**Link to GitHub:** <https://github.com/WilliamsDave/Shopping-AI>

User Manual

**Purpose of the System**

Our program is a shopping AI system that provides catered item recommendations to users based on their current cart and selected aisles. It utilizes market basket analysis and Apriori algorithm association rules to suggest recommended items seamlessly and dynamically.

**Features**

* **Aisle Selection:** Users can select an aisle using the buttons at the top to display the items within that aisle.
* **Item Addition:** Users can add items to the cart by using the cart icon within the item tiles.
* **Cart:** Users can view items within the cart on the right side of the screen. Items can be removed using the “remove” button.
* **Cart Recommendations:** Users can click the “Continue to checkout” button above the cart, which will pop up with item recommendations based on their current cart, which they can add directly to their cart. No checkout screen is implemented as our project is an item recommendation system, not a functioning storefront.
* **Aisle-Based Recommendations:** Within each aisle, the user will see three recommended item tiles, which are created dynamically and can be added directly to the cart. These items update whenever the aisle is changed or certain items are added to their cart.

**Initial Application Setup (Windows)**

1. Clone the repository via the provided GitHub link.
2. Within the local repository initialize a virtual Python environment using the command “python -m venv .venv”.
3. Once created, you can launch the virtual environment using the command “venv/Scripts/activate.ps1”.
4. Install the correct dependency versions using the commands “pip install mlxtend==0.23.1” and “pip install flask==3.0.3” as well as pip install pandas
5. You can now run the app using the command “python app.py” and then click the hyperlink within the console message “Running on <http://127.0.0.1:5000>”. This will bring you to the locally hosted environment.

(Mac/Linux setup should be similar, but commands may differ, this was created and tested in a Windows environment)

Implementation Manual

**Problem statement:**

Create a grocery shopping web application that recommends items based on a predictive artificial intelligence algorithm using user-specific data based on the Apriori Algorithm and Market Basket Analysis.

**System requirements**

* OS: Only tested on Windows
* Browser Compatibility: Chrome, Firefox, and Edge
* Frontend: HTML, CSS, JavaScript
* Backend: Python (Flask framework)
* Database: Pandas with CSV as a dataset
* Other Libraries: Mlxtend for Apriori and association rules
* Dependencies: Python libraries: flask, pandas, and Mlxtend

**Conceptual design**

The design of this web application is based on websites that allow the user to purchase objects online, except it uses predictive AI techniques to recommend items for the user currently in session. Our project has 2 main layers: the frontend and the backend. The user-friendly frontend displays item recommendations dynamically and provides an interactive user interface with cart management. The backend processes user input and generates the recommendations as well as handling the dataset preprocessing and Apriori MBA analysis.

**Functional requirements**

* Display items dynamically within each selected aisle.
* Ability to add and remove items from the cart.
* Generation of recommendations based on the current user cart and using the aisle selection in conjunction with the user’s cart.
* Recommendations are displayed dynamically within each aisle and through a modal when the checkout button is selected.

**Implementation**

Frontend:

* + readJson(): Fetches data from the JSON file and initializes the webpage with information
  + handleSelectChange(aisle): Updates item list and recommendations for currently selected aisle
  + generateItemBlock(): Creates item blocks dynamically based on the JSON information we read previously
  + addToCart(name, price, image) & removeFromCart(name): Adds and removes items from cart
  + getRecommendations(): Retrieves cart-based recommendations from the backend
  + getAisleRecommendations(aisleName): Retrieves cart and aisle-based recommendations from the backend
  + Various other functions are there to assist with display functionality

Backend:

* Data is passed between the front and back end using GET and POST methods
* find\_relevant\_rules(cart, rules): Finds relevant Apriori rules based on the current cart
* find\_aisle\_recommendations(aisle\_name, grocery\_data, cart): Retrieves aisle based recommendations
* get\_top\_consequents(rules, top\_n=3): Retrieves a certain amount of recommended items based on rules found previously

**Testing**

Testing was crucial for our project since it shows accuracy, efficiency, and development over time. This was especially necessary when passing data through the front and backend and ensuring the data was accurate throughout. This thorough testing helped us identify bugs and ensured that the final implementation met the desired quality standards. Testing was necessary within the Apriori algorithm to confirm data in and out was accurate, and also within the communication between front and back end, on top of that testing was required to ensure dynamic item recommendations were accurate and displayed correctly. For future scopes, we think implementing ML techniques in testing can bring up more ideas for future improvements.

