BGC Engineering Inventory Store Project Requirements and Specification Document

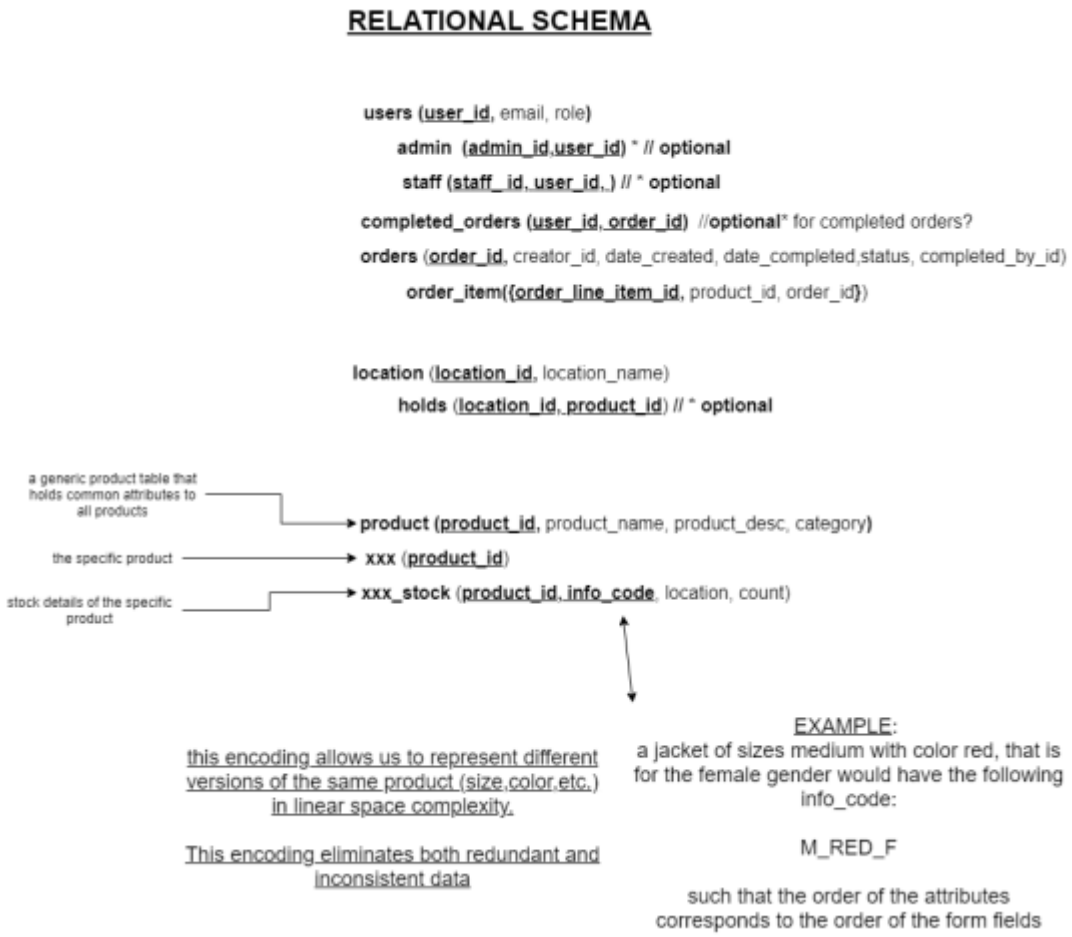
2021/07/23
Iteration 2

Abstract:
BGC internal website designed for staff members to request promotional items and gear for both personal and client benefits. An E-commerce website with two user roles views: Staff and Administrator. The 2 main features are for staff members to be able to make requests for available items, and for administrators to fulfill those requests. There are also many other individual features such as generating reports.

Customer:
The new website will provide a better user experience for BGC staff members and normal users. On the staff (member) side, the website will be easier to navigate and use. On the administrative (admin team) side, users will no longer have to manually correct invalid entries in the database with new valid server-logic, reducing human errors.

Competitive Analysis:
This project consists of a complete revamping of the existing site. Front-end technologies such as HTML/CSS/JavaScript will be used to provide a better user experience. The backend will use Node.js as a server-side language with a PostgreSQL database. The new web application will provide the most integral functions of the existing inventory store along with additional features if time permits.

