inch Silver	1	0
Baby Bracelets - Medium Rose Gold	1	0
Rose Stud Earrings	1	0
Gold Mens Wedding Band Size 10	1	0
White Gold Mens Wedding Band Size 10	1	1

Yearly Report Testing

	Initial State Value	Expected Value After Persona Orders	Actual Value After Persona Orders
2018 Sales	19,021.75	19,492.65	19,492.65
2018 Profit	15,064.35	15,353.25	15,353.25
2018 Rings Sold	11	12	12
2018 Bracelets Sold	10	10	10
2018 Earrings Sold	4	8	8

Monthly Report Testing

	Initial State Value	Expected Value After Persona Orders	Actual Value After Persona Orders
November Sales	1,489.30	1,489.30	1,489.30
November Profit	1,188.30	1,188.30	1,188.30
November Rings Sold	1	1	1
November Earrings Sold	1	1	1
November Bracelets Sold	4	4	4
December Sales	N/A	470.90	470.90
December Profit	N/A	288.90	288.90
December Rings Sold	N/A	1	1
December Earrings Sold	N/A	4	4
December Bracelets Sold	N/A	0	0

Inventory List

	Initial State Value	Expected Value After Persona Orders	Actual Value After Persona Orders
Black Stud Earring Quantity	100	96	96
Grand Mothers Bracelets - 8.5 Inch Gold	100	100	100
Mothers Bracelets - 7 Inch Silver	100	100	100
Baby Bracelets - Medium Rose Gold	100	100	100
Rose Stud Earrings	3	3	3
Gold Mens Wedding Band Size 10	100	100	100
White Gold Mens Wedding Band Size 10	100	99	99

Customer List

	Initial State Value	Expected Value After Persona Orders	Actual Value After Persona Orders
Count of Customers	25	27	27

Report Screenshots (Taken After Completing Test Paths)

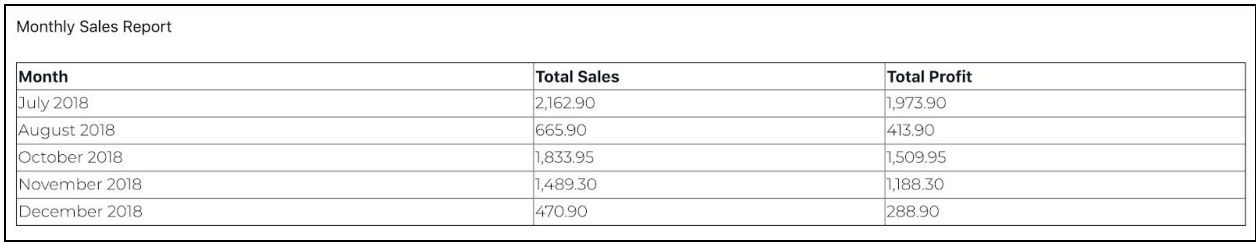
Monthly Sales Report

Requirements Addressed

5.1 - Generate a Monthly/Yearly Sales Report which includes
5.1.1 - The ability to specify the start/end date of the reporting period
5.1.2 - The ability to prevent invalid reporting period dates
5.1.3 - The total dollar sales for the specified period
5.1.4 - The total dollar profit for the specified period
5.1.5 - The total number of items sold by category

Monthly Sales Query

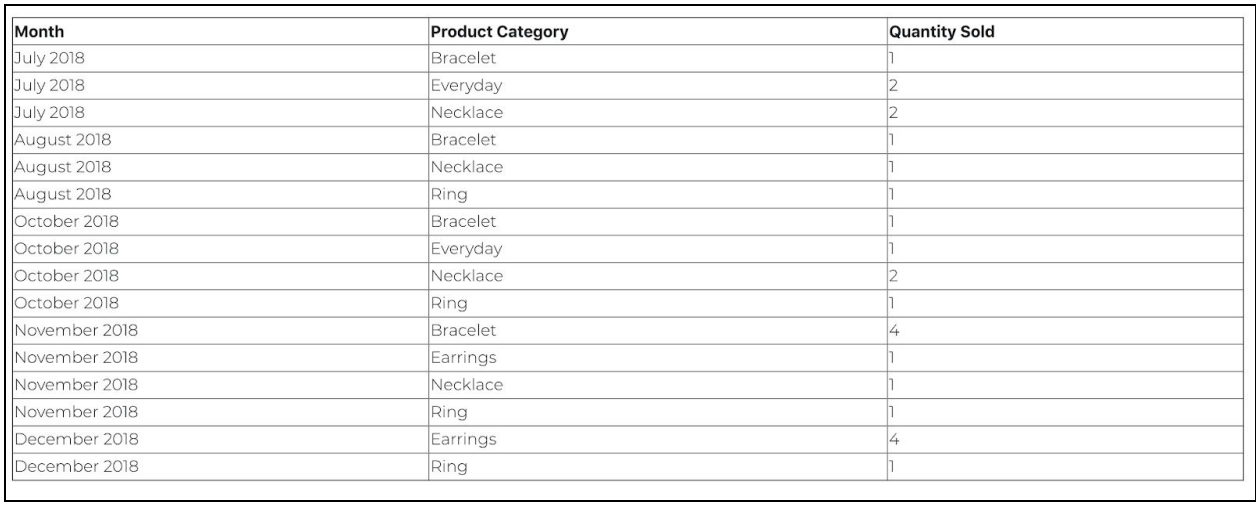
Monthly Sales Image 1



Monthly Sales Report

Month	Total Sales	Total Profit
July 2018	2,162.90	1,973.90
August 2018	665.90	413.90
October 2018	1,833.95	1,509.95
November 2018	1,489.30	1,188.30
December 2018	470.90	288.90

Monthly Sales Image 2



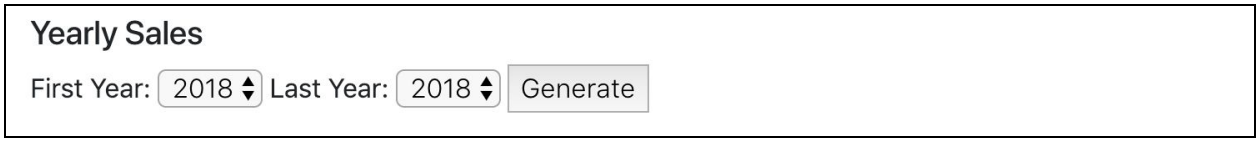
Month	Product Category	Quantity Sold
July 2018	Bracelet	1
July 2018	Everyday	2
July 2018	Necklace	2
August 2018	Bracelet	1
August 2018	Necklace	1
August 2018	Ring	1
October 2018	Bracelet	1
October 2018	Everyday	1
October 2018	Necklace	2
October 2018	Ring	1
November 2018	Bracelet	4
November 2018	Earrings	1
November 2018	Necklace	1
November 2018	Ring	1
December 2018	Earrings	4
December 2018	Ring	1

Yearly Sales Report

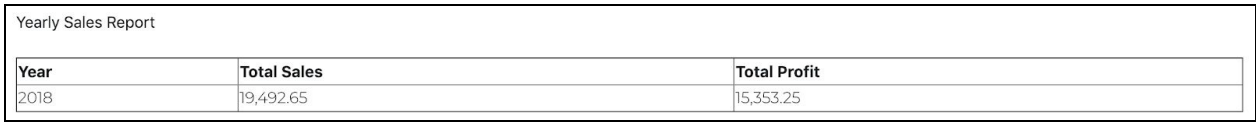
Requirements Addressed

5.1 - Generate a Monthly/Yearly Sales Report which includes
5.1.1 - The ability to specify the start/end date of the reporting period
5.1.2 - The ability to prevent invalid reporting period dates
5.1.3 - The total dollar sales for the specified period
5.1.4 - The total dollar profit for the specified period
5.1.5 - The total number of items sold by category

Yearly Sales Query



Yearly Sales Report - Image 1



Yearly Sales Report

Year	Total Sales	Total Profit
2018	19,492.65	15,353.25

Yearly Sales Report - Image 2

Year	Product Category	Quantity Sold
2018	Bracelet	10
2018	Earrings	8
2018	Everyday	3
2018	Necklace	11
2018	Ring	12
2018	Set	17

Inventory Report

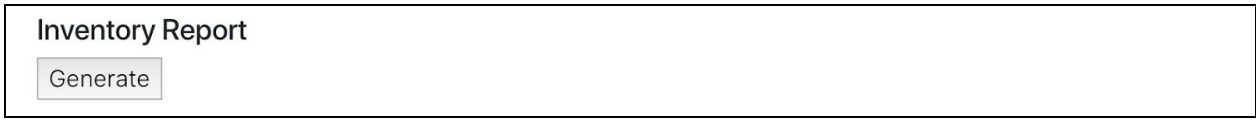
Requirements Addressed

5.2 - Generate a report of Inventory Levels and Costs which includes

5.2.1 - The names and amounts of items currently in inventory

5.2.2 - The total cost of items currently in inventory

Inventory Query



Inventory Report - Image 1

Inventory Report

Product ID	Product Name	Color	Size	Wholesale Price	Stock	Inventory Cost
20001	Mother's Bracelet	Gold	7 inch	30.00	100	3,000.00
20002	Mother's Bracelet	Silver	7 inch	30.00	100	3,000.00
20003	Mother's Bracelet	Gold	8.5 inch	30.00	100	3,000.00
20004	Mother's Bracelet	Silver	8.5 inch	30.00	100	3,000.00
20005	Grandmother's Bracelet	Gold	7 inch	25.00	100	2,500.00
20006	Grandmother's Bracelet	Silver	7 inch	25.00	100	2,500.00
20007	Grandmother's Bracelet	Gold	8.5 inch	25.00	100	2,500.00
20008	Grandmother's Bracelet	Silver	8.5 inch	25.00	100	2,500.00
20009	Baby Bracelet	Gold	Small	22.00	100	2,200.00
20010	Baby Bracelet	Rose Gold	Small	22.00	100	2,200.00
20011	Baby Bracelet	Silver	Small	22.00	100	2,200.00
20012	Baby Bracelet	Gold	Medium	22.00	100	2,200.00
20013	Baby Bracelet	Rose Gold	Medium	22.00	100	2,200.00
20014	Baby Bracelet	Silver	Medium	22.00	100	2,200.00
20015	Wedding Band	Gold	7	110.00	100	11,000.00
20016	Wedding Band	White Gold	7	110.00	100	11,000.00
20017	Wedding Band	Silver	7	110.00	100	11,000.00
20018	Wedding Band	Gold	8	110.00	100	11,000.00
20019	Wedding Band	White Gold	8	110.00	100	11,000.00
20020	Wedding Band	Silver	8	110.00	100	11,000.00
20021	Wedding Band	Gold	9	110.00	100	11,000.00

Inventory Report - Image 2 (some records skipped between previous image and this one)

20055	Pendant Necklace	Silver	16 inches	60.00	100	6,000.00
20056	Pendant Necklace	Silver	18 inches	60.00	100	6,000.00
20057	Pendant Necklace	Silver	20 inches	60.00	100	6,000.00
20058	Pendant Necklace	Silver	22 inches	60.00	100	6,000.00
20059	Pearl Necklace	Pearl	16 inches	85.00	100	8,500.00
20060	Pearl Necklace	Pearl	18 inches	85.00	100	8,500.00
20061	Pearl Necklace	Pearl	20 inches	85.00	100	8,500.00
20062	Pearl Necklace	Pearl	22 inches	85.00	100	8,500.00
20063	Crystal Necklace	Crystal	16 inches	100.00	100	10,000.00
20064	Crystal Necklace	Crystal	18 inches	100.00	100	10,000.00
20065	Crystal Necklace	Crystal	20 inches	100.00	100	10,000.00
20066	Crystal Necklace	Crystal	22 inches	100.00	100	10,000.00
20067	Owl Pendant Necklace	Owl	16 inches	22.00	100	2,200.00
20068	Owl Pendant Necklace	Owl	18 inches	22.00	100	2,200.00
20069	Owl Pendant Necklace	Owl	20 inches	22.00	100	2,200.00
20070	Owl Pendant Necklace	Owl	22 inches	22.00	100	2,200.00
20071	Rose Stud Earings	Rose	One Size	10.00	3	30.00
20072	Cat Ring	Silver	5	11.00	100	1,100.00
20073	Cat Ring	Silver	6	11.00	100	1,100.00
20074	Cat Ring	Silver	7	11.00	100	1,100.00
20075	Cat Ring	Silver	8	11.00	100	1,100.00
20076	Silver Bangles	Silver	One Size	44.00	0	0.00
					Total:	**549,748.00**

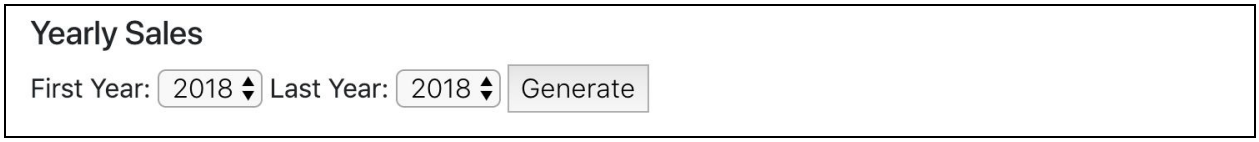
Customer List Report

Requirements Addressed

5.3 - Generate a report of Customers which includes

5.3.1 - The names, addresses, email, and phone number of all customers

Customer List Query Button



Customer List Image

Customer List

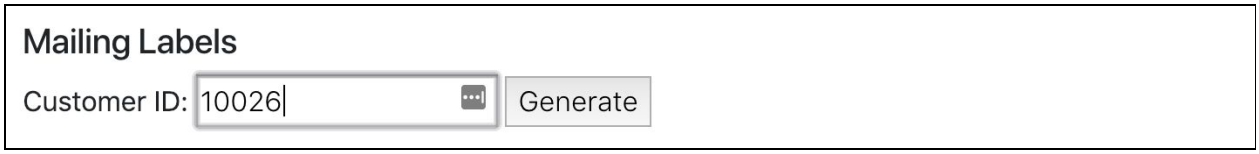
Customer ID	First Name	Last Name	Address	City	State	Zip	Phone Number	Email
10001	Goraud	Sharpe	4710 Corscot Place	Littleton	CO	80126	(303) 8018142	gsharpe0@japanpost.jp
10002	Ronny	Lawey	99363 Kenwood Parkway	Wilmington	DE	19886	(302) 2977840	rlawey1@weather.com
10003	Drucy	McGillecole	5114 Paget Lane	Milwaukee	WI	53263	(414) 2892128	dmcgillecole2@hc360.com
10004	Jewell	Giff	984 Maple Terrace	Birmingham	AL	35290	(205) 9020053	jgiff3@mlb.com
10005	Hallie	Basili	5438 Fisk Junction	Silver Spring	MD	20918	(240) 4076537	hbasili4@walmart.com
10006	Cordie	Scotfurth	3 La Follette Junction	Pasadena	TX	77505	(713) 4356930	cscotfurth5@usnews.com
10007	Jahzeel	Atwood	4212 Blair Court	Hardin	MO	64035	(714) 7701799	jesse@live.com
10008	Donatienne	Dermott	1259 Upton Avenue	Westbrook	ME	40923	(628) 5815827	singh@verizon.net
10009	Servaas	Acquati	2024 Hanifan Lane	Dunwoody	GA	30338	(598) 6580459	euice@live.com
10010	Sylvi	Simon	155 Kelly Street	Gastonia	NC	28052	(931) 2823729	afifi@icloud.com
10011	Zsiga	Daube	4648 Waterview Lane	Pena Blanca	NM	87041	(377) 6790729	jsnover@live.com
10012	Adonis	Ramsay	1049 Fincham Road	San Diego	CA	82103	(248) 5254405	fraser@optonline.net
10013	Porfastr	Sierkert	2643 Tetrick Road	Fort Myers	FL	33901	(527) 9001858	caronni@att.net
10014	Alexei	McNee	1711 Oak Lane	Hume	MO	64752	(273) 7768676	jesse@outlook.com
10015	Castor	Ola	486 Tori Lane	Salt Lake City	UT	84116	(340) 6788712	leocharre@me.com
10016	Jaksa	Chowdhury	4302 Amethyst Drive	Adrian	MI	49221	(630) 9180833	alhajj@yahoo.com
10017	Abel	Keller	4137 Brookside Drive	Birmingham	AL	35209	(327) 3594970	boein@hotmail.com
10018	Photina	Brankovic	1446 Longview Avenue	New York	NY	10004	(218) 962-0306	avalon@yahoo.ca
10019	Cezar	Osmund	626 South Border Ave.	Riverdale	GA	30274	(777) 306-0607	keijser@aol.com
10020	Lea	Hadi	77 Alton St.	Lincolnton	NC	28092	(282) 488-2823	chlim@aol.com
10021	Soraya	Torbjorn	68 Amerige St.	New Bern	NC	28560	(862) 890-1405	fwiles@msn.com
10022	Mpho	Erna	199 Sierra Rd.	Newington	CT	61141	(674) 611-5460	conteb@msn.com
10023	Marilenna	Blaanid	19 Saxon Street	Morristown	NJ	79603	(656) 501-9126	khris@yahoo.com
10024	Su	Berhtoald	490 Country Lane	Peoria	IL	61604	(304) 498-8284	mleary@me.com

Mailing Label Report

Requirements Addressed

5.4 - Generate mailing labels for a customer

Mailing Label Query



Mailing Label Images

Junction Jewelers Data Dictionary

Authors:
Kendra Bach, Mark Ferrall, Cooper Stansbury
Nolan Gage, Nicholas Crooker

Data Dictionary

The data dictionary is a one-stop shop for technical details regarding the website and supporting applications. This document contains initial project specification and granular details regarding technical implementation. The data dictionary is written at various levels of technicality, starting at a high-level and moving towards specific examples of code. This document is intended for various users. It is, however, decidedly technically focused. We encourage readers with questions about the document to reach out they have questions. Contact information is available in the document introduction.

Important Note

Our first priority, driven by our mission statement, is to provide an excellent customer experience. This priority influenced critical decisions throughout the development process, and where trade-offs were made, priority was given to solutions that would deliver a quality product to the client in the given time frame. A critical example of this trade-off is the normalization state of the database. The database structure conforms to first normal form *in order to support the business logic* and provide a simple interface with the web application. This makes subtle, but real differences in the customer experience. A highly normalized database would have resulted in messy communication protocol between the Java application and the actual database instance that may have resulted in bugs, or lackluster functionality. Our team includes highly skilled Java developers, and it was in the customer's best interest to build the database as a support structure, rather than the logic driver for our customers.

There are other examples of these trade-offs described in this document. We feel that a functioning solution with well-documented areas for opportunities is stronger than a solution achieves everything, but is undocumented and untraceable.

For developer documentation of important decisions, please see the GitHub Issue Tracker:

- https://github.com/KendraMBach/CIS556_Project/issues

Data Dictionary Contents

Affiliation

University of Michigan Dearborn
College of Engineering and Computer Science
Department of Computer and Information Science
4901 Evergreen Rd, Dearborn, MI 48128

Development Team Skills Overview

The skills of each team member contributed to the design choices that were made throughout the project. Development time was reduced by by choosing technologies that team members were already comfortable with. This required integrating several technology stacks into the final project, dependent on each team member's area of responsibility.

- Kendra Bach: Background in software engineering and web application development. Built Java Spring MVC web application to integrate with team members' database.
- Nick Crooker: Background in software engineering and embedded systems. Handled emailing receipts and the newsletter.
- Mark Ferrall (Team Lead): Experienced with Microsoft Access, limited SQL, and documentation. Project management experience and comfort managing teams informed his role as Team Lead.
- Cooper Stansbury: Background as ETL developer. Built database and loaded initial state.
- Nolan Gage: Background in software development including developing SQL queries and front-end code for corporate finance reporting. Wrote SQL queries and report pages to run our reports and render results.

Development Team Contact Information

- Mark Ferrall: mdferral@umich.edu
- Cooper Stansbury: cstansbu@umich.edu
- Kendra Bach: kmbach@umich.edu
- Nicholas Crooker: ncrooker@umich.edu
- Nolan Gage: nolang@umich.edu

Business Mission Statement

Customer experience is our North Star and our first priority. We engage our customers by creating a simple, efficient, and no-nonsense shopping experience. This principal guides our choices from inventory, to web-design, to our hiring processes.

Implementation Statement

Our small business requires a website to facilitate e-commerce retail. Our website must handle both the front end customer interface and backend inventory processes related to transactions, including reporting. The front end will handle customer accounts, sales orders and shipping cost calculation. The backend must handle all relevant inventory and reporting functionality. Core customer types must be used to validate the design.

Additional Functionality

The assignment requires the project team to identify two additional features beyond the project requirements. The project team chose **emailed receipts** to the user and **product pagination** of the products on the product display webpage.

Email receipts

Our product supports direct customer communication via newsletters and emails of completed transactions. The SMTP scripts are written in Python and managed by the Java application.

Pagination

Pagination, in our context, refers to separating products into a number of discrete pages in order to make a cleaner ‘look and feel’ for customers when shopping. This was handled by the business layer of the application, but depends on the results of HQL queries executed against the MySQL instance.

Source Code Repository

The application source code, supporting applications, database structure, and initial database load can be found at the following public GitHub repository:

- https://github.com/KendraMBach/CIS556_Project

Notes on Development Strategy

- We use Git as our version control system for this implementation. The latest release and developer documentation can be found here: https://github.com/KendraMBach/CIS556_Project
- To track in-development issues and feature requests we use the GitHub Issue tracker, which can be found here: https://github.com/KendraMBach/CIS556_Project/issues

Fundamental Implementation Assumptions

Below is a list of fundamental assumptions made by the project team while designing and implementing the Junction Jewelers website solution.

- All non-ecommerce related processes are functioning, and covered elsewhere.
- There is a window of time after testing to migrate local processes to production servers.
- Production servers have all required software.
- Only one customer shops at any given time.
- Junction Jewelers conducts business only in the United States of America, including Hawaii and Alaska. In the future we plan to expand business globally.
- Junction Jewers accepts only the US Dollar (not bitcoin). As such, not transaction details are necessary through the website. (We do not store credit cards).
- All transactions are tendered with the same payment type.
- Timestamps for orders are always specified in EST, regardless of where order is placed.
- Charms and birthstones have no limit to their supply.
- Charms and birthstones have have no cost to Junction Jewelers, there inventory is not important to track.
- Junction Jewelers has domain that can be used instead of localhost.

