Team Deliverable 4 - The Core 4

Meeting Notes

This document contains updated Phase 2 requirements, UML design, user manual, compile and run instructions, reflection, and member contribution table. Content builds upon the previous Team Deliverable 3.

A) Requirements

Functional Requirements Implemented:

1) The system must allow users to create an account by entering their name, email, and billing information.
    a) Creates a unique profile that allows for future functionality with rewards, orders, and lists.
2) The system must display a product catalog showing grocery items with names, prices, and images.
    a) This is an essential functionality because if users cannot access the catalog, no orders can be created.
3) The system must allow users to add, remove, and update quantities of products in their shopping cart.
    a) This is a core functionality that enables users to manage their groceries and adjust quantities before checkout.
4) The system must calculate subtotal, total price, and discounts if applied in the shopping cart automatically.
    a) Provides accurate order totals for users before checkout.
5) The system must allow users to enter coupon codes and apply the corresponding discount to the order total.
    a) This incentivizes purchases and returning customers. Users may use “Welcome10” (10% off), and “Welcome5” ($5 off $25 orders) when using the program.
6) The system must award users reward points at the rate of one point per $10 spent on groceries.
    a) This functionality encourages users to use BasketBuddy and create a loyalty program for every time they return.
7) The system must update and display the reward points tracker on the home page after each successful checkout when logged in.
    a) Registered users must be logged in, displaying their points tracker on the home page.
8) Users must be able to sort and filter items while looking through the catalog of goods.
    a) Gives users more efficient feedback and sortability to navigate the catalog and find the desired groceries.
9) Users must be able to create, manage, or delete lists.
    a) This gives users the ability to make note of important items or favorites, ensuring that no item is forgotten during checkout.
10) Users must be able to see past orders after they checkout.
    a) This allows users to see a list of what they have already purchased in the past, which may be useful repeat orders.
11) Users must be able to edit their profile and delete their account.
    a) Gives users the functionality to change their specific information and delete the contents if it needs to be modified.

Functional Requirements Changed/Removed:

1) The system was supposed to store all data in a SQLite database, but we decided to go with a local storage configuration.

    a) We were planning on implementing a working off site database using SQLite, however, it seemed much more complicated to connect everything together. Using a local storage solution made it much easier to work and functioned with everything we needed for this deliverable.
2) The system was supposed to have an admin account to add or remove products.
    a) We cut this feature to focus on the shopper's experience instead of the back end management side.

Non-Functional Requirements:

1) All pages must load within two seconds on a standard internet connection.
2) When the catalog is loaded, sorting and filtering must be completed in under two seconds after the user requests it.
3) When a user interacts with any button or menu, responses must occur within one second of the request.
4) When the cart page loads, updating quantities or removing an item must refresh the total within one second without reloading the page
5) When the user clicks on the light or dark mode toggle, the change must occur in under half a second.

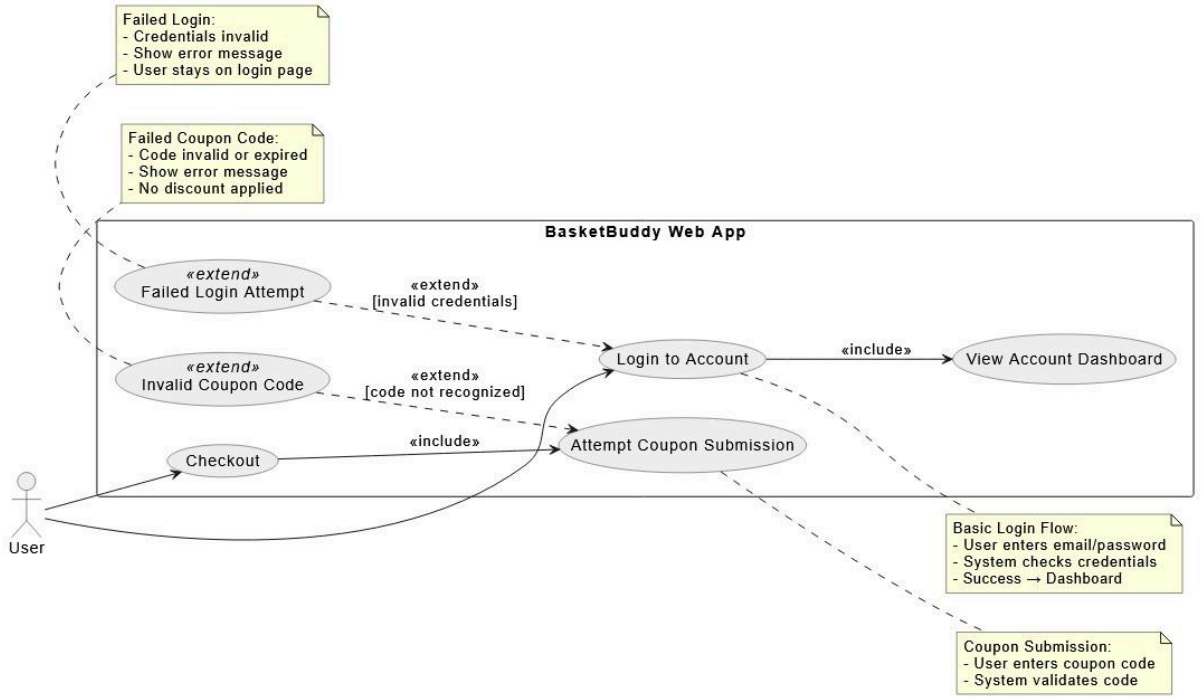
Requirements Reflection:

- Our functional requirements did not change from Deliverable 3 because the feedback we received was focused on migrating from a local host to a database server. A nonfunctional requirement was added to accommodate users and to improve usability with light & dark mode.

