BGC Engineering Inventory Store Project Requirements and Specification Document

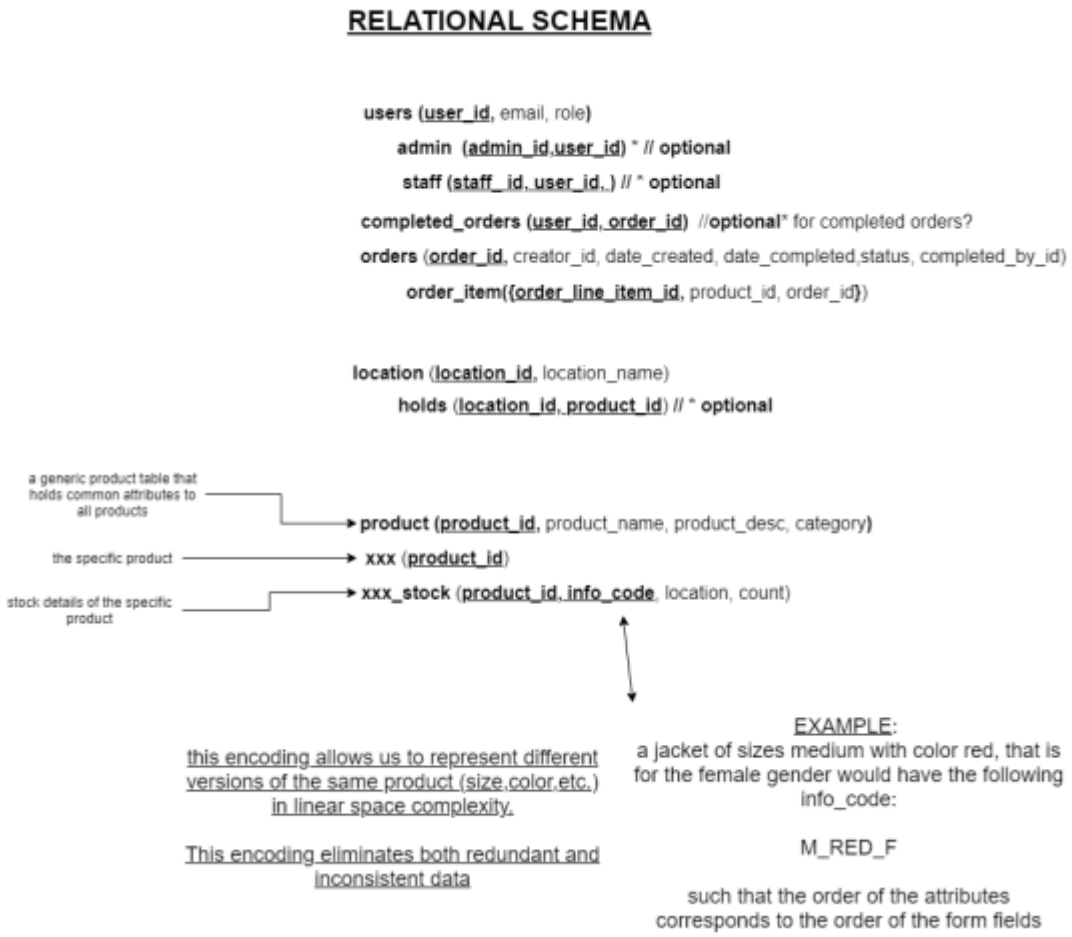
2021/07/09
Version 1

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.