**Application Programs**

This section documents the helper programs used within our system that mainly assist in data processing and image creation. The first program processes our grocery dataset CSV into a more usable JSON file, the second retrieves and saves images for grocery items using the Unsplash API.

*Program 1 CSV to JSON Converter:*

This program reads our CSV dataset of grocery items and converts it into a more functional JSON file. Within this items are grouped by their respective aisles and duplicate items are removed. Each item is stored with default details including a placeholder image path which is later updated through the image retrieval program.

Key functions:

* Item Class:
  + Creates item objects with the necessary default attributes
  + Gets called through the process\_items function to create each item from the dataset
* process\_items():
  + Reads from the CSV input file and uses the data to create Item’s via the Item Class
  + Sorts items using their aisles and stores them into a JSON file for further use

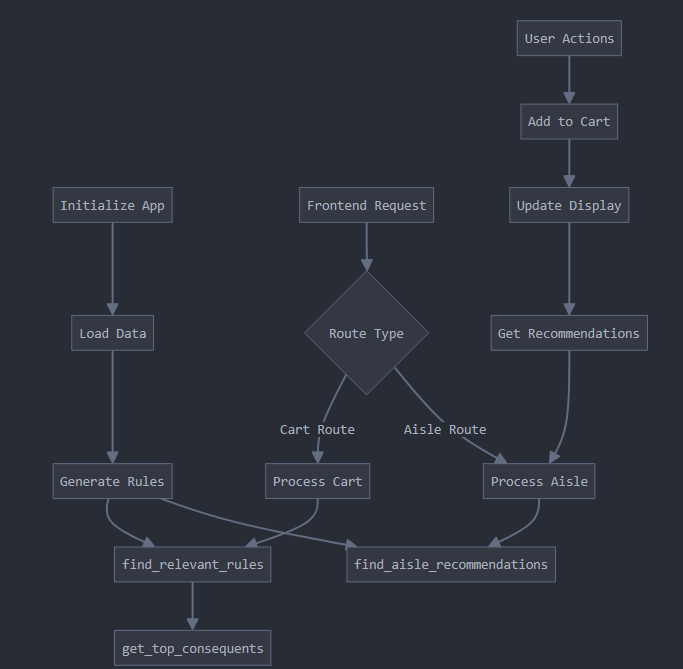
*Program 2 Image Retriever:*

This program downloads images for our grocery items from the Unsplash API and updates the JSON file with corresponding image file paths. This ensures that the images are correctly named and in the appropriate file directory allowing us to continue to use dynamic item displays within our main program. This program has built in limits to comply with the API’s usage limits.

Key functions:

* download\_image():
  + Loops iterating through the JSON data
  + Calls the API to retrieve images
  + Handles images downloads and saves them to our file directory
* JSON Processing:
  + Reads the initial JSON file to decide which images need to be downloaded
  + Updates the JSON file with the newly updated file paths

*Main Program Control Flow:*

****

**User Interface Design**

The user interface is designed as a website that allows users to have real-time predictions of items that they might be interested in buying. From the “Recommended Items” the shopper has the ability to directly add that item to their curt. Each aisle is displayed through an HTML button, and the aisles are based on a column that is contained in the dataset. The majority of this code is done through the UI, where a JavaScript function will turn each item from the dataset into a JSON object, which is then displayed as a picture that can be clicked. Underneath, each picture also has the name of the object from the dataset (taken from the JSON objects list). As an item gets clicked, it is added to the “shopping cart.” The “cart” is essentially a list of concatenated strings that get sent to the back end.

