Software Requirements Specification For

09/19/2024

Team Members: Danny Elhamalawy, Kevin Gawinauth, Alex Pateroulakis, Elijah Philip, Gayathri Suresh

TABLE OF CONTENTS

1. Introduction:

The GreenGrocer app is a grocery shopping application that allows users to browse different product categories, view product details, and manage their shopping carts. The app is designed with Bootstrap, HTML structure, and Python code to provide a simple yet effective user interface for users to select items such as fruits, meats, drinks, and frozen foods. The application will soon integrate with Kroger's API to fetch real-time grocery store data, providing users with up-to-date product information, including availability, pricing, and item details from Kroger's vast inventory. This app is designed to improve the grocery shopping experience by offering convenience and efficiency through easy-to-navigate features.

1.1 Purpose:

This paper serves as a guide for the technical requirements and specifications needed to construct the GreenGrocer grocery shop application. The goal of this project is to develop a platform that allows users to interact with the Kroger API to retrieve pricing and product information while also viewing store locations, perusing product categories, adding items to a cart, and managing their favorites. The program supports basic e-commerce features like cart management and user authentication.

1.2 Project Scope:

The project scope is building a user-friendly online application that interfaces with the Kroger API to receive and display product and price information particular to individual stores is part of the project's scope. This will allow users to: Register, access, and control their accounts. Other functions include seeing the open stores and the things that they offer, putting goods in a cart, keeping track of their top picks, completing their shopping list by using a virtual cart to manage it and illustrating product availability and pricing variations between store locations as the application will concentrate on five Long Island sites.

1.3 Target Audience:

This document aims to provide developers involved in the project with a thorough understanding of the functionality and technology stack and for project managers to monitor advancement. This document is also intended for End Users who are curious about the technical aspects involved in creating the GreenGrocer application. The application infrastructure is put up and maintained by system administrators.

1.4 Document Conventions:

The document uses common conventions for naming:

Words surrounded by “ ” indicate buttons.

2. Technology and the Operating Environment:

2.1 Product Features:

GreenGrocer offers a variety of features that streamline the grocery shopping process:

- **Category Browsing**: Users can browse through product categories like Fruits, Meats, Drinks, and Frozen items. Each category displays related products with images, names, and prices.
- **Cart Management**: Items can be added or removed from the cart, and the app dynamically updates the total price. A cart button on the home page displays the current number of items in the cart.
- **Product Details**: Each item has a detailed display, including an image, description, and price. The app plans to integrate real-time product data using Kroger’s API to ensure users receive accurate product information.
- **Checkout**: Users can review their cart, adjust items, and view a total cost before proceeding to checkout.

2.2 Product Description:

2.2.1 Front-End:

The GreenGrocer application's front end is constructed using CSS3 and HTML5 for markup and styling. Bootstrap 5 is employed to create a responsive layout and UI elements, while JavaScript handles client-side logic and interactivity. Dynamic content is rendered from the backend using Jinja2 templating. The front-end user interface allows users to easily explore product categories, navigate store locations, and interact with their cart and favorites.

2.2.2 Rear End:

The Flask web framework and Python are used in the development of the application's back end. Routing, session management, and communication with the Kroger API are all handled using Flask. It also manages user authentication and session management, using Flask-Login. For secure forms with CSRF protection, use Flask-WTF. For sign-ups and logins, basic user data will be stored in SQLite (or a file-based JSON system).

2.2.3 API:

The Kroger's product and location API is integrated by the application to retrieve: Product information unique to certain regions. Details about the store, such as product availability and costs specific to a given area. Requests are used to make API calls to retrieve: Products sorted by location and category. Pricing details are based on the location ID of that particular store.

2.3 User Classes and Characteristics:

The application includes three main types of users, all with different functionalities based on their intended uses for the program. The types include Unregistered User, Registered User, and Guest.

- **Unregistered Users:** Users who plan on registering and creating an account. Users can become members by signing up with their email and creating a password which they will have to type twice. After completing the registration, users will have to verify their emails to activate their accounts and log in to the application.
- **Registered Users:** Returning users who have previously created and logged into their accounts. Once verified, members will have access to a range of exclusive features such as transaction/order history, promotions and coupons, and the ability to favorite products.
- **Guest:** Users can browse through a collection of items and make purchases without creating an account. However, they won't be able to view their order history, save their address, and cannot access promotional offers along with other features that registered users have access to.

2.4 Operating Environment:

- **Supported Platforms:** The application will be able to operate on Windows, macOS, and Linux with support for Chrome, Firefox, Safari, and Edge browsers.
- **Hardware Requirements:** The application is designed to run efficiently on standard consumer devices with at least 2 GB RAM and basic connectivity.
- **Network Requirements:** A stable internet connection is required for the application to function properly and efficiently. More specifically for signing up and logging in, managing the cart (adding and removing items), and completing the checkout process.
- **Server Environment:** The application's server will be hosted on Github Pages which will manage content that changes with the user's interaction, and operate efficiently regardless of user traffic levels.

