 TheGroceryStore

The Grocery Store

Documentation

By: Ayaz Nakhuda, David Ferris, Muhammad Alvi

Table of Contents

Purpose	3
What is The Grocery Store?	3
Inspiration for Creating The Grocery Store	3
Timeline	4
Goals	5
SMART Goal #1	5
SMART Goal #2	6
Pages	7
Sign Up	7
Login	8
Products Page	8
Checkout	8
Return Policy	8
Order History	9
Terms Of Service	9
About Us	9
Grocery List Maker	9
Nutritional Facts Provider	9
Grocery Store Finder	10
Functionality	10

Purpose

What is The Grocery Store?

At its core, The Grocery Store is an ecommerce site that allows users to shop for their groceries and various other day to day necessities from the safety of their home during these difficult and unexpected times, but, in reality, The Grocery Store is much more than this. The Grocery Store is the consumer's ultimate grocery shopping tool that provides users with various tools to shop, manage, and plan all their grocery needs in one spot. The Grocery Store combines all the common functionality of an ecommerce site such as, creating and accumulating a shopping cart, ordering items, and viewing previous orders, with additional functionality such as, a built in grocery list builder, a store locator for all your favourite stores, and a nutrition facts page for all your favourite products, to bring consumers comfort throughout the shopping process during this stressful time.

Inspiration for Creating The Grocery Store

The inspiration for this project came from the impact that COVID-19 has had on people's ability to shop for their day to day necessities. Due to COVID-19 the simple task of going to get groceries has become a stressful nightmare for millions of people. This caused us to want to make a tool for users that could bring back some of that simplicity and comfort to the shopping process which we lost. This led us to create The Grocery Store, the consumer's ultimate grocery shopping tool, providing comfort and support no matter which way you choose to shop your necessities.

Timeline

To ensure everything is completed on time, we have a weekly check-in system where we assign tasks on a weekly basis. Here is the 5-week timeline we followed for this project:



Week 1: We all worked on and submitted the project proposal.

Week 2: Research on competitors such as NoFrills, Food Basics, and Loblaws was done

Week 3: Get the HTML markup completed for almost all pages. Header and Footer were made using PHP. The database was set up using MySQL.

Week 4: CSS was implemented in all the pages. The following JavaScript APIs/libraries were tested and used; Ajax, jQuery, HTML Geocoder, Mapbox, and Edamam Food and Grocery Database.

Week 5: In our final meeting, all the pages were put together. We worked on the presentation and rehearsed it. Presentation was done on November 25th, 2020.

Goals

SMART goals are a good way to complete our objectives efficiently while appropriately delegating work to the team.

SMART Goal #1

What: Design a footer that is...

- (a) visually appealing & responsive
- (b) links to other pages
- (c) provides helpful information

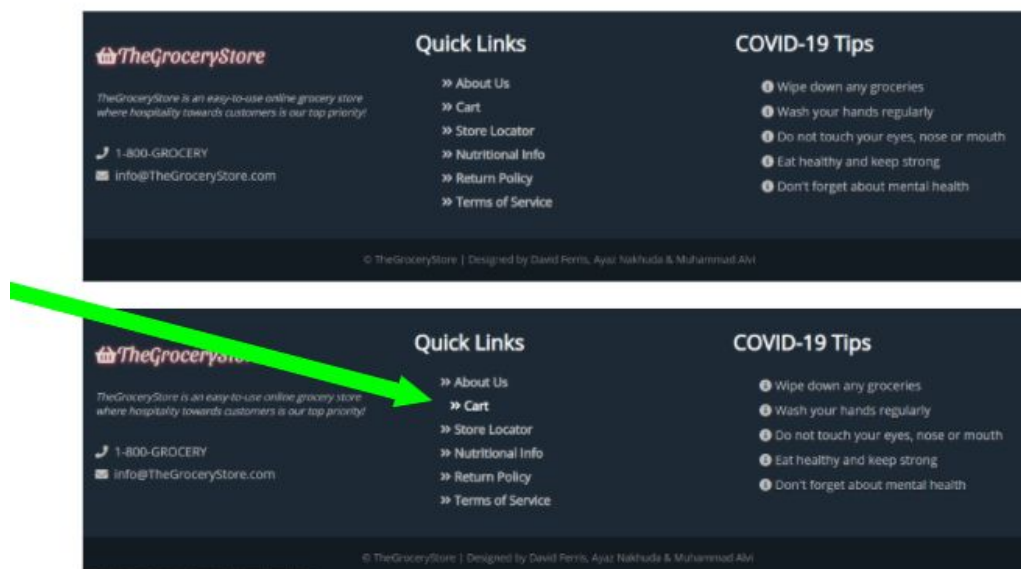
Who: Each part assigned to a different group member.