**Implementation Plan**

David Williams - Front-end, Back-end, Data processing

Stela Opingari - Back-end, Project Architecture

Ethan Cooper - Front-end, handled image API calls

Ming Murray - Front-end, Back-end

Sammy Nouadir - Front-end, assisted with image API calls

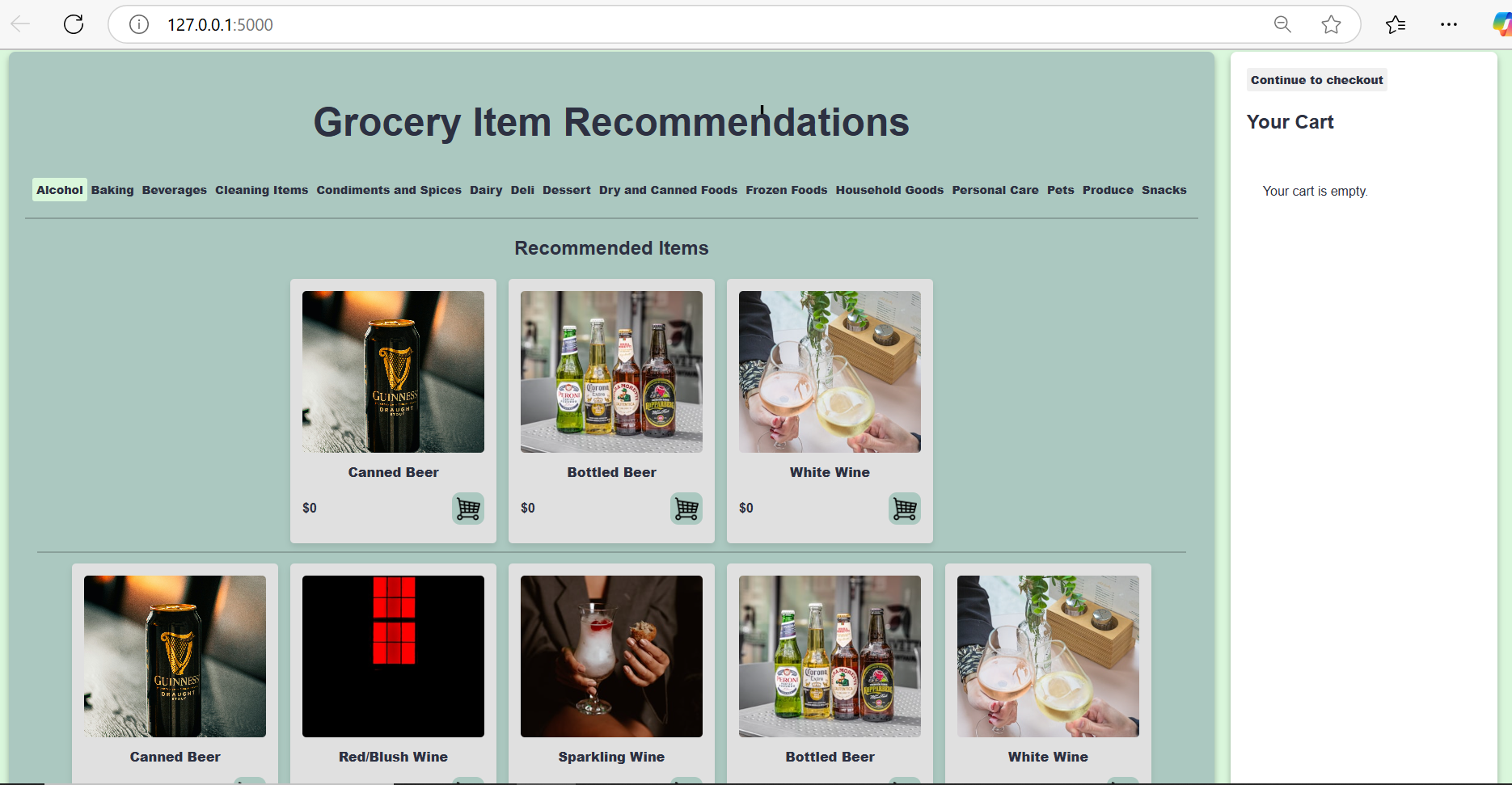
Trevor Small - Back-end, worked on JSON converter