2.5 Design and Implementation Dependencies, Constraints, and Assumptions:

2.5.1 Dependencies:

- **Third-Party API Dependency:** The application's integration with the Kroger API means that it is dependent on its availability. If the Kroger API is shut down or changes, then the functionality of the website may be impacted.

2.5.2 Constraints:

- **Limited Mobile Compatibility:** The application is designed specifically for desktop and laptop devices and is not optimized for mobile platforms. Users who try accessing the application on a mobile device may encounter layout issues and reduced functionality.
- **Screen Resolution:** The application is optimized for larger devices (e.g. 1280x720 resolution or higher). Users with smaller resolution devices may experience a less optimal visual experience.

2.5.3 Assumptions:

- **Sufficient Hardware:** The device that the user is running the application on has sufficient hardware (2 GB RAM) and adequate processing power to ensure the best possible user experience.
- **Stable Connection:** Users will have access to a stable connection while using the application so there can be a smooth experience while adding and removing items to the cart and checking out.
- **Availability of Kroger API:** The Kroger API will remain available and provide accurate real-time data for product listings, inventory, and pricing.

2.6 User Documentation:

The application has a FAQ section that answers common questions that users may have. Users will also have access to email support for any issues or inquiries about the application.

3. System Users and Features:

3.1 User Authentication and Registry:

3.1.1 Description:

All three types of users will be able to access this feature, which retrieves or creates the account information of the user and grants them access to the app's features. However, the services of the feature are determined by user class. The system differentiates the features' applications based on user type. The following aspects of the GreenGrocer application facilitate user authentication:

- Login Page: Registered users can access the login page by entering their login credentials. The system ensures that only authenticated users may access specific functionalities, like adding things to the cart, by using Flask-Login to manage user sessions.
- Sign-up Page: By entering the necessary account information (name, address, phone number, special username, password), users can register for a new account. Passwords are kept safe in encrypted formats.
- Guest Login: Users do not need to create an account to browse the program. They can click a button on the Login page and access the Homepage of the application.

3.1.2 System-user Interaction:

Unregistered User:

1. The user clicks the Sign-Up button.
2. The system redirects users to a page where they can enter the appropriate information to create an account.

3. The user submits new account information such as email address, name, phone number, and password.
4. The system redirects them to the login page. The process continues as a Registered User.

Registered User:

1. Users enter their username and password into the appropriate fields.
2. If either entry is incorrect, the system presents an error.
3. If the user forgot their login information, they may click “Forgot Password”.
4. The system directs them to a page to reset details.
5. If their entry is correct, the system retrieves and loads the user’s profile data.
6. The system presents a home page.

Guest:

1. The user clicks the “Continue as Guest” button.
2. The system presents a home page.

3.1.3 *Functional Requirements:*

R-1: Create Sign Up/Login page with allocated space for Registered Users to enter their username and password

R-2: Create a Login button to submit completed login information

R-3: Present appropriate error messages when incorrect login information is submitted

R-4: Create a Forgot Password link for Registered Users to reset their password

R-5: Produce a new Forgot Password interface for Registered Users to change their appropriate login credentials

R-6: Create a Sign-Up button for Unregistered Users to make an account

R-7: Create a new Sign-Up interface for Unregistered Users with allotted space to enter information and establish login credentials

R-8: Create a Continue as a Guest button for Guest Users to continue to the application home page

3.2 Home Page:

3.2.1 *Description:*

The home page is presented to all users and serves as a central page to easily navigate through different pages and tabs of the application. It is the starting point for using and accessing all functionalities of the program. It has the following key features:

- **Navigation Bar:** This is a high-priority feature of the home page. It displays buttons to allow the user to switch to the cart, favorites, and promotions tabs.
- **User Profile:** An icon representing the user's profile is displayed at the top left corner of the page, allowing the user to access account details, purchase history, and FAQ (as described in Sections 2.6 and 3.3).
- **Store Selection:** Allows users to select the store location they wish to shop from. Accordingly, a change in location selection will change the availability of products (as described in Section 3.4).
- **Search Bar:** Users can search for an item they are looking for. The feature will return items that match the same keyword or return similar items in case the intended item is not available (as described in Section 3.6).

- **Product Categories:** Users can browse food categories like fruits, meats, vegetables, and drinks displayed as tiles (as described in Section 3.5). Users can favorite items by clicking the star icon in the corner of each product tile.

3.2.2 *System-User Interaction:*

This section will only describe the system-user interaction for the navigation bar. The interactions for the other features on the home page will be explained in their respective sections as mentioned above in Section 3.2.1.