Initial Database Design:

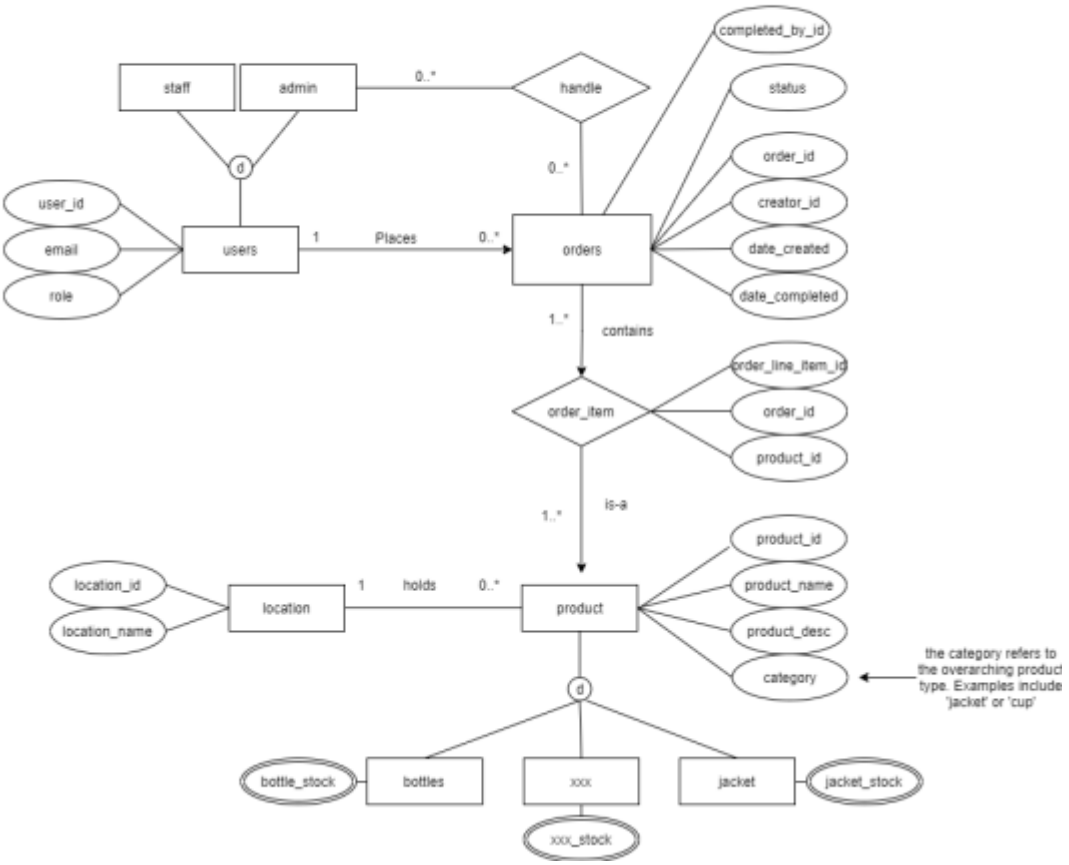


The figure above represents a tentative relational schema for our backend. This will be edited and revised alongside pending review from the employer who will provide feedback ASAP.

Revisions were made to the database schema to the following for Iteration 2:

Users (user_id, username, email, role)
Locations (location_name)
Products (product_id, product_name, product_desc, category, value)
Product_details (product_id, sku_id, size, gender, color, product_location, product_count, product_img)
Order (order_id, requester_id, fulfiller_id, status, request_type, date_created, date_completed)
Order_lines (order_id, sku_id, order_count)

Entity-Relationship Diagram



The Entity-Relationship Diagram above demonstrates the relationships between the tables and the attributes/columns of each table. This will inevitably need to be updated and optimized as implementation progresses.

Epics aimed for Iteration 2:

1. Ability to login using the Azure Active Directory API **(Iteration 1)**
2. Ability to view items on a shopping page (client) **(Iteration 1)**
3. Ability to insert an item and its variations inside the database **(Iteration 2)**
4. Ability to create a cart if no cart (server side), and add item(s) to the cart **(Iteration 2)**
5. Ability to checkout an order to be handled (to the checkout page) **(Iteration 2)**

User Stories:
For this project two different types of users are going to be involved: staff members and administrators.

Due to the lack of access and authority to interact with BGC's tenant/organization on Microsoft Azure, we cannot enforce that all users entering the application must be part of the BGCEngineering domain. (We also don't have our own domain).

Our project is currently only run locally, and will be imported into the BGC's staff intranet when completed. Currently, the redirect to Microsoft's login API is triggered when a user makes a request to a local server running on port 3000, and any personal, school, or work email with correct credentials entered will lead them to the landing page provided the database has an entry of their email.

