BGC Engineering Inventory Store Project Requirements and Specification Document

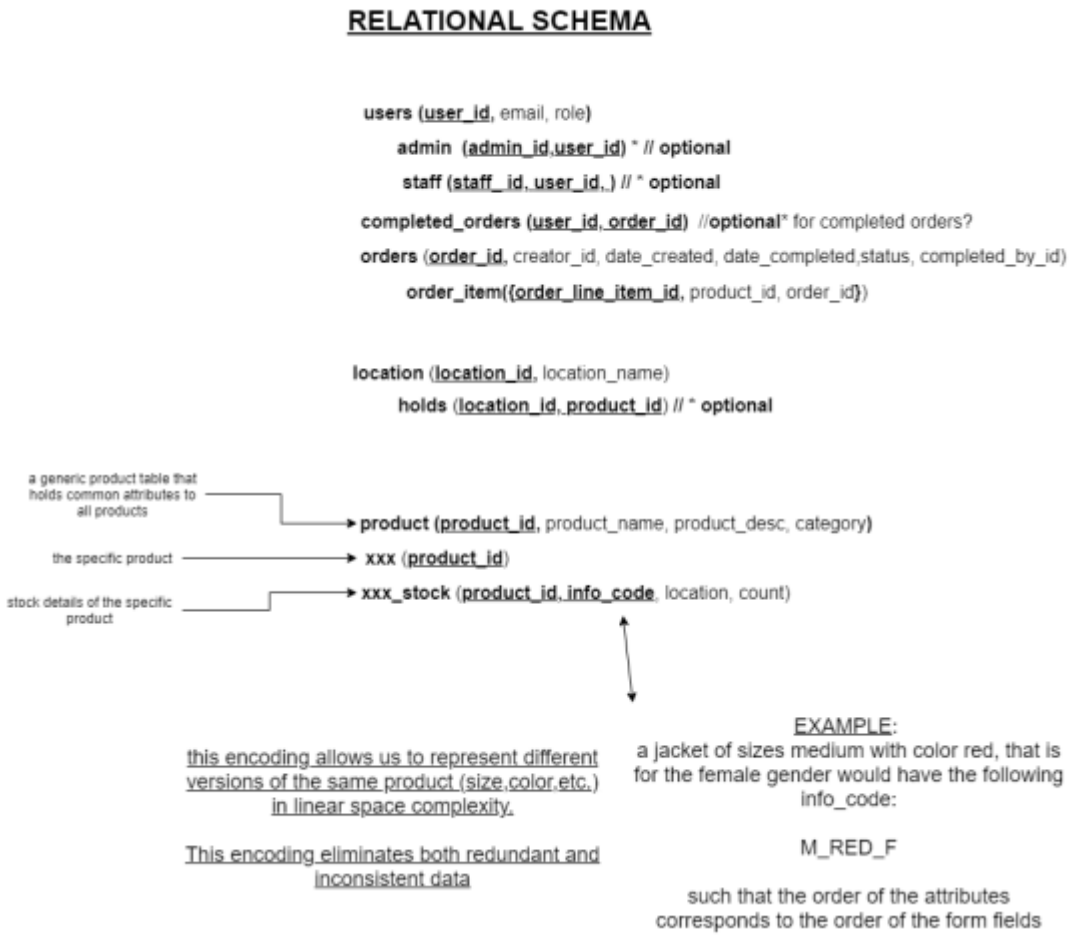
2021/08/06
Iteration 3

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:

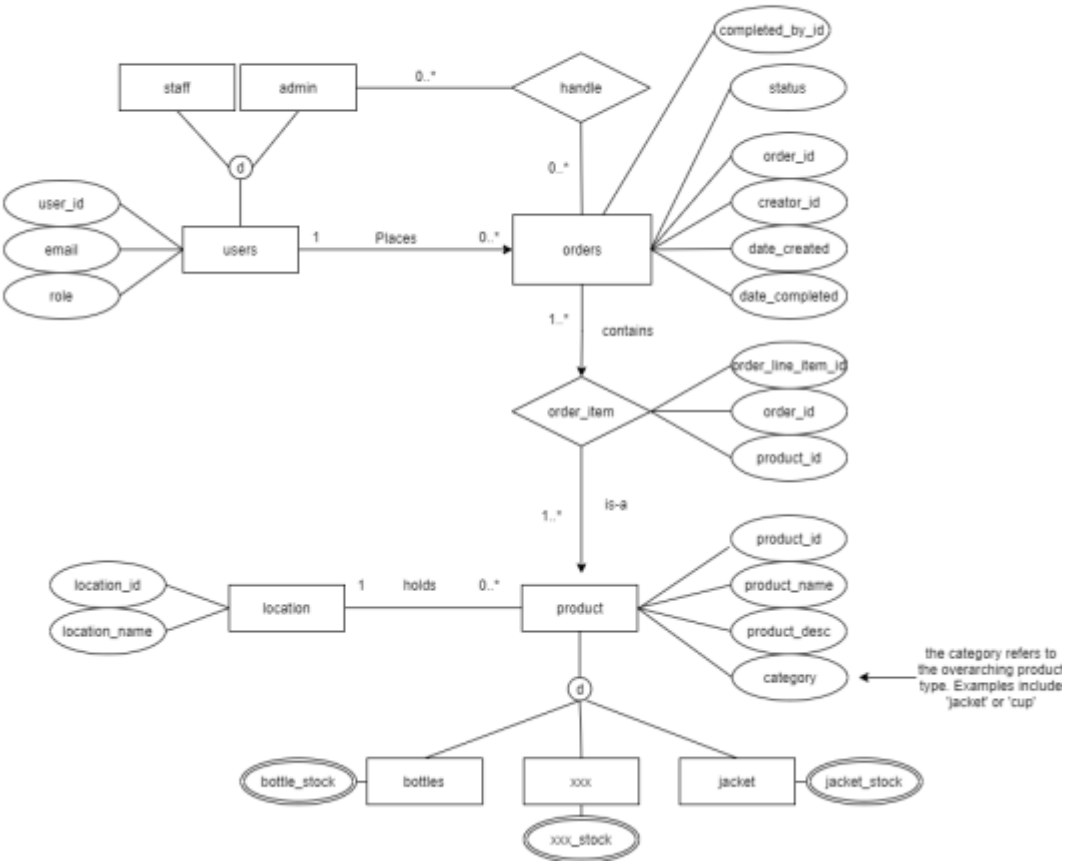


The figure above represents a tentative relational schema for our backend. This will be edited and revised alongside pending review from the employer who will provide feedback ASAP.

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 3:

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)**
6. Ability to modify an order's request status as an Administrator **(Iteration 3)**
    - Status: Waitlist/Completed/Submitted
7. Implement server-side logic to handle updating of values (item quantity) **(Iteration 3)**
8. Ability to view all items in the database (shopping page - filtered by category, staff) **(Iteration 3)**
9. Ability to view/edit items in the database (admin) **(Iteration 3)**
10. Ability to view all requests (admin-side). Perhaps filter by status. **(Iteration 3)**
11. Ability to view some more pages (contact, faq, guideline...) Do not keep useless links **(Iteration 3)**

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

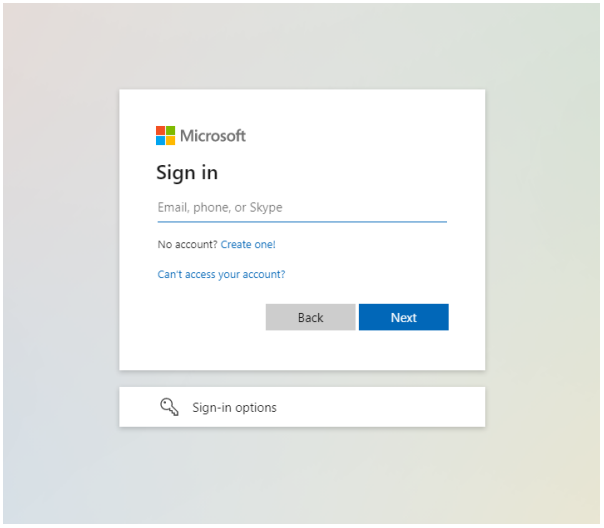
Actors: Any user visiting the application's URL