All Users

1. The user clicks the “Cart” button on the Navigation Bar.
2. The system directs them to the Cart tab.
3. The user clicks the “Promotions and Coupons” button on the Navigation Bar.
4. The system directs them to the Promotions and Coupons tab.
5. The user clicks the “Home” button on the Navigation Bar.
6. The system directs them to the Home tab once again.

Registered Users

1. The user clicks the “Favorites” button on the Navigation Bar.
2. The system directs them to the Favorites tab.

3.2.3 *Functional Requirements:*

This section will only describe the functional requirements for the navigation bar. The functional interactions for the other features described on the home page will be explained in their respective sections as mentioned above in Section 3.2.1.

R-9: Create a navigation bar that appears on the bottom of all pages of the application.

R-10: Create clickable buttons on the navigation bar for Home, Favorites, Cart and Promotions, and Coupons such that once the icon is clicked, it redirects the user to the appropriate tab.

R-11: Add appropriate icons for the “Home”, “Favorites”, “Cart”, and “Promotions and Coupons” buttons.

R-12: Ensure the Favorites tab only appears for Registered Users.

3.3 Account Management:

3.3.1 *Description:*

The account management page is only available to Registered Users and its main function is to allow users to see and change their personal account details. Additionally, users can see their history and edit their accounts as they see fit. It has the following key features:

- **Profile Information:** Users can update personal information such as their name, email, phone number, address, and password.
- **Saved Information**: Members can choose to save and edit their address, phone number, payment method, and payment information.
- **Order History:** Members can view their past purchases which includes the date the purchase was made, method of payment, items purchased, and any discounts applied.
- **Loyalty and Rewards:** Users can view how many points are obtained and their equivalence in USD.

3.3.2 *System-User Interactions:*

1. The user clicks on the “Account” button on the top left side of the Home page.
2. The system redirects users to the Account Management page.
3. The user clicks on “View and Edit Account Information”.
4. The system presents the user’s profile information.
5. Users can view their account information. The user clicks the “Edit” button on the top right corner.
6. The system allows users to edit their information.
7. Users change their information and click the “Save” button on the bottom middle of the page.
8. The system saves the newly entered information and prevents users from editing further information. The system presents the Account Management page.
9. The user clicks “View Order History”.
10. If the customer has made previous orders, the system presents all receipts of past orders made by the customer.
11. The user clicks on a receipt.
12. The system displays the order information including price and coupons applied.
13. The user clicks the “Done” button.
14. The system redirects users to the account management page.
15. If the customer has not made previous orders, the system shows an error message and redirects the user to the account management page.
16. Users click on the “Loyalty and Rewards” button.
17. The system redirects the user and displays the points the user currently has.

3.3.3 *Functional Requirements*

R-13: Create an Account Button on the top left corner of the Home Page.

R-14: Create an Account management Interface with 3 buttons for “View and Edit Account Information”, “View Order History”, and “Loyalty and Rewards”

R-15: Create a “View and Edit Account Information” page that displays the account profile information

R-16: Includes an “Edit” button that allows the user to change their information.

R-17: Include a “Save” button that allows users to save their edited information.

R-18: Create a “View Order History” page that shows the past order receipts of the customer

R-19: Create a “Loyalty and Rewards” page that shows the customer’s points

R-20: Create a “Done” button on the View and Edit Account Information”, “View Order History”, and “Loyalty and Rewards” pages that allows users to return to the Account Management Page

3.4 Location Management:

3.4.1 *Description*

This feature allows all users to pick the location of the store they wish to order from using a dropdown menu. Due to changes in supply between regions, the availability of groceries will be affected by changes in location. The GreenGrocer application is limited to Long-Island Kroger’s locations as of this version.

3.4.2 *System-User Interactions:*

1. The user clicks the “Select a Location” button.
2. The system opens a drop-down menu including the different available locations.
3. The user selects a certain location from the menu.
4. The system registers the change and internally changes the grocery item information according to the inventory of the selected location using Kroger’s API.

3.4.3 Functional Requirements

R-21: Implement a store selection dropdown selection that allows users to select from several different stores preferably Long Island-based locations.

R-22: Link Kroger Location API to the dropdown menu such that a location change reflects a change in the grocery product data displayed

3.5 Product Categories:

3.5.1 Description

This feature is present on the Home page and takes up a majority of the screen. This feature allows users to easily navigate through categories to find products and is available to all users. Based on the location and availability, the products will be organized into clearly defined categories as per the Kroger's API such as:

- **Food & Beverages:** Fruits, Vegetables, Dairy, Meat, Snacks, etc.
- **Household Items:** Cleaning Supplies, Paper Products, Detergents.
- **Personal Care:** Shampoo, Soap, Skincare.
- **Baby Products:** Diapers, Baby Food.
- **Pet Products:** Pet Food, Pet Accessories.