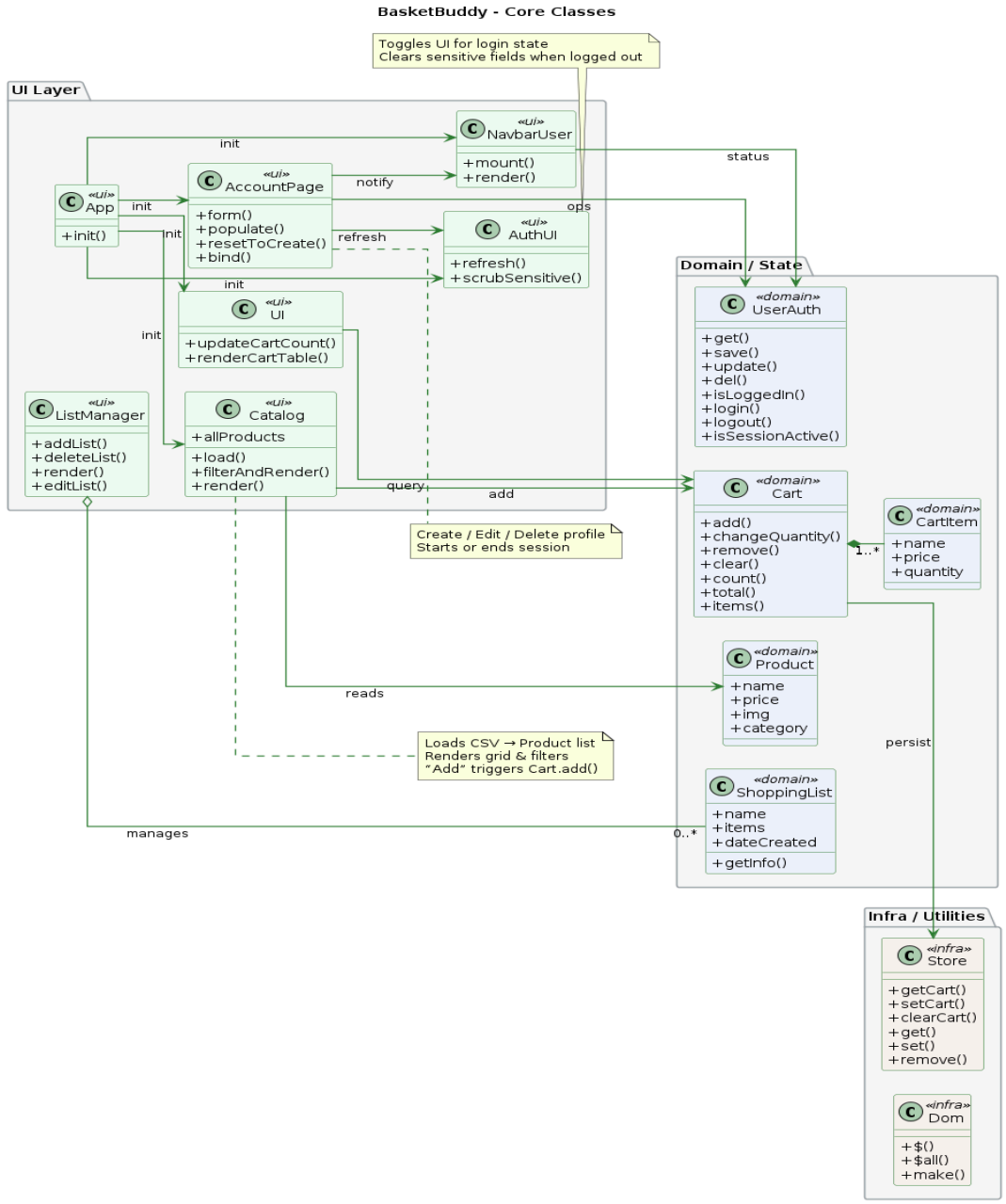
B) UML Design

Use Case Diagram

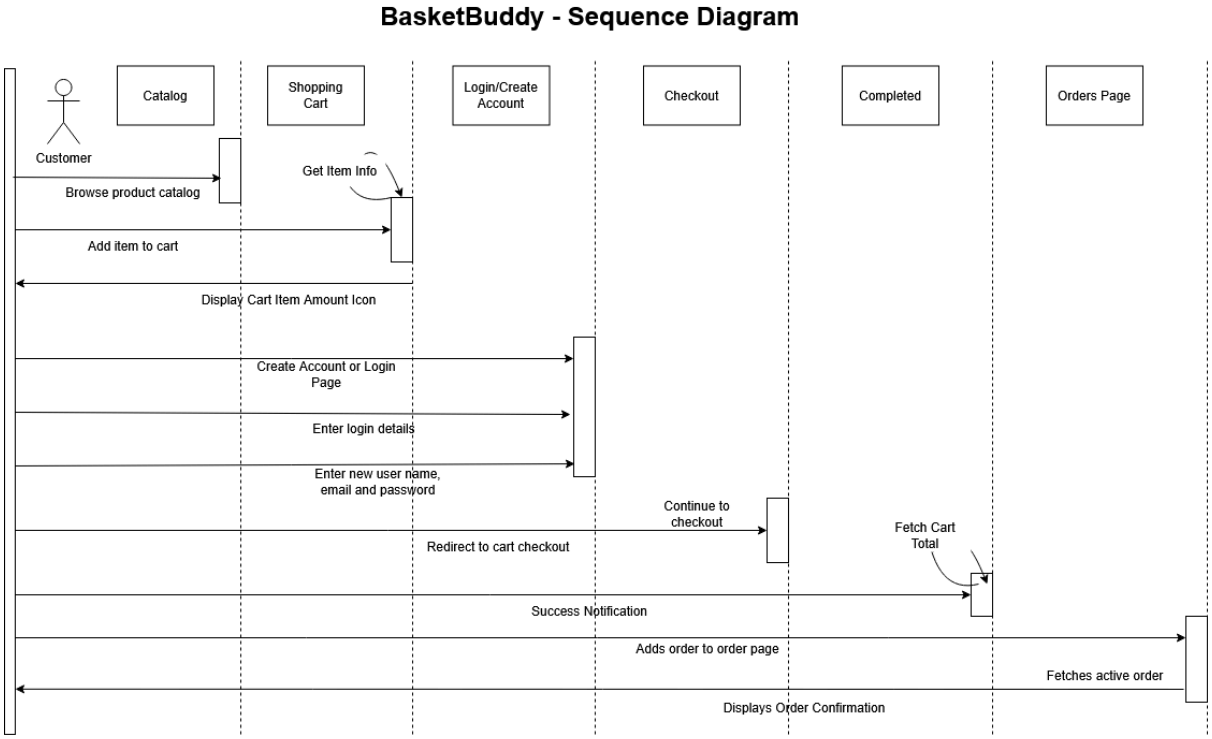
Error Case


Failed Login:
- Credentials invalid
- Show error message
- User stays on login page
Failed Coupon Code:
- Code invalid or expired
- Show error message
- No discount applied
BasketBuddy Web App
«extend»
Failed Login Attempt
«extend»
[invalid credentials]
Login to Account
«include»
View Account Dashboard
«extend»
Invalid Coupon Code
«extend»
[code not recognized]
Attempt Coupon Submission
«include»
Checkout
User
Basic Login Flow:
- User enters email/password
- System checks credentials
- Success → Dashboard
Coupon Submission:
- User enters coupon code
- System validates code

Class Diagram



Sequence Diagram



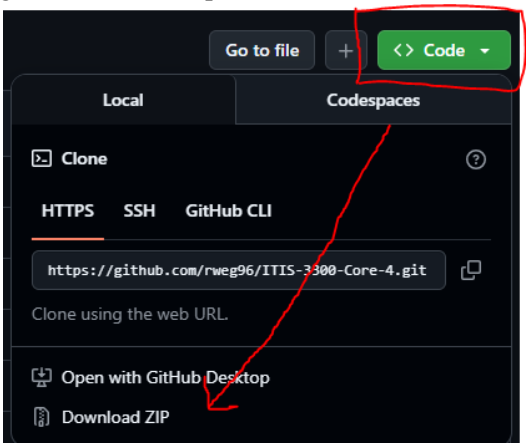
C) User Manual - Install & Use

Installation

To install BasketBuddy, there are two options to begin running the website. There are hyperlinks throughout this section to directly access specific relevant pages. Overall, this demonstrates the flow and guides a user that's unfamiliar with GitHub.

Primary Method: The website is successfully hosted utilizing GitHub Pages, built into GitHub, so that the user can access it on browsers such as Google Chrome, Edge, and Safari. There is no need to install anything, just access the link **HERE.**

Secondary Method: If the primary link fails, you will need to navigate to the GitHub repository for our group. You can access the repository by clicking **HERE.** Once on the page, you can click the green "Code" drop down button and download the ZIP file.