Business Specifications

This section details the project requirements established to ensure the implementation performs correctly. Project requirements are broken into (1) functional requirements and (2) non-functional requirements.

Functional Requirements

The Functional Requirements were developed upon review of the needs provided by Junction Jewelers and review of the products carried by the store. This review revealed several logical groupings of requirements: Website, Product Pages, Shopping Cart, Database, Reports, and User Account. The project team must document addressing all functional requirements and their acceptance criteria for the project to be complete.

Requirement	Acceptance Criteria
Website	1. The website must include: 1.1. An initial homepage, including names and images of the team members 1.2. The ability to create a new customer account 1.3. The ability to log in to an existing customer account 1.4. A newsletter signup, which will send the user an immediate confirmation/welcome email 1.5. A page listing all products carried by the store 1.5.1. Views of products must be paginated, with no more than 6 products per page 1.6. A list of the most popular products carried by the store 1.7. A product page for all products (see product page requirements) 1.8. The ability to place an order 1.9. Ability to place multiple orders 1.10. Ability to select customizations for some products 1.11. Ability to order more than 1 of an item 1.12. Ability to search for items by item name 1.13. Ability to view products by their category 1.14. An interface for generating reports 1.15. The ability to add multiple items to the cart 1.16. The ability to log out of an account
Products Pages	2. The Products Pages Must 2.1. Provide the ability to select distinct sizes, color, or material for products with those options 2.2. Provide the ability to add a product to the cart 2.3. Display the description and image of each product

	2.4. Provide the ability to specify engravings, birthstones, or charms for products with those options
Shopping Cart	3. The Shopping Cart must 3.1. Provide the ability to add/remove items from the cart 3.2. Provide the ability to change the quantity of an item in the cart 3.3. Provide the ability to checkout, adding the cost of shipping to the final total 3.4. Prevent checkout with an empty cart, instead redirecting the customer to the products page 3.5. Prevent checkout without a valid user account 3.6. Send users an emailed receipt upon checkout 3.7. Prevent purchase of an out of stock item 3.8. Prevent purchasing more of an item than there are items in stock
Database	4. The database must include 4.1. Table to support all products carried by store 4.2. Table to support the cart 4.3. Table to support all customers 4.4. Table to support orders 4.5. Table to support birthstone options 4.6. Table to support charm options 4.7. Table to support shipping locations and costs 4.8. Support for customizations of some products (engravings, birthstones, and charms) 4.9. Support reporting of sales, inventory, and customer information (see report requirements) 4.10. Constraints to prevent 4.10.1. Improperly specified new product listings 4.10.2. Improperly specified orders 4.10.3. Improperly specified user accounts
Reports	5. The Reports must: 5.1. Generate a Monthly/Yearly Sales Report which includes 5.1.1. The ability to specify the start/end date of the reporting period 5.1.2. The ability to prevent invalid reporting period dates 5.1.3. The total dollar sales for the specified period 5.1.4. The total dollar profit for the specified period 5.1.5. The total number of items sold by category 5.2. Generate a report of Inventory Levels and Costs which includes 5.2.1. The names and amounts of items currently in inventory 5.2.2. The total cost of items currently in inventory

	5.3. Generate a report of Customers which includes 5.3.1. The names, addresses, email, and phone number of all customers 5.4. Generate mailing labels for a customer
User Account	6. The User Account Must 6.1. Allow customers to create password protected accounts to order from the store 6.2. Allow customers to store their name and shipping information 6.3. Constrain improperly specified user account information

Non-Functional Requirements

Review of the business needs of Junction Jewelers revealed several types of non-functional requirements: Aesthetic, User Stories, and Content. Developing User Personas, sample target customers of Junction Jewelers, will help clarify development of features and testing strategies around customer needs.

Aesthetic	● Website should look and feel modern ● Reports should be polished (but simple)
Support User Stories:	● Persona 1: New Male customer using site to find a specific product ● Persona 2: New female customer browsing, no endpoint ● Persona 3: Returning Male customer using site to find a specific product
Content	● Must include information about company ○ Company name ○ Location ○ Contact information

Implementation Test Plan

The persona test paths test the application from the perspective of three target customers of the business. By focusing on these users, the project team better targeted the development and testing of the application with the real-world demands placed on it. Three personas are described below.

Persona 1

New male customer using site to find a specific product.

- He is getting married soon and needs to pick out a wedding band.
- He knows he needs to purchase a wedding ring, but isn't sure which one he wants, or what his ring size is.

Persona 2

New female customer expecting a baby in April, just browsing for items

- She's looking for a series of bracelets for her, her mother, and the baby
- She'd like a customized engraving of the baby's name
- She wants her own bracelet to feel unique
- She might want to look at some other items for herself while she's shopping

Persona 3

Returning male customer purchasing groomsman gifts

- He is purchasing some matching black stud earrings for his groomsmen
- He previously used the site to purchase his wedding band

Constraint Tests Specification

The constraint tests demonstrate the functionality of the limitations placed on the ordering process, and the protection of the database. These limits prevent ordering quantities of products greater than the store inventory. The test cases will show the user messages upon entering invalid information.

Constraints Tested

- Prevent Adding Items to Cart without Required Options (Size)
- Prevent Registration without Required Information
- Non-Matching Search
- Prevent Registration with Invalid Phone Number
- Prevent Registration with Invalid Email
- Order Quantity Exceeding Inventory
- Prevent Ordering Items Out of Stock
- Prevent Ordering Without an Account

Reporting Tests Description

The constraint tests demonstrate the functionality of the reporting features built into the application. Test cases will show the anticipated and actual outputs using test data, and the changes to those reports after completing the Persona Test Paths.

Technology Overview

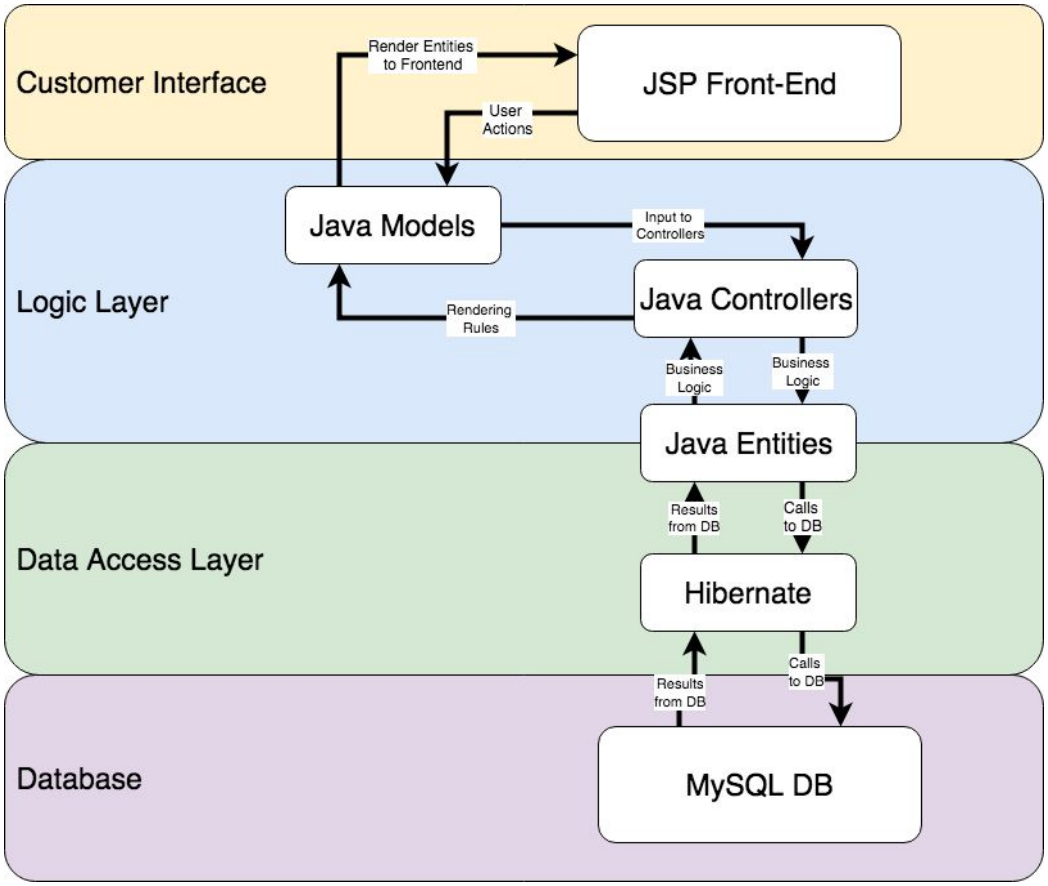
Each technology is described in detail in the section labeled ‘Technology Stack’. This section is intended for readers without any knowledge of the technologies described, or for readers looking for justification of a technology choice. Links to technology-specific documentation can be found under each section for more advanced readers.

For developer details, the source code repository found here:

- https://github.com/KendraMBach/CIS556_Project

The following diagram shows the technical infrastructure of Junction Jewelers website and database. There are four abstract layers of the technology stack: (1) The customer facing layer, (2) the logic layer, (3) the data access layer, and the (4) database. Each layer is summarized briefly below:

1. Customer Layer: Responsible for rendering core functionalities to customers.
2. Logic Layer: Responsible for translating between physical storage and customers.
3. DAO Layer: Responsible for accessing data in the database based on logic needs.
4. Database: Physical storage of data.



Technology Stack

This section provides summaries of each technology in each of the four abstract layers of the application. Links are provided directly to the developer documentation for each, where possible.

JavaServer Pages (JSP)

We use JSP to render our inventory to the users and to mitigate user requests with the system logic and constraints. JSPs are user-facing components of Java servlets. We chose to work with JSP because it integrates well with basic web (HTML) pages via tags and communicate with the Java servlets via HTTP requests and responses. As a result of this communication, they support dynamic content based on user input, and remain essential for rendering our database contents to site visitors. JSP enables a core functionality, pagination, used to control the number of products per page.

- JSP Documentation Link: https://www.oracle.com/technetwork/java/index-jsp-138231.html
- Java Pagination Documentation Link: https://www.baeldung.com/rest-api-pagination-in-spring

HTML/CSS

We use HTML and CSS to display our web pages. We chose to work with HTML and CSS because they are internet standards. Both the Hypertext Markup Language (HTML) and Cascading Style Sheets (CSS) are core technologies of web application development. HTML provides the foundation of each user page and CSS provides the layout and design. As an extension of both, we utilized Bootstrap 3, a powerful open source frontend framework, that gave us the ability to design our UI with a clean, modern appearance.

- CSS Documentation Link: https://developer.mozilla.org/en-US/docs/Web/CSS
- HTML Documentation Link: https://developer.mozilla.org/en-US/docs/Web/HTML
- Bootstrap Documentation Link: https://getbootstrap.com/docs/3.3/

Java

Java is the main driver of communication between layers of our application and is the language that the business logic is written in. We chose to work with Java because we have experienced Java developers on our team. Java is a robust object-oriented language with numerous libraries and frameworks built specifically with web application development in mind. We used Java servlets combined with the Spring Model-View-Controller (MVC) and Security frameworks, to allow for quick loading and rendering of our site's user interface pages, as well as the creation of login functionality for all users.

- Java Documentation Link: https://docs.oracle.com/en/java/
- Java 1.8 Documentation Link: https://docs.oracle.com/javase/8/docs/api/
- Java Spring MVC Documentation: https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html
- Java Security Frameworks: https://docs.oracle.com/javase/8/docs/technotes/guides/security/

Python

We use Python in order to manage email communications with our customers. Python is an interpreted object-oriented programming language. There are two core functions that make use of Python: (1) email receipts and (2) customer newsletters. We use Python version 3 to make calls to the MySQL database instances and to email customers from our operational gmail account via Simple Mail Transfer Protocol (SMTP). We chose to use Python because of its support for rapid development and its simplicity. Future iterations of this functionality may be implemented in Java.

- Python Documentation Link: https://docs.python.org/3/
- Python SMTP Documentation Link: https://docs.python.org/3/library/smtplib.html

MySQL

Our database is a MySQL instance. We chose to work with MySQL and MySQL workbench because of the ease of integration with Java web applications and because they support rapid database development. MySQL is an open source database driven by Oracle and optimized specifically for web application development. It is easily scalable to schemas of all sizes as its internal InnoDB engine maintains its exceptional performance standards regardless of the workload. By choosing to work with my MySQL, we were able to use MySQL Workbench as our RDBMS, which provided us with an easy to use graphical user interface tool for all of our database needs.

- MySQL Documentation Link: https://dev.mysql.com/doc/
- MySQL Workbench Documentation Link: https://dev.mysql.com/doc/workbench/en/

Hibernate

We used Hibernate in conjunction with its built-in query language (HQL) for definition of all relations and their attributes in the web application, as well as persistence of user transactions in the logic layer of the application. We chose Hibernate because it allows for robust communication and data access to the data layer from a Java web application. Hibernate is a Java framework built specifically for the mapping of Java classes (entities) to their corresponding relational database tables. When combined with the Java Persistence API (JPA),

annotation functionality becomes available; this enabled us to define cardinality for each entity to match what was defined in the database.

- Hibernate Documentation Link: https://hibernate.org/orm/documentation/5.3/
- HQL Documentation Link: https://docs.jboss.org/hibernate/orm/3.3/reference/en-US/html/queryhql.html

Apache Tomcat

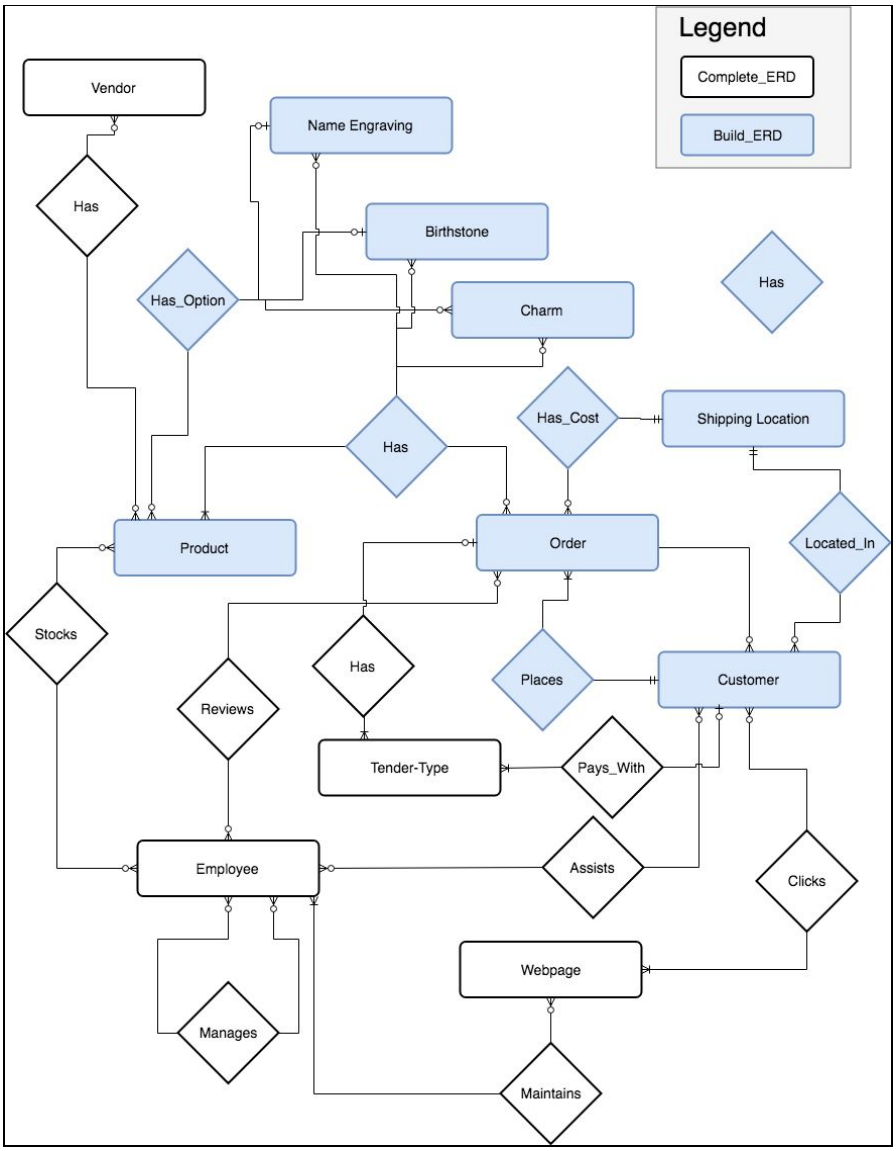
Apache Tomcat is a Servlet and JSP Container that allows for seamless deployment for Java Web Applications. We chose to use Tomcat to develop our web application locally with Eclipse (a Java IDE).

- Tomcat Documentation Link: http://tomcat.apache.org/tomcat-8.0-doc/
- Eclipse Documentation Link: https://www.eclipse.org/documentation/

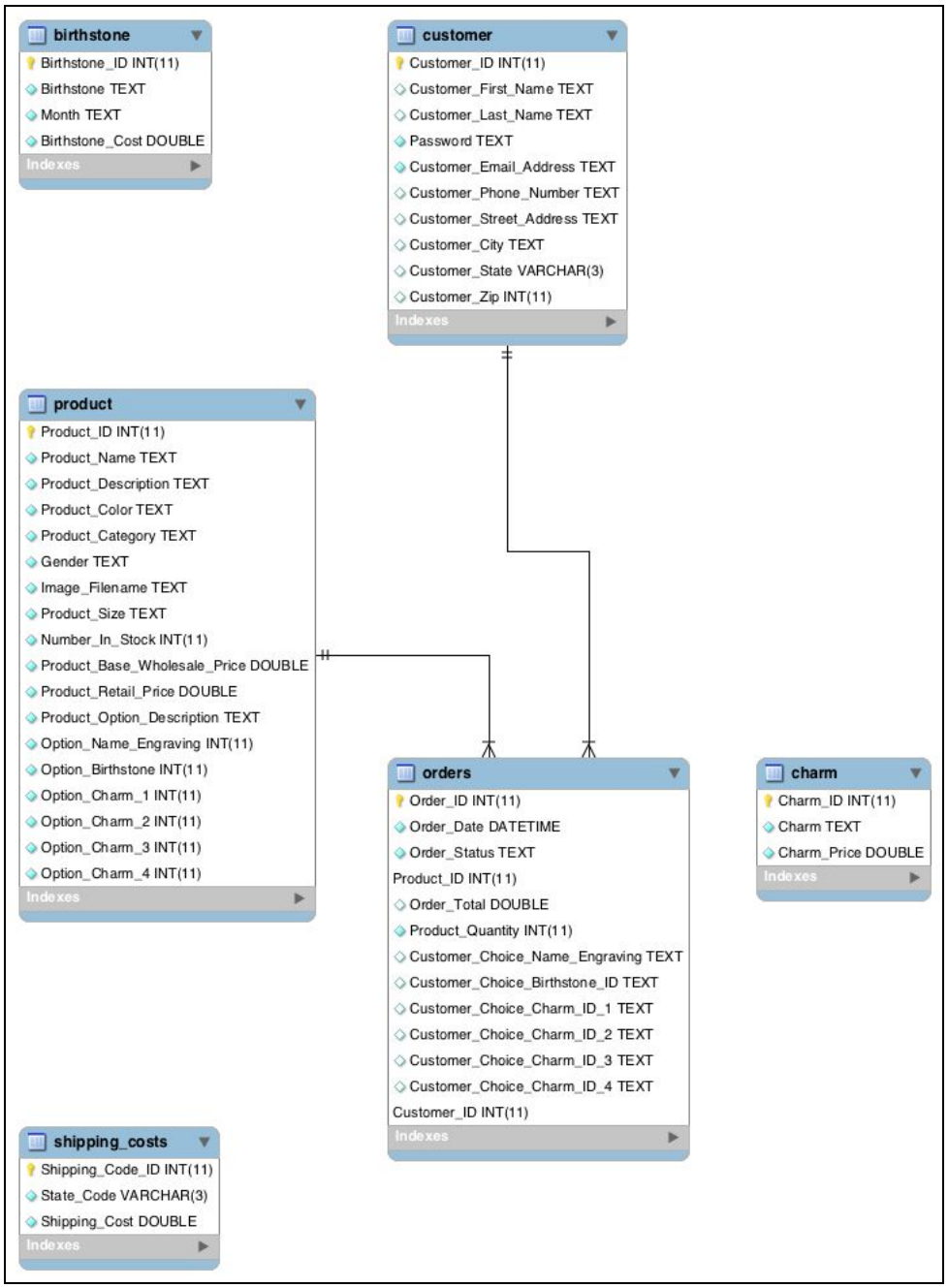
High-Level Diagrams

The entity relation diagram is the highest-level abstraction of our data. This section is intended for readers looking to understand Junction Jewelers data domain at various levels of granularity. Each diagram has a short description.

The first diagram shows how each entity relates to one another in our domain. The first diagram consists of two critical components: (1) the entities directly related to the implementation process and (2) the full scope of the organization. Each entity from (1) is fully specified in the corresponding section of the data dictionary and has a more detailed image describing attributes in this section.



This diagram shows is auto-generated ERD based on the actual database implementation. **There is an important note here:** Not all relationships are fully specified in the implementation. This is intentional. The web application uses Hibernate, an access control mechanism to make calls to the database in a query language called HQL. Further information can be found in the Technology Stack section. These relations are left out of the database, but covered in the business layer (in the Java application). This is done primarily to avoid (1) having two separate specifications of constraint logic, and (2) to make updating the database itself much easier.

