BasketBuddy System Specifications & Requirements

Introduction

Purpose

This document identifies the requirements and features that will be used for the *BasketBuddy* project, specifying the components and system structure that the application will contain. This includes the functional and non-functional requirements, as well as the interfaces that will be implemented. This document will be used to guide the focus during development, and to better understand the aspects the project will encompass.

Scope

The scope of *BasketBuddy* allows users to login, browse groceries, add items, apply coupons and earn or redeem rewards for future purchases. The application will allow users to purchase everything they need prior to arriving at the store, ensuring their groceries will be ready for them upon arrival.

Overall Description

Despite the busy lives of people, between jobs, schooling, or caring for family, everyone must eat. *BasketBuddy* puts the time back in the hands of the shopper, allowing them to provide for themselves or their family in a timely manner. Purchasing groceries ahead of time will save a shopper hours throughout a given month. Utilizing the website or application will facilitate this transaction, enabling the products to be ready upon arrival at the physical store location.

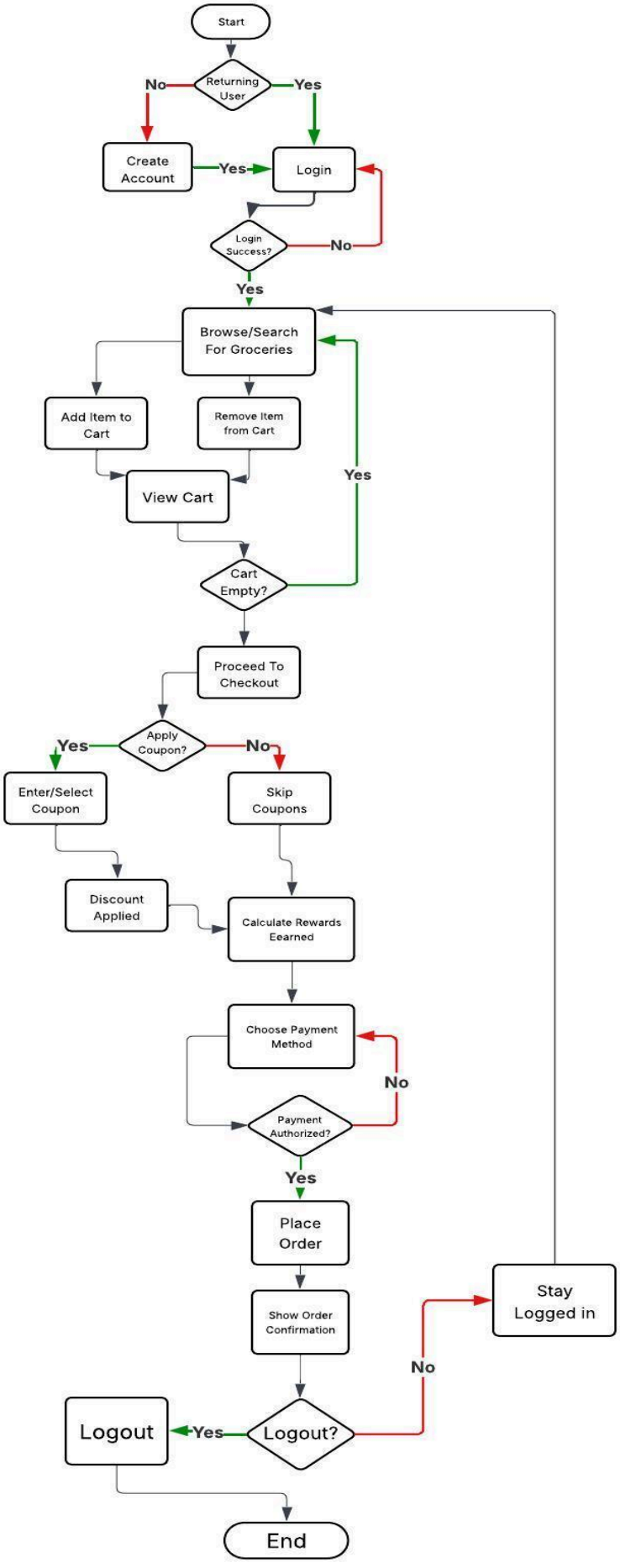
Product Perspective & Features

BasketBuddy offers a mobile and desktop platform, both operating on the back of an SQL database which holds and stores grocery items, user accounts, and user lists. The software for *BasketBuddy* doesn't rely on anything else besides the locally operated SQL server. Moreover, the application will include a shopping cart functionality, manage payments and purchases, create user accounts, apply coupons, and a customer rewards system.