Once downloaded, you will see the folder named **“phase-1”** in the **src** directory**.** This is the folder that contains all of the files for the project and **“BasketBuddy”**

D) Compile & Run Instructions

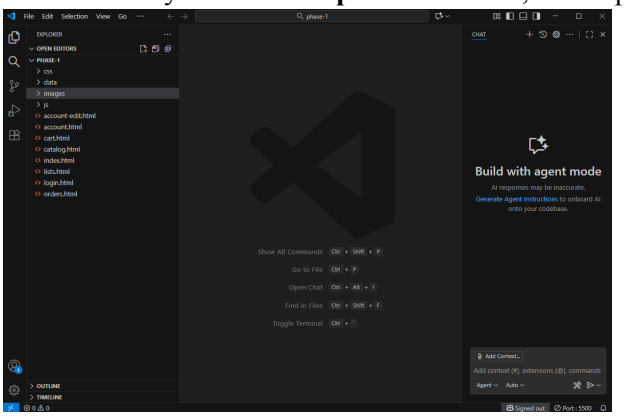
Running the App

To run “BasketBuddy,” there are two options that coincide with whichever installation method you choose:

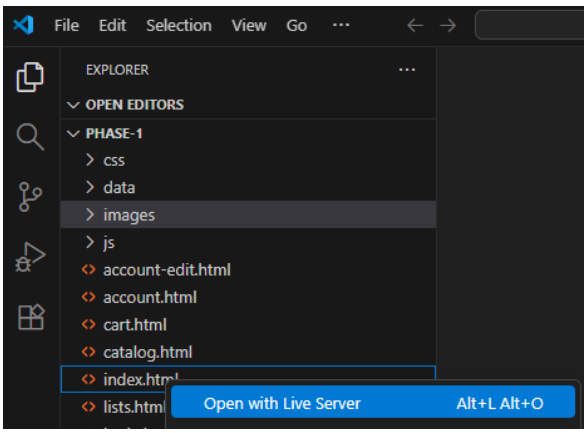
Primary Method to Run Application: As previously mentioned above, you may run the webpage without any sort of installation needed. Access it by clicking **HERE**. This is the easiest and most efficient method of getting to the webpage.

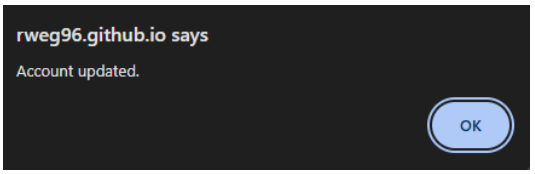
Secondary Method to Run Application: If you want to download the ZIP file of our project files, there are just a few more steps involved.

1) Download the ZIP file and extract all the contents from the ZIP to a place on your computer you will remember.
2) After extracting the ZIP file, open any program that allows you to test code. We as a group recommend using **VS Code** because this is typically what students and faculty recommend in CCI classes at UNCC.
3) Once you have opened VS Code, add a folder by selecting in the top left **File** → “**Open Folder**”. Then, navigate to the place you extracted the ZIP file and navigate through the file paths to eventually get to our ZIP folder name “ITIS-3300-Core-4-main” → src → **“phase1”**. Once you click the “**phase-1**” folder, click open. Your screen should look like this.



4) Now that you have the files you need to access “BasketBuddy”, right click on **“index.html”** and click “**Run With Live Server**”. You can now see our website, provided you have the Live Server extension downloaded in VS Code.

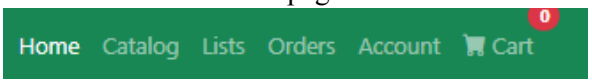




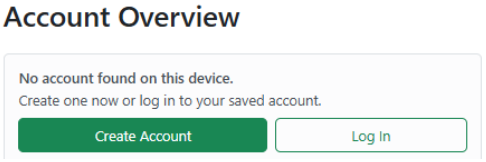
Using the App

Once everything is set up, navigate to the website either locally or over the internet. Below are the steps of a user's flow as they navigate through BasketBuddy to complete a purchase.

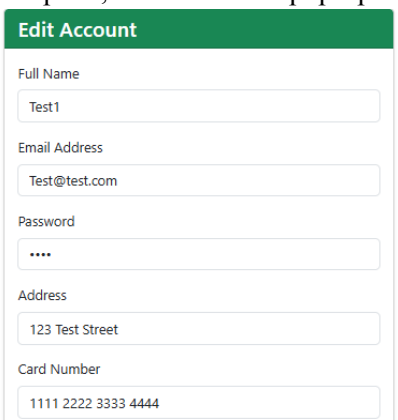
1) Once on the home page, you can easily begin shopping. However, a rewards system gives registered users points after ordering, so you are going to want to create an account first. Navigate to the header in the top right and click on the “Account” button. This will direct you to the account overview page.



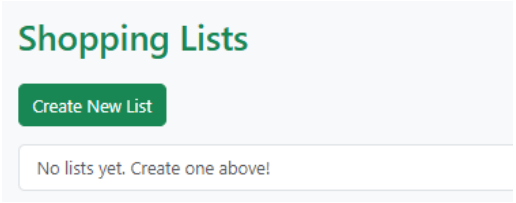
2) Next, click the “Create Account” button. Once you are a registered user, you would click “Log In” as a returning user instead.



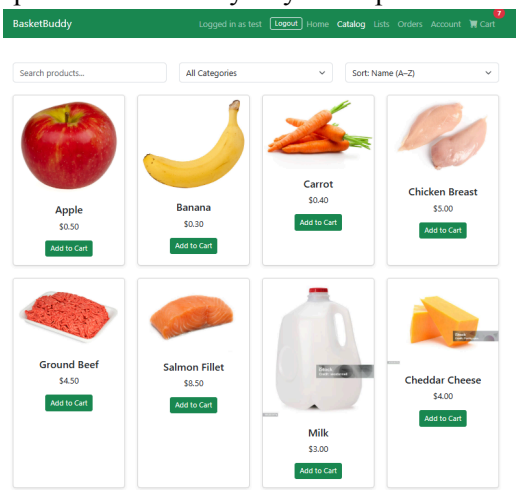
3) On the account creation screen, fill out all necessary fields with personal information. Once complete, there will be a pop-up that notifies the user that their account was created.