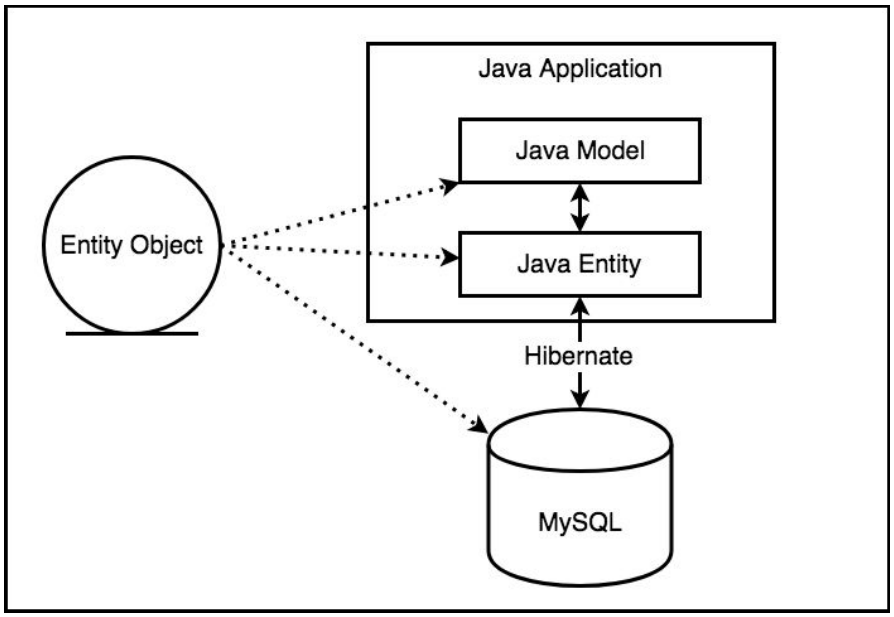


Abstract Entity Specification

This section details the data types, the attributes, and other important information about each entity in our domain. We start with an abstract overview of the entity types and there database implementation and move towards specific schematic definitions and code examples for interested readers. At this time we only specify the entities included in the BUILD ERD. As new business models are constructed, new sections will be added to this document. Each attribute is discussed.

Important Note Regarding Representation

Each entity is represented differently in different layers of the application. The core entity-type and it's database representation are discussed in this section. However, different technologies and needs impact the *physical representation* of each entity in different layer of the application. We have **separate sections** for the MySQL and Java representations. The entity, then, should be seen as a logical construct that persists through different layers of the application space, as shown below:



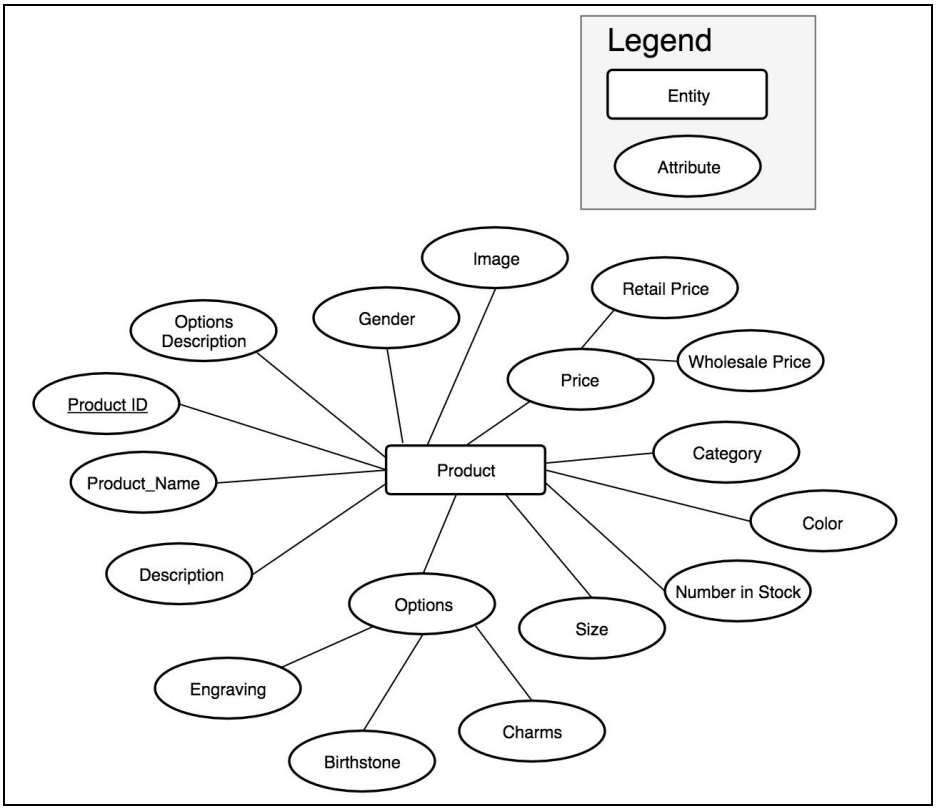
Product

The product entity represents all instances of our core products. As such, the product must have attributes that are necessary and sufficient to differentiate each product from one another. We do not consider charms and birthstones to be products.

Candidate Keys:

- Product_ID: Product_ID is the only uniquely identifying attribute for a product of a specific size and color that is suitable for our application. This key was created to manage uniqueness of products. By creating a designated key we maintain a single PK, which important for references to this table.
- (Product_Name, Product_Size, Product_Color): the combination of product name, it's size and it's color could have been used. This was avoided because product name may not be future proof.

This diagram represents an abstract product, and the attributes of the product. Product size and material/color are chosen as attributes in order to ensure that there are no multi-valued attributes. This has the implication that some other attributes are redundant, for example 'Options.' Each product with the same name will have the same options across all sizes and materials. This will be mitigated in a future version through further normalization.



Below is a list of the attributes that have been implemented in the MySQL instance along with brief descriptions.

TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_COMMENT
product	Product_ID	int	All product IDs start with '2'.
product	Product_Name	text	The English name of the product. Note: the name spans all sizes and material options for this product class.
product	Product_Description	text	The English description of the product used on the website.
product	Product_Color	text	Color or material.
product	Product_Category	text	The category of product for reporting purposes and menu rendering on the website.
product	Gender	text	The intended gender for the product, but in 2018 you never really know…
product	Image_Filename	text	Filename for the image. Used for rendering website.
product	Product_Size	text	The size, if any. 'One Size' should be used if there are no distinct size options for the product.
product	Number_In_Stock	int	Current inventory levels.
product	Product_Base_Wholesale_Price	double	Based wholesale price.
product	Product_Retail_Price	double	Product selling price (displayed on the website and used when calculating order total).
product	Product_Option_Description	text	Description of options.
product	Option_Name_Engraving	int	Binary signal indicating this option is available for this product. '1' indicates that this option IS available.
product	Option_Birthstone	int	Binary signal indicating this option is available for this product. '1' indicates that this option IS available.
product	Option_Charm_1	int	Binary signal indicating this option is available for this product. '1' indicates that this option IS available.
product	Option_Charm_2	int	Binary signal indicating this option is available for this product. '1' indicates that this option IS available.

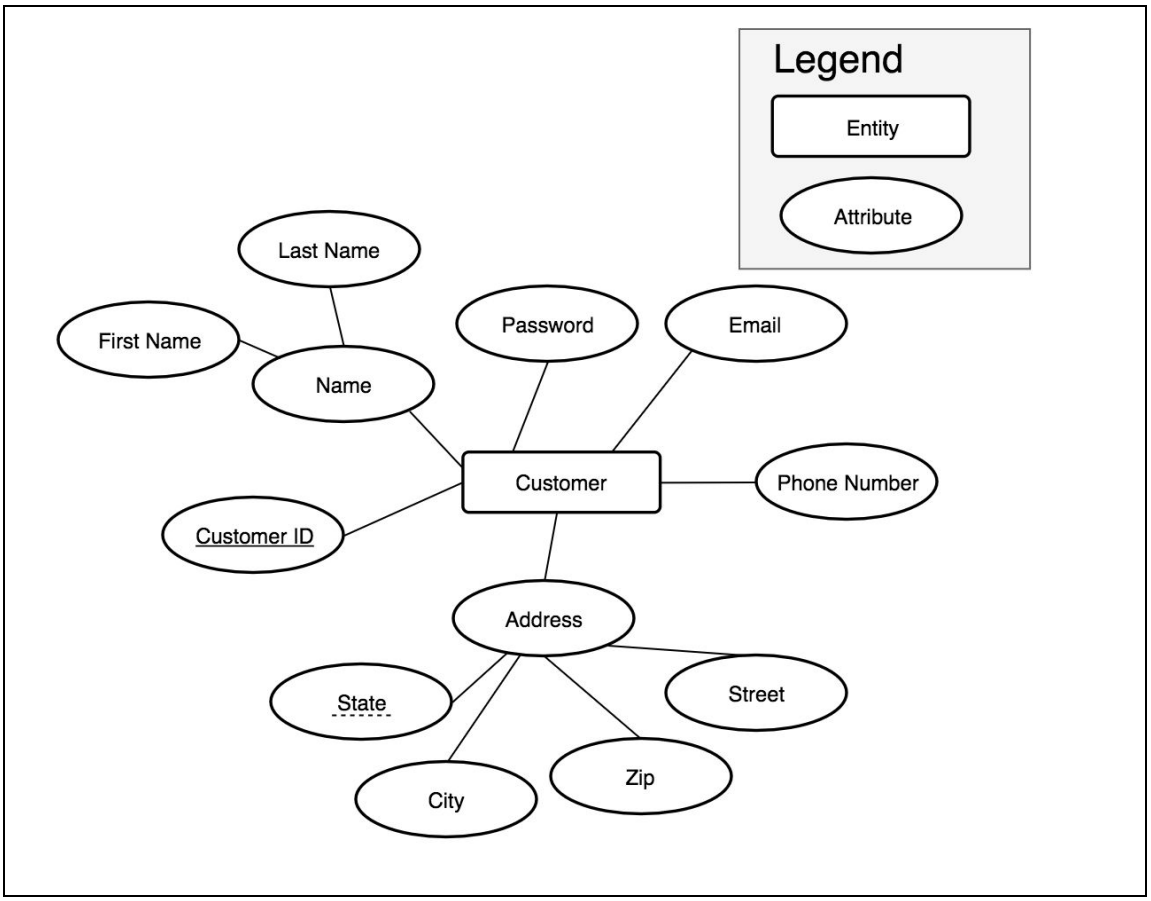
product	Option_Charm_3	int	Binary signal indicating this option is available for this product. ‘1’ indicates that this option IS available.
product	Option_Charm_4	int	Binary signal indicating this option is available for this product. ‘1’ indicates that this option IS available.

Customer

The customer entity represents all instances of our customers. It is important to note that ‘State’ contains a reference to the shipping cost entity.

Candidate Keys:

- Customer_ID: a uniquely identifying number created to manage customer uniqueness. This has been chosen to avoid common pitfalls of other customer keys.
- Other keys include various combinations of customer attributes. By creating a designated key we maintain a single PK, which important for references to this table.



Below is a list of the attributes that have been implemented in the MySQL instance along with brief descriptions.

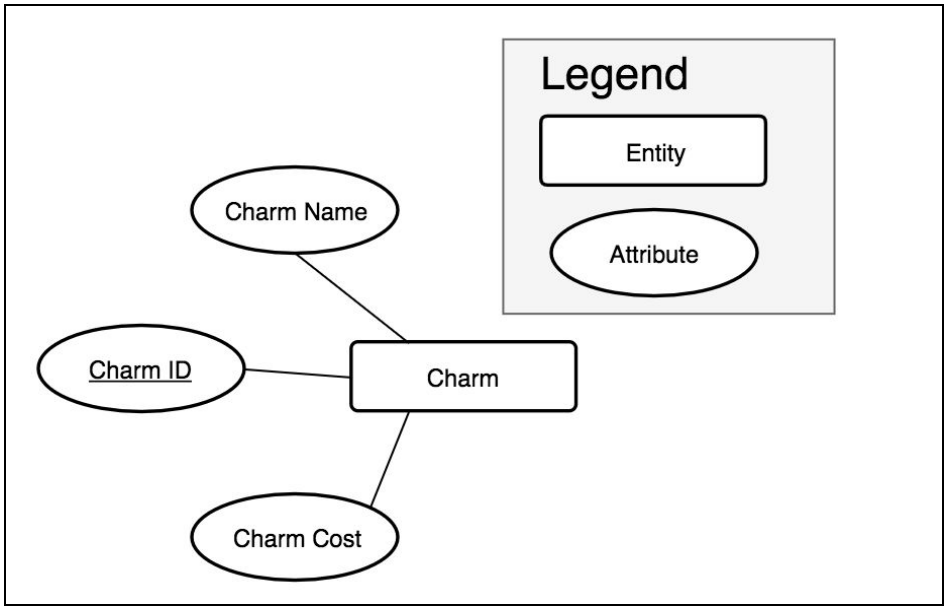
TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_COMMENT
customer	Customer_ID	int	Customer ID always starts with ‘1’.
customer	Customer_First_Name	text	Given name for the customer. This is driven by customer input.
customer	Customer_Last_Name	text	Surname name for the customer. This is driven by customer input.
customer	Password	text	Customer password. Plans to encrypt this in the future.
customer	Customer_Email_Address	text	Email address. Driven by user input.
customer	Customer_Phone_Number	text	Customer phone number.
customer	Customer_Street_Address	text	Shipping address.
customer	Customer_City	text	Customer City (US only for now).
customer	Customer_State	varchar	Customer State. This must be a two character stated code (upper-case) to match for shipping costs.
customer	Customer_Zip	int	Customer zip code.

Charm

Charms are selectable options that customer can add to some products in order to make them more unique. Charms have varied costs to the customers. For our current needs, charms are fairly simple.

Candidate Keys:

- Charm_ID: By creating a designated key we maintain a single PK, which important for references to this table.
- Charm: Charm name may be suitable, but is not future proof.



Below is a list of the attributes that have been implemented in the MySQL instance along with brief descriptions.

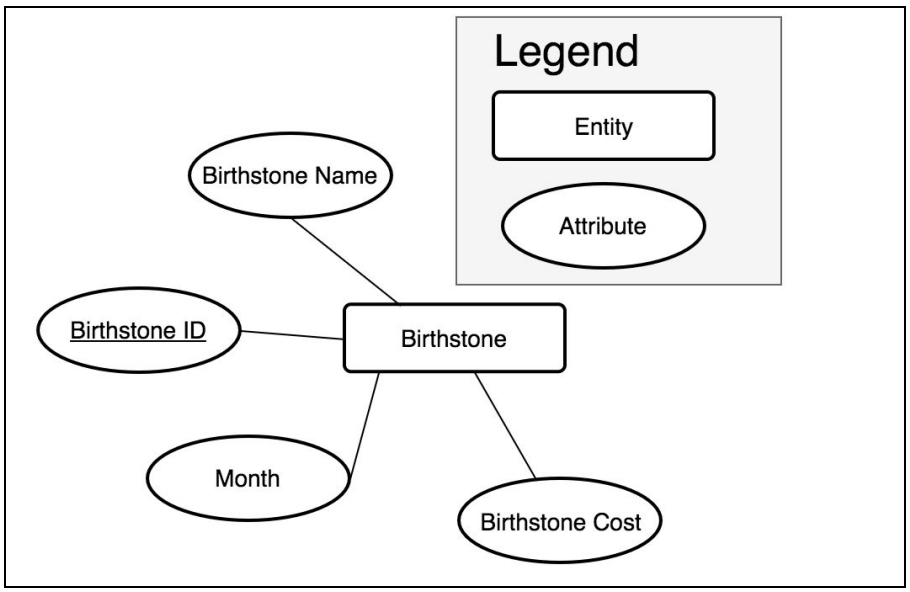
TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_COMMENT
charm	Charm_ID	int	First record is the default value used by the java model when rendering drop down options.
charm	Charm	text	The English name of the charm that is rendered in the drop down options.
charm	Charm_Price	double	The price of the add-on charm in USD.

Birthstone

Birthstones are selectable options that customer can add to some products in order to make them more unique. Birthstones have varied costs to the customers. For our current needs, Birthstones are fairly simple.

Candidate Keys:

- Birthstone_ID: By creating a designated key we maintain a single PK, which important for references to this table.
- Birthstone: Birthstone name may be suitable, but is not future proof.
- Month: At this time, Month is suitable. But we may wish to offer multiple choices for each month in the future.



Below is a list of the attributes that have been implemented in the MySQL instance along with brief descriptions.

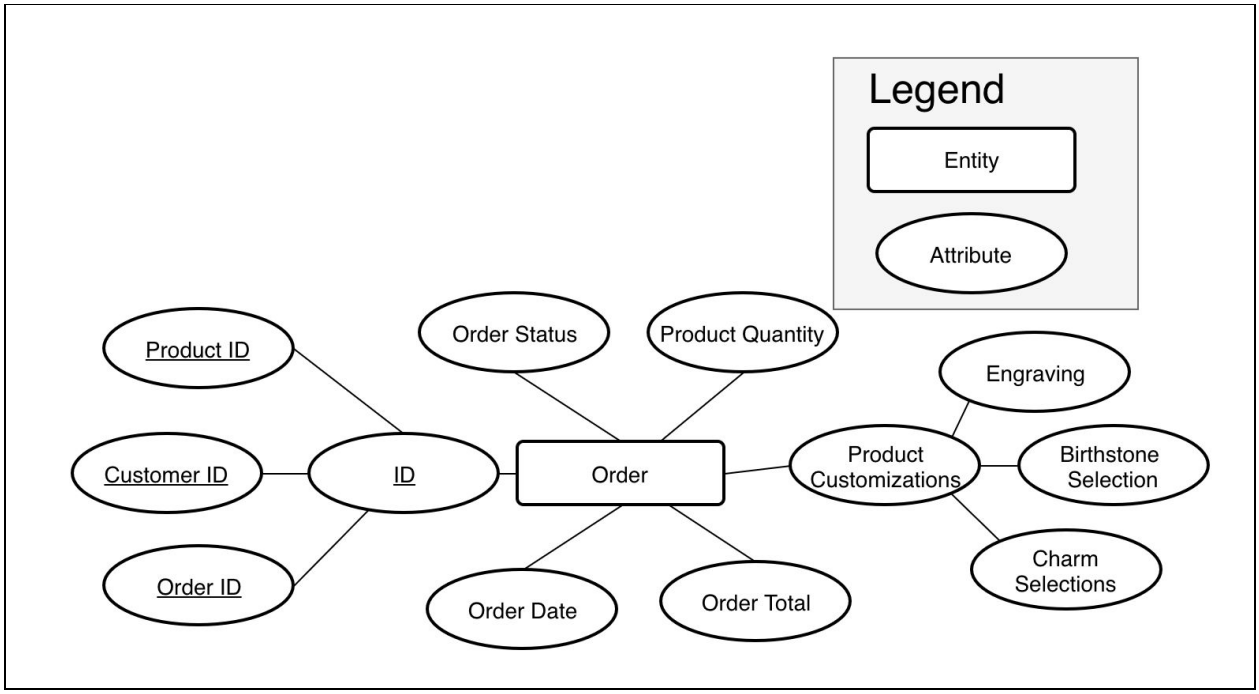
TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_COMMENT
birthstone	Birthstone_ID	int	First record is the default value used by the java model when rendering drop down options.
birthstone	Birthstone	text	The English name of the birthstone.
birthstone	Month	text	The English month for the birthstone. We do allow duplicates for inventory expansion purposes.
birthstone	Birthstone_Cost	double	The cost of the birthstone in USD.

Order

Orders are a many-to-many relation between customers and products. In order to enforce uniqueness in this table we chose to implement a composite key. Orders is purposefully limited to a single table (instead of creating a transitive relation between customers and orders, and orders and products) to minimize the number of HQL queries that needed to be written to interface with the database. As such, the order entity represents that data related to a customer's choices.

Candidate Keys:

- (Customer_ID, Product_ID, Order_ID): composite key to group products, customers, and orders.



Below is a list of the attributes that have been implemented in the MySQL instance along with brief descriptions.

TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_COMMENT
orders	Order_ID	int	Order ID is used to group all items (records) for an order. Order ID does not constitute a PK by itself.
orders	Order_Date	datetime	The date the order was SUBMITTED.
orders	Order_Status	text	Current status of the order. 'Open' means that 'Submit order' has not yet been clicked.
orders	Product_ID	int	The FK for the product being purchased.

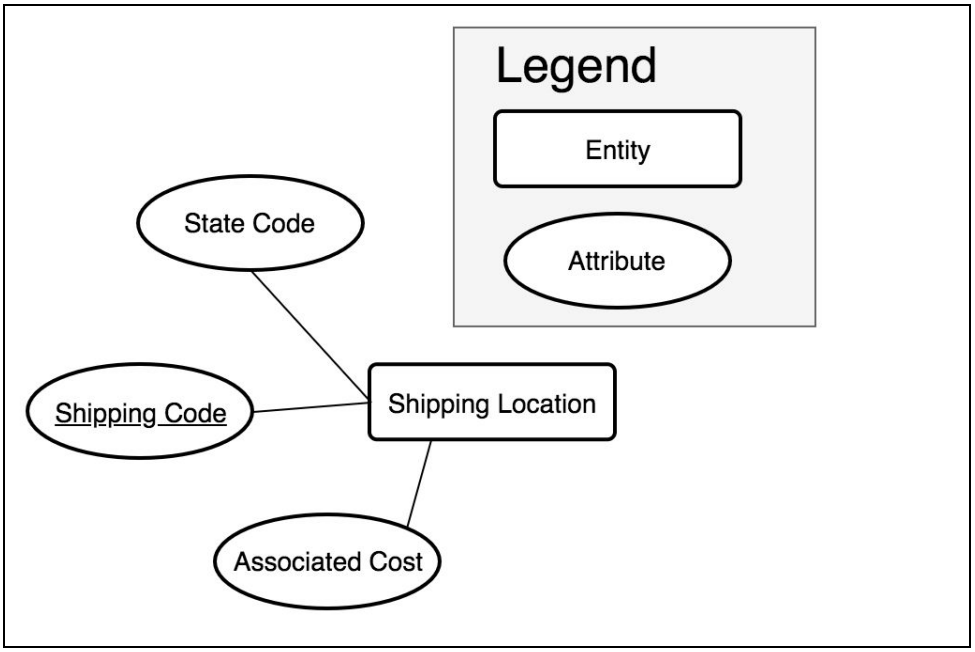
orders	Order_Total	double	The order total. Note: this is duplicated for each product record on an order.
orders	Product_Quantity	int	The number of exact items on the order. For example, some customers may order two wedding rings…..
orders	Customer_Choice_Name_Engraving	text	Engraving based on user input.
orders	Customer_Choice_Birthstone_ID	text	Birthstone selection based on user input.
orders	Customer_Choice_Charm_ID_1	text	Charm selection based on user input.
orders	Customer_Choice_Charm_ID_2	text	Charm selection based on user input.
orders	Customer_Choice_Charm_ID_3	text	Charm selection based on user input.
orders	Customer_Choice_Charm_ID_4	text	Charm selection based on user input.
orders	Customer_ID	int	Customer making the purchase.