3.5.2 System-User Interactions:

1. The user is presented with the Home page. The user clicks on a category button.
2. The system opens a page that includes the multi-level categories (e.g., Dairy → Milk, Cheese, Butter).
3. The user clicks on a subcategory.
4. The system opens the subcategory and displays available products based on location.

3.5.3 *Functional Requirements:*

R-23: Display category buttons for users to browse products on the Home page by certain categories, such as Fruits, Drinks, Frozen, and Meats.

R-24: Display product categories with icons or images to enhance usability.

R-25: Display subcategory buttons once a category is chosen

R-26: Provide a "See All" button for viewing the full product directory.

3.6 Product Search:

3.6.1 *Description*

This feature is a high-priority functionality of the application available to all features. It is also on the Home page and allows the customer to search for a certain product by name in a search bar. Users can input product names, categories, and brands through the search bar. The auto-complete feature suggests products based on partial input. The search bar will save recent searches for easy user access and navigation. Other search filters include price range, availability (to show in stock only), and dietary preferences (e.g., gluten-free, vegan, organic). Once the user has searched for their item, this feature provides the functionality to sort the results by relevance, price (low to high, high to low), rating/popularity, and discount/promotion. The results of the search are displayed on the screen as a tile including important information such as name, image, price (with and without discount), unit/weight, brand (if applicable), availability (in stock or not), and customer ratings.

3.6.2 *System-User Interactions*

1. The user enters the item name into the search bar and edits the search by using a search filter.
2. The system uses the Kroger's API to retrieve items that match the search criteria and displays the appropriate items as tiles.

3. The user opens the dropdown menu to sort the searches using the filters mentioned above.
4. The system sorts the searches by those criteria.
5. The user clicks on the product tile that fits their search.
6. The system opens a detailed page that includes product information and related items.
7. The user selects an item.
8. The system opens the product page.
9. The user views the items and presses the “Add to Cart” button.
10. The system adds the items to the cart.

3.6.3 *Functional Requirements*

R-27: Create a search bar that allows users to search products by product names, categories, and brands.

R-28: Add functionality that allows search history to be accessed by the user and select past searches

R-29: Create a section for users to select the search filters such as price range, availability (to show in stock only), and dietary preferences (e.g., gluten-free, vegan, organic)

R-30: Create sorting filters to sort the products by price, availability, discounts, and ratings.

R-31: Create product tiles that display product name, image, price (with and without discount), unit/weight, brand (if applicable), availability (in stock or not), and customer ratings using information from the Kroger’s API.

R-32: Create a display page for each product that includes a full product description, nutritional information, ingredient (if applicable), user reviews, availability in-store or for delivery, and similar and alternative products.

R-33: Create an “Add to Cart” button on product pages

3.7 Checkout:

3.7.1 Description:

Checkout is a high priority for the GreenGrocer application. The functionalities of the checkout vary between Registered Users and Guest Users, as guest users cannot save their name and home address or get access to loyalty points/coupons during the checkout process. From the Cart tab, users can click the “Checkout” button to enter the checkout page. The page displays a list of products added to the cart, including additional information regarding the cost and quantity of the item. Using buttons, users can modify the product quantity, and remove products from the cart, and if they are registered users, they can save items for later in favorites. Once finalized, users can move onto the order finalization page that displaces a detailed breakdown of the total cost as well as a designated area to enter promotions and discount codes. Guests are provided with the option to register for an account to earn points on the order and for easier future checkouts. Lastly, users move onto the payment page to enter their payment details (e.g. credit card information) and select the pick-up or delivery to get their order. Due to the scope of the project, GreenGrocer will not include financial/transactional functionalities.

3.7.2 System-User Interactions

1. The user clicks the “Cart” button in the navigation bar.
2. The system redirects the user to the cart page.
3. The user clicks the “Checkout” button.
4. The system displays the checkout page.
5. The user edits the cart. The user clicks the “Payment” button.
6. The system saves the changes made to the order and proceeds to display the payment page.

7. The user enters the payment information and selects a delivery option for the order.
8. The system saves and “secures” the payment (see section 3.7.1) and processes the order.

3.7.3 *Functional Requirements*

R-34: Create a cart counter at the bottom of the navigation bar that updates when adding items to the cart.

R-35: Create a cart page that includes the product name, image, price, quantity, total price for each item (quantity x price), and availability status (in stock/out of stock).

R-36: Implement a “Checkout” button that allows the user to proceed from the card to checkout to finalize the order.

R-37: Provide -/+ buttons on the checkout page to allow users to edit product quantity and remove items.

R-38: Enable registered users to add products to their favorite section of the 3-button layout functionality to save products for later.

R-39: Create a button for guests to select to make a registered account and link to the account creation page featured in User Authentication and Registry (Section 3.1.3)

R-40: Create a “Payment” button that displays the payment page.

R-41: Create a section for users to input discount codes and promotions.

