BasketBuddy Grocery Ordering

“The Core 4”

Team Members

- Tyler Niemonen: Team Lead, Web developer, UI/UX
- Ryan Wegener: Back end developer, GitHub Lead
- Carson Carmody: Web & Back end developer, JavaScript, Frameworks
- Jeffrey Pautrat: Front End Developer, UI/UX

Project Description

Overview

The BasketBuddy grocery ordering system enables shoppers to remotely purchase groceries online, allowing them to shop around their busy lives or schedule. This project contains three distinctive parts: the web interface, mobile interface, and database to store grocery items. All three components will communicate to ensure that shoppers can browse items and submit or cancel orders effectively on either platform. BasketBuddy will synchronize across multiple devices, so that if a shopper begins their order on the website, they may continue on their phone. This underscores the importance of designing a single, consistent database that both platforms can connect to reliably.

User Interface

The website will be built with HTML and CSS, along with frameworks such as Bootstrap for styling and layout. JavaScript will be used for interactive portions of the site. Both the web and mobile interfaces will adhere to similar design, ensuring that swapping between them is familiar for the shopper. To achieve this, a common set of design elements such as fonts, button styles, and color schemes will ensure the feel of the application remains the same regardless of the device being used. Figma wireframes of the interfaces will ensure we have the proper design and functionality, optimized for user experience and accessibility to many different users. The WCAG accessibility guidelines include adjustable font sizes and toggling dark or light mode. These features will let users tweak the application to their preferences or needs to make it easier to use.

Item Catalog

A list of available grocery items, organized by categories: Produce, meat, dairy, and non-perishables. Shoppers can browse, search with keywords, filter, add, and remove items

from their cart. Additionally, shoppers can add an item to their favorites list. Every item will display its name, price, and how many are added to the cart. To make the catalog user friendly, each item will also include a picture so it can be quickly recognized. Users will be able to sort results by category and price. The catalog will also suggest related items that have a similar keyword.

Shopping Cart

A collection of grocery items that the user has selected for purchase. Users will be able to adjust item quantities, remove items, or add an item to a list they have created. The shopping cart will also show the total number of items and the total cost as they are updated. This makes it easier for users to keep track of how much they are spending before checking out. Alternatively to lists, a shopper may keep items in their cart across sessions, meaning that if they close the app or website, the cart will still be saved for later. This makes it convenient for users who like to plan their shopping over time instead of all at once.

Lists

A set of personalized lists where shoppers can put items aside for favorites or for later shopping. This allows shoppers to quickly purchase household items such milk, bread, or eggs to their cart without the need to search each item individually. Saving items for later lets shoppers create an order gradually without submitting it for purchase.

Orders

Shoppers can view their active or past purchases which displays total cost, date of checkout, and list of items purchased. Active orders which haven't been fulfilled can be cancelled. Past order history will be accessible for convenience so users can easily reorder common items by selecting "Reorder."

Account

Shoppers can create or login to accounts, manage their payment and billing information, and add contact information needed to notify the shopper when their order is ready. The account system will also allow users to set preferences like default delivery addresses, saved payment methods, and their preferred way of communication. These features ensure that users can quickly check out without re-entering their information each time.

Website

The website will allow shoppers to browse groceries, construct their cart, create lists, and manage purchases. HTML and CSS will be used for the front-end design, with JavaScript handling interactivity. Bootstrap and other tangential frameworks may be used to make the layout more responsive and to keep the design consistent across different screen sizes. This will help ensure that the website works well on desktops, laptops, and tablets without needing

separate designs. Python with the Flask framework will serve as the backend for routing, user accounts, and other related logic or functions. Flask was picked for supporting features like sessions and database access. The backend will be developed in JavaScript with Node.js and SQLite. This will allow both the back end and front end to use a single language, making development straightforward. For both interfaces, the backend will act as the connection point between the website, database, and mobile app, maintaining the consistent system design across platforms.

Mobile

The mobile interface hosts all of the same functionalities as the website, but it will be scaled to fit the smaller screens of phones and tablets. The design will be mirrored from the website so that shoppers have a consistent shopping experience no matter which platform they shop from. Buttons, menus, and text will be adjusted to fit on a touchscreen, but the overall look will stay the same. Users will not need to relearn how to use the app, so that whether they are on their phone or laptop, browsing items and managing the cart or lists will act in the same way.

Database

The database will be constructed using SQLite. It will store basic item information, shopping carts, lists, order history, and user accounts. This database will connect both the website and the mobile application so that they can be used alongside each other. By keeping everything in one place, the system will stay consistent no matter which device the user is accessing the data from. For example, if a shopper adds items to their cart on the website, those same items will appear when they open the app on their phone. SQLite is a simple but reliable choice for this project, making it possible to set up tables for all of the information we need without being too complex.

Risk Management:

Late Submissions:

- **Probability:** Low
- **Impact:** Significant
- **Monitoring:** Members will monitor deadlines and communicate daily on the weekdays.
- **Mitigation:** If the group ends up nearly making a late submission, we can meet and determine a better timeline that fits all of our schedules to prevent close calls in the future. Additionally, reassessing all deliverables and ensuring the project scope is still feasible for the timeline will enable future success.