Shipping Location

Shipping costs vary by state, and are provided in the table below. Costs are flat regardless of order size. The cost is added at the time of checkout.

Candidate Keys:

- Shipping_Code_ID: Created to avoid collisions should business expand internationally.
- State_Code: Currently, this would satisfy the requirements.



Below is a list of the attributes that have been implemented in the MySQL instance along with brief descriptions.

TABLE_NAME	COLUMN_NAME	DATA_TYPE	COLUMN_COMMENT
shipping_costs	Shipping_Code_ID	int	"Shipping ID. Used a generated PK because cannot ensure state codes will be unique if we expand beyond the US. "
shipping_costs	State_Code	varchar	"State code to match with customer state. 2 characters
shipping_costs	Shipping_Cost	double	"Cost of shipping am order to this state. "

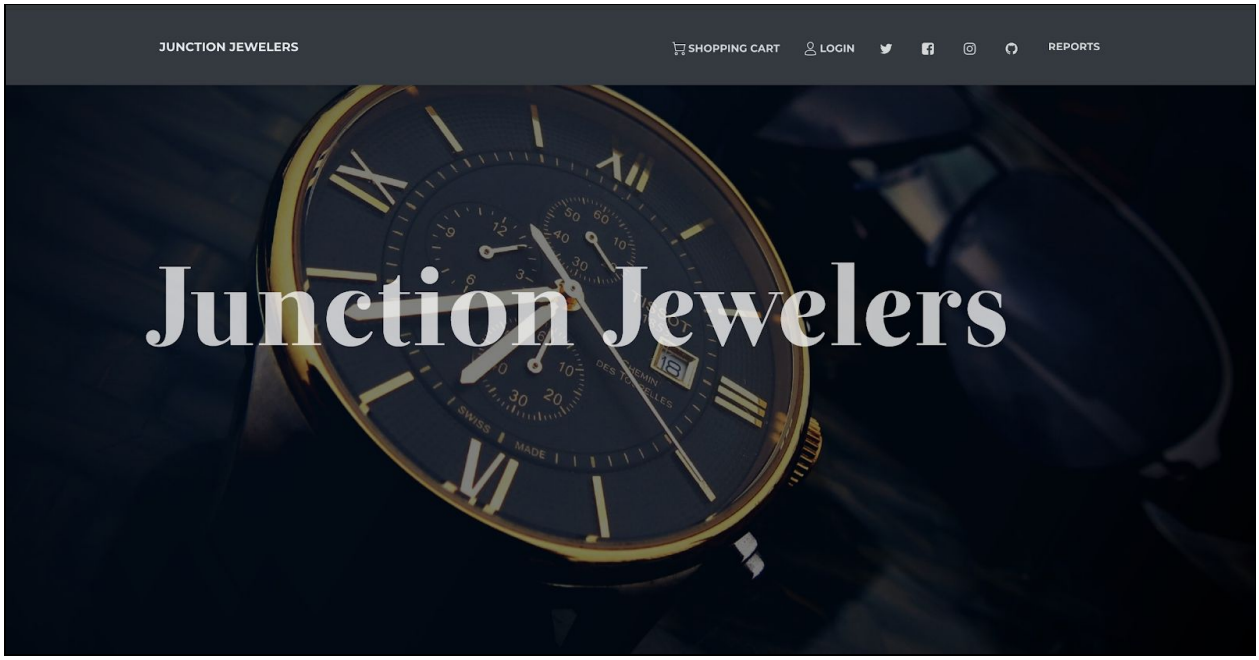
Front-End Website Specification

This section contains detailed descriptions of the customer-facing layer of the web application, including the functionality that is expected from each respective web page. For more information about the technology driving the web pages please see the ‘Technology Stack’ section of the Data Dictionary. This section is intended for front-end developers and readers familiar with the web functionality. For more information about the business logic and the database system please see the sections: ‘Business Logic’ and ‘Data Model’, respectively. This section outlines only what the end-user can see and what behaviors the user should expect when navigating Junction Jewelers website.

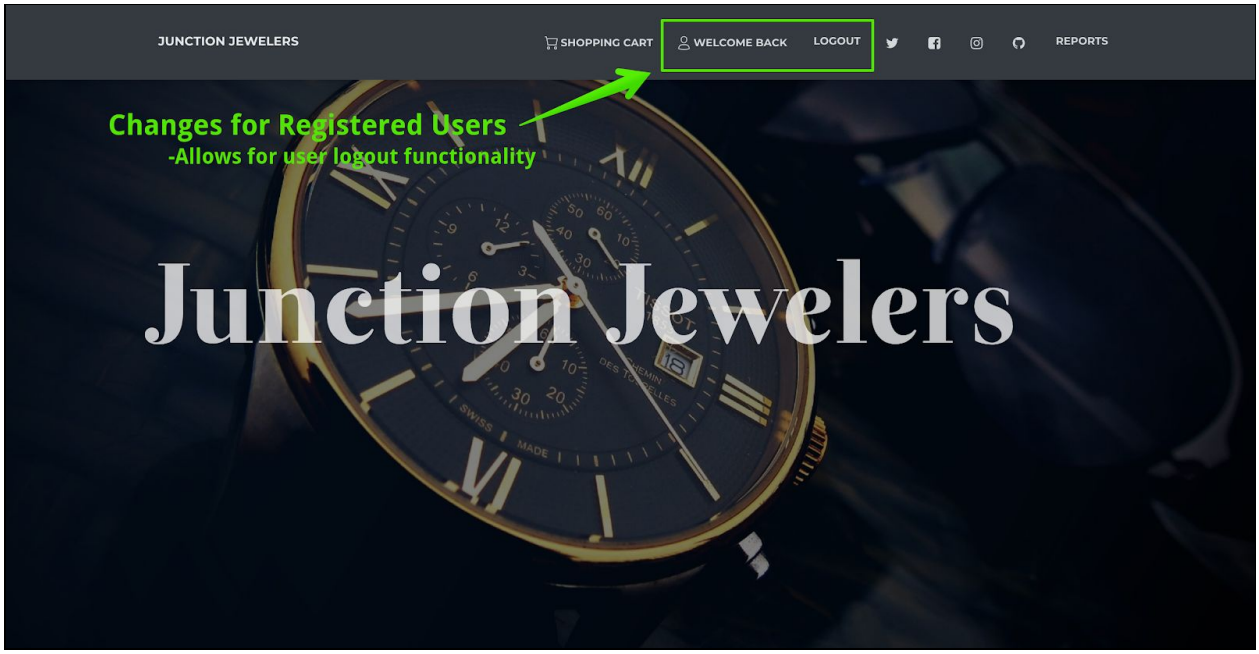
Homepage

The initial landing page for Junction Jewelers (URL to come once deployed). The homepage presentation is based on whether the user is logged in. If the user has previously logged in during the current session they will be directed to (2), otherwise they will see (1). By scrolling the user can view (3) links to product pages, the (4) ‘About Us’ section, and the (5) newsletter sign-up.

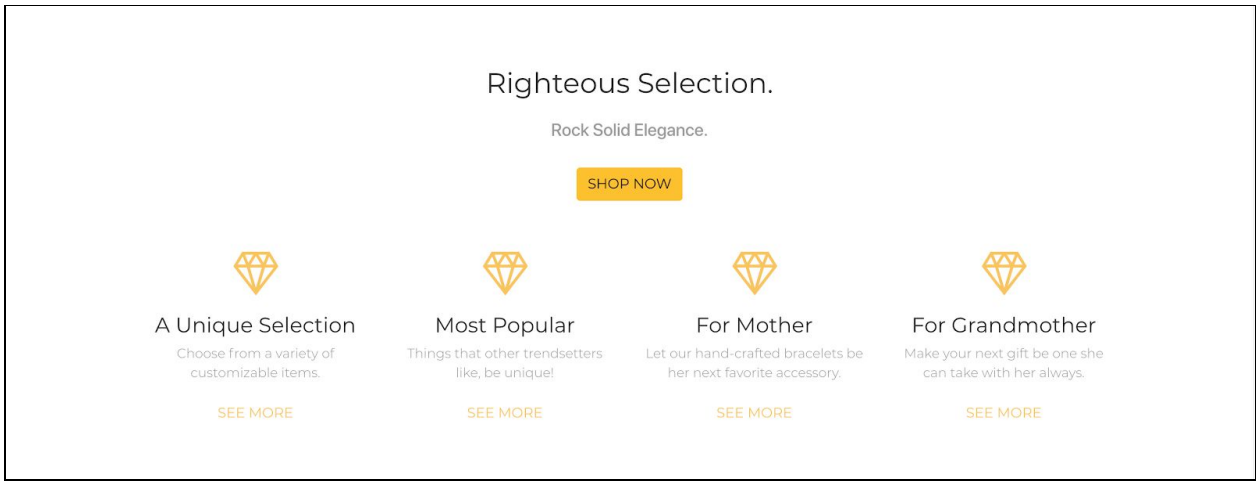
1. Landing Page View for Unregistered User



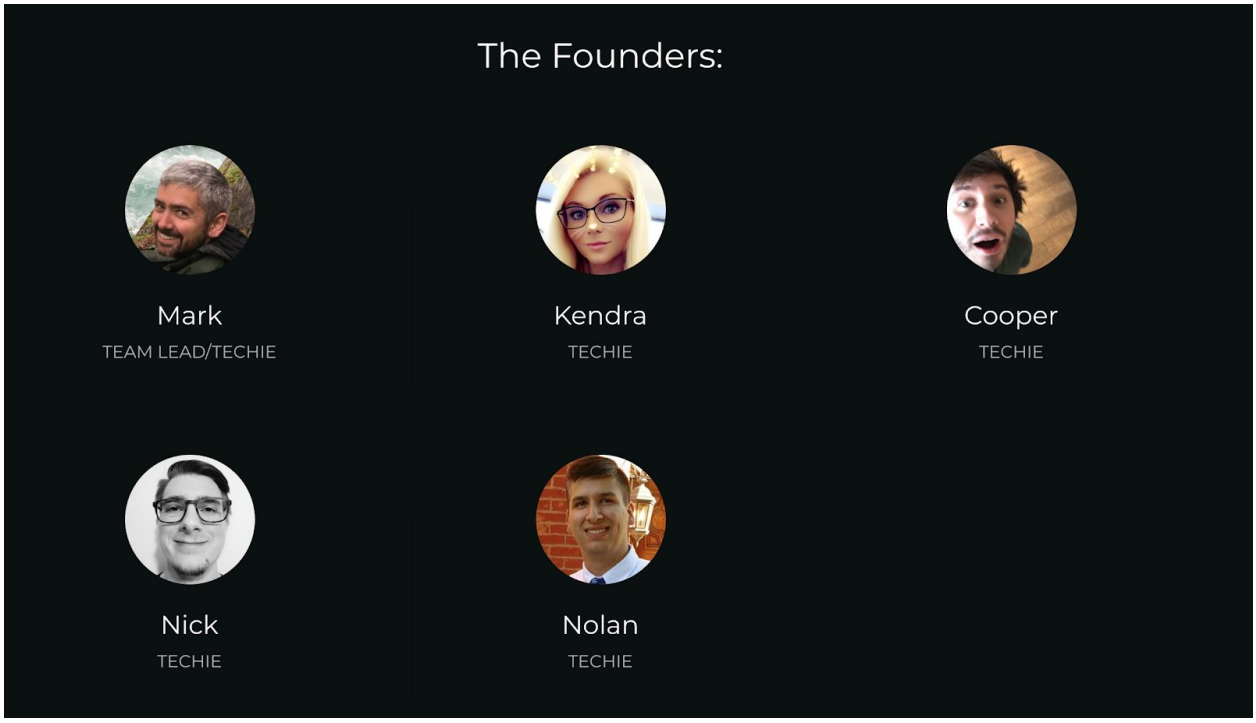
2. Landing Page for Logged-In User



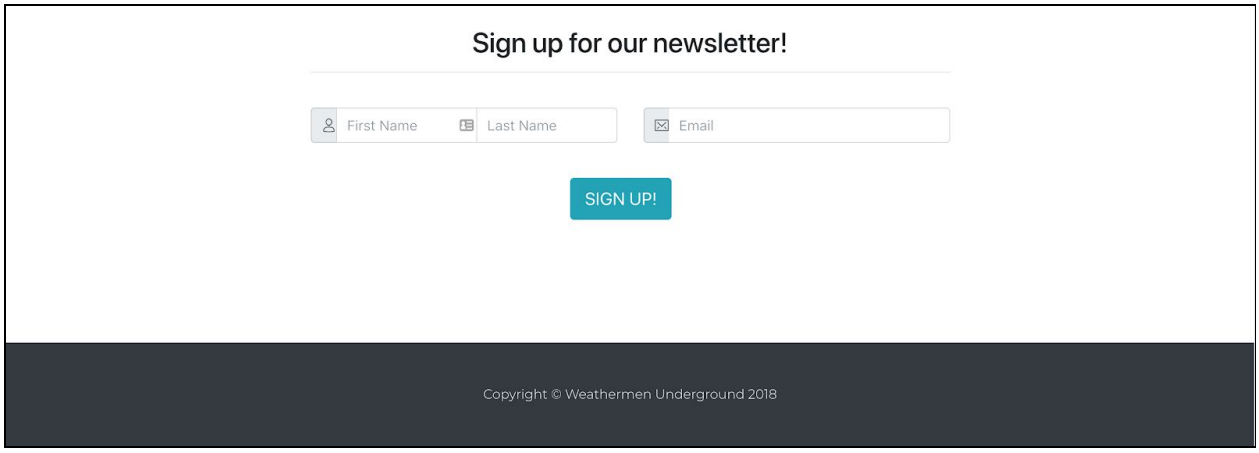
3. Link to Products



4. Team photos and Information



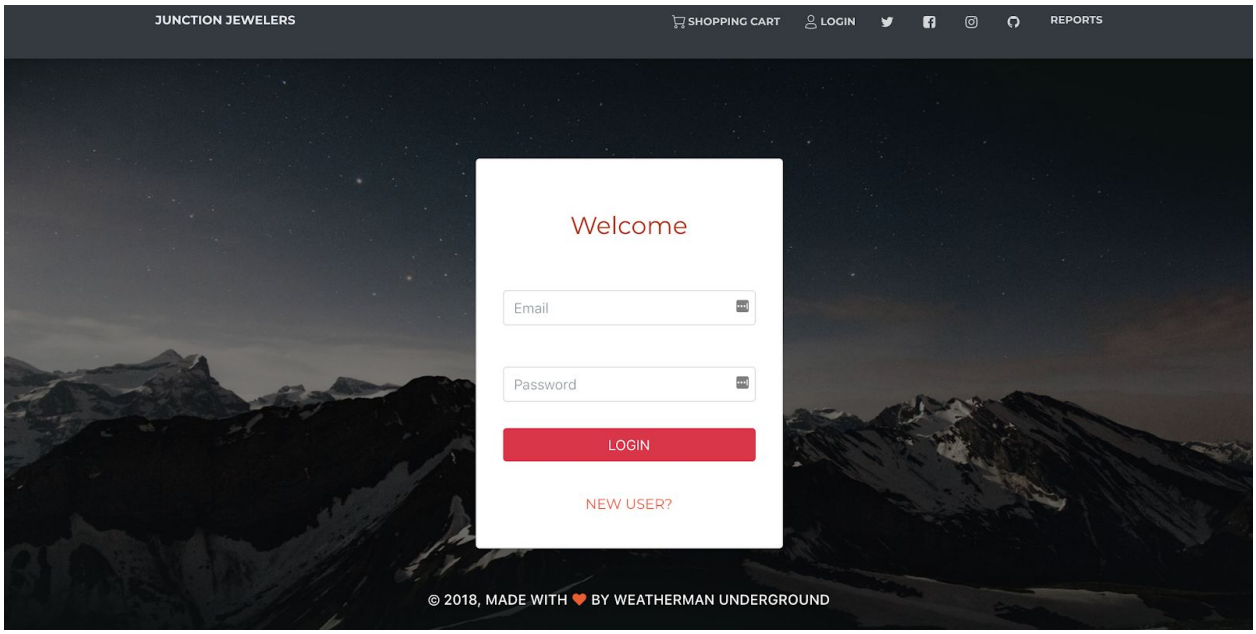
5. Email signup



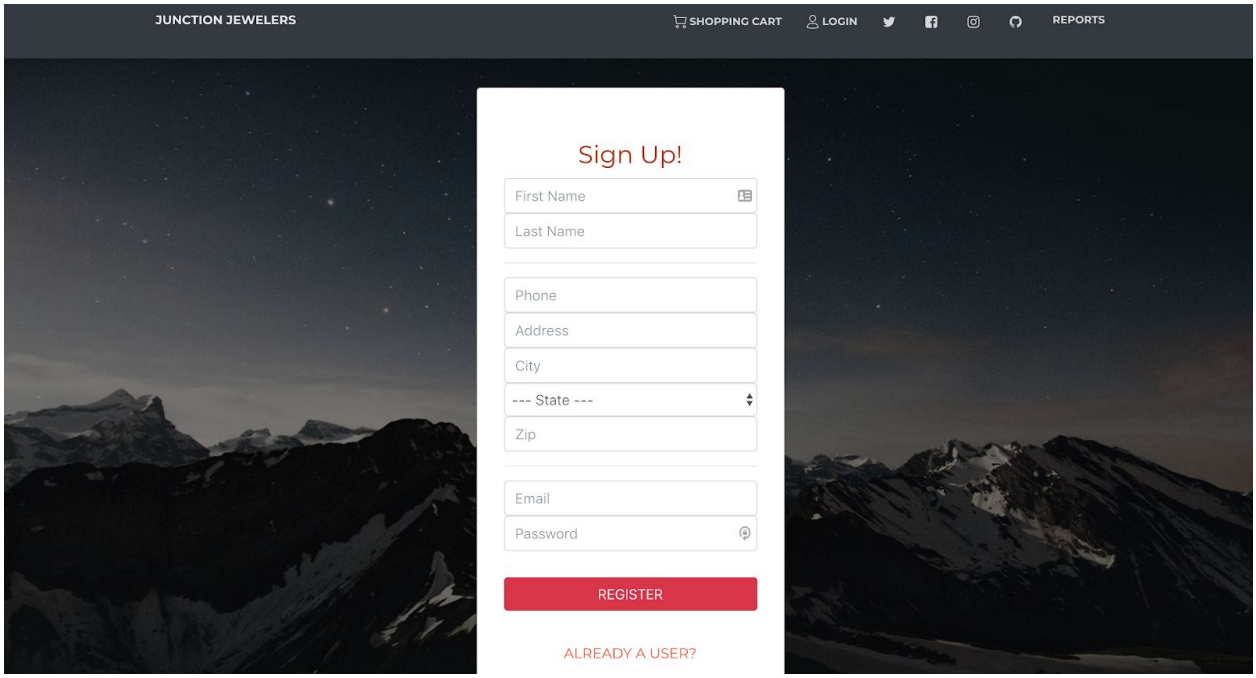
Customer Login and Signup

These pages enable users to login to their existing account (1), or create a new account if one doesn't exist (2). New users are prompted for their name,address, email, and password. After login, the customer is redirected back to the homepage. If the user enters and invalid username or password, the page refreshes and they can re-enter their information. Note: the users' email acts as their username

1. Login Functionality



2, Create New Account Page

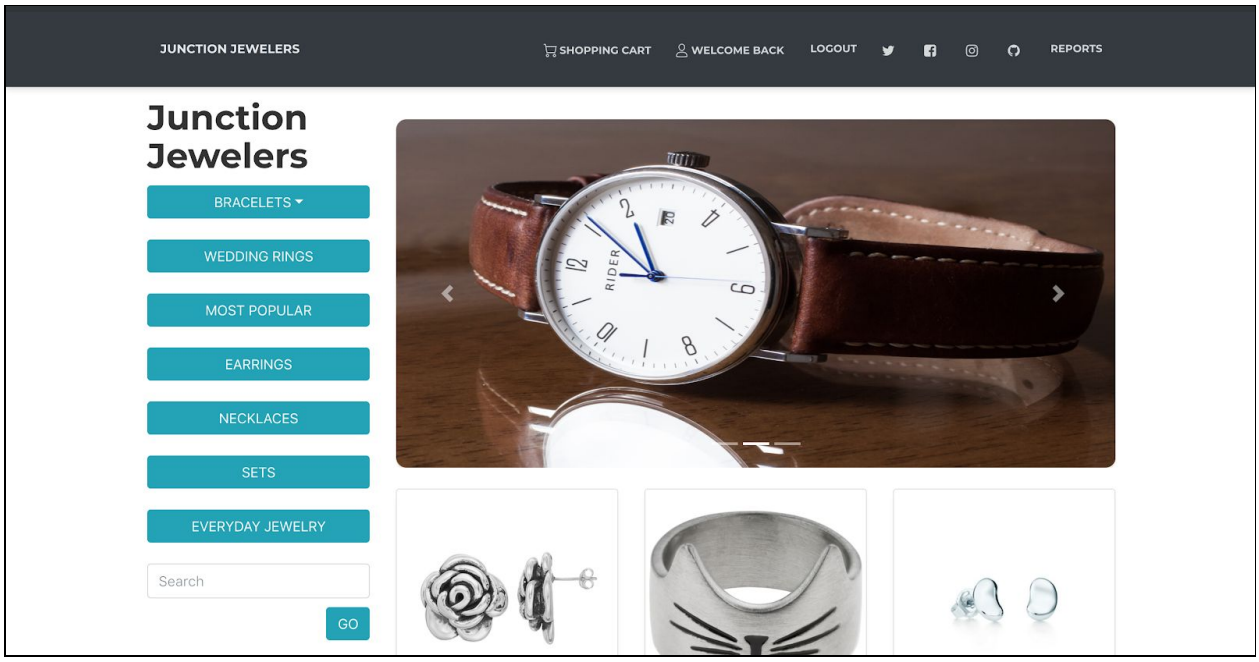


Product List Page

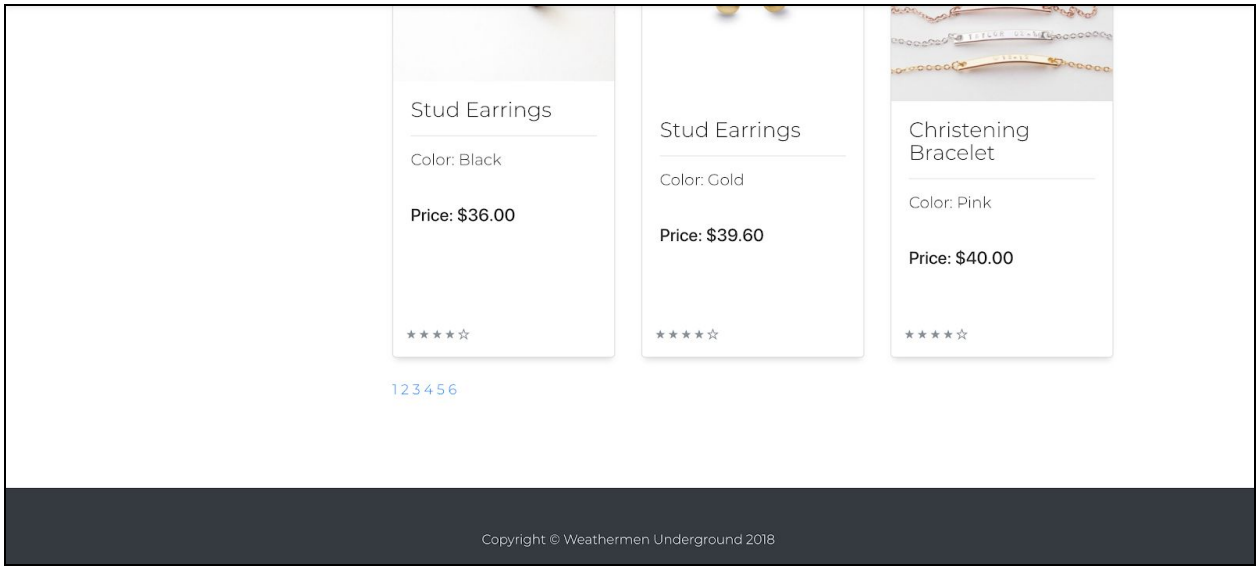
The storefront is the primary navigation for the store, all products are be available through this page. The first view of the product page (1) shows all products carried by the store, but paginates the products or only six are shown at a time (2). Clicking on any product category button will filter the products, displaying only those in that category (3 and 4). Clicking on any product will bring the user to the product page for that product. When the user clicks go on a search, any products matching that search term are displayed in the product area (5). Note: Search is limited to Product Name and Category.