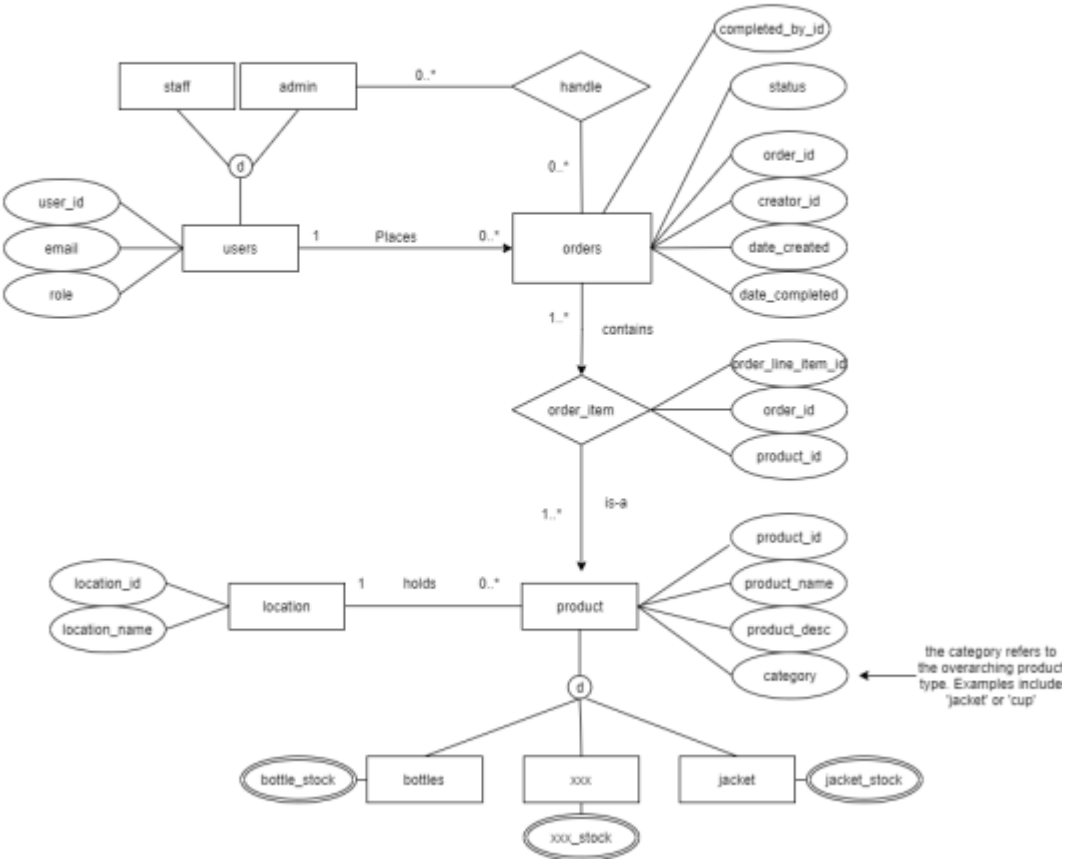
Competitive Analysis:
This project consists of a complete revamping of the existing site. Front-end technologies such as HTML/CSS/JavaScript will be used to provide a better user experience. The backend will use Node.js as a server-side language with a PostgreSQL database. The new web application will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

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 1:

1. Ability to login using the Azure Active Directory API **(Iteration 1)**
2. Ability to view/filter items on a shopping page (client) **(Iteration 1)**

Epic number 2 is not fully completed and will have to be delivered as part of iteration 2. The ability to browse by category is there but a filter will need to be implemented. The project is very complex and a lot of time was required to set up the database schema, configure local environments, and set up the project in an MVC design pattern for modularity. In short, a lot of work was done that is not shown in the views.

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

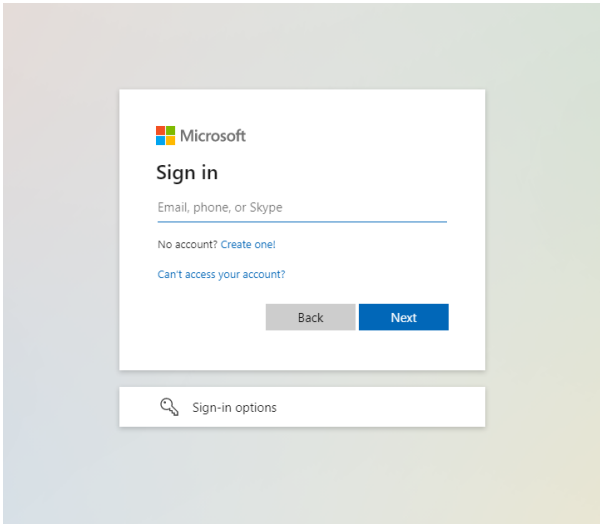
Actors: Any user visiting the application's URL