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

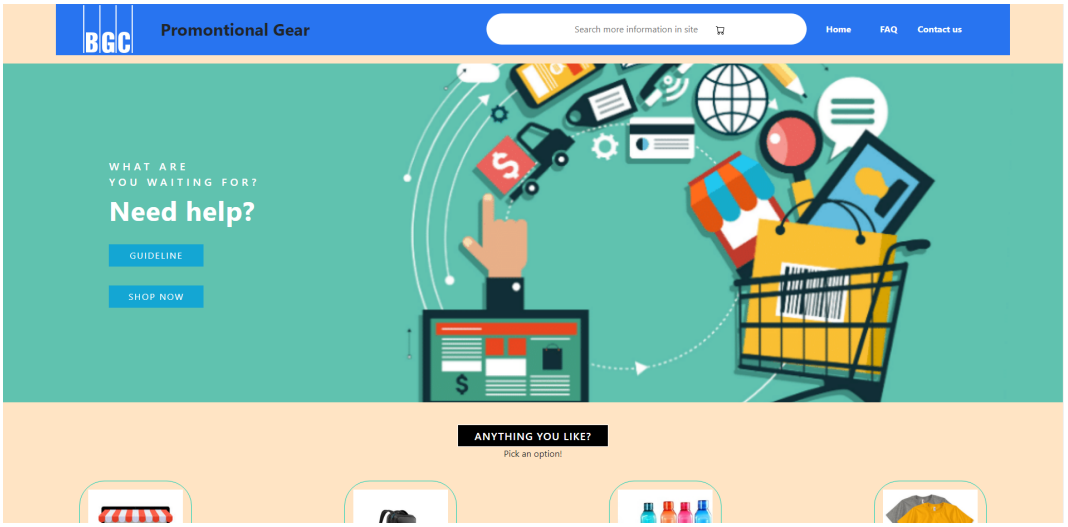
Name: Staff Landing Page

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

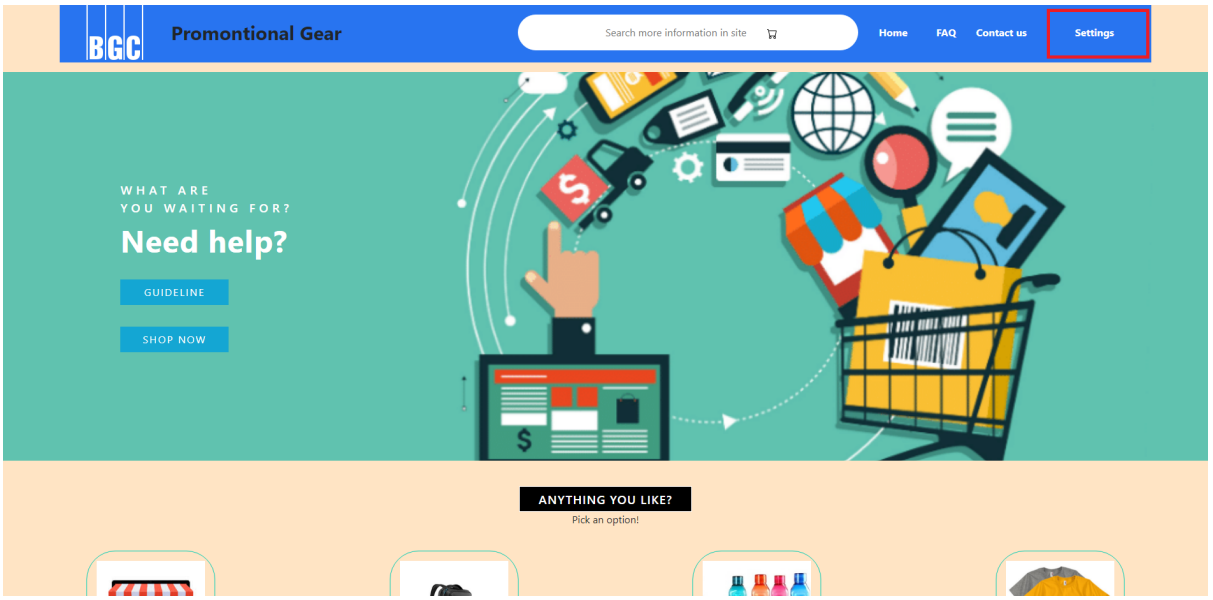
Name: Administrator Landing Page

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