User Story 1:

Iteration #: 1

Name: Redirect to Microsoft Login

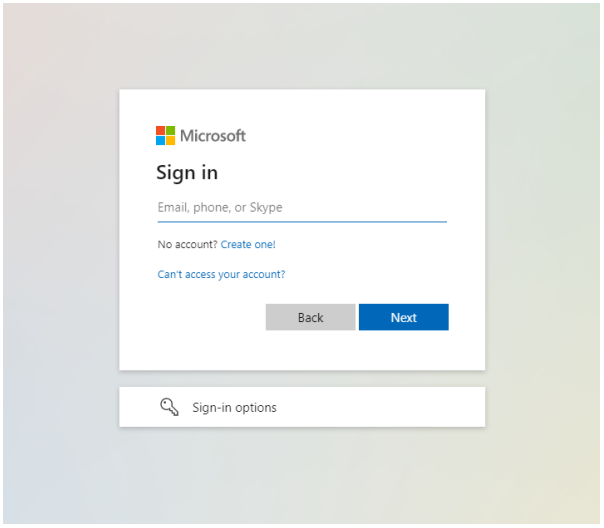
Actors: Any user visiting the application's URL

Triggers/Preconditions: The user knows the application's URL (currently set to localhost:3000) and visits the page using their browser.

Actions/Postconditions:The user is redirected to Microsoft's Login page

Acceptance tests:

Success: Visiting the application URL using different browsers such as: Chrome, Mozilla Firefox, Edge.



User Story 2:

Iteration #: 1

Name: Staff login

Actors: Members of the company (Currently testing anyone with personal/work email as we don't have a domain)

Triggers/Preconditions: The user has been redirected to Microsoft's Login page, and has entered a valid email/password combination. The user's email also needs to be in the users table.

Actions/Postconditions: The user is redirected back to our application after entering valid credentials. Information regarding the user (username, email) is displayed on the server side to ensure the user's data can be pulled based on their login.

Acceptance tests:

Success: Email: Outlook email with Password, Email: Gmail with Password, SFU email with password

Fails: When the user is not present in the database despite correct credentials

User Story 3:

Iteration #: 1

Name: Admin login

Actors: Members of the company (Currently anyone with personal/work email)

Triggers/Preconditions: The user has been redirected to Microsoft's Login page, and has entered a valid email/password combination. The user's email is also stored in a *users* table, with a column 'role' with the value 'administrator'. (There is no signup option, administrators will have to be manually entered in the database by someone with database privileges)

Actions/Postconditions: The user is redirected back to our application after entering valid credentials. Information regarding the user (username, email) is displayed on the server side to ensure the user's data can be pulled based on their login.

Acceptance tests:

Email: Logged in using Outlook email, with email in the database table

Email: Logged in using Gmail, with email in the database table

User Story 4:

Iteration #: 1

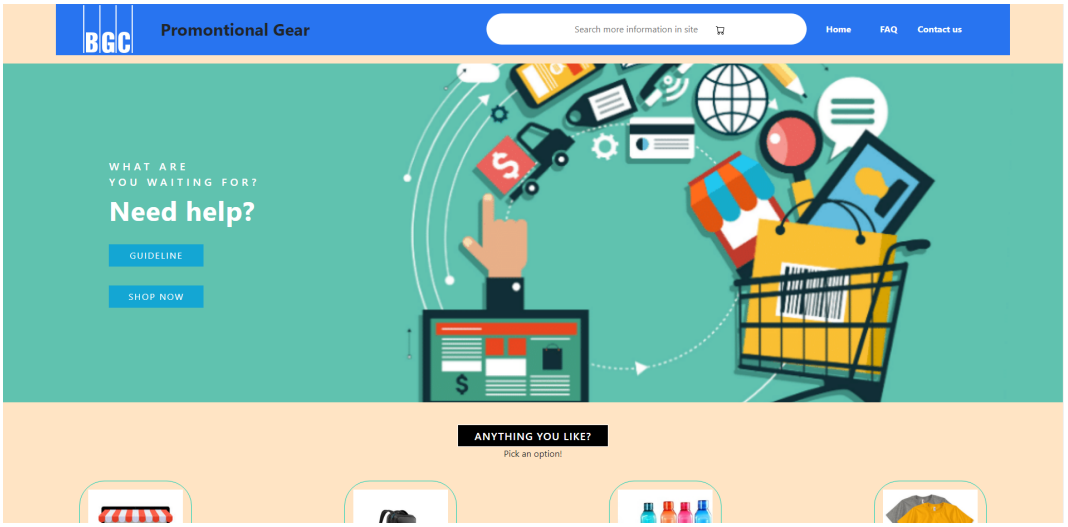
Name: Staff Landing Page

Actors: Staff (Currently anyone with personal/work email)

Triggers/Preconditions: The user has been redirected to Microsoft's Login page, and has entered a valid email/password combination. The user's email is also stored in a *users* table, with a column 'role' with the value staff. (There is no signup option, administrators and/or staff will have to be manually entered in the database by someone with database privileges)

Actions/Postconditions: Users are redirected to the landing page after they log in. Users are presented with a navbar, banner, and several product categories that will route them to a page that will display products of that category Within the navbar, there are some menu items that lead users to different information pages.