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

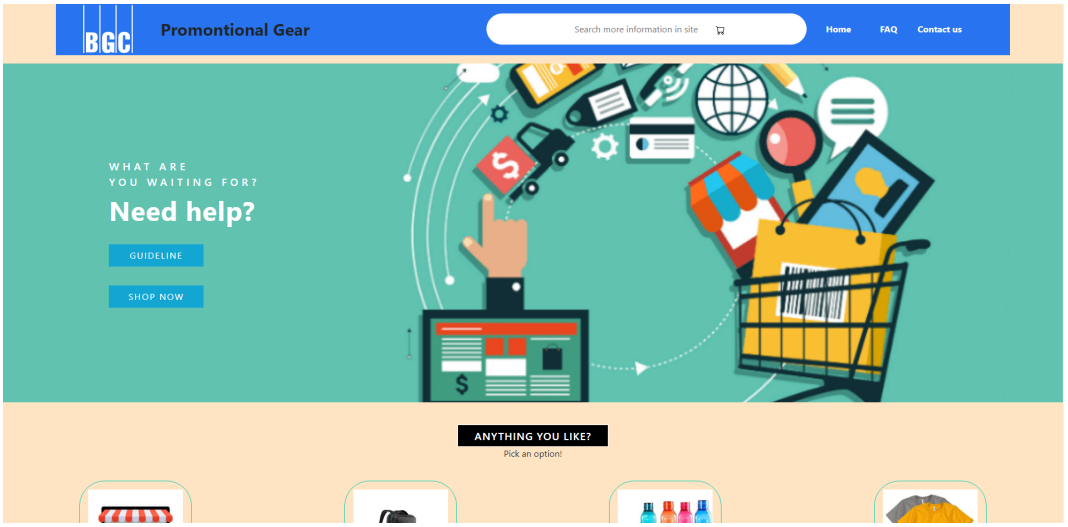
Name: Staff Landing Page

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

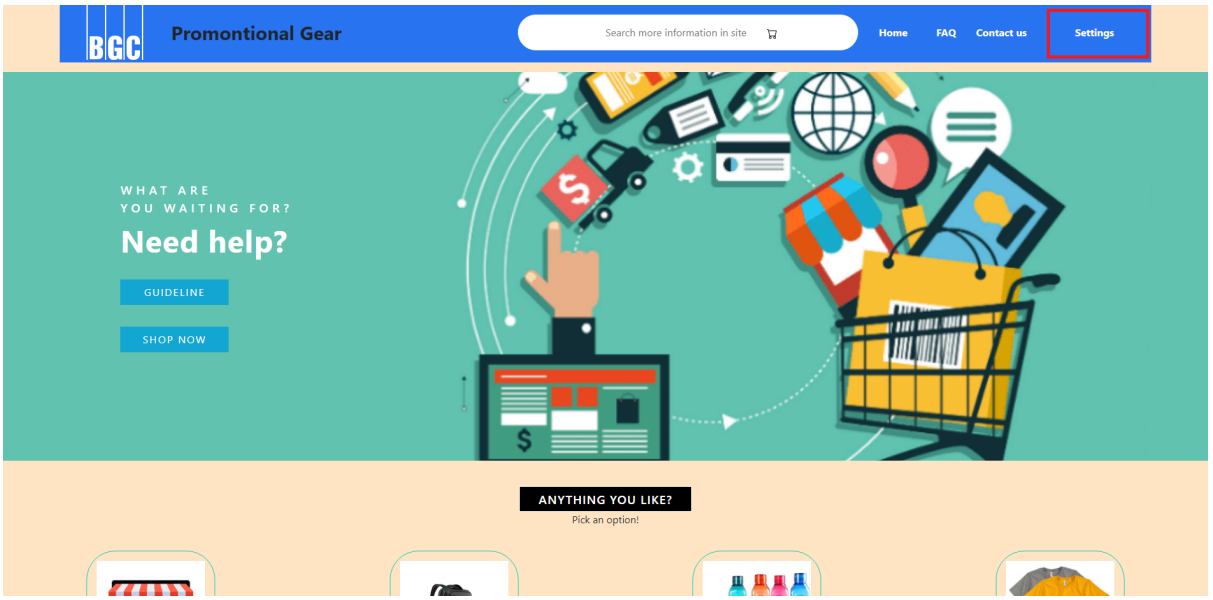
Name: Administrator Landing Page

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