System Interfaces

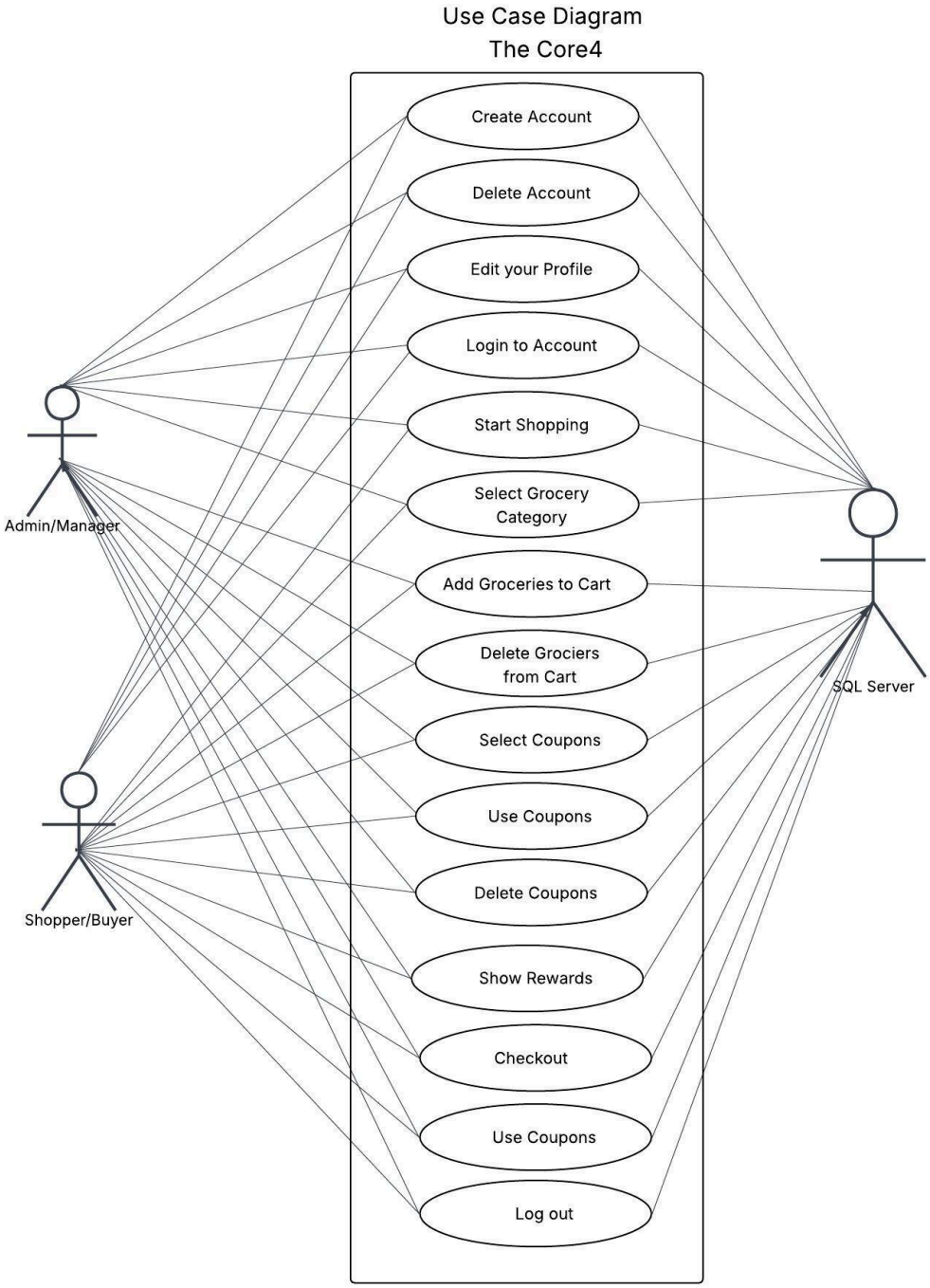
BasketBuddy will operate smoothly on all desktop platforms, such as Mac, Windows, or Linux. Further, the mobile system will be designed for compatibility with android devices.

Flowchart


Start
Returning User
No
Yes
Create Account
Yes
Login
Login Success?
No
Yes
Browse/Search For Groceries
Add Item to Cart
Remove Item from Cart
View Cart
Yes
Cart Empty?
Proceed To Checkout
Apply Coupon?
Yes
No
Enter/Select Coupon
Skip Coupons
Discount Applied
Calculate Rewards Eearned
Choose Payment Method
No
Payment Authorized?
Yes
Place Order
Show Order Confirmation
Stay Logged in
No
Logout
Yes
Logout?
End

The flowchart depicts our process of how *BasketBuddy* will guide the user along their shopping experience. The primary objective is for the shopper to create an account to be able to utilize all of the features the system provides. User accounts enable coupons and rewards, which can be applied to future orders for later discounts. The shopper will be able to add and delete items from their cart, where they may apply their coupons or opt to skip this step. Then, the shopper will be prompted to enter their payment information, which will be saved along with their user account. After the transaction is successfully completed, the user may logout and end their session, or can stay logged in to view their active order status.

Use Case Diagram



The use case diagram guides software development to outline the relationship between the actors, such as the admin or manager, shopper or buyer, and the SQL data server. The admin or manager acts as the overarching role that can control the entire system. This includes the ability to make changes to the design and auditing that the correct groceries are provided to the correct shopper. Additionally, coupons can be managed and supported in the event of processing issues.

Shoppers or buyers have full control over their user account. They may create, edit, or delete their account. They are able to add and remove items from their shopping cart and also ensure that they have the ability to apply the coupons if they have any. Once the shopper has submitted their order, they can log out until they are ready to purchase groceries.