Where: Bottom of every page on website.

When: Have it done by the end of 4th week.

Why: It distinguishes us from our competitors, and improves overall user experience. Gives the website a finished look and feel.

This goal was achieved as the group was able to produce a professional-looking footer that complements the website. The quick links have smooth animations.



SMART Goal #2

What: Improve our website accessibility by...

- (a) providing nutritional facts
- (b) giving tips to deal with COVID-19
- (c) helping customers find nearby stores

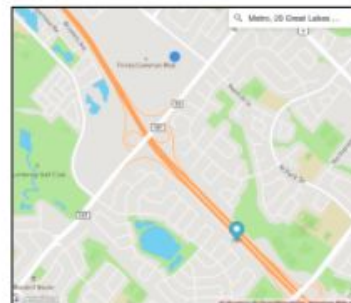
Who: Each part assigned to a different group member.

Where: Separate pages scattered throughout the site, and on the footer.

When: Have it done by the end of 3rd week.

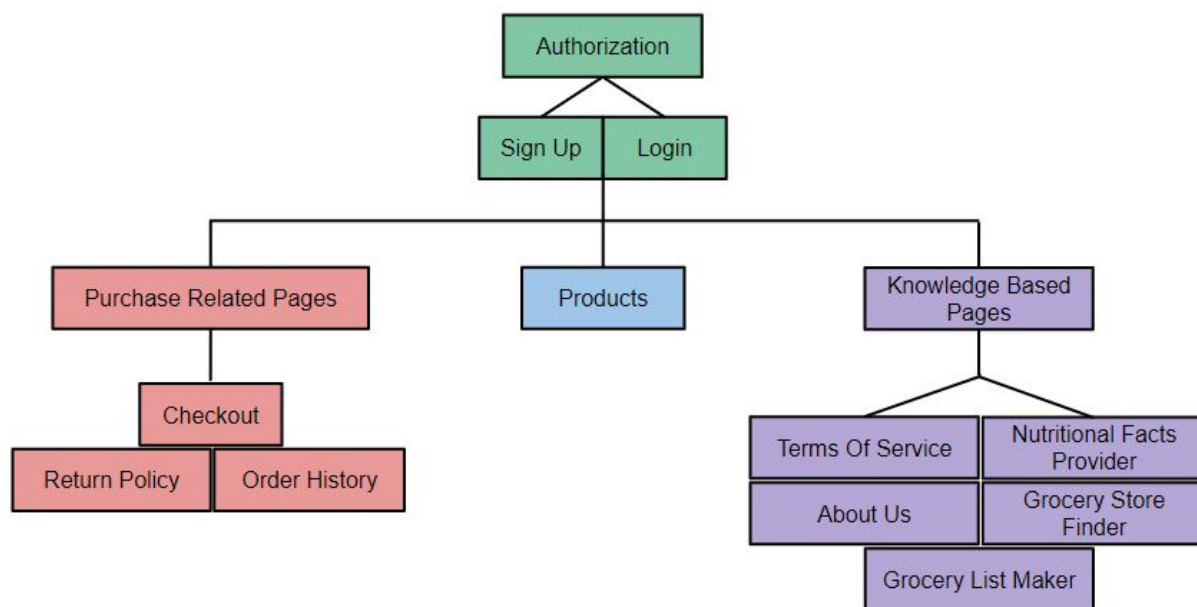
Why: Due to COVID-19, the demand for accessibility needs has increased, and our website must be able to fulfill them. People are concerned for their health, and they want to make sure they know how to get to the store safely.

We succeeded in improving the accessibility of our website by adding COVID-19 tips, Edamam API, and Mapbox API (See “Functionality”).



Pages

The following sections of this document will give brief descriptions of each page. For more specific information behind the functionality of the pages, refer to the “Functionality” section. In addition, all the files with code have been commented to improve readability. Here is the overall structure of the website:



Sign Up

The sign up page asks the user for their username, email, and password. The password is then encrypted using the built-in [password_hash](#) function which hashes a password using the bcrypt algorithm that is provided by the php language. All of this information is stored in a MySQL database.

Login

The login page requests the user to input their username and password. The MySQL database is checked to see if the account exists, using prepared statements for security purposes, and if the details given are correct. If these conditions are met, the user gets access to the website and is redirected to the products page.