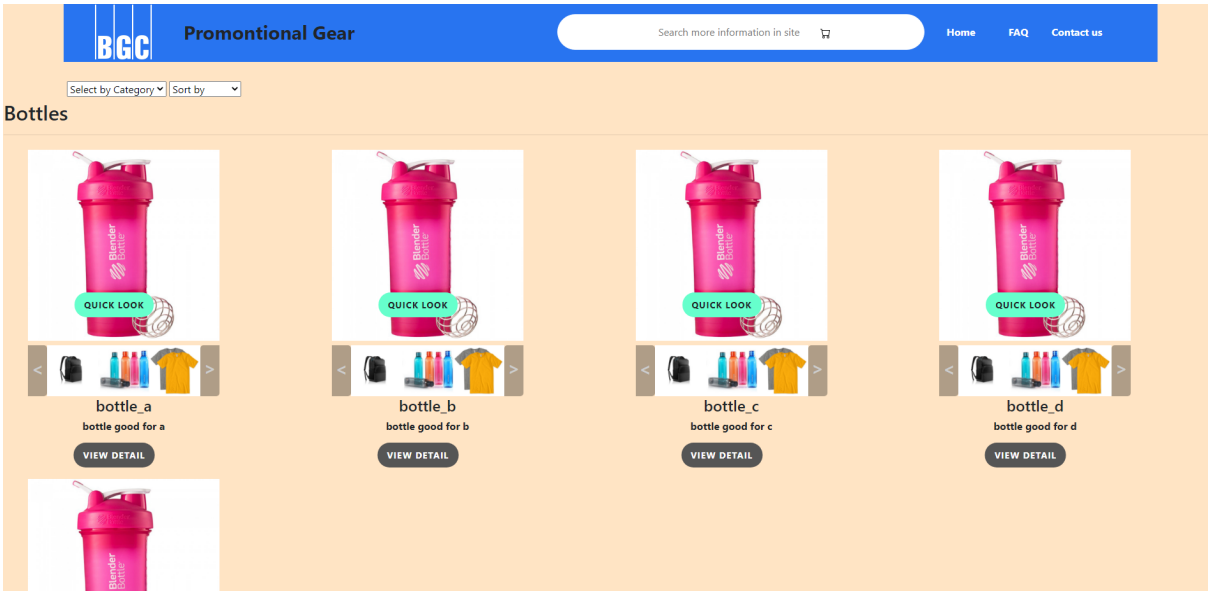
Triggers/Preconditions: The user has valid credentials, already logged in and redirected to the landing page. There are bottles product information present in the database

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.

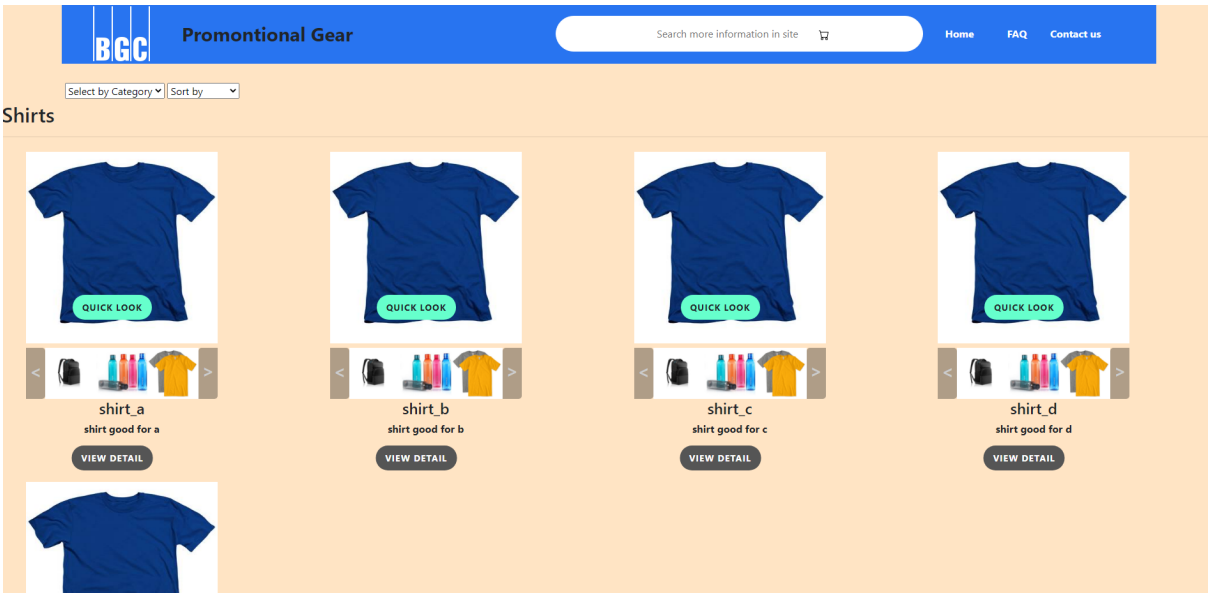
Acceptance test:

Success:

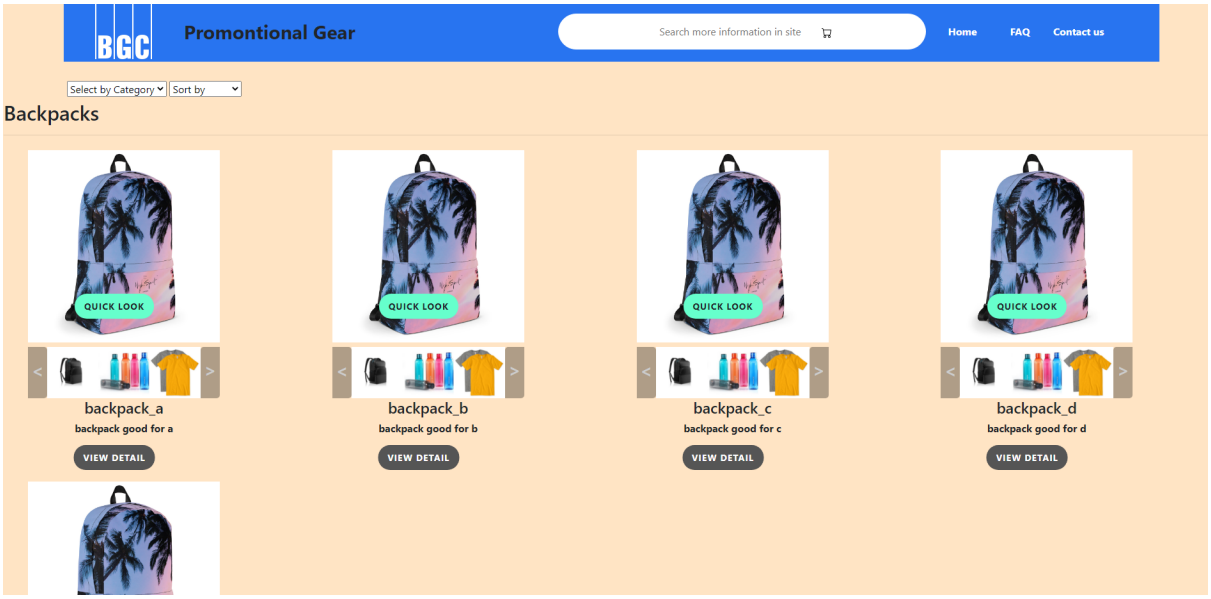
Selected the “Bottles” category, and a list of products within the Bottle category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all bottles.



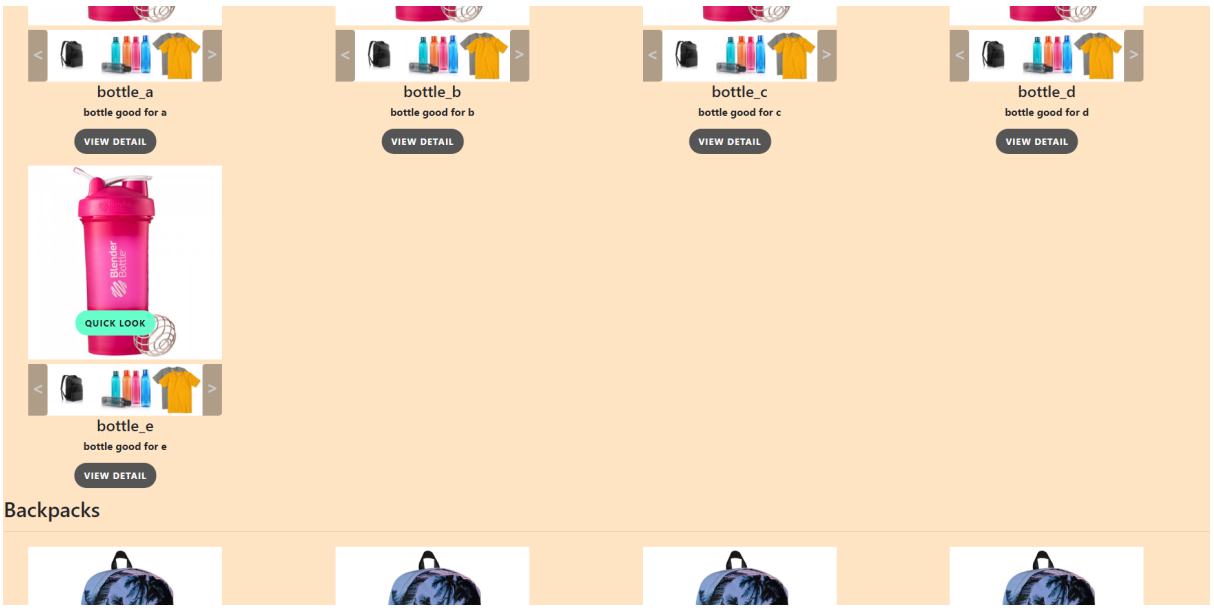
Selected the “Shirts” category, and a list of products within the Shirts category were pulled from the database and rendered to the screen. Each bottle has its own name and description. Currently the same image is displayed for all shirts.