R-42: Create a receipt that displays a detailed breakdown of the total cost, including the subtotal, taxes (based on user location), and discounts applied.

R-43: Create a section for users to input their credit card and payment information.

R-44: Create an option for users to click Delivery or Pick-up to receive their order.

3.8 Order Summary:

3.8.1 *Description*

This feature allows Registered Users to receive a summary of their orders and keep track of their transactions. Users will be emailed a receipt of their order, tracking number, and details regarding the pick-up/delivery of their purchase. The order summary will not be provided to Guests.

3.8.2 *System-User Interactions*

1. The user clicks the “Submit and Confirm” button.
2. For Guest users, The system submits the order and displays a dialog box with an order success message with the order number, delivery/pick-up details, and receipt. For Registered Users, the system submits the order and displays the dialog box like for the Guest User, but also sends the user a summary of their order with transactions.
3. The user receives the email and views the details of their order.

3.8.3 *Functional Requirements*

R-45: Create a “Submit and Confirm” button to submit the payment information and order.

R-46: Generate an order number and transaction number for each order.

R-47: Create a dialog box that contains the order number, estimated delivery or pickup time, and payment confirmation (transaction ID, amount paid).

R-48: Add a functionality where the system sends Registered Users an email that includes all the same information as the dialog box mentioned in R-47.

3.9 Loyalty, Rewards, and Promotions:

3.9.1 *Description*

This feature allows Registered Users to attain loyalty points from making purchases and these points allow users to access additional discounts and promotions. The amount of points

earned is based on how much they spend ($1 = 10 points). There is no limit to the amount of points they can accumulate. This feature is not available to Guest Users.

3.9.2 *System-User Interactions*

1. The Registered User spends X dollars.
2. The system confirms the transaction and adds X * 10 points to the user's profile.

3.9.3 *Functional Requirements*

R-49: Create a functionality where the system can convert the final transaction amount to points.

R-50: Provide the system with the ability to edit, subtract, and add points to a Registered User's profile.

R-51: Display the changes to the user's points on their profile on the Loyalty and Rewards part of Account Management as described in Section 3.3.1.

3.10 Accessibility:

3.10.1 *Description*

This feature is of high priority and available to all users of the GreenGrocer web application. Members will be able to navigate through language selection and proper visual compatibility for all devices. Forms and webpages will include buttons and labels, as well as images to adapt to certain devices.

3.8.2 *System-User Interactions*

1. The user clicks the floating "Accessibility Button" found on all pages.
2. The system opens the accessibility page.
3. The user clicks on a setting and changes the setting.

4. The system saves the changes and modifies the user interface to reflect the changes.

3.8.2 *Functional Requirements*

R-52: Create an Accessibility button that all users can access.

R-53: Create an Accessibility page with a menu

R-54: Provide an option to change the language of the application to cater to a diverse user base.

R-55: Provide an option to change the colors of the page to account for color blindness and other visual disabilities.

4. External Interface Requirements:

4.1 Example Interfaces for Users (UI):

The web-based user interface of the GreenGrocer program is made to be user-friendly and responsive on desktop and mobile devices. Important UI components consist of:

4.1.1 *Sign-Up/Log-In Pages:*

Provide a straightforward form with fields for a username and password so that users may register and authenticate themselves (Section 3.1).

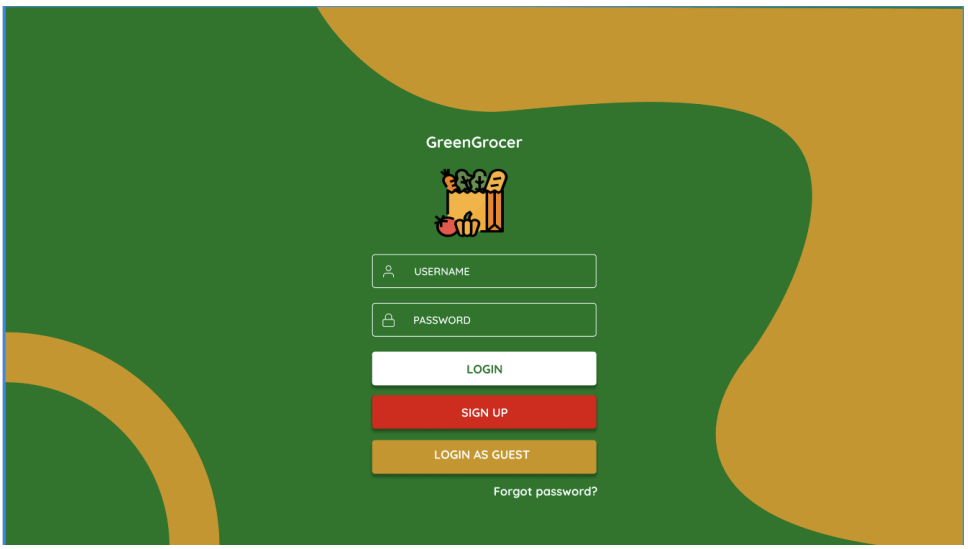

Figure 1: Login & Signup

4.1.2 Home Page:

Gives customers the option to choose from five pre-identified Long Island stores for their chosen store location via a dropdown menu. This establishes which store's merchandise is on display and accessible for purchase. Product Categories: Users can browse products in each category (Fruits, Meats, Drinks, and Frozen) by clicking on buttons that have clear labels.