Acceptance tests: Logged in using an email with staff role in the database. Login page is successfully rendered



User Story 5:

Iteration #: 1

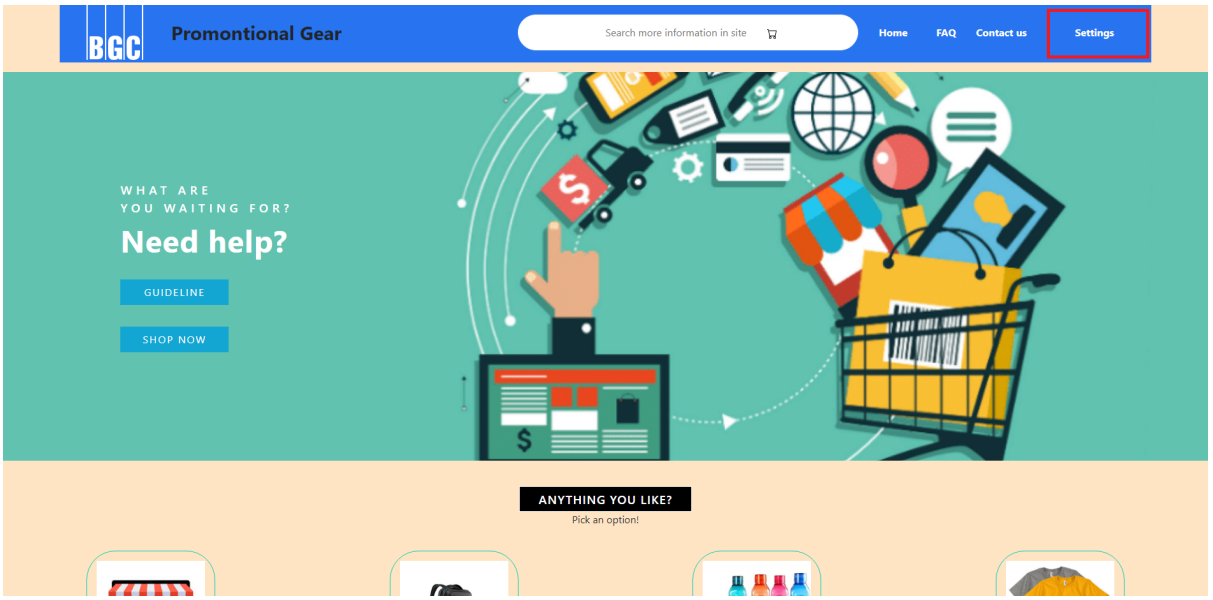
Name: Administrator Landing Page

Actors: Administrator (email with role administrator)

Preconditions: The user has valid credentials, logs in, and is redirected back to our application's home page. The user's email is also in a users table with administrator role

Actions/Postconditions: Users could be redirected to the landing page after they log in. Users could see some product categories briefly. Also, there will be some buttons or headings that could lead users to a shopping page, guideline, and other information. The administrator page will have an additional menu item called "Settings" that will redirect them to a **page with additional controls** (to be implemented)

Acceptance tests: Logged in using an email with administrator role in the database. Success : Login page is successfully rendered with an additional "Settings" option in the navbar.



User Story 6:

Iteration #: 1

Name: Viewing products from a specific category

Actors: Administrator and staff

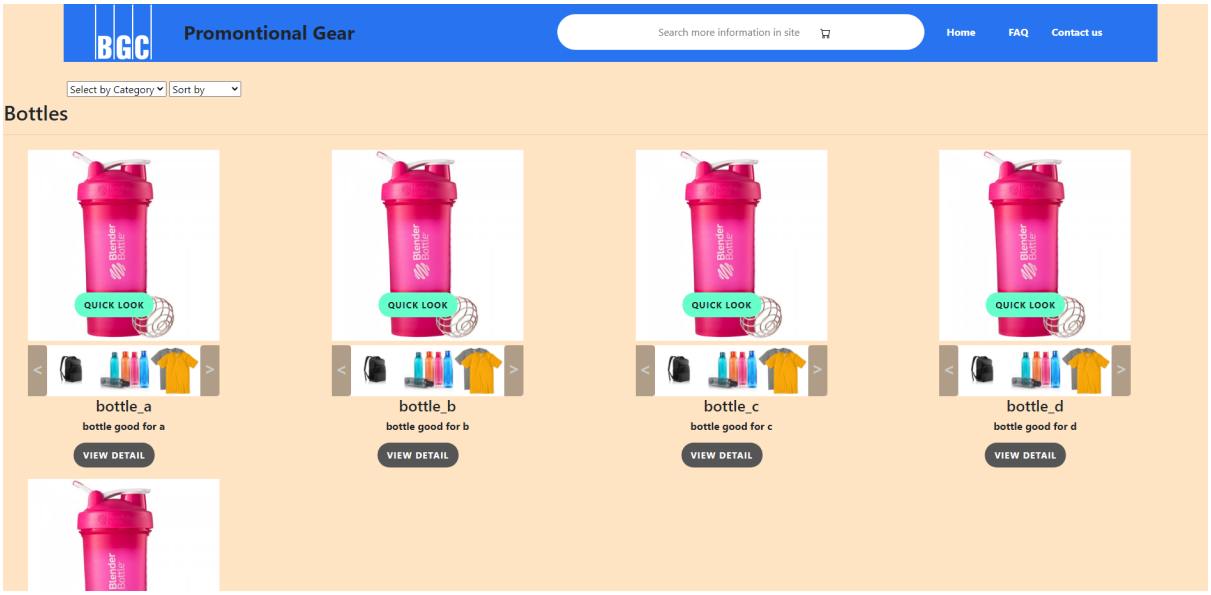
Triggers/Preconditions: The user has valid credentials, already logged in and redirected to the landing page. There are bottles product information present in the database

Actions/Postconditions: User selects on one of the three categories (more will be added) and is redirected to a page where all the bottles in the database are listed. Each item can be viewed close up in a small popup.

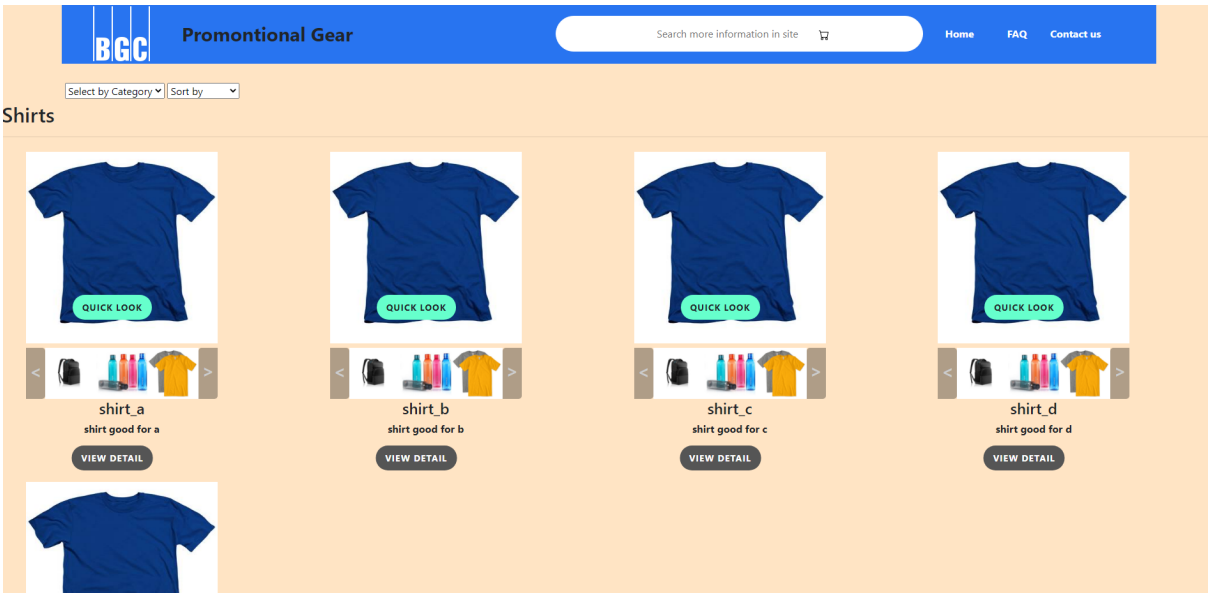
Acceptance test:

Success:

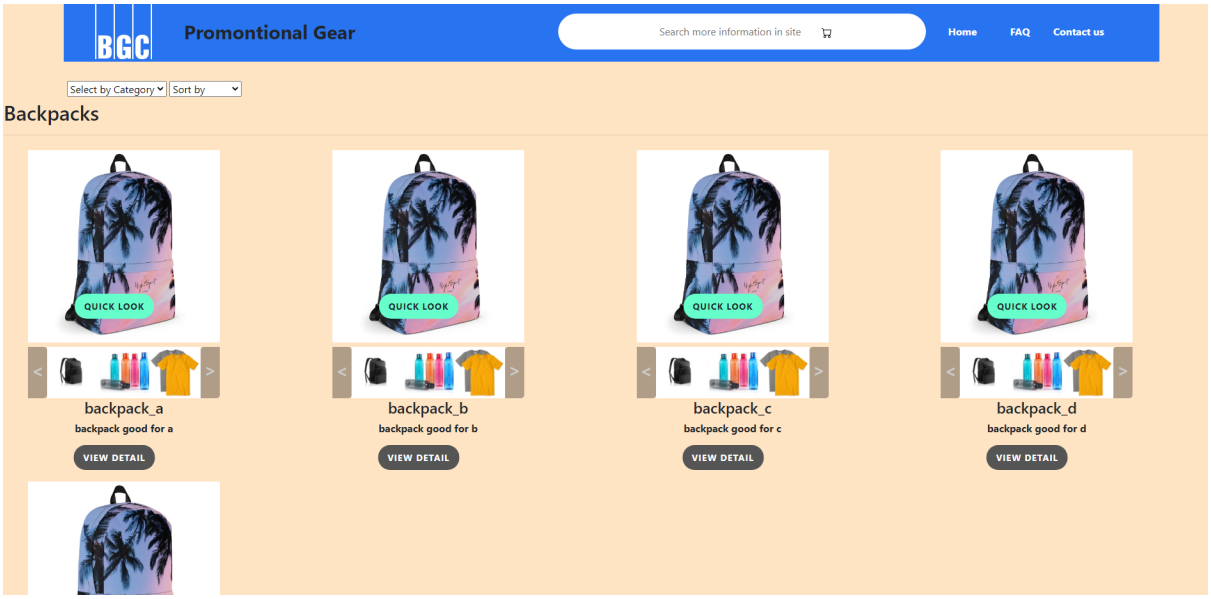
Selected the “Bottles” category, and a list of products within the Bottle category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all bottles.



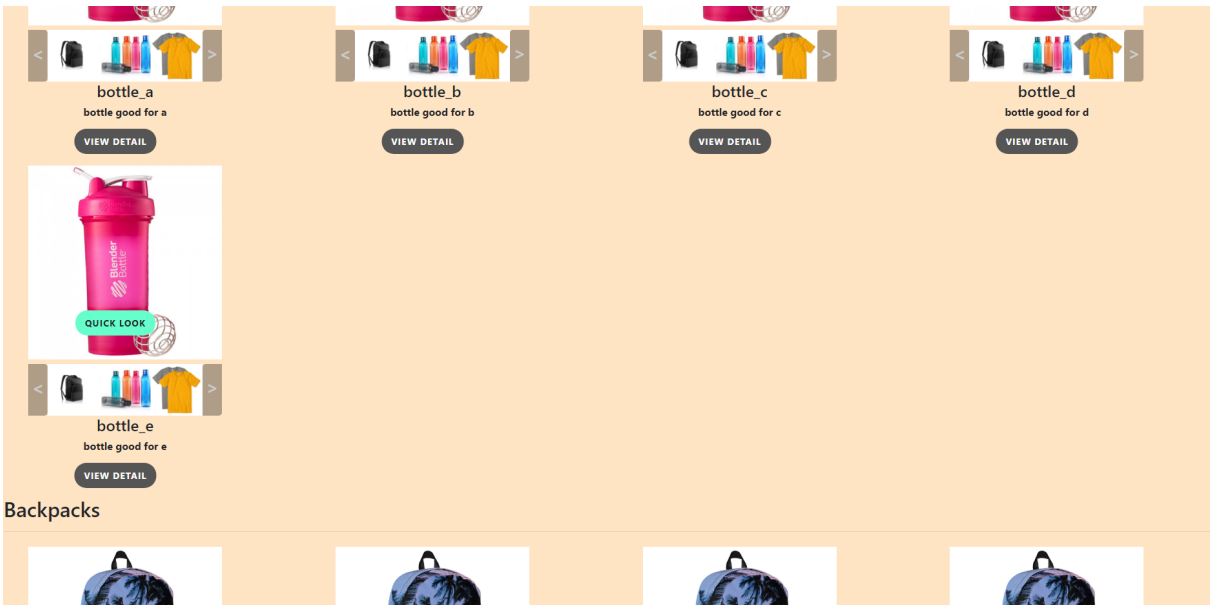
Selected the “Shirts” category, and a list of products within the Shirts category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all shirts.