Products Page

This is the main landing page of the website. It displays the featured products. The reason behind this is purely monetary. The first thing our customers are exposed to are our products, thus they would be more inclined to add products to their cart quickly.

Checkout

Once the user has added products to their cart, they can see them on the checkout page. The checkout page is very important for businesses as it is where the customer pays for the items. It is important to keep the page simple and accessible to ensure payments are smoothly sent to the business.

Return Policy

In the online shopping era we live in, customers want the ability to return or exchange products they may not be satisfied with. Our comprehensive return policy mentions all the details the customer would want to know relating to this process.

Order History

Having the ability to view previous orders is very useful for customers to keep a record of past purchases. The customer can use their order number to search for a specific order. Orders are stored in the MySQL database.

Terms Of Service

In the modern day, with the growing concerns of security and transparency of websites, we made sure to develop a terms of service for our users to refer to when using our website. Whether or not they decide to read it is up to them, but our responsibility is to have it available.

About Us

The about us page provides a brief summary of the company behind the website.

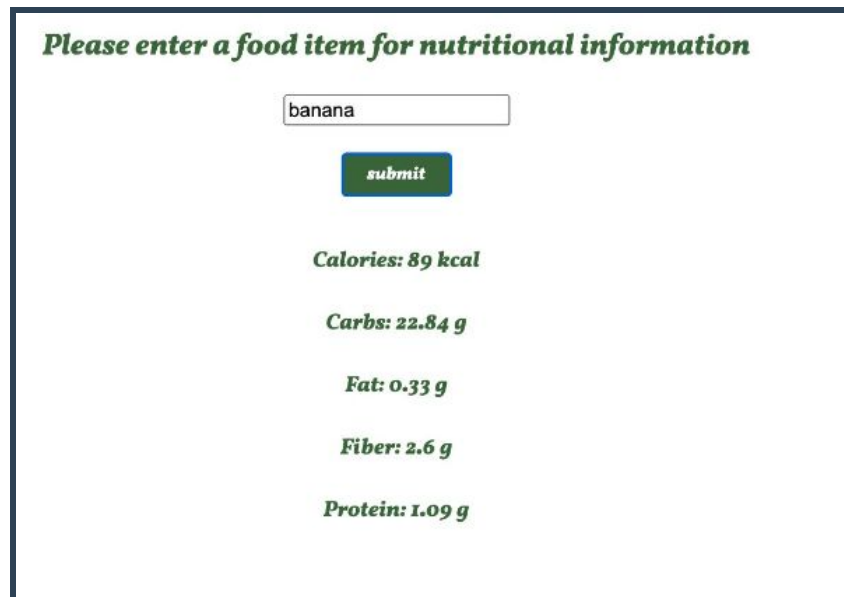
Grocery List Maker

This page allows users to create a shopping list. They can check off items as they purchase them. When they are done with the list, they can double click the items to remove them entirely off the list.

Nutritional Facts Provider

A user is able to search for a food item which gets sent to the Edamam Food and Grocery Database API. A JSON response body will then be parsed

retrieving nutritional information for the searched food item. The nutritional information will then be displayed to the user.



A screenshot of a web form titled "Please enter a food item for nutritional information". The form contains a text input field with the word "banana" entered, a green "submit" button, and a list of nutritional values for banana: "Calories: 89 kcal", "Carbs: 22.84 g", "Fat: 0.33 g", "Fiber: 2.6 g", and "Protein: 1.09 g".

Please enter a food item for nutritional information	
Input:	banana
Submit:	submit
Calories:	89 kcal
Carbs:	22.84 g
Fat:	0.33 g
Fiber:	2.6 g
Protein:	1.09 g

Grocery Store Finder

This page allows users to search for a store, address or postal code near their location. Mapbox API is used for displaying the map and retrieving the search results. Search results will also appear to the right of the map.

Functionality

For functionality APIs were used. The first API used was the HTML Geocoder API. This API asks the user to use their address. Then this API will send a response body that contains the user's latitude and longitude. The latitude and longitude will then be sent to the next API namely, Mapbox API. Below is a demonstration of how the HTML Geocoder API was implemented:

Calling the method to retrieve the latitude and longitude.

Making the map.

Centering on the user's position.