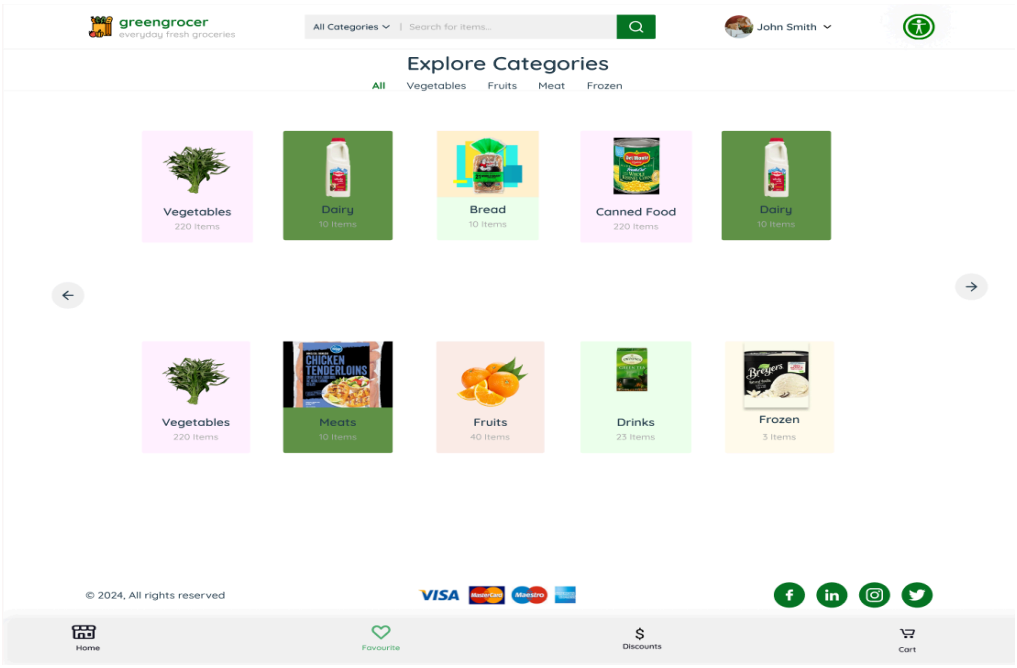

Figure 2: Home Page

4.1.3 Favorites and Cart:

The favorites, cart, and home icons are all conveniently located on the bottom navigation bar. For simple navigation, these buttons employ the visual indicators "home," "cart," and "star." The amount of goods in the cart at any given time is shown by the cart icon.