Rituparna Ghosh - Front-end

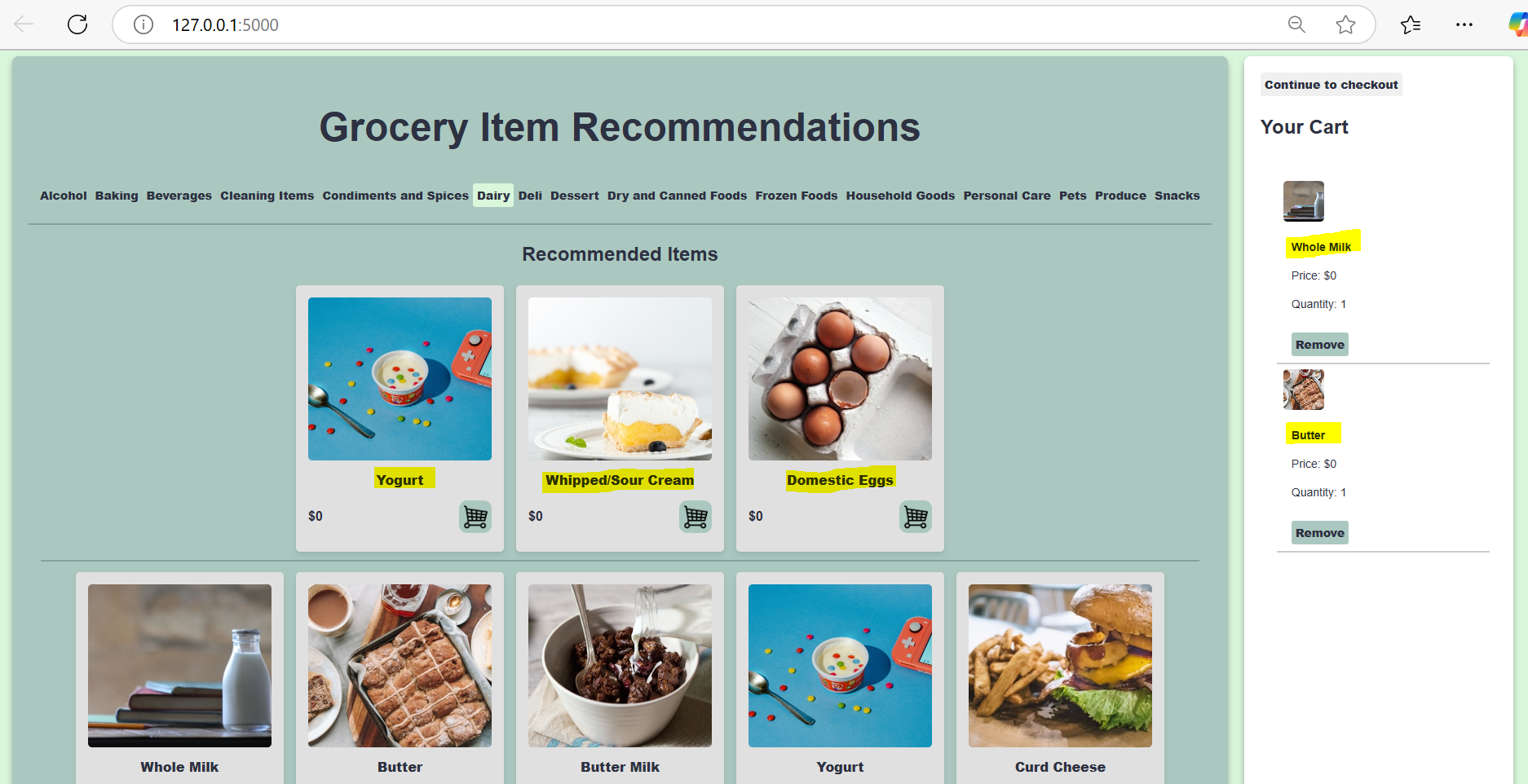
**Code Listing**

<https://github.com/WilliamsDave/Shopping-AI>

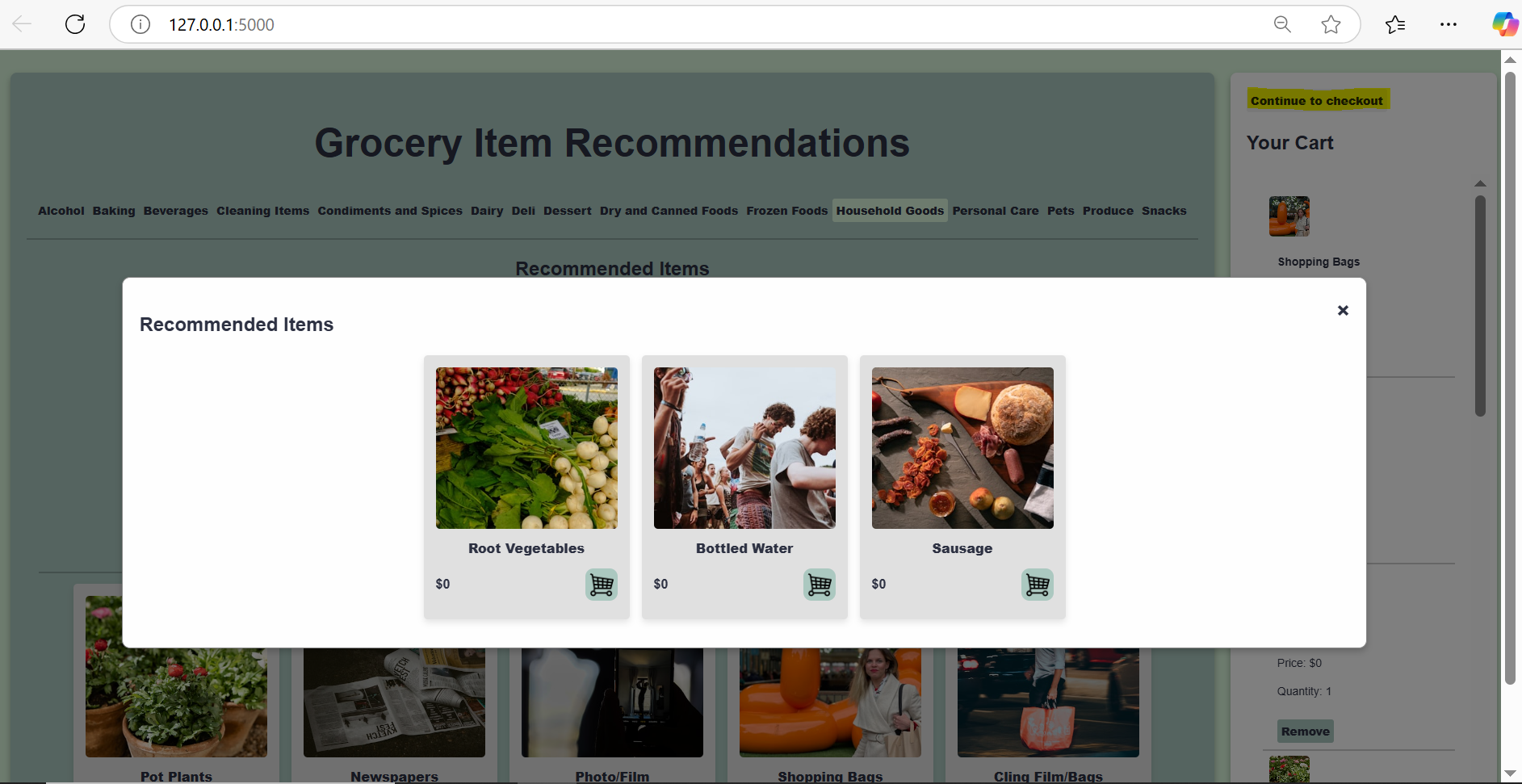
**Sample Output**

****

The first three items under the "Recommended Items" section are dynamically generated based on the current aisle's location. The aisles are arranged alphabetically, with "Alcohol" as the first aisle, which is why items like Canned Beer, Bottled Beer and White Wine are initially recommended. Users can seamlessly navigate between aisles, browse all available items within the selected aisle, and add products to their cart instantly by clicking the cart icon.



As depicted above once the user has added Whole Milk and Butter to their cart the recommended items are updated based on the current items in the cart and the aisle location which in this scenario is Dairy.



After clicking the “Continue to checkout” a pop-up window appears with three additional recommended items based solely on the user’s cart. These items can be added to the current cart by clicking the cart icon.