Note on pagination:
For every query that returns a product list greater than 6 items, partitioning of the itemset across multiple pages, or *pagination*, will occur. The user will see the first six items returned by the query on the first page, and a list of hyperlinked page numbers at the bottom. When selected by the user, each number will direct the user to the following 6 items in the returned product list.

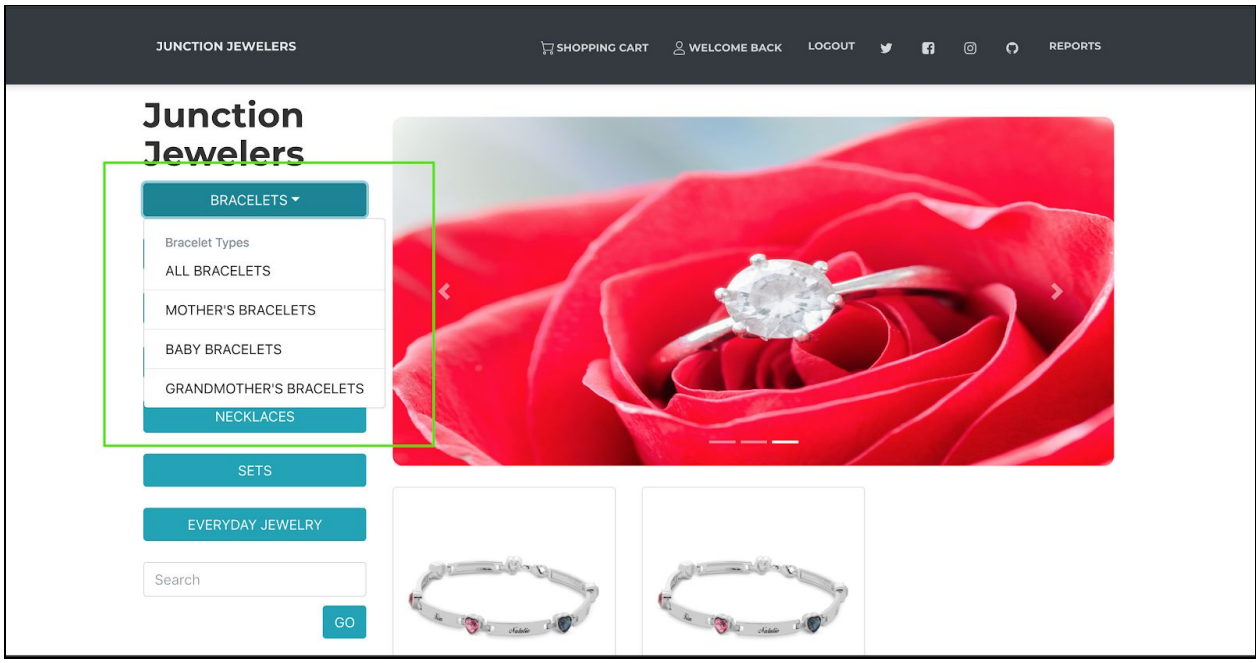
1. Initial Product List Page



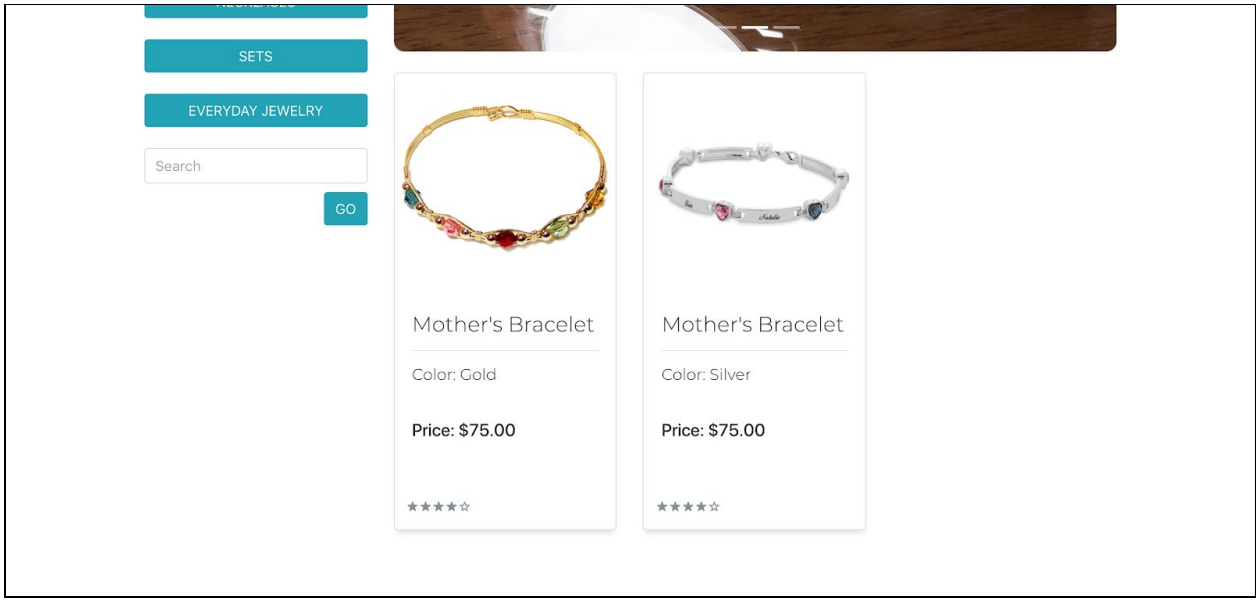
2. Product Pagination



3. Bracelet Dropdown View

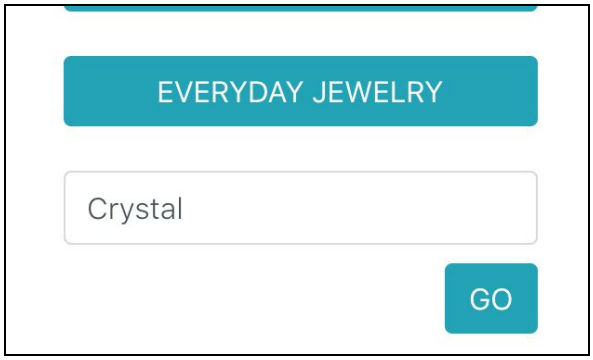


4. Category Filtering Demonstration

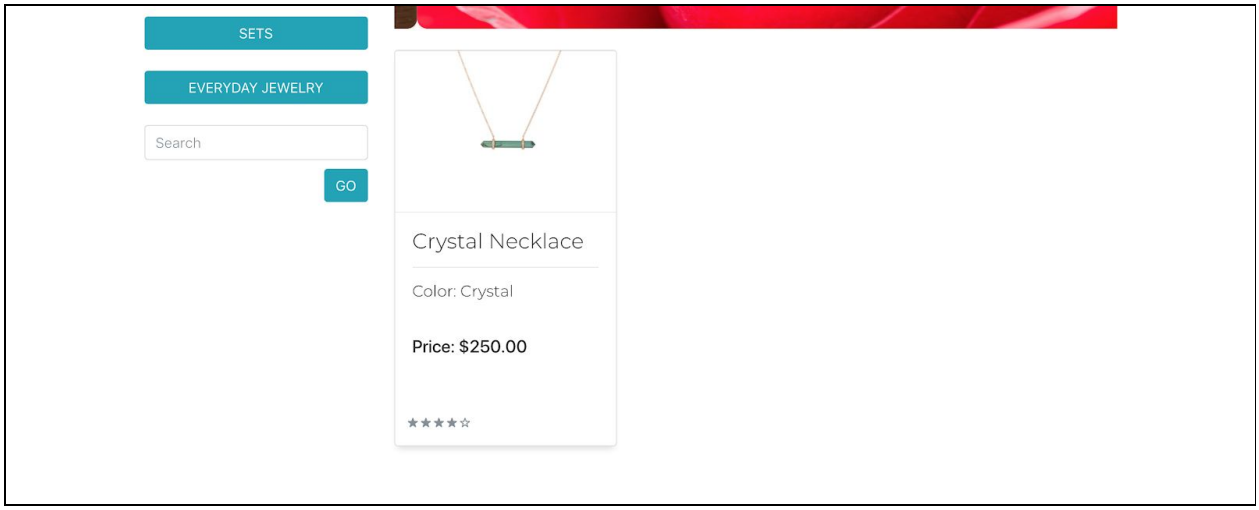


5. Search Demonstration

Search for Crystal



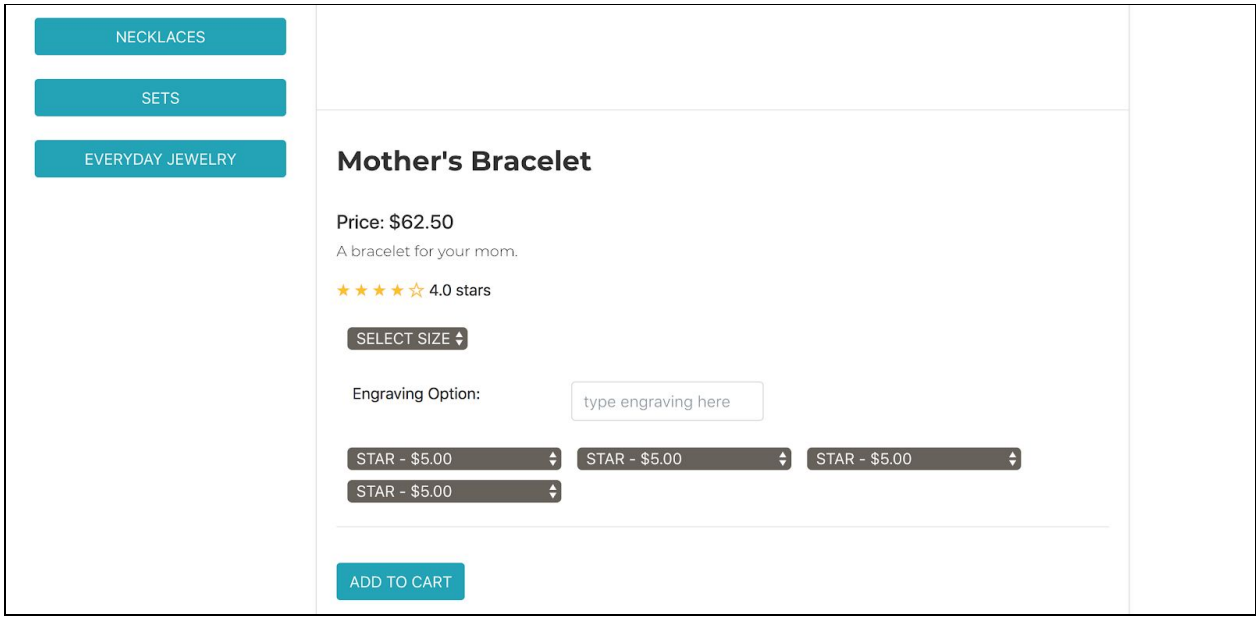
Results for Crystal



Product Pages

The product page displays information for all products and enables the user to view and purchase the item. Product customizations and options are available here (1). Once the user has created chosen a valid configuration of a product, they can add the item to their cart. If the user has chosen customizations, those are stored as part of their order and can be seen in their cart. Optional customizations (birthstone, engravings, charms) can be left blank.

1. Product Selection Page



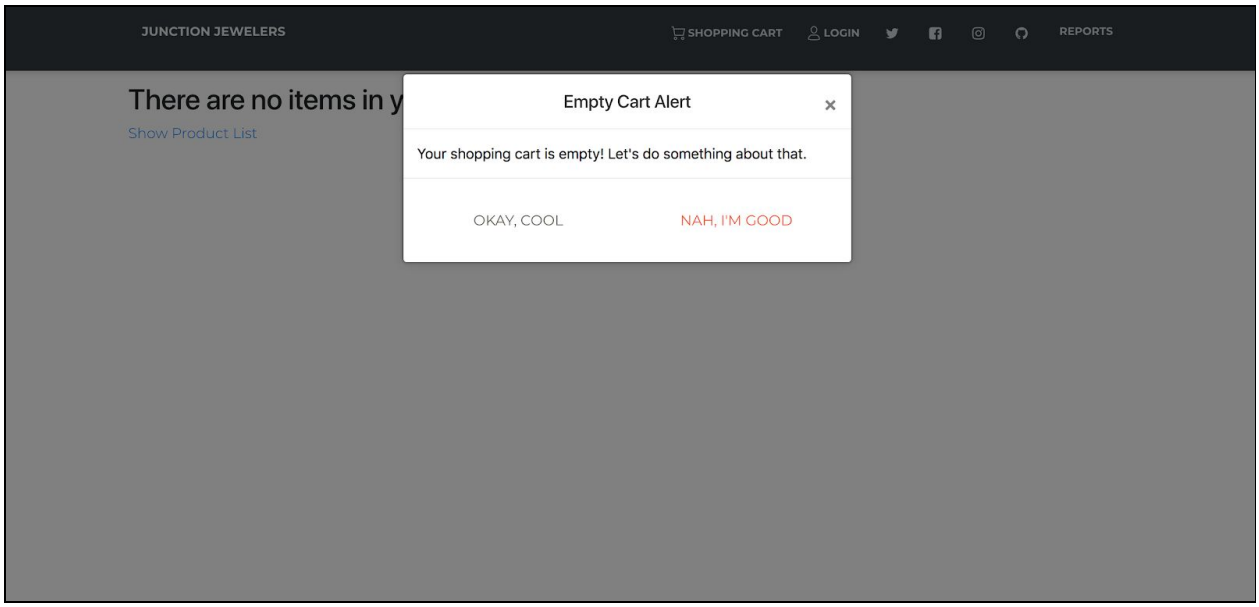
Shopping Cart

The shopping cart enables the user to review their cart, change the quantity of items in the cart, remove items, and checkout to complete their order (3). Prior to adding items to the cart, the cart is empty and will prompt the user to view the Product List page (1). Items can be added to the cart whether or not the user has logged in, but they will be required to login prior to checkout (2). When the user clicks checkout, the transaction is completed and marked complete in the database. All customizations for the product are displayed, alongside the cost of those options. The cart is emptied. To change the quantity of items in the cart, the user can:

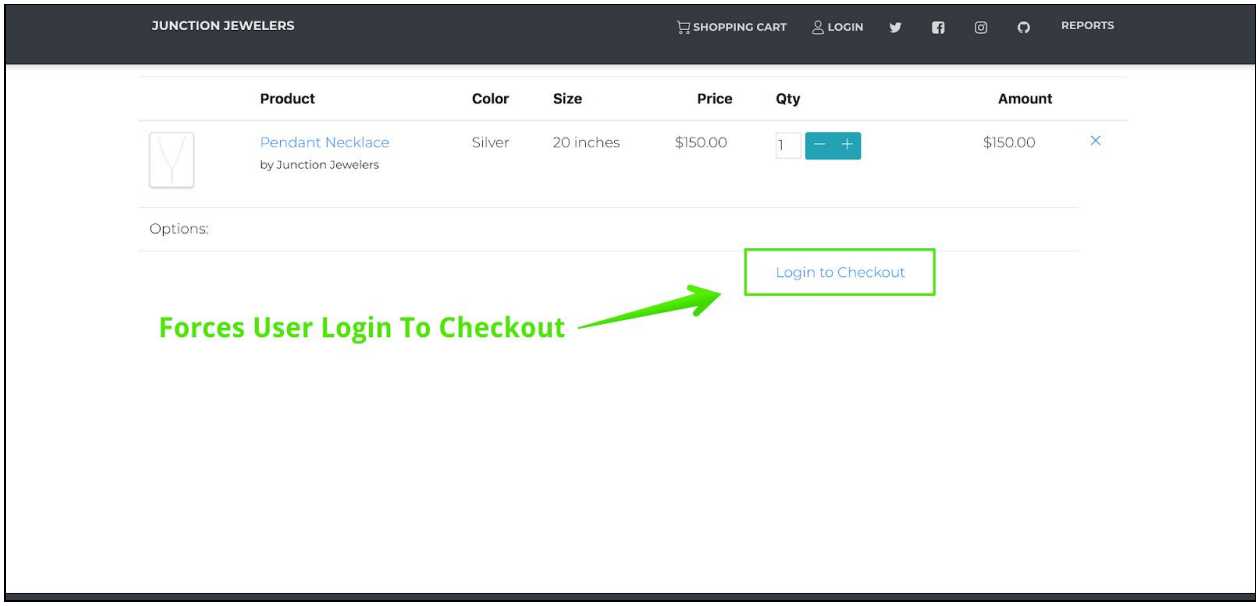
- Change the text in the quantity text box and click ‘Update’
- Click the ‘x’ button to delete an item from the cart

After completing their purchase, the user is shown a receipt page as order confirmation (4). A receipt is sent to their email address as well.

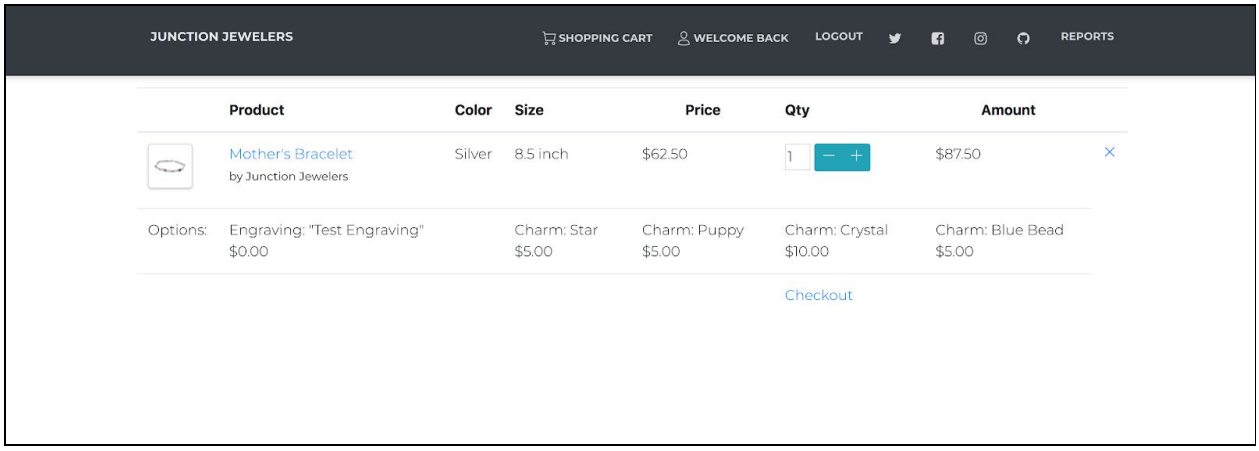
1. Empty Shopping Cart



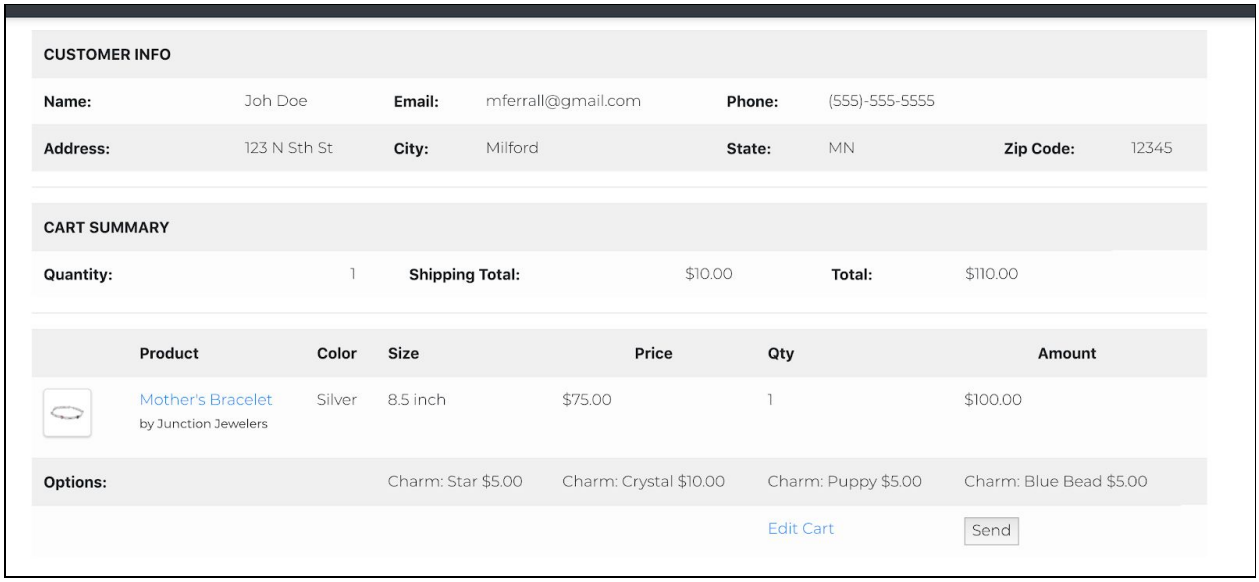
2. Shopping Cart for Non-Registered User