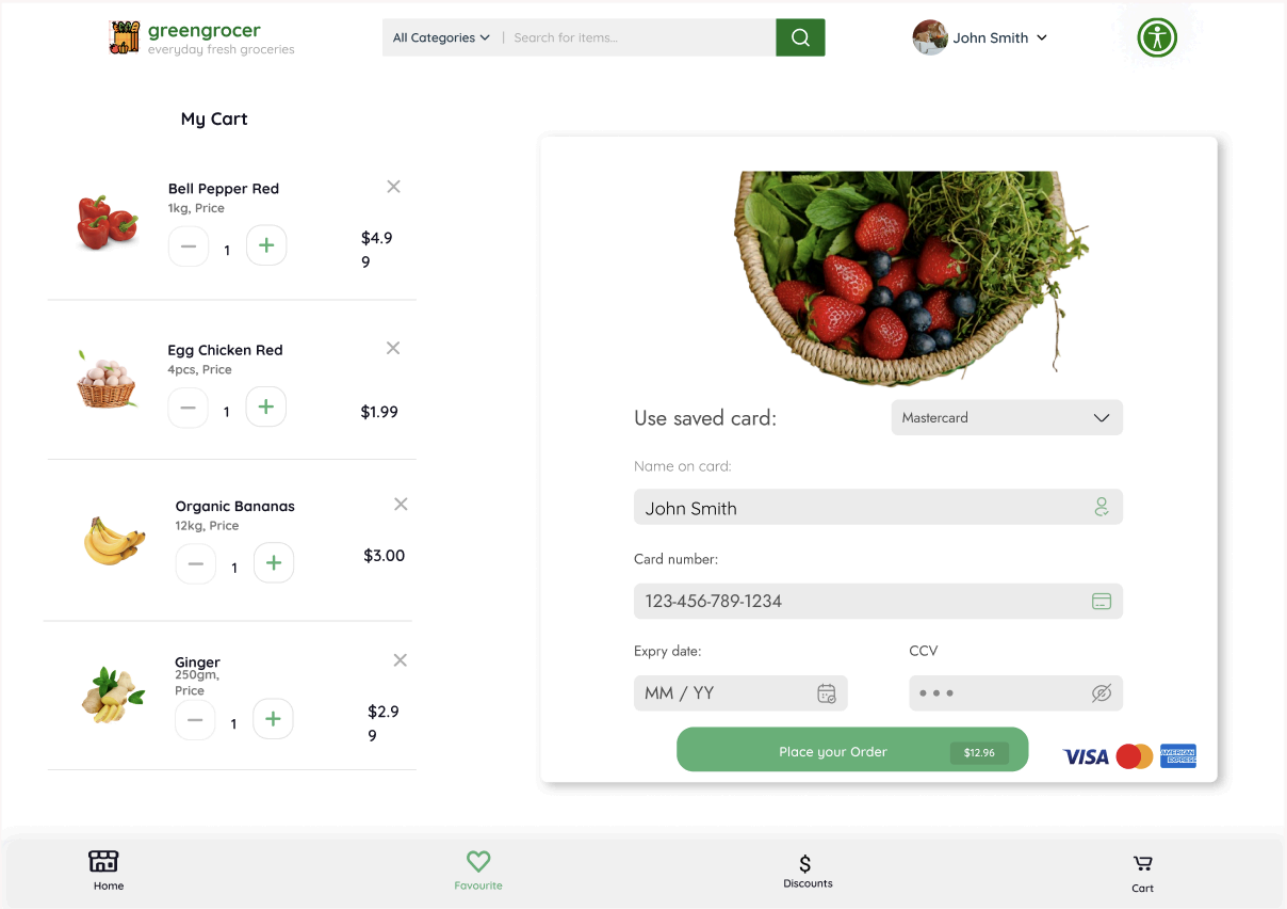

Figure 3. Favorites and Cart Pages

4.2 Interfaces for APIs:

The Kroger API is accessed by the GreenGrocer application, which uses it to retrieve and display real-time product data. Important exchanges with APIs consist of:

- Product API: Retrieves information about a product, such as its name, category, picture, and price according to location. Products are obtained using the location ID of the chosen retailer.
- Location API: Ensures the user sees only pertinent product data for the specified store by retrieving store details to fill the dropdown list of potential locations.
- Pricing Data: Location-specific prices are returned by the Kroger API when the store's location ID is sent to it.

4.3 Interfaces for Hardware:

Since it is a web application, no direct hardware interaction is needed. Every contemporary web browser can be used to access the program on the following devices:

- Computer (Windows, macOS, Linux) or laptop
- Tablets or smartphones (iOS, Android)

4.4 Interfaces for Software:

The following software is integrated with the application:

- Flask Web Framework: Oversees server-side API interaction, routing, and session management.
- The Bootstrap CSS Framework makes sure that layout and design are adaptable to many screen sizes.
- Flask-Login and Flask-WTF: Oversee form security and user authentication, guaranteeing appropriate user session management and CSRF prevention.
- For secure communication between the client (browser) and the server, the program employs HTTPS. API requests to Kroger are sent over HTTPS utilizing secure RESTful API calls, guaranteeing the secure transmission of sensitive information like product pricing.

5. Non-Functional Requirements:

5.1 Needs for Performance:

5.1.1 *Response Time*: To guarantee a seamless user experience, API queries to the Kroger API will be fulfilled in two to three seconds. Rendering pages and adding things to the cart will be completed locally in less than a second.

5.1.2 *Scalability*: The program must be able to support at least 50 users at once without experiencing appreciable performance lag. To handle more users and API calls, the backend needs to be scalable.

5.1.3 *Throughput:* The program must be able to process at least 100 API calls in a minute without experiencing any outages.

5.2 Security Conditions:

5.2.1 *Authentication:* Flask-Login is used by the application to safely authenticate users. Best practices for encryption are followed when storing passwords.

5.2.2 *CSRF Protection*: To guard against Cross-Site Request Forgery (CSRF) threats, the application uses Flask-WTF.

5.2.3 *Secure API Calls:* Sensitive information, like product prices, is communicated securely since all interactions with the Kroger API take place over HTTPS.

5.2.4 *Session Management:* To avoid unwanted access, user sessions will be safely maintained and automatically logged out after a predetermined amount of inactivity.

5.3 Usability:

5.3.1 *User-Friendly Design*: Users can effortlessly navigate between various functionalities (store selection, category browsing, cart management) thanks to the UI's simple design.

5.3.2 *Responsive Layout*: Regardless of screen size, the interface is suited for desktop and mobile devices, guaranteeing a smooth user experience.