Selected the “Backpacks” category, and a list of products within the Backpacks category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all backpacks.



Selected the “All Categories” image and all products of different categories are displayed on the page.



Users Stories for Iteration 2

User Story 7:

Iteration #: 2

Name: Filter products on an All Products page

Triggers/Preconditions: Products are in the database

Actions/Postconditions: User is able to click on filter options at the top of a shopping page and filter by items

Acceptance test: User is able to filter by Category, size etc.

Success cases:

Filtering products by bottles, backpacks and shirts

User Story 8:

Iteration #: 2

Name: Specific detail page for a specific type of product

Triggers/Preconditions: Product is in the database, user is logged in

Actions/Postconditions: User selects “View detail” of a product on the “All products” page. The user is redirected to a page with general/specific information about the product.

Acceptance test: Inserting an Item with category bottle.

Success cases:

Users are able to view detailed information of a product when clicking on “View Detail”.

Fail cases:

View detail functionality is missing in other pages such as a specific category page.

User Story 9:

Iteration #:2

Name: Uploading product of a particular category, with attributes such as Name, Description, and Value

Triggers/Preconditions: User is an administrator and clicks on the “Settings” header

Actions/Postconditions: User enters in general product information such as name, description, value.

Acceptance test: To be tied in with uploading specific types of the product in the user story below

Success cases:

Any text within the database character limit for that attribute. Description textbox does not contain apostrophes or quotes.

Fail cases:

When an apostrophe is entered in the description textbox, it needs to be preprocessed before inserting into database (need to prepend extra) or insertion will result in an error

User Story 10:

Iteration #:2

Name: Uploading different types of a product

Triggers/Preconditions: User is an administrator and clicks on the “Settings” header

Actions/Postconditions: User enters in specific product attributes such as SKU_ID, color, gender, size, location, count, image path

Acceptance test: User is able to select a category, enter in general product information, and specific types of that product. Users can add rows and delete rows.

Success cases:

User is able to insert a general name, description, and value for the product. Users are also able to add new rows for a specific type of product. Finally, users can upload up to 10 images to the server file system.

Fail case:

When an apostrophe is entered in the description textbox, it needs to be preprocessed before inserting into the database (need to prepend extra) or insertion will result in an error.

Improvements: Make categories field a select option instead of simply text box to restrict category types.

User Story 11:

Iteration #: 2

Name: Adding products to order

Triggers/Preconditions: User navigates to all category, upon selection of the product, pressing add to the cart button will add the product to the cart, which is in fact an order upon submis

Actions/Postconditions: User is able to add an item to the order

Acceptance test:User able to see the new added product in the cart. Furthermore, the user is able to modify the quantity of the product.

Success cases:

Adding any product in the “All Categories” page was successfully added to the shopping cart. If the user had no

Fail cases:

Unable to add products that are not displayed in “All Categories” page (no functionality)

User Story 12:

Iteration #: 2

Name: Review order and insert final details

Triggers/Preconditions: User is logged in and has added products to the order

Actions/Postconditions: User is brought to a page where the items are listed alongside a Terms and Conditions segment. User email is automatically inputted in an editable input box, with the user selecting a location.

Acceptance test: Order status is changed to Pending, reflected in the database with time of submission.

Success cases:Currently our iteration 2 stops with press the submit button. In our future iteration, we expect a success case to be an update in the admin order table as well as a notification of the order sent to the admin.

Fail cases: currently, no fail cases have been reported as the implementation is a work in progress.

User Stories for Iteration 3

User Story 13:

Iteration #: 3

Name: Complete an Order

Triggers/Preconditions: User is an administrator and there is an outstanding order that is pending

Actions/Postconditions: Administrator is able to fulfill the order and change its status.

Acceptance test: Order status is completed, reflected in database. Corresponding products in the order have their quantity decremented based on the amount ordered.

User Story 14:

Iteration #: 3

Name: Complete an Order

Triggers/Preconditions: User is an administrator and there is an outstanding order that is pending

Actions/Postconditions: Administrator is able to fulfill the order and change its status.

Acceptance test: Order status is completed, reflected in database. Corresponding products in the order have their quantity decremented based on the amount ordered.