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.



------TO BE IMPLEMENTED------

User Story 7:

Iteration #:2

Name: Filter products

Triggers/Preconditions: Products are in the database. User is logged in successfully.

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.

User Story 7:

Iteration #:2

Name: Uploading product of a particular category, with corresponding attributes to the database

Triggers/Preconditions: User is an administrator

Actions/Postconditions: Item is successfully inserted in the the corresponding tables

Acceptance test: Inserting a bottle into the database (attributes to be finalized). The corresponding record can be found in the database and pulled.

User Story 8:

Iteration #:2

Name: Uploading product of a particular category, with corresponding attributes to the database

Triggers/Preconditions: User is an administrator

Actions/Postconditions: Item is successfully inserted in the the corresponding tables

Acceptance test: Inserting a bottle into the database (attributes to be finalized). The corresponding record can be found in the database and pulled.

User Story 9:

Iteration #: 2

Name: Adding product to order

Triggers/Preconditions: User is logged in

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

Acceptance test: Orders page contains the product. Orders table and ordersLineItem table contains the newly added item. Order status is incomplete.

User Story 9:

Iteration #: 2

Name: Review and Submit order

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 uneditable, greyed input box, with the user selecting a location.

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

User Story 10:

Iteration #: 3

Name: Complete an Order

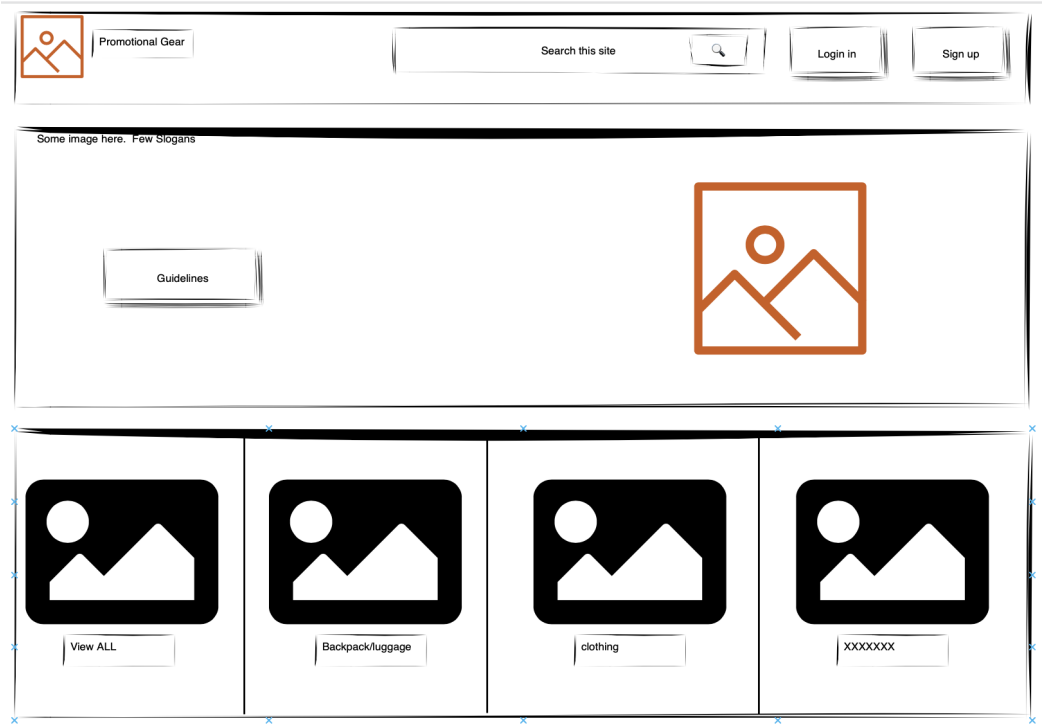
Triggers/Preconditions: User is an administrator and there is an outstanding order that is pending