5.3.3 *Accessibility*: The program must adhere to fundamental online accessibility guidelines, such as keyboard navigation and contrast ratio criteria, in order to be usable by people with impairments.

5.4 Availability:

5.4.1 *System Uptime*: With the exception of planned maintenance and upgrades, the application ought to be accessible 99.9% of the time.

5.4.2 *Error Handling*: The system ought to alert users politely and permit them to attempt activities again without experiencing a crash in the event that the Kroger API is unavailable. Error messages must be easy to read and offer concise guidance.

5.5 Maintainability:

5.5.1 *Code Readability*: To guarantee that future developers can quickly comprehend and maintain the system, the project code will include proper comments and adhere to common Flask/Python norms.

5.5.2 *Modularity*: Modular components will be used in the design of the program to ensure that modifications to one area (such as API interaction) do not adversely affect other areas (such as UI rendering).

5.6 Accuracy of Data:

5.6.1 *Data Consistency:* Throughout the user's engagement with the application, the cart and user sessions will be kept up to date. Even if the user leaves the website, the items they have added to the cart will stay there until they are deleted.

5.6.2 *Database Integrity:* The system must guarantee the safe and uncorrupted storage of user data, including login credentials and cart information.

5.7 Adherence to Rules and Regulations:

5.7.1 *GDPR Compliance*: The program must abide by data privacy regulations to guarantee that user information is only gathered and kept after the user's express consent. It will be possible for users to ask for their data to be deleted.

5.7.2 *ACM Code of Ethics Compliance*: The GreenGrocer web application follows the community's code of ethics to ensure the application is working for the greater good, not ensuing harm, and accessible to all people.

6. Project Management:

6.1. Methodology:

The Scrum framework has been adopted, an agile methodology that emphasizes flexibility and iterative progress. The team operates in one-week sprints, with each sprint aiming to produce a working increment of the app. Brief team check-ins are held twice a week to ensure progress remains on track and to address any challenges promptly.

6.2. Team Structure:

The role of meeting facilitator is rotated amongst group members weekly to ensure equal and shared leadership. Communication is primarily handled through a shared Slack (or Discord), in addition to twice-weekly check-ins, with additional meetings scheduled as needed.

6.3. Tools and Technologies:

For project management, the free version of Trello is used to create and assign tasks, as well as track progress. Trello also provides an activity chart for an overview of the project's progress, helping to stay on track and establish the expected endpoint from the beginning. If necessary, the information from the activity chart is used to identify areas that might require project crashing. GitHub is used for version control, enabling team collaboration on code, conducting peer reviews, and managing different versions of the app efficiently.

6.4. Risk Management:

At the start of the project, potential risks were assessed and a risk register was created to track and add new risks as they emerge. This list is reviewed and updated during weekly meetings. For each identified risk, a basic mitigation strategy is developed. For example, if difficulties are foreseen with a particular technology, time is allocated for group study sessions or advice is sought from the course instructor.

6.5. Quality Assurance:

To maintain code quality, a peer review system has been implemented where code changes must be reviewed by at least one other team member before merging into the main branch. Pair programming is also practiced for complex features to enhance both code quality and knowledge-sharing.

6.6. Deliverables and Milestones:

After determining the tasks required for the project, benchmarks for code completion are established using similar code from GitHub repositories. The number of lines of code needed for each task is assessed along with the code-per-day rate of the member working on the task. These metrics are used to estimate how long a task will take and whether the member is on schedule. As more tasks are completed, past tasks are used to refine future estimates.

6.7. Change Management:

Given the project's scope and timeline, changes are handled through group consensus. During twice-weekly meetings, any team member may propose changes, and their impact on the timeline and workload is discussed. If a change is approved, it is added to the Trello board for implementation in future sprints.

Appendix A: References and Resources

1) Kroger Product API
    a) Use: The Kroger Product API provided access to the Kroger's product database, which helped us integrate products into our application. The API features names, prices and images of the items listed in the database for our grocery application.
    b) Reference: Kroger. (n.d.). Product API - Partner. Kroger Developers. Retrieved from https://developer.kroger.com/reference/api/product-api-partner#tag/Products/operation/productGet.

2) Ngrok Tunnels API
    a) Use: Ngrok's Tunnels API allowed us to use a secure platform to tunnel a local server to the web. Additionally, this allowed us to share the local environment publicly without deploying it on a hosting application.
    b) Reference: Ngrok. (n.d.). *Tunnels API Documentation*. Ngrok. Retrieved from https://ngrok.com/docs/api/resources/tunnels/.

3) Flask Installation Documentation

    a) Use: Flask was used to build the backend of our application while handling responses and requests. We used this to help us set up a web server for our project.
    b) Reference:
    Flask. (n.d.). Flask Installation. Flask Documentation. Retrieved from https://flask.palletsprojects.com/en/3.0.x/installation/.