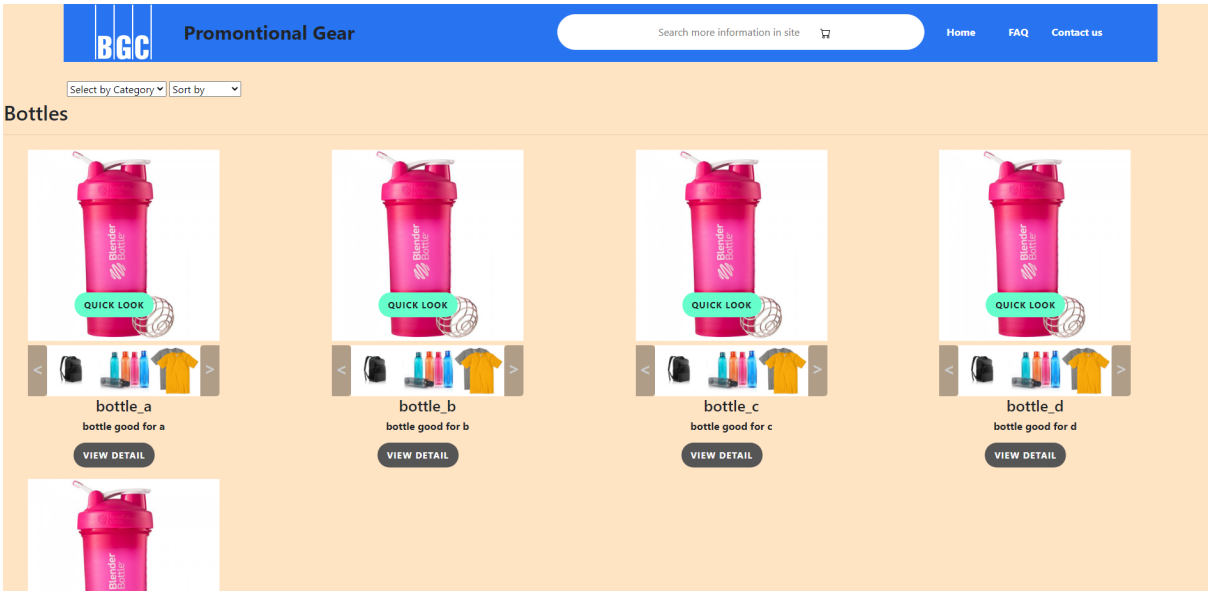
Triggers/Preconditions: The user has valid credentials, already logged in and redirected to the landing page. There are bottles product information present in the database

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.