3. Shopping Cart for Registered User



4. Order Receipt and Confirmation



CUSTOMER INFO

Name:	Joh Doe	**Email:**	mferrall@gmail.com	**Phone:**	(555)-555-5555		
Address:	123 N Sth St	**City:**	Milford	**State:**	MN	**Zip Code:**	12345

CART SUMMARY

Quantity:	1	**Shipping Total:**	$10.00	**Total:**	$110.00

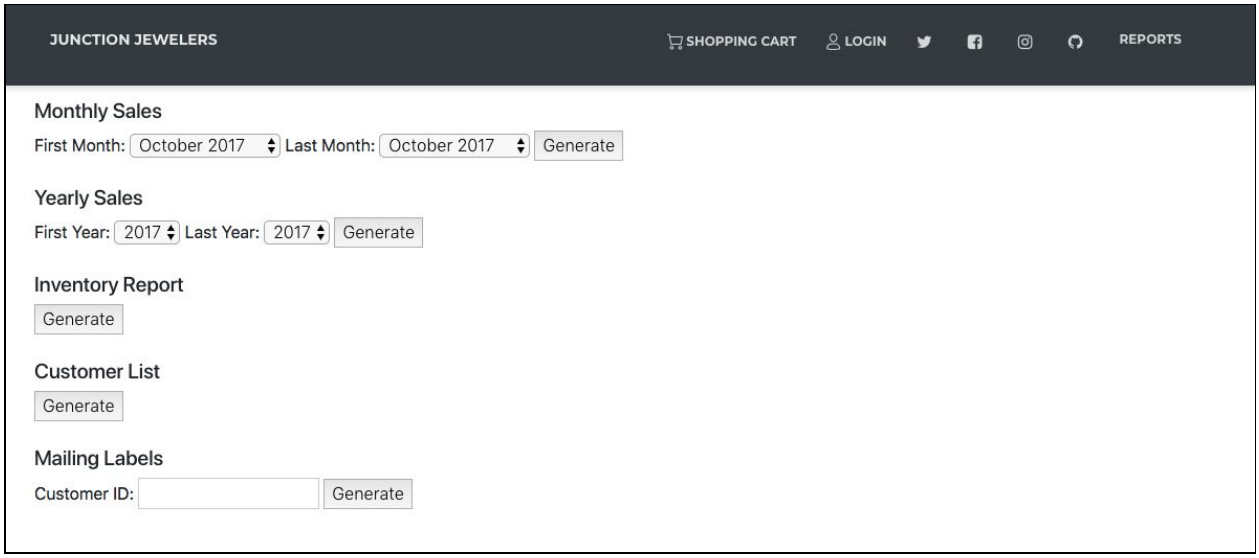
	Product	Color	Size	Price	Qty	Amount
	Mother's Bracelet by Junction Jewelers	Silver	8.5 inch	$75.00	1	$100.00
Options:			Charm: Star $5.00	Charm: Crystal $10.00	Charm: Puppy $5.00	Charm: Blue Bead $5.00
					Edit Cart	Send

Reports

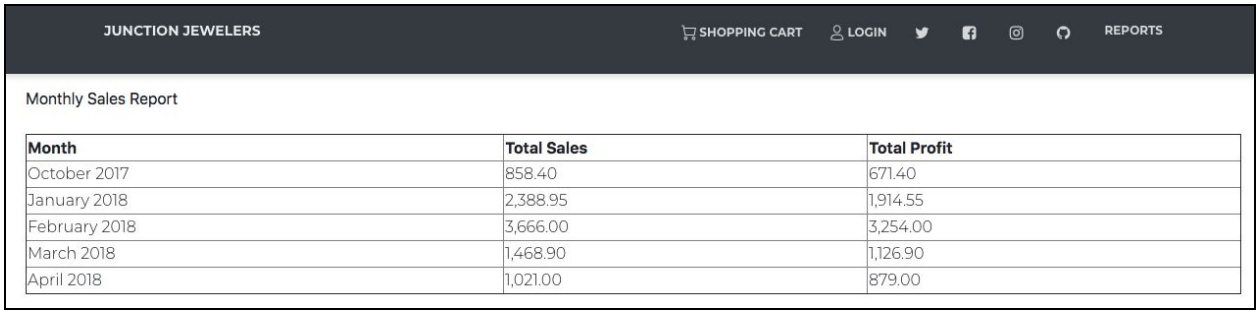
The reporting page provides five reports that summarize the sales of the store, inventory, and customer information. On the main reports page (1), the user is provided the option to select between five reports:

- Monthly Sales - Outputs two separate tables that summarize sales over the user entered time period from “First Month” and “Last Month”. Table one (2) shows the total revenue and profit for each month in the given range, while table two (3) shows the total number of products sold each month (results are grouped by product category).
- Yearly Sales - This report is very similar to the monthly sales report, except the results in each table are grouped by calendar year rather than month. User inputs are given by “First Year” and “Last Year”, and they specify which calendar years are to appear on the report. You can see the same tables for total revenue/profit (4) and total number of items sold (5).
- Inventory Report - Prints a list (6) of each product in the inventory and shows the current number in stock, along with the wholesale cost of each item and the total inventory cost of that item (determined by multiplying the quantity by unit wholesale cost). The total inventory cost is also give on the last line of the report (7). This amount signifies how much money could be made by liquidating all items in stock for their respective wholesale prices.
- Customer List - Prints a list of each customer in the sales database (8). Output gives the customer ID, customer name (first and last), and full customer address. No user input is required, as this report always prints an exhaustive list of the customers.
- Mailing Labels - Outputs a sheet of mailing labels for a given customer (9). The customer ID is provided by the user in an input prompt. The format for the results is a 3x3 table of mailing labels, which contain the typical 3-line US postal address format.

1. Main "Reports" Page



2. Monthly Sales - Total Revenue/Profit



Month	Total Sales	Total Profit
October 2017	858.40	671.40
January 2018	2,388.95	1,914.55
February 2018	3,666.00	3,254.00
March 2018	1,468.90	1,126.90
April 2018	1,021.00	879.00

3. Monthly Sales - Number of Each Product Category Sold

Month	Product Category	Quantity Sold
October 2017	Bracelet	2
October 2017	Everyday	3
October 2017	Necklace	1
January 2018	Earrings	3
January 2018	Ring	2
January 2018	Set	1
February 2018	Bracelet	1
February 2018	Ring	3
March 2018	Bracelet	1
March 2018	Necklace	1
March 2018	Ring	2
April 2018	Bracelet	1
April 2018	Necklace	2

4. Yearly Sales - Total Revenue/Profit

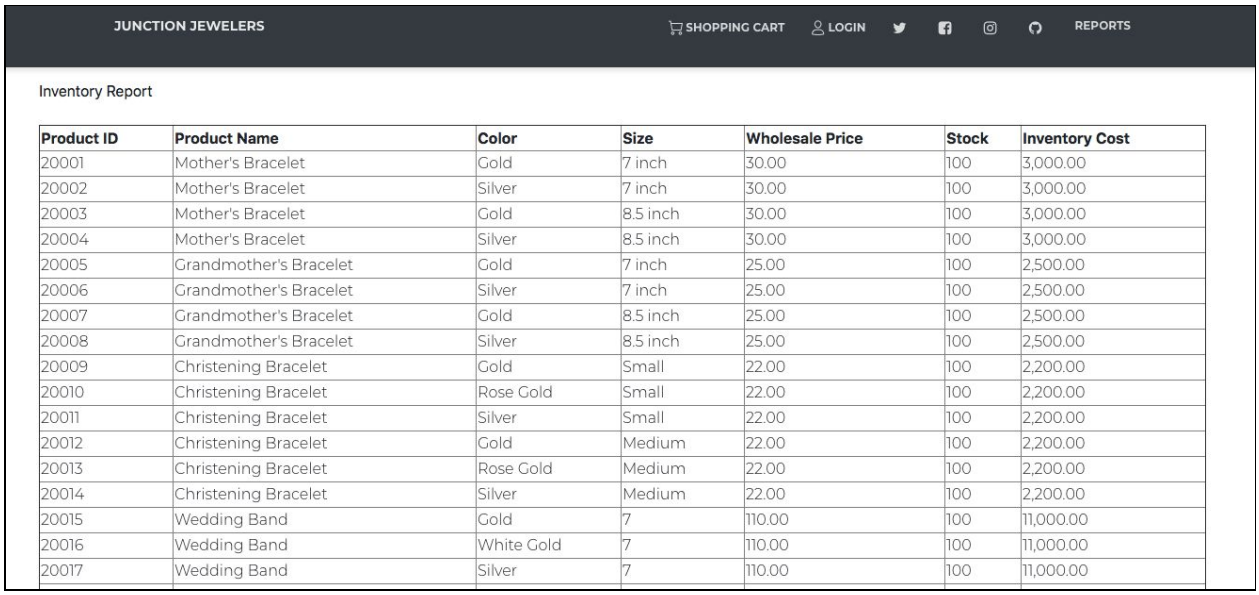
Yearly Sales Report

Year	Total Sales	Total Profit
2017	858.40	671.40
2018	18,508.80	14,745.40

5. Yearly Sales - Number of Each Product Category Sold

Year	Product Category	Quantity Sold
2017	Bracelet	2
2017	Everyday	3
2017	Necklace	1
2018	Bracelet	10
2018	Earrings	3
2018	Everyday	3
2018	Necklace	10
2018	Ring	10
2018	Set	17

6. Inventory Report - Title and Sample Rows



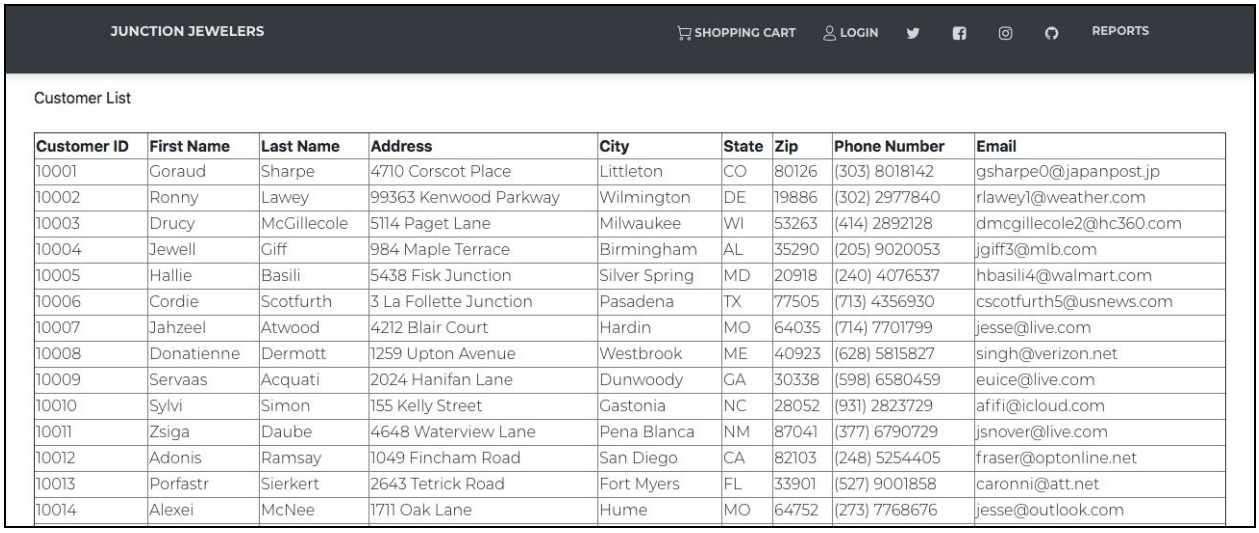
Inventory Report

Product ID	Product Name	Color	Size	Wholesale Price	Stock	Inventory Cost
20001	Mother's Bracelet	Gold	7 inch	30.00	100	3,000.00
20002	Mother's Bracelet	Silver	7 inch	30.00	100	3,000.00
20003	Mother's Bracelet	Gold	8.5 inch	30.00	100	3,000.00
20004	Mother's Bracelet	Silver	8.5 inch	30.00	100	3,000.00
20005	Grandmother's Bracelet	Gold	7 inch	25.00	100	2,500.00
20006	Grandmother's Bracelet	Silver	7 inch	25.00	100	2,500.00
20007	Grandmother's Bracelet	Gold	8.5 inch	25.00	100	2,500.00
20008	Grandmother's Bracelet	Silver	8.5 inch	25.00	100	2,500.00
20009	Christening Bracelet	Gold	Small	22.00	100	2,200.00
20010	Christening Bracelet	Rose Gold	Small	22.00	100	2,200.00
20011	Christening Bracelet	Silver	Small	22.00	100	2,200.00
20012	Christening Bracelet	Gold	Medium	22.00	100	2,200.00
20013	Christening Bracelet	Rose Gold	Medium	22.00	100	2,200.00
20014	Christening Bracelet	Silver	Medium	22.00	100	2,200.00
20015	Wedding Band	Gold	7	110.00	100	11,000.00
20016	Wedding Band	White Gold	7	110.00	100	11,000.00
20017	Wedding Band	Silver	7	110.00	100	11,000.00

7. Inventory Report - Total on Last Line

20071	Rose Stud Earings	Rose	One Size	10.00	3	30.00
20072	Cat Ring	Silver	5	11.00	100	1,100.00
20073	Cat Ring	Silver	6	11.00	100	1,100.00
20074	Cat Ring	Silver	7	11.00	100	1,100.00
20075	Cat Ring	Silver	8	11.00	100	1,100.00
20076	Silver Bangles	Silver	One Size	44.00	0	0.00
					Total:	**549,930.00**

8. Customer List

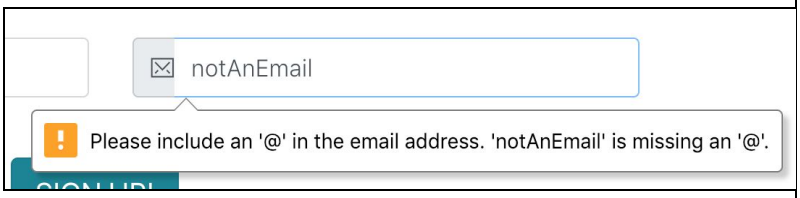
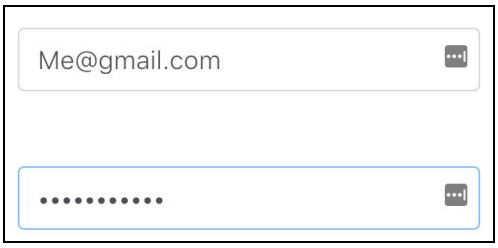
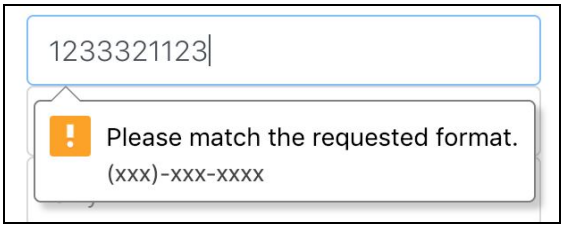


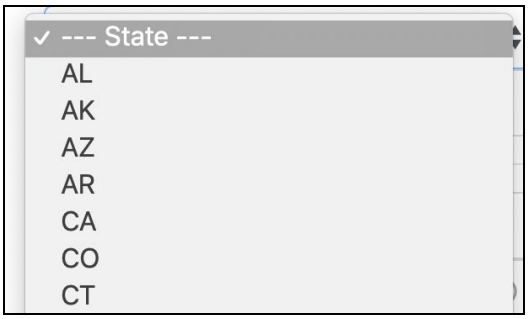
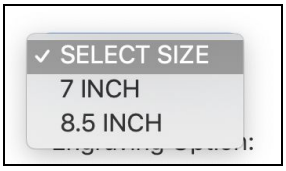
Customer ID	First Name	Last Name	Address	City	State	Zip	Phone Number	Email
10001	Goraud	Sharpe	4710 Corscot Place	Littleton	CO	80126	(303) 8018142	gsharpe0@japanpost.jp
10002	Ronny	Lawey	99363 Kenwood Parkway	Wilmington	DE	19886	(302) 2977840	rlawey1@weather.com
10003	Drucy	McGillecole	5114 Paget Lane	Milwaukee	WI	53263	(414) 2892128	dmcgillecole2@hc360.com
10004	Jewell	Giff	984 Maple Terrace	Birmingham	AL	35290	(205) 9020053	jgiff3@mlb.com
10005	Hallie	Basili	5438 Fisk Junction	Silver Spring	MD	20918	(240) 4076537	hbasili4@walmart.com
10006	Cordie	Scotfurth	3 La Follette Junction	Pasadena	TX	77505	(713) 4356930	cscotfurth5@usnews.com
10007	Jahzeel	Atwood	4212 Blair Court	Hardin	MO	64035	(714) 7701799	jesse@live.com
10008	Donatienne	Dermott	1259 Upton Avenue	Westbrook	ME	40923	(628) 5815827	singh@verizon.net
10009	Servaas	Acquati	2024 Hanifan Lane	Dunwoody	GA	30338	(598) 6580459	euice@live.com
10010	Sylvi	Simon	155 Kelly Street	Gastonia	NC	28052	(931) 2823729	afifi@icloud.com
10011	Zsiga	Daube	4648 Waterview Lane	Pena Blanca	NM	87041	(377) 6790729	jsnover@live.com
10012	Adonis	Ramsay	1049 Fincham Road	San Diego	CA	82103	(248) 5254405	fraser@optonline.net
10013	Porfastr	Sierkert	2643 Tetrick Road	Fort Myers	FL	33901	(527) 9001858	caronni@att.net
10014	Alexei	McNee	1711 Oak Lane	Hume	MO	64752	(273) 7768676	jesse@outlook.com

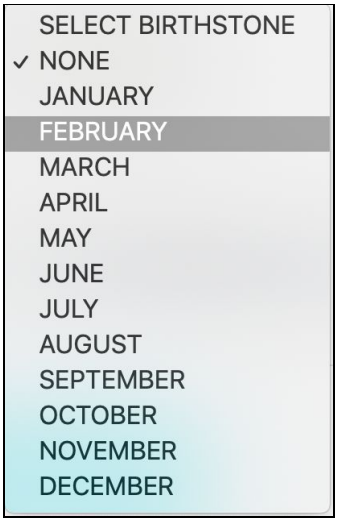
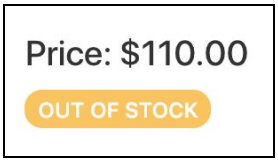
9. Mailing Labels

Front-End Data Validation

This section outlines and demonstrates data validation in the database. These data fields include those related to user accounts, products, and order details. The constraints in place prevent invalid data from entering and corrupting the database.

Data Field	**Expected Behavior**	**Demonstration**
Email Address Entry	On click with invalid email address, display “Please include an ‘@’ in the email address [user entry] is missing an ‘@’.” Applies to Newsletter Email, Login Email, and New Account Email	
Non Existing Account	On click of Sign In, page refreshes	Invalid Entry  After Refresh 
Phone Number Entry	On Click of ‘Register’, if format does not match requirement, display “Please match the requested format. (xxx)-xxx-xxxx”	

Data Field	Expected Behavior	Demonstration
Account Information Entry	On Click of 'Register', if information is empty, display "Please fill out this field." Applies to Address, City, Zip, Email, and Password	
Address State Entry	State validated by restricting user to the options provided	
Size Entry	Product size entry limited to sizes available in database. Applies to all products.	
Charm Selection	Charm options are limited to those available in the database.	

Data Field	Expected Behavior	Demonstration
Birthstone Selection	Birthstone options are limited to those in the database.	
Out of Stock Items	Out of stock item product pages cannot be accessed. Message “Out of Stock” is displayed	
Order Quantity Exceeding Stock Quantity	On click of ‘Update’, if quantity exceeds stock, page refreshes to previous value	*Prior to Clicking ‘Update’*  *After Clicking ‘Update’* 

Data Field	Expected Behavior	Demonstration
Monthly Report Month Entry	Only months with orders in the database are available for selection via dropdown menu.	
Yearly Report Year Entry	Only years with orders in the database are available for selection via dropdown menu.	

Business Layer Specification

This section contains details of the business logic in the logic layer of the application stack. The logic layer acts as a mitigating layer between the database and the customer-interface. As such, the Java application requires a set of representations for the entities described in the ‘Entity Specification’ section of this document. These entities help facilitate a number of core operations including communication with the database.

Create User

- When a new user visits the site and navigates to the login page, they will be able to create an account by selecting the “Register Now” link at the bottom of the login form.
- The user is then required to fill all out fields prior to submitting.
- Upon passing Spring Security validation, a new customer is created and persisted to the database.
- Note a personalized session is not present until the user officially logs in via the login page.

Login

- When a returning user visits the site, they are considered anonymous by the HTTP response, and therefore are unable to checkout items.
- By navigating to the login page, a user is able to login with their email and correlating password.
- Upon validating the credentials, a personalized user session is created and becomes active for 30 minutes, or until the user logs off.
- This session is stored in the HTTP response along with the user’s unique id.
- The active session allows users to checkout products as frequently as desired within the 30 minute duration.

Create New Cart

- A new cart is created for each site session, and persists between user authentication states.
- It is configured to automatically add to the HTTP request, via a customized HTTP utility method, allowing it to be independent of the user personalized session.
- This functionality ensures all items in the cart - prior to login/register - remain until user action is taken.
- Once a user places an order and receives confirmation of such, the cart is then cleared in the request.