User Story 15:

Iteration #: 3

Name: Update an Order

Triggers/Preconditions: User is an administrator and there is an outstanding order that is pending

Actions/Postconditions: Administrator is able to cancel or wait list the order and change its status according to the inventory .

Acceptance test: Order status is updated, reflected in database (state, Inventory quantity). Corresponding products in the order have their quantity decremented based on the amount ordered.

User Story 16:

Iteration #: 3

Name: Track employees benefit

Triggers/Preconditions: User is an administrator and he navigates to settings page

Actions/Postconditions: Administrator is able to track each user's past orders and benefits via a table.

Acceptance test: ability to view the table of users and their total benefit.

User Story 17:

Iteration #: 3

Name: Email notification for low stock

Triggers/Preconditions: A product's stock has reached a certain threshold (<= 5)

Actions/Postconditions: an email is sent from mailvelope to an admin team email, notifying that the product iis on low stock.

Acceptance test: after a product's count is decremented and hits the threshold, admin receives email with product's information.

User Story 18:

Iteration #: 3

Name: Out of stock notification (on views)

Triggers/Preconditions: The Inventory quantity of each item in a different category is less than or equal to 0.

Actions/Postconditions: That item will show the "out of stock", and add to cart button change to not available.

Acceptance test: Check that when each inventory number is 0, the front end needs to display the word "out of stock" and some buttons become unselected.

User Story 19:

Iteration #: 3

Name: Modify specific product information

Triggers/Preconditions: User is an administrator

Actions/Postconditions: User is able to open a page that lists all items and is able to modify and update the products general information, as well as specific information pertaining to a type of product

Acceptance test: Values are successfully modified in the database, and refreshing the page will result in updated values.

How we improved our process from iteration 1 to iteration 2

More meetings were held. User stories were broken down and developed by team members. Working with node and github is a learning experience. Improvements from last iteration using the node and github version control tool. Incorporated members who handled front end in Iteration 1 to familiarize and contribute to back-end server-side logic and code.

Velocity Measurements *(v = avg number story points / week)*

1 → straightforward

2 → medium sized

3 → complex story

5 → EPIC

8 → EPIC

Using the rubric above for story points, our team has evaluated our average productivity for the past 2 weeks, given our results.

Week Number	Story Point Velocity	Explanation
Week 1	$2 + 3 = 5sp$ $5sp \,/\, 1\,week = 5sp/week$	⇒ we finalized our database schema; in order to complete this task, we had to integrate the sku_id attribute as well as discover the most memory-efficient way to track our products and orders = **3sp** ⇒ most of our meetings, time and effort has been dedicated towards understanding user stories and researching the development tools(node mainly) that will allow us to achieve such user stories. Achieved pseudocode that facilitated implementation phase = **2sp**
Week 2	$2 + 2 + 5 + 8 = 17sp$ $17sp \,/\, 1\,week = 17sp/week$	⇒ all the server-side logic + database logic regarding the shopping cart functionalities and checkout, required large amounts of time and resources.

		Pair programming was applied for risk-mitigation = **8sp** ⇒ interactive front-end views were officialized in week 2. These file commits allowed our backend developers to implement the necessary shopping cart functionalities = **5sp** ⇒ A product 'details' view that furthers user experience, was created = **2sp** ⇒ rework for our database schema was done for accuracy purposes. We wanted to ensure efficiency and modularity. = **2sp**
Week 1 & 2	$5 + 17 = 22sp$ $22sp \,/\, 2\,week = 11sp/week$	Overall, we were successful in achieving a reasonable velocity of **11 story points per week**. Although there was room for improvement during the implementation phase - such as a need for clearer communication in terms of keeping track of change - we were extremely successful in overcoming hurdles, and working together in an efficient, orderly fashion.

What we need to improve on for iteration 3

- *Write tests using a testing framework*
- *More tests*
- *Increase User Story output significantly*
- *Merging version on github coherently(still quite a bit of inconsistency of branch versions vs main).*
- *Improve our communication to allow more efficient progress.*
- *Improve User interface and styling consistency*
- *In regards to iteration 3, the group is aiming to increase the story velocity for the first week to 15. Very little architecture, pseudocode and algorithm left to be developed. Most work needs to be implemented. Hence, the increase of speed. In regards to week 2 of iteration 3, the group aims to increase the story velocity to 20 sp/week. Such an increase will allow us enough time to refine the user experience and interface. Communication is a key tool to achieve*

such speed. Already implemented a plan to hold a meeting and pair coding session every other day.

Github Link: https://github.com/tommychang97/BGCInventoryStore/tree/main