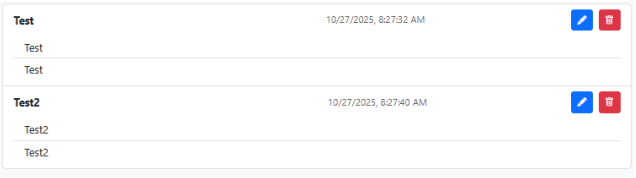


4) Now that you have created an account, you may take advantage of all features within BasketBuddy. You can start by creating lists, containing your favorite items or weekly essentials. Lists are handy when referencing items that you order often from the catalog.

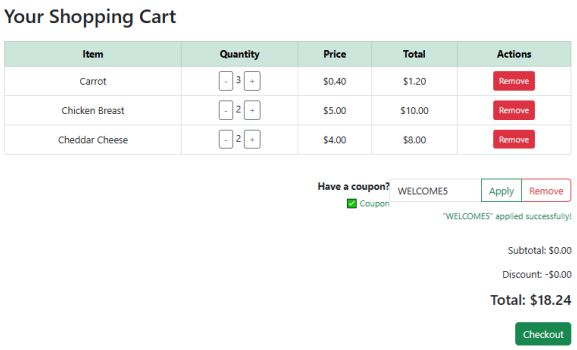


5) Next, you can order groceries! Navigate to the catalog page, displaying all available items for purchase. Simply click “Add to Cart” on the items you want in your order. You can also use the built in search or filtering function to make it easier to navigate the catalog for a particular item. The cart in the top right displays the total amount of items you have in your cart and will update automatically as you shop.

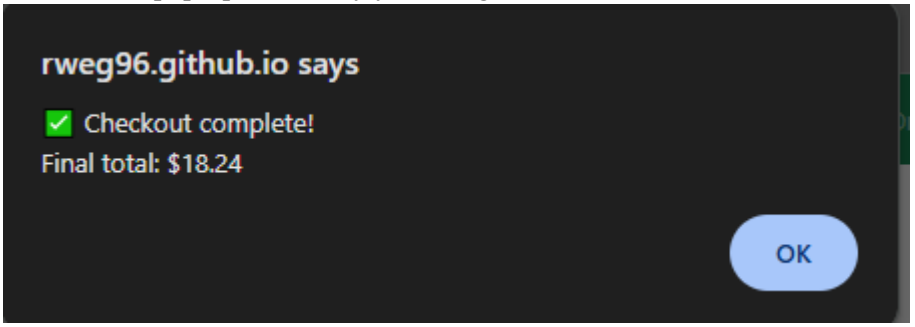




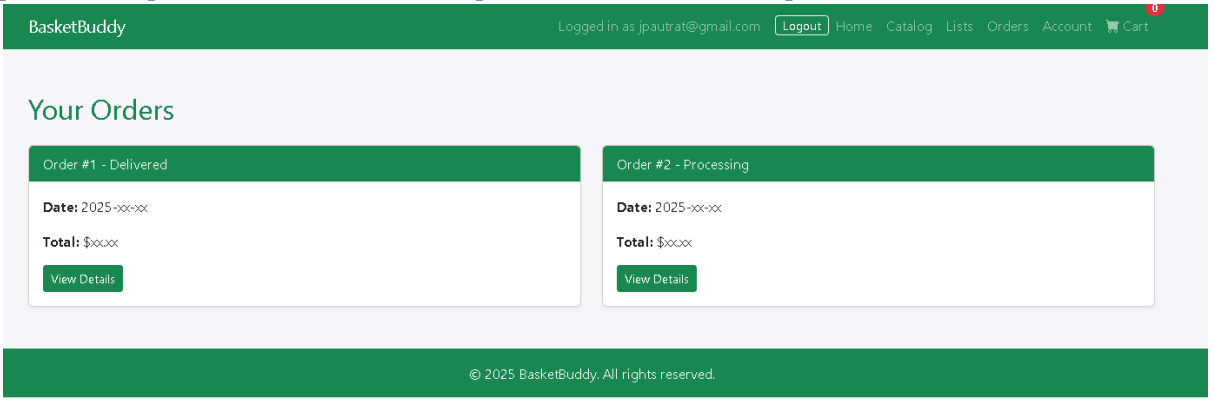
6) If you have a cart full of groceries, it’s time to check out. By clicking the cart icon in the top right corner, you can see all the items you have added along with a total price. If you have a coupon code, add it! It will automatically apply and adjust the total cost of your order.



7) With payment information already filled out, you may click the “Checkout” button. If successful, a pop-up will notify you along with the final total that the checkout is completed.



8) Navigate to the “Orders” page along the top header. This will reflect your orders, past or present, and provides details about date, price, and items that were purchased.