In regards to the SQL data server, this will store all necessary information such as item inventory, user account information, billing and physical addresses, coupons, and user made shopping lists.

System Features

- **Personal Shopper Account**
  Provides secure user authentication and profile management. Stores account data such as login credentials, order history, and saved shopping lists in the SQLite database.
- **Shopping Cart**
  Implements a temporary data structure linked to the user session. Tracks selected items, quantities, and real-time price totals before checkout.
- **Coupon System**
  Applies discount logic at checkout. Validates codes against a database table of coupons and adjusts the order total accordingly.
- **Rewards System**
  Maintains a points or credits balance for each user account. Updates balances after purchases and provides logic for redemption during checkout.
- **Checkout System**
  Handles order finalization. Confirms cart contents, applies discounts or rewards, processes payment data, and creates a persistent order record in the database.
- **Creating, Modifying, and Deleting Shopping Lists**
  Allows Create, Read, Update, Delete operations on user created lists. Lists are stored in the database and linked to a specific user account.
- **System Notifications**
  Implements event-driven messages for key actions such as order confirmation or account changes. Notifications may be displayed in-app or sent via email.

- **Order Management**
  Provides functionality to view, update, and cancel orders. Uses database queries to display active and historical order data to the user.
- **Product Catalog**
  Maintains a master table of items with fields for name, description, price, category, and image URL. Supports item browsing and retrieval via SQL queries.
- **Search and Filter Options**
  Implements SQL query logic for keyword search and filtering by category, price, or popularity. Results are returned dynamically to the user interface.
- **Session Management**
  Tracks user activity using session tokens. Ensures carts, lists, and login status persist during the user's interaction with the system.
- **Multi-platform Support**
  Ensures consistent functionality across web and mobile view. Both pages connect to the same SQLite database.
- **Security and Privacy**
  Implements password hashing, and restricted database access. Protects sensitive user data from unauthorized access.
- **Scalability and Performance**
  Optimizes database queries and system architecture for efficiency.

External Interface

Keyboard

- Users will be able to use a standard keyboard that will allow them to provide inputs to the system on whichever part they are in.

Monitor

- The monitor will be needed to be able to view the website and see the functions they are performing with other devices.

Mobile Device

- Android specific devices will be able to utilize the page to scroll, add and delete items, and any other functions that are necessary to purchase goods.

Internet Connectivity

- Users will need to have a stable internet connection or access to their mobile network to be able to access *BasketBuddy* on their personal computers or mobile devices.

Functional Requirements

- User must be able to create account
- Users must be able to view/edit account information
- User must be able to delete account

Catalog & Search

- System shows product catalog(name, image, price, unit/size, availability
- Search, with filters and sorting
- Admin can create/read/update/archive products

Shopping Cart & Lists

- Users can add/remove/update quantities of products in cart
- Cart maximum number of item enforcement
- Users can create, rename, duplicate, delete shopping lists
- Cart shows running subtotal, discounts, tax, and total in real time
- Users can add/remove coupon codes at cart/checkout
- System validates coupons(eligibility, expiration date, product/brand applicable) apply correct math to totals

Rewards

- System tracks reward points
- Users can view point balance and redeem points
- enter/manage pickup contact info and preferred pickuptime
- Payment is captured/authorized and recorded

Non-functional Requirements

Performance

- The database should be able to return search results in at least 3 seconds, and load and return saved shopping carts and orders in 1-3 seconds

Security

- User login data should not be stored in plain-text
- Only authenticated users can access personal data
- Users should be timed out after extended periods of inactivity

Quality & Reliability

- System should be easy to use across web and mobile view
- Database should be reliable and accessible
- System should feel responsive and consistent

Usability

- Users should be able to learn the interface without spending lots of time on it
- Buttons and menus should be labeled in plain language
- Font size and dark mode should be adjustable

Development Plan

Catalog Phase

- The catalog phase of design will deploy the entire catalog of grocery items that are offered through *BasketBuddy*. This includes product names, photos, cost, and other information relevant to the shopper.

User Accounts & Lists Phase

- The user accounts and lists phase of design will be outlined prior to ordering. *BasketBuddy* requires shoppers to create an account prior to submitting orders. This ensures that the shopper can access their active or previous orders which will be outlined in the third design phase. Lists will also be associated with the user account, where shoppers can create, edit, or delete customs lists of items from the catalog.

Ordering Phase

- The ordering phase of design will deal with implementing the shopping cart functionality, along with placing, cancelling, or tracking active orders that are associated with a user account. Adding payments, addressing, and contact or billing information will also be enabled during this phase.

Group Member	Contributions	Percentage %
Tyler Niemonen	**Constructed use case diagram, flowchart, and description. Helped with introduction and functional requirements.**	**20%**
Carson Carmody	**Created the three-phase project design section. Formatted document & wording.**	**20%**
Jeff Pautrat	**Functional & non functional requirements**	**20%**
Ryan Wegener	**Features & requirements**	**20%**

Jakia Gary	**Helped with system features & requirement ideas**	**20%**