Add/Remove/Update Quantity from Cart

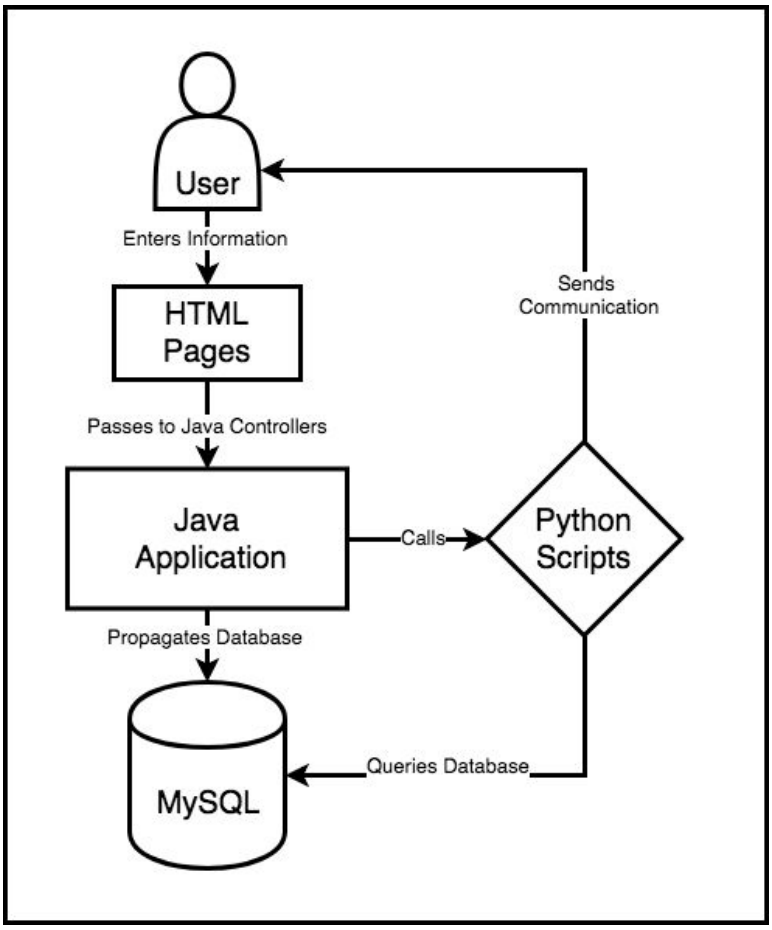
- Any item in our store that is not out of stock has the ability to be added to the session cart.
- Likewise, any item that exists in a cart at a given time has the option to be removed.
- Upon addition of an item, the controller creates a new “line” that stores all item features as specified by the customer.
- This line is then added to the cart object, after it is retrieved from the session, and the new state is persisted within the response.
- When a customer requests the deletion of a cart item, the cart is retrieved from the session, and the line corresponding to the requested item is deleted.
- The new state of the cart is then persisted to back to the request.
- Users may change the quantity of each selected item in the shopping cart page by adding the desired number and pressing “update”.
- When update is selected, the POST HTTP response returns the user’s new specified quantity, as well as the correlating item ID, to the controller as response attributes.
- The attributes are then used to lookup the specified item in the cart via a DAO call, and the quantity is updated within the cart.
- Once again, the new state of the cart is persisted.

Checkout

- When an authenticated user decides to checkout, the option will be made available to them at the bottom of the shopping cart page, via a check from the response for the current principal ID.
- By selecting this option, the user will be directed to a confirmation page rendering all user shipping information (via retrieval of their auto-generated ID from the request principal, and DAO lookup based on this) and their soon-to-be order details.
- If the user chooses to confirm this order, they will select the “Send” button at the bottom of the page.
- This will trigger two actions in the following order:
    - 1) The creation and persistence of the order to the database
    - 2) The call of a customized python script that sends the user a receipt of their order.
- Action #1 is performed by retrieval of the shopping cart from the request’s session which will be returned in its most recently updated state.
- Then through a DAO call, multiple order instances are created, one for each product in the user’s cart.
    - This functionality is allowed by the composite primary key that is composed of the generated Order ID, the Customer’s ID, and Product ID.
- After the creation of each, they are persisted to the database, and the session is flushed.

Email Receipt

This section provides a description of the Emailing functions, and how they are implemented. (See following page)



Technology

The below features, Email Receipt and Customer Newsletter, utilize python and two main libraries to achieve the desired functionality. The first library, MySQL Connector, is used to connect to our database. It enables generating SQL queries to retrieve the necessary data. The second library is the SMTPLib. This allows us to connect to an SMTP server with a set of given credentials to generate outgoing email messages.

Email Receipt

Creating a receipt requires a single input of OrderID. Given this input, we can derive the Customer's Address information. We can also retrieve which products, as well as added options, were selected in the order. With this information, we format the customer's information into a shipping address, and create a formatted itemized list, including options and pricing. This data is fed into the the SMTP library to send the appropriate customer an email.

Customer Newsletter

A newsletter is sent to a user when they register for the website. We collect the email address, first name, and last name of the customer on the registration page and feed these three inputs into the newsletter function. Within this function, we check the current month and look up the corresponding birthstone. We then generate a welcome email using the customer's first name as a greeting and include the name of the birthstone that is associated with the current month.

Database Entity Specification

This section details the MySQL instance specifications. This section is intended for developers familiar with SQL looking to see exact create statement specification for each table in the database. Each entity from above is represented as a relation, with attributes and data types defined congruently, but sometimes differently to the Java entity representation.

Charm


```
CREATE DATABASE  IF NOT EXISTS `jewlz` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `jewlz`;
-- MySQL dump 10.13  Distrib 8.0.13, for macos10.14 (x86_64)
--
-- Host: localhost    Database: jewlz
-- ------------------------------------------------------
-- Server version 8.0.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
```

```
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `charm`
--

DROP TABLE IF EXISTS `charm`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
 SET character_set_client = utf8mb4 ;
CREATE TABLE `charm` (
  `Charm_ID` int(11) NOT NULL COMMENT 'First record is the default value
used by the java model when rendering drop down options.',
  `Charm` text NOT NULL COMMENT 'The English name of the charm that is
rendered in the drop down options. ',
  `Charm_Price` double NOT NULL COMMENT 'The price of the add-on charm in
USD. ',
  PRIMARY KEY (`Charm_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-12-04 10:43:58
```

Birthstone

Table: birthstone

Columns:

Birthstone_ID	int(11) PK
Birthstone	text
Month	text
Birthstone_Cost	double

```
CREATE DATABASE  IF NOT EXISTS `jewlz` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `jewlz`;
-- MySQL dump 10.13  Distrib 8.0.13, for macos10.14 (x86_64)
--
-- Host: localhost    Database: jewlz
-- ------------------------------------------------------
-- Server version 8.0.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `birthstone`
--

DROP TABLE IF EXISTS `birthstone`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
```

```
 SET character_set_client = utf8mb4 ;
CREATE TABLE `birthstone` (
  `Birthstone_ID` int(11) NOT NULL COMMENT 'First record is the default
value used by the java model when rendering drop down options.',
  `Birthstone` text NOT NULL COMMENT 'The English name of the birthstone.
',
  `Month` text NOT NULL COMMENT 'The English month for the birthstone. We
do allow duplicates for inventory expansion purposes. ',
  `Birthstone_Cost` double NOT NULL COMMENT 'The cost of the birthstone in
USD. ',
  PRIMARY KEY (`Birthstone_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-12-04 10:43:58
```

Customer

Table: customer

Columns:

Customer_ID	int(11) AI PK
Customer_First_Name	text
Customer_Last_Name	text
Password	text
Customer_Email_Address	text
Customer_Phone_Number	text
Customer_Street_Address	text
Customer_City	text
Customer_State	varchar(3)
Customer_Zip	int(11)

```
CREATE DATABASE  IF NOT EXISTS `jewlz` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `jewlz`;
-- MySQL dump 10.13  Distrib 8.0.13, for macos10.14 (x86_64)
--
-- Host: localhost    Database: jewlz
-- ------------------------------------------------------
-- Server version 8.0.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 SET NAMES utf8 ;
```

```
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `customer`
--

DROP TABLE IF EXISTS `customer`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
 SET character_set_client = utf8mb4 ;
CREATE TABLE `customer` (
  `Customer_ID` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Customer ID always
starts with '1'. ',
  `Customer_First_Name` text COMMENT 'Given name for the customer. This is
driven by customer input. ',
  `Customer_Last_Name` text COMMENT 'Surname name for the customer. This is
driven by customer input. ',
  `Password` text NOT NULL COMMENT 'Customer password. Plans to encrypt
this in the future. ',
  `Customer_Email_Address` text NOT NULL COMMENT 'Email address. Driven by
user input. ',
  `Customer_Phone_Number` text COMMENT 'Customer phone number. ',
  `Customer_Street_Address` text COMMENT 'Shipping address.',
  `Customer_City` text COMMENT 'Customer City (US only for now). ',
  `Customer_State` varchar(3) DEFAULT NULL COMMENT 'Customer State. This
must be a two character stated code (upper-case) to match for shipping
costs. ',
  `Customer_Zip` int(11) DEFAULT NULL COMMENT 'Customer zip code. ',
  PRIMARY KEY (`Customer_ID`)
) ENGINE=InnoDB AUTO_INCREMENT=10026 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
```

```
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-12-04 10:43:57
```

Product

Table: product

Columns:

Product_ID	int(11) PK
Product_Name	text
Product_Description	text
Product_Color	text
Product_Category	text
Gender	text
Image_Filename	text
Product_Size	text
Number_In_Stock	int(11)
Product_Base_Wholesale_Price	double
Product_Retail_Price	double
Product_Option_Description	text
Option_Name_Engraving	int(11)
Option_Birthstone	int(11)
Option_Charm_1	int(11)
Option_Charm_2	int(11)
Option_Charm_3	int(11)
Option_Charm_4	int(11)

```
CREATE DATABASE  IF NOT EXISTS `jewlz` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `jewlz`;
-- MySQL dump 10.13  Distrib 8.0.13, for macos10.14 (x86_64)
--
-- Host: localhost    Database: jewlz
-- ------------------------------------------------------
-- Server version 8.0.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `product`
--

DROP TABLE IF EXISTS `product`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
 SET character_set_client = utf8mb4 ;
CREATE TABLE `product` (
  `Product_ID` int(11) NOT NULL COMMENT 'All product IDs start with '2'. ',
  `Product_Name` text NOT NULL COMMENT 'The English name of the product.
Note: the name spans all sizes and material options for this product class.
',
  `Product_Description` text NOT NULL COMMENT 'The English description of
the product used on the website. ',
  `Product_Color` text NOT NULL COMMENT 'Color or material. ',
  `Product_Category` text NOT NULL COMMENT 'The category of product for
reporting purposes and menu rendering on the website. ',
  `Gender` text NOT NULL COMMENT 'The intended gender for the product, but
in 2018 you never really know...',
  `Image_Filename` text NOT NULL COMMENT 'Filename for the image. Used for
rendering website. ',
```

```
  `Product_Size` text NOT NULL COMMENT 'The size, if any. 'One Size' should
be used if there are no distinct size options for the product. ',
  `Number_In_Stock` int(11) NOT NULL COMMENT 'Current inventory levels. ',
  `Product_Base_Wholesale_Price` double NOT NULL COMMENT 'Based wholesale
price. ',
  `Product_Retail_Price` double NOT NULL COMMENT 'Product selling price
(displayed on the website and used when calculating order total). ',
  `Product_Option_Description` text NOT NULL COMMENT 'Description of
options. ',
  `Option_Name_Engraving` int(11) NOT NULL COMMENT 'Binary signal
indicating this option is available for this product. '1' indicates that
this option IS available.',
  `Option_Birthstone` int(11) NOT NULL COMMENT 'Binary signal indicating
this option is available for this product. '1' indicates that this option
IS available.',
  `Option_Charm_1` int(11) NOT NULL COMMENT 'Binary signal indicating this
option is available for this product. '1' indicates that this option IS
available.',
  `Option_Charm_2` int(11) NOT NULL COMMENT 'Binary signal indicating this
option is available for this product. '1' indicates that this option IS
available.',
  `Option_Charm_3` int(11) NOT NULL COMMENT 'Binary signal indicating this
option is available for this product. '1' indicates that this option IS
available.',
  `Option_Charm_4` int(11) NOT NULL COMMENT 'Binary signal indicating this
option is available for this product. '1' indicates that this option IS
available.',
  PRIMARY KEY (`Product_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-12-04 10:43:58
```

Order

Table: orders

Columns:

Order_ID	int(11) PK
Order_Date	datetime
Order_Status	text
Product_ID	int(11) PK
Order_Total	double
Product_Quantity	int(11)
Customer_Choice_Name_Engraving	text
Customer_Choice_Birthstone_ID	text
Customer_Choice_Charm_ID_1	text
Customer_Choice_Charm_ID_2	text
Customer_Choice_Charm_ID_3	text
Customer_Choice_Charm_ID_4	text
Customer_ID	int(11) PK

```
CREATE DATABASE  IF NOT EXISTS `jewlz` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `jewlz`;
-- MySQL dump 10.13  Distrib 8.0.13, for macos10.14 (x86_64)
--
-- Host: localhost    Database: jewlz
-- ------------------------------------------------------
-- Server version 8.0.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `orders`
--

DROP TABLE IF EXISTS `orders`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
 SET character_set_client = utf8mb4 ;
CREATE TABLE `orders` (
  `Order_ID` int(11) NOT NULL AUTO_INCREMENT COMMENT 'Order ID is used to
group all items (records) for an order. Order ID does not constitute a PK
by itself. ',
  `Order_Date` datetime NOT NULL COMMENT 'The date the order was SUBMITTED.
',
  `Order_Status` text NOT NULL COMMENT 'Current status of the order. 'Open'
means that 'Submit order' has not yet been clicked. ',
  `Product_ID` int(11) NOT NULL COMMENT 'The FK for the product being
purchased. ',
  `Order_Total` double DEFAULT NULL COMMENT 'The order total. Note: this is
duplicated for each product record on an order. ',
  `Product_Quantity` int(11) NOT NULL COMMENT 'The number of exact items on
the order. For example, some customers may order two wedding rings.....',
```

```
  `Customer_Choice_Name_Engraving` text COMMENT 'Engraving based on user
input.',
  `Customer_Choice_Birthstone_ID` text COMMENT 'Birthstone selection based
on user input. ',
  `Customer_Choice_Charm_ID_1` text COMMENT 'Charm selection based on user
input. ',
  `Customer_Choice_Charm_ID_2` text COMMENT 'Charm selection based on user
input. ',
  `Customer_Choice_Charm_ID_3` text COMMENT 'Charm selection based on user
input. ',
  `Customer_Choice_Charm_ID_4` text COMMENT 'Charm selection based on user
input. ',
  `Customer_ID` int(11) NOT NULL COMMENT 'Customer making the purchase. ',
  PRIMARY KEY (`Order_ID`,`Product_ID`,`Customer_ID`),
  KEY `Product_ID_idx` (`Product_ID`),
  KEY `Customer_ID_idx` (`Customer_ID`),
  CONSTRAINT `Customer_ID` FOREIGN KEY (`Customer_ID`) REFERENCES
`customer` (`customer_id`),
  CONSTRAINT `Product_ID` FOREIGN KEY (`Product_ID`) REFERENCES `product`
(`product_id`)
) ENGINE=InnoDB AUTO_INCREMENT=30025 DEFAULT CHARSET=utf8mb4
COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-12-04 10:43:59
```

Shipping Cost

Table: shipping_costs

Columns:

Shipping_Code_ID	int(11) PK
State_Code	varchar(3)
Shipping_Cost	double

```
CREATE DATABASE  IF NOT EXISTS `jewlz` /*!40100 DEFAULT CHARACTER SET
utf8mb4 COLLATE utf8mb4_0900_ai_ci */;
USE `jewlz`;
-- MySQL dump 10.13  Distrib 8.0.13, for macos10.14 (x86_64)
--
-- Host: localhost    Database: jewlz
-- ------------------------------------------------------
-- Server version 8.0.13

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
 SET NAMES utf8 ;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `shipping_costs`
--

DROP TABLE IF EXISTS `shipping_costs`;
/*!40101 SET @saved_cs_client     = @@character_set_client */;
 SET character_set_client = utf8mb4 ;
```

```
CREATE TABLE `shipping_costs` (
  `Shipping_Code_ID` int(11) NOT NULL COMMENT 'Shipping ID. Used a
generated PK because cannot ensure state codes will be unique if we expand
beyond the US. ',
  `State_Code` varchar(3) NOT NULL COMMENT 'State code to match with
customer state. 2 characters, uppercase. ',
  `Shipping_Cost` double NOT NULL COMMENT 'Cost of shipping am order to
this state. ',
  PRIMARY KEY (`Shipping_Code_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
/*!40101 SET character_set_client = @saved_cs_client */;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2018-12-04 10:43:58
```

Database Normalization

This section documents the normalization decisions for the database. This section is intended primarily for future database designers and developers looking to understand the current design of the database. Database normalization is an approach to managing redundancy and data organization within a database. Our normalization choices are made primarily in the service of the web application, which handles most of the business logic needed to create our customer experience. This has some important implications:

- Few Tables: The database is designed to have a small number of relations (6). The reason for this decision is the complicated mapping between the MySQL instance and the Java application via Hibernate. In order to reduce development time and make interfacing simple, it was desirable for us to consolidate data where possible. Database normalization typically mandates a larger number of relations to manage possible redundancy.
- Large Tables: The decision above results in tables with larger than desirable numbers of attributes. Each attribute for all 6 relations is fully dependant on the primary key, but there are opportunities for higher normal forms in future projects.
- Important Exceptions: This section documents areas that break standard normalization rules. We explain our choices and mitigation strategies for future projects.
    - Order Total: In the ‘order’ relation there is a field named ‘Order_Total.’ This field violates first normal because it is redundant for each record for each order. This has the potential to anomalies for improperly specified transactions. In order to address this design flaw we would create a separate table to order totals using the order_ID as a foreign key.
    - Product Metadata: Several fields in the product table are redundant at the ‘Product_Name’ level. For example, all ‘Grandmother’s Bracelets’ have the same metadata (price, description, gender, custom options). This can be mitigated by creating a ‘Product_Metadata’ table and using something like ‘Product_Name’ as a foreign key (though it makes some sense to develop a new numeric code for this purpose).

Java Entity Specification

This section contains the Java representations of the entities. This section is intended for developers looking to work with the web application or communication with the MySQL instance.

Product

```
      public Product(int id, String name, String description, String color,
String category, String gender, String size,
                  int inStock, double priceWholeSale, double priceRetail,
String image, String prodOptDesc, int optEngrave,
                  int optBirthstone, int optCharm1, int optCharm2, int
optCharm3, int optCharm4) {
            super();
            this.id = id;
            this.name = name;
            this.description = description;
            this.color = color;
            this.category = category;
            this.gender = gender;
            this.size = size;
            this.inStock = inStock;
            this.priceWholeSale = priceWholeSale;
            this.priceRetail = priceRetail;
            this.image = image;
            this.prodOptDesc = prodOptDesc;
            this.optEngrave = optEngrave;
            this.optBirthstone = optBirthstone;
            this.optCharm1 = optCharm1;
            this.optCharm2 = optCharm2;
            this.optCharm3 = optCharm3;
            this.optCharm4 = optCharm4;

      }
```

Customer

```
	public Customer(String password, String firstName, String lastName,
String email, String phone, String address,
			String city, String state, String zip) {
		super();
		this.password = password;
		this.firstName = firstName;
		this.lastName = lastName;
		this.email = email;
		this.phone = phone;
		this.address = address;
		this.city = city;
		this.state = state;
		this.zip = zip;
	}
```

Charm

```
	public Charm(int id, String month, String type, Double cost) {
		this.id = id;
		this.type = type;
		this.cost = cost;
```

Birthstone

```
public Birthstone(int id, String month, String type, Double cost) {
		this.id = id;
		this.month = month;
		this.type = type;
		this.cost = cost;
```

Order

```
      public Order(Date orderDate, String orderStatus, int amount, Double
prodRetailPrice,
                  String customerName, String customerAddress, String
customerEmail, String customerPhone,
                  String nameEngraving, String birthstoneID, String
charmId1, String charmId2, String charmId3,
                  String charmId4, Product product, Customer customer) {
            super();

            Date date = new Date();
            this.orderDate = date;
            this.orderStatus = orderStatus;
            this.amount = amount;
            ;
            this.orderTotal = prodRetailPrice;


            this.nameEngraving = nameEngraving;
            this.birthstoneID = birthstoneID;
            this.charmId1 = charmId1;
            this.charmId2 = charmId2;
            this.charmId3 = charmId3;
            this.charmId4 = charmId4;
            this.customerId = customer;
            this.prodId = product;
      }
```

Shipping Location

```
	public ShippingCost(int id, String state, Double cost) {
		this.id = id;
		this.state = state;
		this.cost = cost;
	}
```

Data Access Specification (Hibernate)

The data access logic is the connection between the entity representation in the MySQL and the entity in the Java Model. Embedded HQL queries control translation logic between representations.