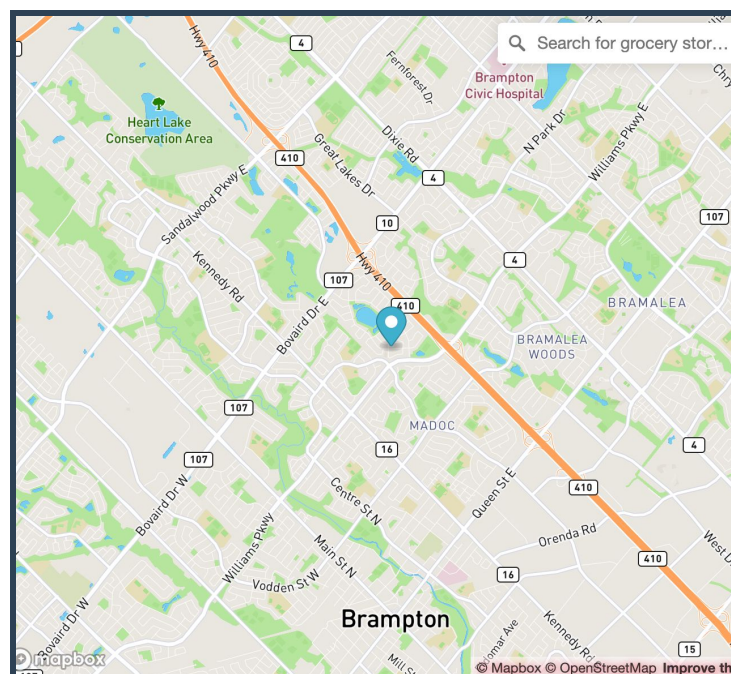
```

<script>
//below is a javascript feature
navigator.geolocation.getCurrentPosition(function(position) {
  var lon = parseFloat(position.coords.longitude);
  var lat = parseFloat(position.coords.latitude);
  mapboxgl.accessToken = 'pk.eyJ1IjoiYX8pY29kaW5nIiwiaYSI...ifQ.v4F1ym2KrUWmJ1-87odWwQ';
  var map = new mapboxgl.Map({
    container: 'map', // Container ID
    style: 'mapbox://styles/mapbox/streets-v11', // Map style to use
    center: [lon, lat], // Starting position [lng, lat] // call the function and have it return the latitude and longitude
    zoom: 12, // Starting zoom level
  });

```

Mapbox API will take the latitude and longitude and center the map on the user's location. The map includes a search bar where the user can search for any store locations, addresses and postal codes near their location. Furthermore, the map allows for markers to be placed on the user's location as well as any locations that the user has searched up.

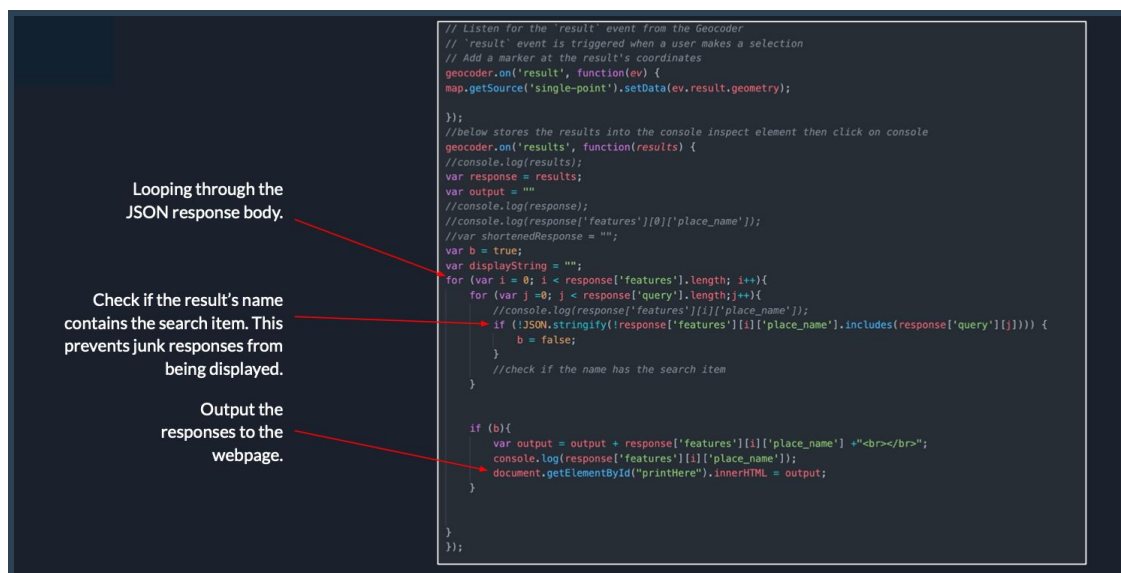
Here is how the map looks like:



Lastly, a JSON response body with the search results is returned from the API and gets displayed to the user as such:



Below is a snapshot of how Mapbox API was implemented:



Below is a sample of the JSON response body:

```
{
  "type": "FeatureCollection",
  "features": [
    {
      "geometry": {
        "type": "Point",
        "coordinates": [
          -76.9750541388,
          38.8410857803
        ]
      },
      "type": "Feature",
      "properties": {
        "description": "Southern Ave",
        "marker-symbol": "rail-metro",
        "title": "Southern Ave",
        "url": "http://www.wmata.com/rider_tools/pids/showpid.cfm?station_id=107",
        "lines": [
          "Green"
        ],
        "address": "1411 Southern Avenue, Temple Hills, MD 20748"
      }
    },
    {
      "geometry": {
        "type": "Point",
        "coordinates": [
          -76.935256783,
          38.9081784965
        ]
      },
      "type": "Feature",
      "properties": {
        "description": "Deanwood",
        "marker-symbol": "rail-metro",
        "title": "Deanwood",
        "url": "http://www.wmata.com/rider_tools/pids/showpid.cfm?station_id=65",
        "lines": [
          "Orange"
        ],
        "address": "4720 Minnesota Avenue NE, Washington, DC 20019"
      }
    },
    {
      "geometry": {
        "type": "Point",
        "coordinates": [
          -76.9281249818,
          38.9786336339
        ]
      },
      "type": "Feature",
      "properties": {
        "description": "College Park",
        "marker-symbol": "rail-metro",
        "title": "College Park",
        "url": "http://www.wmata.com/rider_tools/pids/showpid.cfm?station_id=79",
        "lines": [
          "Green"
        ],
        "address": "4931 Calvert Road, College Park, MD 20740"
      }
    }
  ]
}
```

The final API used is Edamam Food and Grocery Database API. Users can enter a food item into a textbox and upon clicking the enter button, the text will be sent to the API. The JSON response body will then be parsed and nutritional information for the searched food item will be displayed to the user.

Below is a snapshot of how the Edamam Food and Grocery Database API was implemented:

The image shows a code editor with JavaScript code for an API call. Three red arrows point from text annotations to specific parts of the code:

- Call the API and send the search text.** Points to the `$.ajax(settings)` call.
- Check if the response object equals the searched item.** Points to the `if (s2 == x) {` condition.
- Check if this object has a specific nutrient.** Points to the `if (response['hints'][i]['food']['nutrients'].hasOwnProperty('ENERC_KCAL'))` condition.

```
<script type="text/javascript">
$(document).ready(function(){

    $('#button').click(function(){
        var x = document.getElementById("text").value;
        const settings = {
            "async": true,
            "crossDomain": true,
            "url": "https://edamam-food-and-grocery-database.p.rapidapi.com/parser?ingr="+x,
            "method": "GET",
            "headers": {
                "x-rapidapi-key": " ",
                "x-rapidapi-host": " "
            }
        };

        $.ajax(settings).done(function (response) {
            console.log(response);

            var displayString = "";
            for (var i = 0; i < response['hints'].length; i++){

                s = (JSON.stringify(response['hints'][i]['food']['label']));
                s2= s.slice(1,s.length-1);

                if (s2 == x) {
                    console.log(response['hints'][i]['food']['nutrients'])

                    if (response['hints'][i]['food']['nutrients'].hasOwnProperty('ENERC_KCAL')){
                        console.log(response['hints'][i]['food']['nutrients']['ENERC_KCAL']);
                        displayString+="Calories: ";
                        displayString+= response['hints'][i]['food']['nutrients']['ENERC_KCAL'];
                        displayString+=" kcal";
                        displayString+="<br><br>";
                    }
                }
            }
        });
    });
});
```

Below is an example of the JSON response body:

```

    nutritionalPage.html:56
  ▼ Object
    ▶ hints: (22) [{...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}, {...}]
    ▼ parsed: Array(1)
      ▼ 0:
        ▼ food:
          category: "Generic foods"
          categoryLabel: "food"
          foodId: "food_a1gb9ubb72c7snbuxr3weagwv0dd"
          image: "https://www.edamam.com/food-img/42c/42c006401027d35add93113548e..."
          label: "apple"
        ▼ nutrients:
          CHOCD: 13.81
          ENERC_KCAL: 52
          FAT: 0.17
          FIBTG: 2.4
          PROCNT: 0.26
          __proto__: Object
        uri: "http://www.edamam.com/ontologies/edamam.owl#Food_apple"
        __proto__: Object
      __proto__: Object
    length: 1
    __proto__: Array(0)
    text: "apple"
    __links: {next: {...}}
    __proto__: Object

```