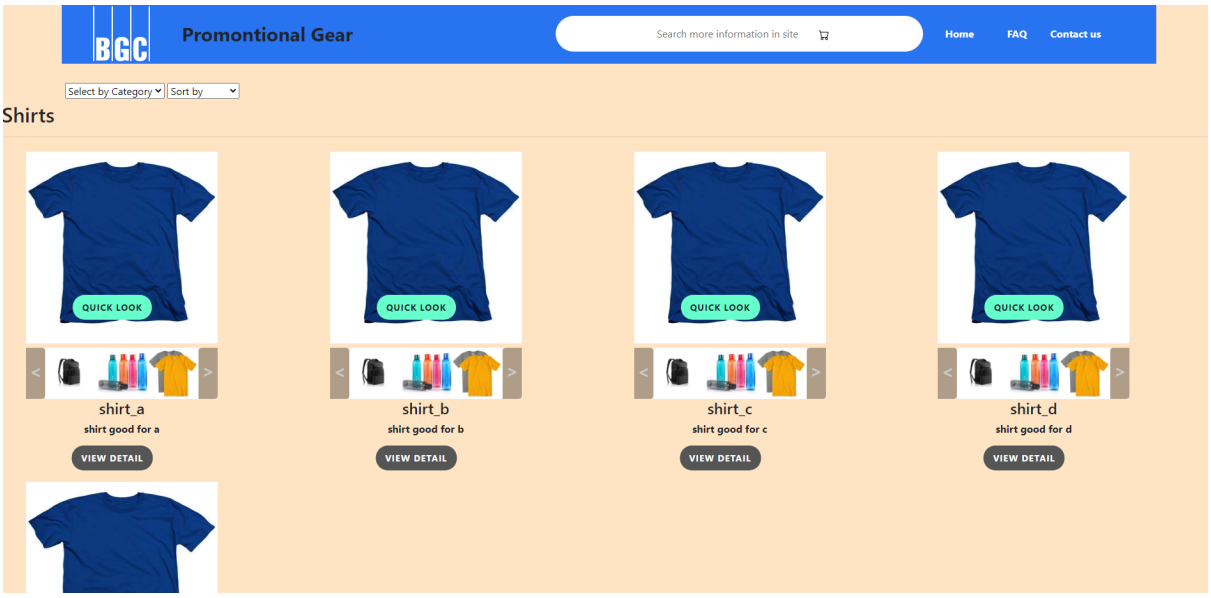
Acceptance test:

Success:

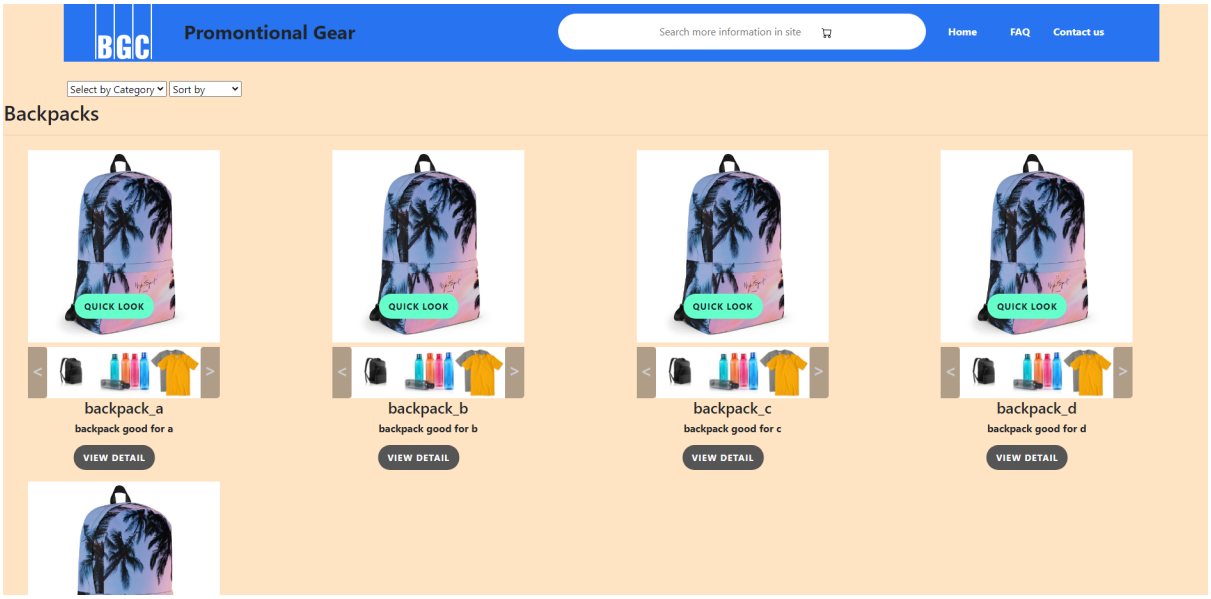
Selected the “Bottles” category, and a list of products within the Bottle category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all bottles.