Product

```
@Transactional
public class ProductDAOImpl implements ProductDAO {

    @Autowired
    private SessionFactory sessionFactory;

    public Product findProduct(int code) {
        Session session = sessionFactory.getCurrentSession();
        Criteria crit = session.createCriteria(Product.class);
        crit.add(Restrictions.eq("id", code));
        return (Product) crit.uniqueResult();
    }

    public Product findProductBySize(String name, String size, String
color, String gender) {
        Session session = sessionFactory.getCurrentSession();
        Criteria crit = session.createCriteria(Product.class);
        crit.add(Restrictions.eq("name", name));
        crit.add(Restrictions.eq("size", size));
        crit.add(Restrictions.eq("color", color));
        crit.add(Restrictions.eq("gender", gender));
        return (Product) crit.uniqueResult();
    }

    public List<String> allSizes(String name) {
      Session session = sessionFactory.getCurrentSession();
      String sql = "select distinct p.size from " + Product.class.getName()
+ " p where lower(p.name) = :name";
      Query query = session.createQuery(sql);
      query.setParameter("name", name.toLowerCase());
      List<String> rows = query.list();

        return rows;
```

```
    }


    public ProductInfo findProductInfo(int code) {
        Product product = this.findProduct(code);
        if (product == null) {
            return null;
        }
        //return new ProductInfo(product.getId(), product.getName(),
product.getPriceRetail());
        return new ProductInfo(product.getId(), product.getName(),
product.getPriceRetail(), product.getDescription(), product.getImage(),
                product.getCategory(), product.getColor(),
product.getInStock(), product.getOptBirthstone(), product.getOptCharm1(),
//
                product.getOptCharm2(), product.getOptCharm3(),
product.getOptCharm4());
    }

    public void save(ProductInfo productInfo) {
        int code = productInfo.getCode();

        Product product = null;

        boolean isNew = false;
        try {
            product = this.findProduct(code);
        }
        catch(Exception e) {
            System.out.println("Null Pointer in Product DAO");
            System.out.println(e);
        }
        if (product == null) {
            isNew = true;
            product = new Product();
            //product.setCreateDate(new Date());
        }
        product.setId(code);
        product.setName(productInfo.getName());
        product.setPriceRetail(productInfo.getPrice());

        if (productInfo.getFileData() != null) {
```

```
            String image = productInfo.getFileData();
            if (image != null) {
                product.setImage(image);
            }
        }
        if (isNew) {
            this.sessionFactory.getCurrentSession().persist(product);
        }
        // If error in DB, Exceptions will be thrown out immediately
        this.sessionFactory.getCurrentSession().flush();
    }

    public PaginationResult<ProductInfo> queryProducts(int page, int
maxResult, int maxNavigationPage,
            String likeName) {
        String sql = "Select new " + ProductInfo.class.getName() //
                + "(p.id, p.name, p.priceRetail, p.description, p.image,
p.category, p.color, p.inStock) " + "from "//
                + Product.class.getName() + " p ";
        if (likeName != null && likeName.length() > 0) {
            sql += "Where lower(p.name) like :likeName and ";
        }
        else {
            sql+= "Where ";
        }
        sql += "p.id IN (select max(p1.id) from " + Product.class.getName()
                  + " p1 Where p1.inStock = (Select max(p2.inStock) from "
+ Product.class.getName() + " p2 where p1.name = p2.name) "
                             + "group by p1.name, p1.color, p1.inStock,
p1.description, p1.priceRetail)";
        sql += " order by p.priceRetail asc ";
        //
        Session session = sessionFactory.getCurrentSession();

        Query query = session.createQuery(sql);
        if (likeName != null && likeName.length() > 0) {
            query.setParameter("likeName", "%" + likeName.toLowerCase() +
"%");
        }
        return new PaginationResult<ProductInfo>(query, page, maxResult,
maxNavigationPage);
    }
```

```
    public PaginationResult<ProductInfo> queryCategoryProducts(int page,
int maxResult, int maxNavigationPage,
            String likeName, String category) {

        String sql = "Select new " + ProductInfo.class.getName() //
                + "(p.id, p.name, p.priceRetail, p.description, p.image,
p.category, p.color, p.inStock) " + " from "//
                + Product.class.getName() + " p " //
                + " Where lower(p.category) = :category";
            sql += " and p.id IN (select max(p1.id) from " +
Product.class.getName()
                  + " p1 Where p1.inStock = (Select max(p2.inStock) from "
+ Product.class.getName() + " p2 where p1.name = p2.name) "
                              + "group by p1.name, p1.color, p1.inStock,
p1.description, p1.priceRetail)";
            sql += " order by p.priceRetail asc ";
        //
        Session session = sessionFactory.getCurrentSession();

        Query query = session.createQuery(sql);
        if (category != null && category.length() > 0) {
            query.setParameter("category", category.toLowerCase());
        }
        return new PaginationResult<ProductInfo>(query, page, maxResult,
maxNavigationPage);
    }

    public PaginationResult<ProductInfo> queryProducts(int page, int
maxResult, int maxNavigationPage) {
        return queryProducts(page, maxResult, maxNavigationPage, null);
    }

    public PaginationResult<ProductInfo> queryBySubCategoryProducts(int
page, int maxResult, int maxNavigationPage,
            String likeName, String mainCategory, String subCategory) {

      Query query = null;


        String sql = "Select distinct new " + ProductInfo.class.getName()
//
```

```
                + "(p.id, p.name, p.priceRetail, p.description, p.image,
p.category, p.color, p.inStock) " + " from "//
                + Product.class.getName() + " p " //
                + " Where lower(p.category) = :mainCategory";
        if(subCategory.equals("all")) {
            sql += " and p.id IN (select max(p1.id) from " +
Product.class.getName()
                        + " p1 Where p1.inStock = (Select max(p2.inStock)
from " + Product.class.getName() + " p2 where p1.name = p2.name) "
                                    + "group by p1.name, p1.color,
p1.inStock, p1.description, p1.priceRetail)";
            sql += " order by p.priceRetail asc ";
        //
        Session session = sessionFactory.getCurrentSession();

        query = session.createQuery(sql);
        query.setParameter("mainCategory", mainCategory.toLowerCase());

        }
        else {

            sql += " and lower(p.name) = :subCategory";
            sql += " and p.id IN (select max(p1.id) from " +
Product.class.getName()
                        + " p1 Where p1.inStock = (Select max(p2.inStock)
from " + Product.class.getName() + " p2 where p1.name = p2.name) "
                                    + "group by p1.name, p1.color,
p1.inStock, p1.description, p1.priceRetail)";
            sql += " order by p.priceRetail asc ";
            //
            Session session = sessionFactory.getCurrentSession();

            query = session.createQuery(sql);
            query.setParameter("mainCategory", mainCategory.toLowerCase());
            query.setParameter("subCategory", subCategory.toLowerCase());
        }
        return new PaginationResult<ProductInfo>(query, page, maxResult,
maxNavigationPage);
    }
```

```
}
```

Customer

The customer data access path includes logic to handle new customers, shown below.

```
@Transactional
public class CustomerDAOImpl implements CustomerDAO {

    @Autowired
    private SessionFactory sessionFactory;

    public Customer findAccount(String email ) {
        Session session = sessionFactory.getCurrentSession();
        @SuppressWarnings("deprecation")
            Criteria crit = session.createCriteria(Customer.class);
        crit.add(Restrictions.eq("email", email));
        return (Customer) crit.uniqueResult();
    }

      @Override
      public void registerNewUser(CustomerInfo newUser) {

            Customer user = null;
            String email = newUser.getEmail();

        boolean isNew = false;
        if (email != null) {
           user = this.findAccount(email);
        }
        if (user == null) {
            isNew = true;
            user = new Customer();
        }
        user.setEmail(email);
        user.setFirstName(newUser.getFirstName());
        user.setLastName(newUser.getLastName());
        user.setPassword(newUser.getPassword());
        user.setAddress(newUser.getAddress());
        user.setCity(newUser.getCity());
        user.setState(newUser.getState());
        user.setZip(newUser.getZip());
```

```
        user.setPhone(newUser.getPhone());


        if (isNew) {
            this.sessionFactory.getCurrentSession().persist(user);
        }
        // If error in DB, Exceptions will be thrown out immediately
        this.sessionFactory.getCurrentSession().flush();
    }

    @Override
    public Customer findAccountWithPass(String email, String password) {
            Session session = sessionFactory.getCurrentSession();
      @SuppressWarnings("deprecation")
            Criteria crit = session.createCriteria(Customer.class);
      crit.add(Restrictions.eq("email", email));
      crit.add(Restrictions.eq("password", password));
      return (Customer) crit.uniqueResult();
    }

    @Override
    public int findCustomerId(String email, String password)
    {
            Session session = sessionFactory.getCurrentSession();
      @SuppressWarnings("deprecation")
            Criteria crit = session.createCriteria(Customer.class);
      crit.add(Restrictions.eq("email", email));
      crit.add(Restrictions.eq("password", password));
      Customer thisCustomer = (Customer) crit.uniqueResult();
      return(thisCustomer.getId());
    }

    @Override
    public Customer lookUpCustomerWithID(int id) {
            Session session = sessionFactory.getCurrentSession();
      @SuppressWarnings("deprecation")
            Criteria crit = session.createCriteria(Customer.class);
      crit.add(Restrictions.eq("id", id));
      return((Customer) crit.uniqueResult());
    }

    @Override
```

```
      public List<Customer> findAll() {
            Session session = sessionFactory.getCurrentSession();
      String sql = "select c from " + Customer.class.getName();
      sql += " c";
      Query query = session.createQuery(sql);
      List<Customer> rows = query.list();

        return rows;
      }

}
```

Charm

```
@Transactional
public class CharmDAOImpl implements CharmDAO {

      @Autowired
    private SessionFactory sessionFactory;

    public List<Charm> getCharmList() {
      Session session = sessionFactory.getCurrentSession();
      String sql = "select c from " + Charm.class.getName();
      sql += " c";
      Query query = session.createQuery(sql);
      List<Charm> rows = query.list();

        return rows;
    }

    public Charm findOne(int id) {
      Session session = sessionFactory.getCurrentSession();
      String sql = "select b from " + Charm.class.getName();
      sql += " b where b.id = :id";
      Query query = session.createQuery(sql);
      query.setParameter("id", id);
      Charm charm = (Charm) query.uniqueResult();

        return charm;
    }
}
```

Birthstone

```
@Transactional
public class BirthstoneDAOImpl implements BirthstoneDAO {

      @Autowired
    private SessionFactory sessionFactory;

      public List<Birthstone> retrieveAll() {
      Session session = sessionFactory.getCurrentSession();
      String sql = "select b from " + Birthstone.class.getName();
      sql += " b";
      Query query = session.createQuery(sql);
      List<Birthstone> rows = query.list();

        return rows;
          }

      public Birthstone findOne(int id) {
      Session session = sessionFactory.getCurrentSession();
      String sql = "select b from " + Birthstone.class.getName();
      sql += " b where b.id = :id";
      Query query = session.createQuery(sql);
      query.setParameter("id", id);
      Birthstone birthstone = (Birthstone) query.uniqueResult();

        return birthstone;
          }
}
```

Order

The Order access path contains references to foreign keys (customer, product, and shipping).

```
@Transactional
public class OrderDAOImpl implements OrderDAO {

    @Autowired
    private SessionFactory sessionFactory;

    @Autowired
    private ProductDAO productDAO;

    @Autowired
    private CustomerDAO customerDAO;

    @Autowired
    private ShippingCostsDAO shippingCostDAO;

    private int getMaxOrderNum() {
        String sql = "Select max(o.id) from " + Order.class.getName() + " o
";
        Session session = sessionFactory.getCurrentSession();
        Query query = session.createQuery(sql);
        Integer value = (Integer) query.uniqueResult();
        if (value == null) {
            return 0;
        }
        return value;
    }

    public void saveOrder(CartInfo cartInfo) {
        Session session = sessionFactory.getCurrentSession();
        ShippingCost shippingCost = null;

        Calendar today = Calendar.getInstance();
            today.set(Calendar.HOUR_OF_DAY, 0);


        int orderNum = this.getMaxOrderNum() + 1;
        OrderInfo order = new OrderInfo();
        order.setOrderNum(orderNum);
```

```
        CustomerInfo customerInfo = cartInfo.getCustomerInfo();

        Customer customer =
customerDAO.findAccount(customerInfo.getEmail());

        order.setCustomer(customer);

        shippingCost = shippingCostDAO.findByState(customer.getState());


        List<CartLineInfo> lines = cartInfo.getCartLines();

        for (CartLineInfo line : lines) {

            String size = line.getProductInfo().getSize();
            String name = line.getProductInfo().getName();
            String color = line.getProductInfo().getColor();
            String gender = line.getProductInfo().getGender();
            Product product = this.productDAO.findProductBySize(name, size,
color, gender);

            order.setProduct(product);
            Order thisItem = new Order(order);


            thisItem.setAmount(line.getQuantity());
            product.setInStock(product.getInStock() - line.getQuantity());

thisItem.setOrderTotal(cartInfo.getFinalizedTotal(shippingCost.getCost()));


            if(product.getOptEngrave() == 1) {

thisItem.setNameEngraving(line.getProductInfo().getEngraving());
            }
            if(product.hasBirthstoneOpt()) {

thisItem.setBirthstoneID(String.valueOf(line.getProductInfo().getBirthstone
Selected().getId()));
            }
            if(product.hasCharmOpt()) {
```

```
thisItem.setCharmId1(String.valueOf(line.getProductInfo().getCharm1()));

thisItem.setCharmId2(String.valueOf(line.getProductInfo().getCharm2()));

thisItem.setCharmId3(String.valueOf(line.getProductInfo().getCharm3()));

thisItem.setCharmId4(String.valueOf(line.getProductInfo().getCharm4()));
            }

            session.saveOrUpdate(product);

            session.persist(thisItem);



        }

        // Set OrderNum for report.

        cartInfo.setOrderNum(orderNum);
    }

    // @page = 1, 2, ...
    public PaginationResult<OrderInfo> listOrderInfo(int page, int
maxResult, int maxNavigationPage) {
        String sql = "Select new " + OrderInfo.class.getName()//
                + "(ord.id, ord.orderDate, ord.orderNum, ord.amount, "
                + " ord.customerName, ord.customerAddress,
ord.customerEmail, ord.customerPhone) " + " from "
                + Order.class.getName() + " ord "//
                + " order by ord.orderNum desc";
        Session session = this.sessionFactory.getCurrentSession();

        Query query = session.createQuery(sql);

        return new PaginationResult<OrderInfo>(query, page, maxResult,
maxNavigationPage);
```

```
    }

    public Order findSingleOrder(int orderId) {
        Session session = sessionFactory.getCurrentSession();
        Criteria crit = session.createCriteria(Order.class);
        crit.add(Restrictions.eq("Order_ID", orderId));
        return (Order) crit.uniqueResult();
    }

    public Order findOrderForCustomer(int orderId, int customerId) {
      Session session = sessionFactory.getCurrentSession();
        Criteria crit = session.createCriteria(Order.class);
        crit.add(Restrictions.eq("Order_ID", orderId));
        crit.add(Restrictions.eq("Customer_ID", customerId));
        return (Order) crit.uniqueResult();
    }
 /*
    public OrderInfo getOrderInfo(int orderId) {
        Order order = this.findSingleOrder(orderId);
        if (order == null) {
            return null;
        }
        Customer customer =
customerDAO.lookUpCustomerWithID(order.getCustomerId().getId());
        return new OrderInfo(order.getId(), order.getOrderDate(), //
                order.getId(), order.getAmount(), customer.getFirstName(),
//
                customer.getLastName(), customer.getAddress(),
customer.getCity(), customer.getState(), //
                customer.getZip(), customer.getEmail(),
customer.getPhone());
    }
    */

    //
    @SuppressWarnings("deprecation")
      public List<Order> listAllOrderItemsForSingleOrder(int orderId) {
        String sql = "Select new " + Order.class.getName() //
                + "(d.id, d.product.id, d.customer.id) "//
                + " from " + Order.class.getName() + " d "//
                + " where d.order.id = :orderId ";
```

```
        Session session = this.sessionFactory.getCurrentSession();

        Query query = session.createQuery(sql);

        query.setParameter("orderId", orderId);

        return query.list();
    }


      public List<Product> listAllOrderItemsForAllOrders() {

              String sql = "select d.prodId "//
                      + "from " + Order.class.getName() + " d group by "
                              + "d.prodId order by count(d.prodId) desc";

              Session session = this.sessionFactory.getCurrentSession();

              Query query = session.createQuery(sql);


              return (List<Product>) query.list();
    }



}
```

Shipping Location

```
    @Autowired
  private SessionFactory sessionFactory;

  public List<ShippingCost> getStateList() {
    Session session = sessionFactory.getCurrentSession();
    String sql = "select s from " + ShippingCost.class.getName();
    sql += " s";
    Query query = session.createQuery(sql);
    List<ShippingCost> rows = query.list();

      return rows;
  }

  public ShippingCost findOne(int id) {
    Session session = sessionFactory.getCurrentSession();
    String sql = "select s from " + ShippingCost.class.getName();
    sql += " s where s.id = :id";
    Query query = session.createQuery(sql);
    query.setParameter("id", id);
    ShippingCost shippingCost = (ShippingCost) query.uniqueResult();

      return shippingCost;
  }

    @Override
    public ShippingCost findByState(String state) {
          Session session = sessionFactory.getCurrentSession();
    String sql = "select s from " + ShippingCost.class.getName();
    sql += " s where s.state = :state";
    Query query = session.createQuery(sql);
    query.setParameter("state", state);
    ShippingCost shippingCost = (ShippingCost) query.uniqueResult();

      return shippingCost;
    }

}
```

Report Specification

The main “Reports” page shows the five reports that are supported by our application. Users can generate a specific report by filling in all applicable parameter prompts and clicking on the “Generate” button, which submits an HTML request for the “Report Rendering” page. This request contains form data that determines which report will be rendered on that page, along with the parameter values (if any) for that particular report. Users can navigate back to this main “Reports” page whenever they would like to enter new values and generate a new report. The five supported reports and their prompt details are listed below:

- Monthly Sales Report - Dropdown list prompts for “First Month” and “Last Month” to determine the date range for the report. These dropdown lists are dynamically populated using the results of a SQL query that returns each month an order was completed from our database. This is done to prevent invalid input by the user and eliminate the need for input validation. However, no validation is done to ensure that “First Month” comes before “Last Month” chronologically. In the case that “First Month” is after “Last Month”, no results are returned.

- Yearly Sales Report - Dropdown list prompts for “First Year” and “Last Year” to determine the date range for the report. These dropdown lists are also dynamically populated using the results of a SQL query that returns each month an order was completed from our database. This is done to prevent invalid input by the user. However, no validation is done to ensure that “First Year” comes before “Last Year” chronologically. In the case that “First Year” is after “Last Year”, no results are returned.

- Inventory Levels and Costs - No prompts are available for this report, as the results give an exhaustive list of all inventory. Total liquidation cost of inventory should be given on the last row of the report output table.

- Customer List - This report also does not contain any prompts. A list of all customers in the database will always be returned.

- Mailing Labels - A text prompt for the customer ID is used to select which customer the mailing labels will be generated for. The customer ID must be entered exactly as it appears on the “Customer List” report. If an invalid Customer ID is entered, no results will be returned.

The “Report Rendering” page actually shows the output of each report, formatted as a table. This page uses a form variable to determine which of the 5 report outputs are rendered. It uses additional form variables to handle any user inputs. Instead of using the “hibernate” method to run HQL queries, we make use of the sql:query and sql:param jsp tags to connect to the

database and run SQL straight from the jsp template code. We also use jsp foreach and if constructs to help iterate over results and choose the desired output. This was done because it eliminates the need to call any unnecessary Java functionality and all of the code for these reports can be modified in one file (“reportRendering.jsp”).

Monthly Sales Report SQL

```
PROMPT: SELECT DISTINCT DATE_FORMAT(Order_Date, '%M %Y') AS Order_Month,
      DATE_FORMAT(Order_Date, '%Y%m') num_month
      from orders
      order by num_month
```

```
REPORT: Profit/Revenue Section
SELECT DATE_FORMAT(Order_Date, '%M %Y') AS Order_Month,
         DATE_FORMAT(Order_Date, '%Y%m') num_month,
         FORMAT(ROUND(SUM(Order_Total), 2), 2) AS Price,
         FORMAT(ROUND(SUM(Order_Total - (Product_Quantity *
Product_Base_Wholesale_Price)), 2), 2) AS Profit
         FROM orders JOIN product ON orders.Product_ID = product.Product_ID
         WHERE Order_Date
         BETWEEN STR_TO_DATE( CONCAT('01 ', ?), '%d %M %Y') AND
LAST_DAY(STR_TO_DATE( CONCAT('01 ', ?), '%d %M %Y'))
         GROUP BY DATE_FORMAT(Order_Date, '%M %Y'),
         DATE_FORMAT(Order_Date, '%Y%m')
         ORDER BY num_month

REPORT: Items/Categories Section
SELECT DATE_FORMAT(Order_Date, '%M %Y') AS Order_Month,
         DATE_FORMAT(Order_Date, '%Y%m') num_month,
         Product_Category,
         SUM(Product_Quantity) AS Quantity_Sold
         FROM orders JOIN product ON orders.Product_ID = product.Product_ID
         WHERE Order_Date
         BETWEEN STR_TO_DATE( CONCAT('01 ', ?), '%d %M %Y') AND
LAST_DAY(STR_TO_DATE( CONCAT('01 ', ?), '%d %M %Y'))
         GROUP BY DATE_FORMAT(Order_Date, '%M %Y'),
         DATE_FORMAT(Order_Date, '%Y%m'),
         Product_Category
         ORDER BY num_month, Product_Category
```

Annual Sales Report SQL

```
PROMPT: SELECT DISTINCT DATE_FORMAT(Order_Date, '%Y') AS Order_Year
      from orders
      order by Order_Year
```

```
REPORT: Profit/Revenue Section
SELECT YEAR(Order_Date) AS Order_Year,
         FORMAT(ROUND(SUM(Order_Total), 2), 2) AS Price,
         FORMAT(ROUND(SUM(Order_Total - (Product_Quantity *
Product_Base_Wholesale_Price)), 2), 2) AS Profit
         from orders JOIN product ON orders.Product_ID = product.Product_ID
         where YEAR(Order_Date) BETWEEN ? AND ?
         group by YEAR(Order_Date)
         order by YEAR(Order_Date)

REPORT: Items/Categories Section
SELECT YEAR(Order_Date) AS Order_Year,
         Product_Category,
         SUM(Product_Quantity) AS Quantity_Sold
         from orders JOIN product ON orders.Product_ID = product.Product_ID
         where YEAR(Order_Date) BETWEEN ? AND ?
         group by YEAR(Order_Date),
         Product_Category
         order by YEAR(Order_Date), Product_Category
```

Inventory Report SQL

```
SELECT Product_ID, Product_Name, Product_Color, Product_Size,
Number_In_Stock, FORMAT(Product_Base_Wholesale_Price, 2) AS
Product_Base_Wholesale_Price,
          FORMAT(Product_Base_Wholesale_Price * Number_In_Stock, 2) AS
Inventory_Cost
from product
```

```
Report Total: SELECT FORMAT(SUM(Product_Base_Wholesale_Price *
Number_In_Stock), 2) AS Total_IC
from product
```

Customer List Report SQL

```
SELECT * from customer
```

Mailing Label Report SQL

```
SELECT CONCAT(Customer_First_Name, ' ', Customer_Last_Name) AS
Customer_Full_Name, Customer_Street_Address,
         CONCAT(Customer_City, ', ', Customer_State, ' ', Customer_Zip) AS
Customer_CSZ from customer where Customer_ID = ?
```