Test Cases

#	Functionality	Input	Output	Pass/Fail
1	A user is able to create an account on the website.	The user navigates to account → create account, and fills out the required fields and clicks create account to make it.	Confirmation appears on screen saying “Account Created”	✓
2	Order History Displayed	Add items to the cart, proceed to checkout while logged in, then open the orders tab.	Orders do not appear under history due to local storage issues	✗
3	Add Item to Cart	When a user clicks the “add to cart button” the total amount of items is properly displayed on the cart icon.	The cart count increases and the item appears in the Cart page.	✓
4	Update Cart Quantity	On the cart page, clicking the + or - besides the item changes the amount	Quantity changes immediately; subtotal and total refresh automatically.	✓
5	Apply Coupon	On the cart page, when ready to checkout, apply a coupon (SAVE10), and click Apply.	Message: “Coupon ‘Save10’ applied successfully!, and total updates to reflect discount	✓
6	Invalid Coupons	Enter an invalid coupon, example (test1)	There should be a message that says “Invalid Coupon” therefore the total doesn't change	✓
7	Checkout Completed	When you are logged in, create an order and then checkout.	Alert shows final total. Cart clears, rewards points increase, and totals reset to $0.00.	✓
8	Rewards display for user on homepage	When a user is logged in, go back to the home page and after an order you should see rewards points. (1 point = $10)	The rewards section is visible with specific point totals based on items purchased	✓

E) In Class Feedback

During the in-class code inspection session, our team received positive feedback overall on the project's structure, layout, and implementation quality. However, two important security concerns were brought up by the other group that relate specifically to authentication.

Feedback Received

- **Local Storage Authentication Can Be Spoofed**
  The reviewing team noted that storing user credentials and login state in the browser's local storage is not secure, as local storage can be easily modified using browser developer tools. For a production-level system that would be deployed for real-world use, this approach would expose the application to account spoofing and unauthorized access.
- **Coupon Array Visible in JavaScript**
  The coupon list is stored client-side in a JavaScript array. This means users could modify these values or create unauthorized discounts by using the browser's console or inspection tools. Similarly, a backend system would be needed for secure validation of this feature.
- **Recommendation for Server-Side Authentication**
  The reviewing team recommended implementing a client-server architecture, where authentication and coupon validation occur on the server using hashing algorithms like Argon2. This would dramatically improve the security and integrity of the application.

Actions Taken

Given the timeline and the intended educational scope of the project, our team elected to keep the original design using local storage rather than implementing a full backend system for Phase II. However, we used the feedback to guide future planning and added the following improvements:

- Documented the security limitations of BasketBuddy's architecture in Deliverable 4 and the presentation.
- Reorganized the authentication code to reduce confusion, which enables future backend integration to be easier to deploy.
- Added guard checks around coupon validation to improve input handling and reduce any user-side manipulation errors.
- Outlined backend migration as a key future improvement in our reflection and limitations section.
- Added toggle dark mode feature to improve usability and accessibility.

Conclusion

- While server-side authentication and validation are beyond the scope of this deliverable and project timeline, this feedback provided valuable insight into how BasketBuddy would need to evolve in order to be deployed securely for public use.

F) Reflection

What We Did: For this project, our group was able to successfully implement many of the features we intended to deploy for BasketBuddy. These features were identified in previous phases and group planning to ensure that end users have a working online grocery shopping medium. We successfully completed the following:

1) A cart that displays all of the items that the user has added to it.

2) Items in the cart are automatically totaled for a total order price, allowing for an easier purchase and checkout process.
3) Implemented a rewards system that applies points after a registered user makes a purchase.
4) Created an item catalog that displays all of the grocery items for sale.
5) Created a search filtering functionality that allows users to search by name (A-Z or Z-A) or price (Low-High or High-Low)
6) Created a login/sign up account creation process.
7) Created a lists section which enables shoppers to add or remove products to lists. Lists can be created, edited, or deleted.
8) Added usability and accessibility dark & light mode toggle to assist users.

What Went Well:
Overall, our group accomplished nearly all of what was set out to deploy from the beginning of planning back in August. We envisioned creating an online grocery shopping solution that would simplify the lives or schedules of users who have little time to shop for groceries. BasketBuddy functions smoothly and uses the local storage to transfer specific data between pages as the user navigates. Our login and account creation process enabled us to implement features that were associated with the account the user created. We were able to keep uniformity between pages when using our UI/UX layout, color scheme, and format of webpages that gave users a pleasing visual experience. Group communication overall was good, and most members took on clear responsibilities and were able to find solutions to issues we were facing during the development phase.

Future Improvements:
1) In the future, it's critical to ensure that smaller details are changed to improve BasketBuddy.
    - Adding more items that encompass the catalog of an entire grocery store.
    - Switch from a local storage solution to implementing a database, like SQL, which can store user accounts, groceries, and payment history.
    - Ensuring that users who are logged in can access all of their previous orders once they have checked out.
    - Adding an order tracking feature that shows the user the current status of their order.
    - Ensure discounts are properly shown on the checkout screen after applying.
    - Adding rewards that people can redeem with the points that they earn by buying groceries.
    - Populate the active order on the home page after it is successfully submitted.