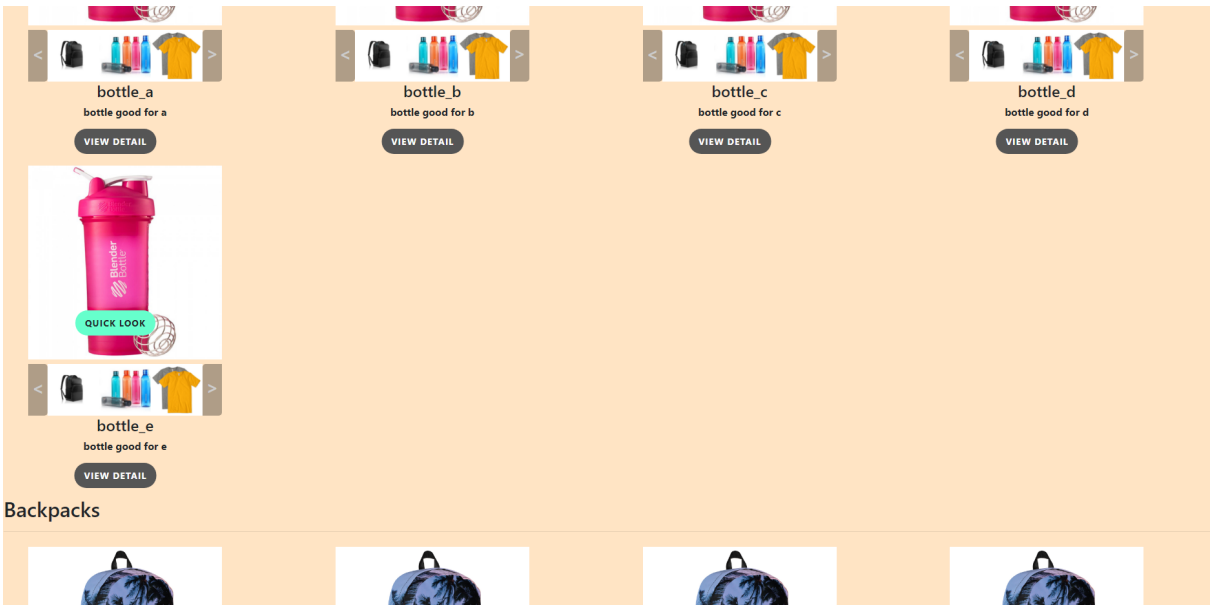
Selected the “Shirts” category, and a list of products within the Shirts category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all shirts.



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.



User 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 12:

Iteration #: 3

Name: View an order's details

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

Actions/Postconditions: Administrator is view an order's details

Acceptance test: Most of the information regarding an order is displayed, such as requester email, order items and their different attributes.

Success cases: Able to view order details for all 4 different types of statuses: submitted, waitlisted, fulfilling, and completed. There are also options to change the status of an order and whether or not you want to manually update values or let the system perform deductions for item count for you. If the order is completed, there is no option to modify status or fulfiller fields (hidden).

User Story 13:

Iteration #: 3

Name: Complete an Order

Triggers/Preconditions: User is an administrator and there is an outstanding order that is submitted with status submitted, waitlisted, or fulfilling

Actions/Postconditions: Administrator is able to complete the order and change its status. If the user selects Automatic deductions, then the item count for that quantity in that location will be decremented based on order count.

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

Success cases:

Statuses change from submitted, waitlisted, or fulfilled to completed. Item quantities are correspondingly deducted based on location of the order.

User Story 14:

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 wait list or fulfill the order and change its status according to the inventory .

Acceptance test: Order status is updated, reflected in database (state, inventory quantity).

Success cases:

Order status can be successfully changed between submitted, fulfilling, waitlisted and vice versa

User Story 15:

Iteration #: 3

Name: Track all orders in the database

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

Actions/Postconditions: Administrator is able to track all the recent orders that have been requested. Also displays the status and date of request.

Acceptance test: A list of all the orders will be shown and have corresponding anchors that will allow you to view the details of that order

Success case: Successfully displays all orders that have been received in chronological order.

Failure cases: no ability to sort by date, status etc.

User Story 16:

Iteration #: 3

Name: Email notification for new order

Triggers/Preconditions: An order has been submitted

Actions/Postconditions: an email is sent from nodemailer package to an admin team email, notifying that there has been a new order received

Acceptance test: Email is received with order number and details.