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

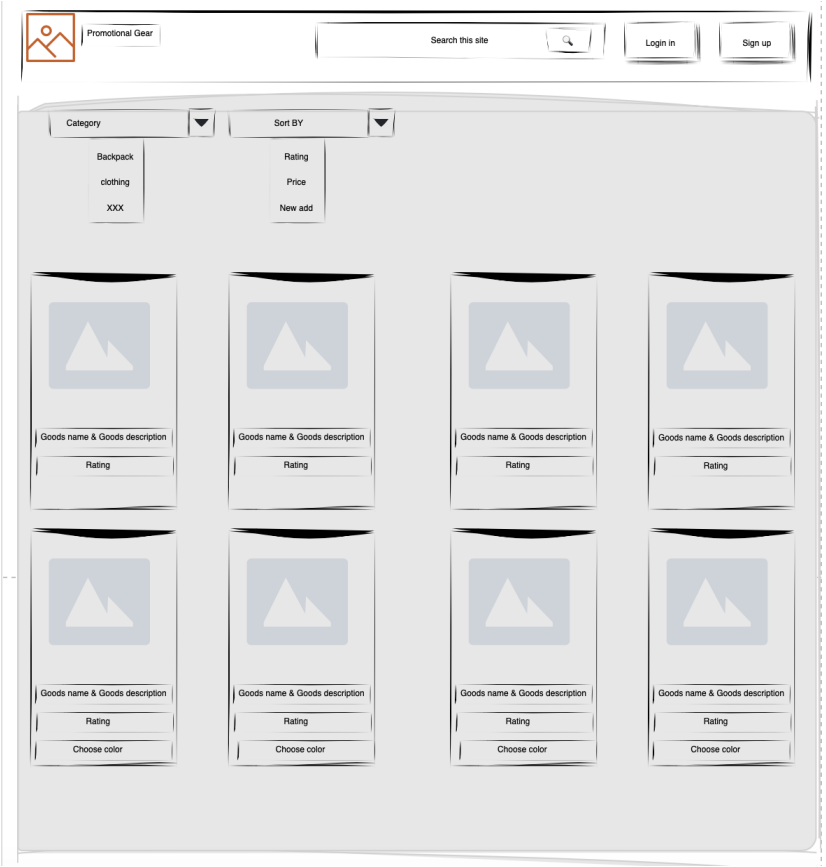
Acceptance test: Order status is completed, reflected in database. Corresponding products in the order have their quantity decremented based on the amount ordered.

User Interface Requirements:

Initial landing page mockup



Category page mockup:



Reflections:

What didn't go right

A better workflow needs to be established, along with more frequent team meetings in order to better meet deadlines, clear any confusion, and discuss stories that need to be broken down and implemented in the next iteration. We shared tasks but it was often unclear who was working on which file or folder and it took some time to integrate everyone's changes. We also experienced issues with GitHub, with some team members encountering difficulties merging branches onto the main branch. A lot of members were not experienced with the node and it took (and is still taking) a considerable amount of time to learn this new language. We believe it will pay off in the future to have a product that is scalable and easy to maintain.

What went well:

System architecture and design is integral to the success of any fairly-complex project, and this is especially true whilst building an e-Commerce site. In order to mitigate software complications and avoid poor engineering practices - such as code rot - we did our best to withhold the IEEE code of ethics and general optimal programming behaviour by following an agreed-upon standard called the Model-View-Controller design pattern. We also held long meetings and thoroughly discussed the protocols between our database architecture and the structural pattern of our codebase.

One great strength that was quite apparent throughout the software engineering process, was a phonenomal display of team coordination and communication. Our team did a great job in sharing tasks and respecting one another’s boundaries if and when it was crossed. For instance, although there was initial disagreement in how the workflow was to be executed, we quickly learned to put aside our differences and cooperate through to the end. One of the ways in which we accomplished this, was to apply pair programming, where one main person performed the development, and 1-2 observers acted as a safety net. This development method allowed us to capitalize on each other's strengths and minimize our weaknesses; it also allowed us to share ideas on different methods of implementation and syntax.

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