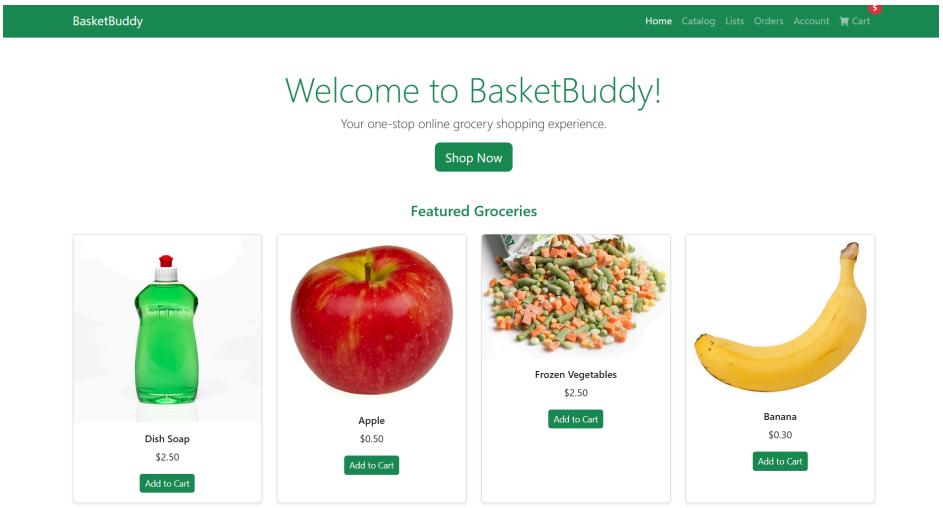
F.) Member Contribution Table

Member Name	Contribution Description	Overall Contribution	Note if applicable
Tyler Niemonen	Created catalog usability, helped build index.html, built the flow chart of overall process flow, worked on updating/changing our functional & NFR requirements helped organize the Deliverable 3 Turn in, installation and how to run portion, completed section C&E, updated meeting notes, created test cases, updated the peer feedback area	20%	N/A
Jeff Pautrat	Initial github repo testing for	20%	N/A

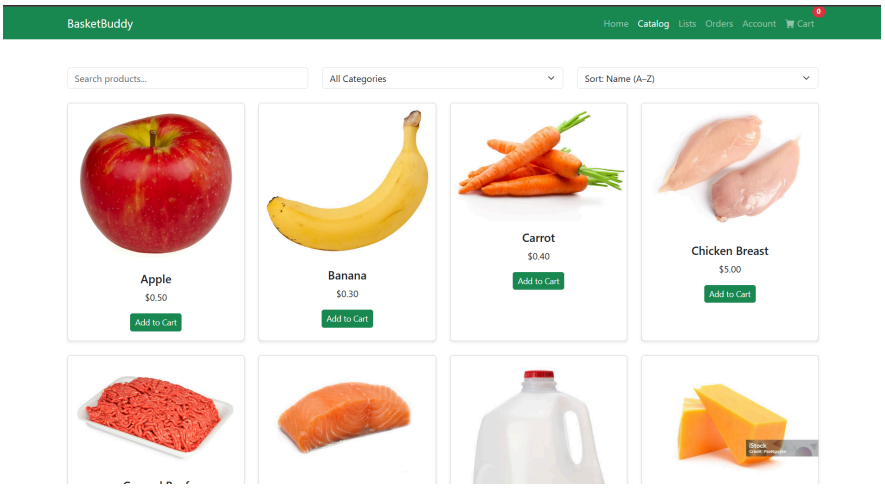
	website deployment options as well as sources and developed the database for the app as well as portions of uml design, created demo video		
Jakia Gary	Implemented the coupons and rewards feature, Contributed to UI interactions involving user reward tracking and discount application	20%	N/A
Carson Carmody	Created the base/initial HTML pages and implemented the shopping list functionality. Created the sequence diagram. Captured UI preview screenshots and formatted deliverable. Worked on the feedback section for deliverable four.	20%	N/A
Ryan Wegener	Implemented catalog search, sorting, and filtering functionality, developed account session management, and produced the class diagram with core code snippets for documentation. Implemented dark & light mode toggle feature.	20%	N/A