Success test: Once an order is submitted, a test email named BGCPromoTeam@gmail.com is used to deliver a new email to personal email 'tommychang97@gmail.com', containing the new order number.

User Story 17:

Iteration #: 3

Name: Out of stock on product details page

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

Actions/Postconditions: Since user selects more details of the product, if the selected combination inventory is less than or equal to 0, the selection key for that attribute will become unselected and it will not be added to the shopping cart (in this case, the sizes will be greyed out and unable to be selected).

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.

Success cases: Sizes are greyed out when they are out of stock and the user is unable to add that specific product with a combination of gender/color/size.

Failure case: There is no notification to the user that the item is out of stock. Instead, an alert is displayed when the user attempts to add a product to cart without selecting all of Gender/Color/Size.

User Story 18:

Name: Modify a product's attributes

Triggers/Preconditions: Admin needs to change some details with any products excluding sku_id and product_id.

Actions/Postconditions: The product information should be changed in the database and can show modified details into all_product pages. All corresponding pages should display updated values pulled from the database.

Acceptance test: Changes are made to any information on any products through the Admin center, as well as to any other page that displays details of the product.

Success cases: The changed information can be displayed correctly on other pages

Failure case: Have not tested changes to product_id and sku_id that could potentially lead to problems in our database schema. Cascading updates to the database will be a solution to this problem.

User Story 19:

Name: Uploading an image(s) to display for specific products

Triggers/Preconditions: User is an administrator and has navigated to the “settings” page.

Actions/Postconditions: The image file uploaded will be saved onto the server’s file structure under the ‘uploads folder’

Acceptance test: The user is able to upload up to 10 image files to the server. If the user attempts to upload more than 10, an alert will be displayed warning the user.

Success cases: User is successfully able to upload jpg,png,jpeg, and other files with different extensions. Users are also successfully warned when attempting to upload more than 10 files via an alert and the upload event does not go through.

User Story 20:

Name: Viewing a product’s specific detail page and adding it to the cart.

Triggers/Preconditions: The product exists in the database and has at least one sku_id (one different variation).

Actions/Postconditions: The product’s different images will be displayed and the user will be able to rotate the image to see them. Different combinations of color and sizes will be generated based on what the user selects of Gender, and then color. After selecting a valid combination and clicking the “Add to cart” button, the product is successfully added to the user’s cart.

Acceptance test: Details about a particular product are displayed as well as specific gender/size/color combinations. Product is successfully added to the cart. If product already exists in the cart, the order count is increased by 1.

Success cases: Different images of that product are able to be displayed and rotated. Colors and sizes options are generated in a tree-like structure once one category has been selected. Products can be successfully added to the cart via the product’s detail page and quantity is increased by 1 each time the product is added.

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: iteration 2 *(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.

Velocity Measurements: iteration 3

Week Number	Story Point Velocity	Explanation
Week 1	$5 + 5 + 3 = 13sp$ $13sp \,/\, 1\,week = 13sp/week$	⇒ our front end development team designed the most challenging UI commit thus far; a collection of views that provides an interface for a wide array of admin functions = **5sp** ⇒ after the creation of the front end interface, our backend development team implemented the settings page, adding new functions to the models and controller files = **5sp** ⇒ our team engineered the modify products page that allows users to change existing product attributes via

		model function invocations = **3sp**
Week 2	$2 + 3 + 5 + 8 = 18sp$ $18sp / 1\,week = 18sp/week$	=> we learned to use the nodemailer package that will allow us to send an email whenever an order is submitted to alert the promotional gear team that a request has come in and needs to be responded to => **2sp** ⇒ we learned the mocha and chai framework. Such tools helped us develop testing units. Jtest has also been considered as an option. However, consensus on the mocha chai=**3sp** ⇒ A product 'details' view that furthers user experience, was created. This allows the user to select different gender/color/size combinations and add the specific product to their cart = **5sp**. => Submitting an order, modifying and updating orders, completing orders. Completed orders will decrease the count of products in the database for that location. **=> 8sp**
Week 1 & 2 (total)	$13 + 18 = 31sp$ $31sp / 2\,week = 15.5sp/wee$	Overall, we were successful in achieving a peak velocity of **15.5 story points per week**.

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