Page Overview Screens

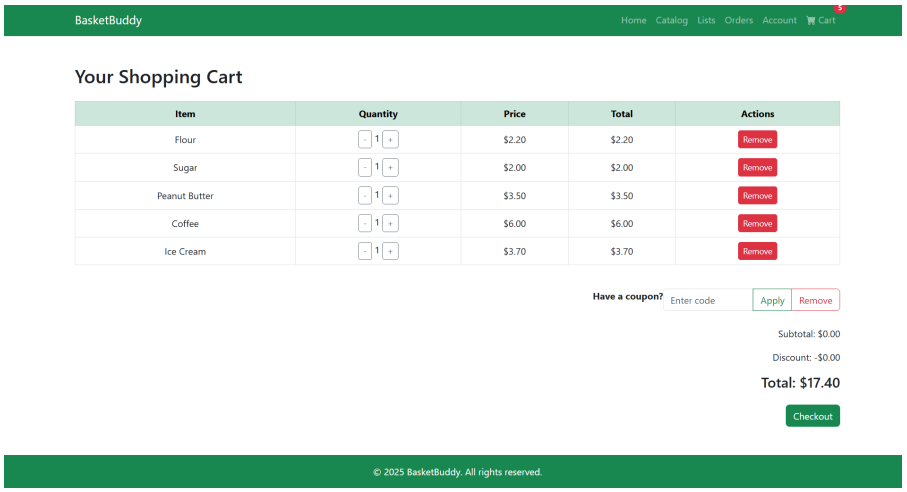
Home



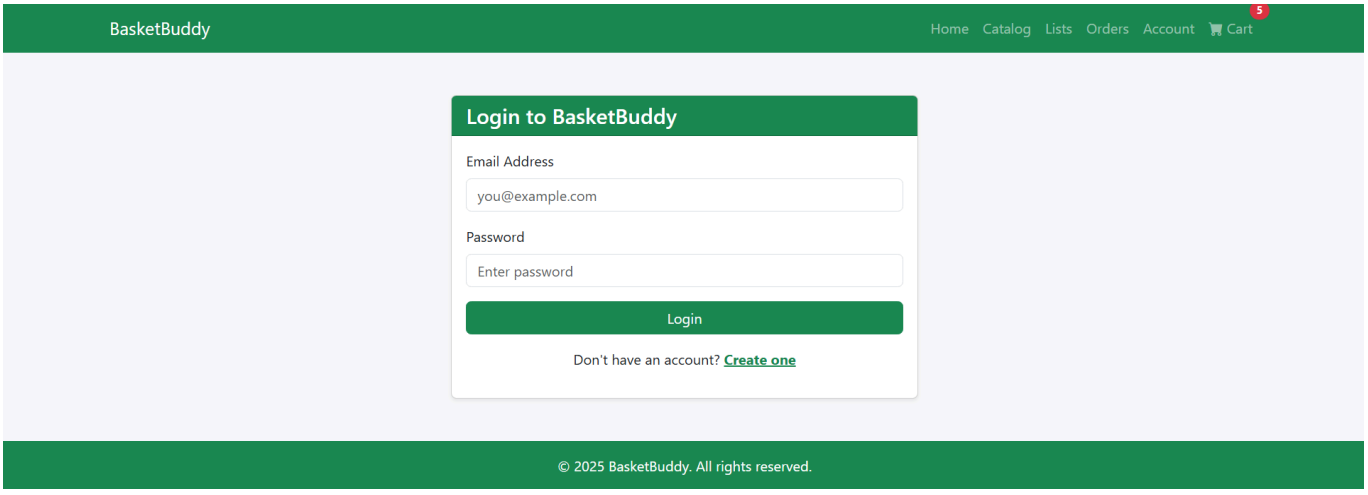
Catalog



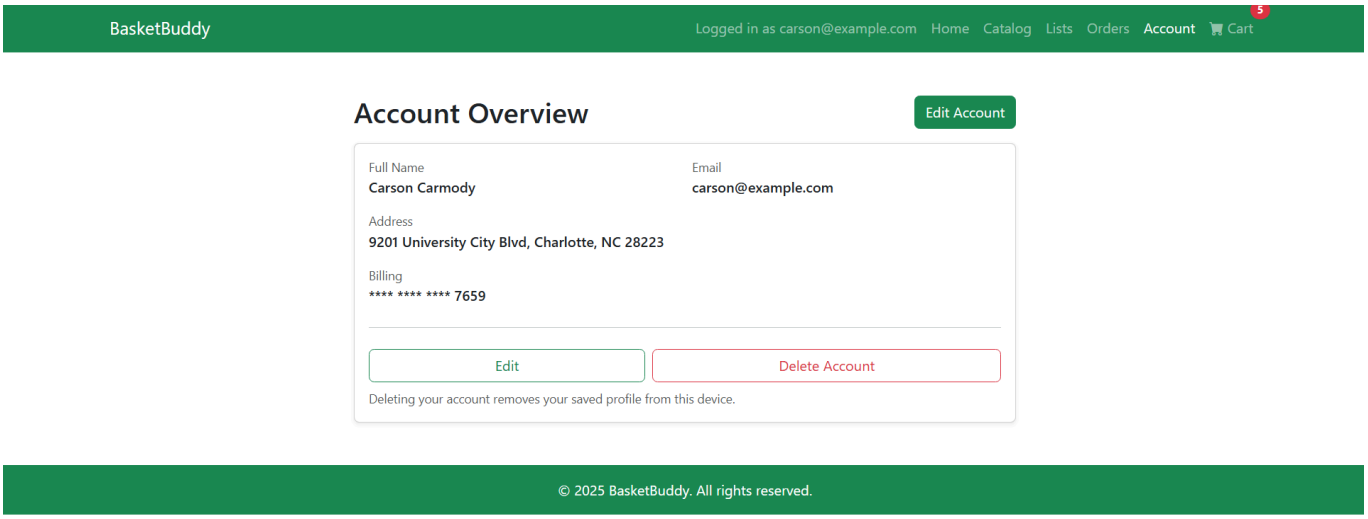
Cart



Login



Account Overview



Orders