Inadequate Deliverables:

- **Probability**: Medium
- **Impact:** Significant
- **Monitoring:** Members will adhere closely to assignment rubrics and instructions.
- **Mitigation:** The group can meet with the TA and discuss how we should more appropriately approach future assignments. This may include reestablishing project objectives or roles.

Schedule Conflicts:

- **Probability:** High
- **Impact:** Medium
- **Monitoring:** Group members will ensure that they work with their jobs and personal life to ensure that there is adequate time for collaboration.
- **Mitigation:** Coordinating specific times to meet virtually will allow us to more conveniently get together and finish an assignment.

Project Schedule

Milestones

Consensus on project details and specifics: 8/28/25

Finalize deadlines: 8/29/25

Document the UI design, have mock-ups: 9/28/25

Have mock-ups of website and mobile application: 9/30/25

Database

- **Gather items**: 10/08/25
- **Create tables**: 10/12/25
- **Host on SQLite**: 10/17/25
- **Completed:** 10/18/25

App

- **Front end core functionality:** 10/01/25
- **Back end core functionality:** 10/09/25
- **Completed:** 10/18/25

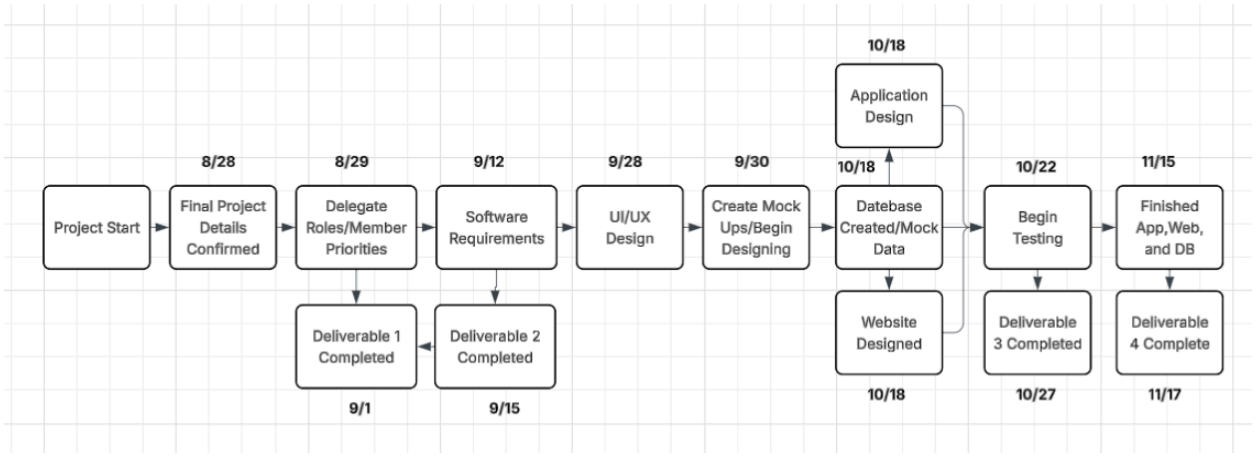
Website

- **Front end functionality**:10/01/25
- **Add CSS/ bootstrap functionality:** 10/06/25
- **All screens finalized:** 10/09/25
- **Adding interactive components to pages:** 10/11/25
- **Back end functionality:** 10/14/25
- **Completed:** 10/18/25

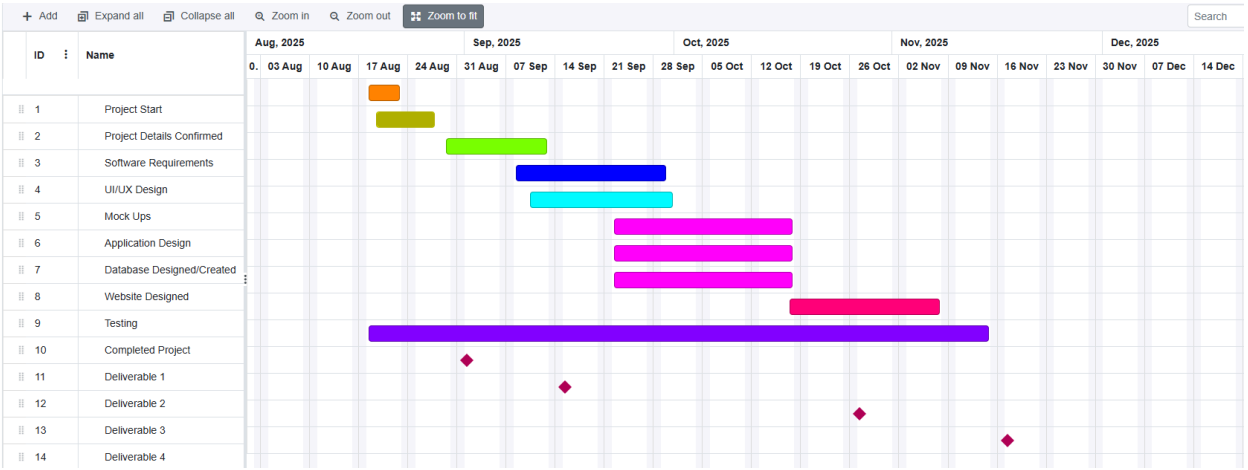
Testing Begins: 10/22/25

Testing Ends: 11/13/25

PERT Chart:



GANTT Chart:



Team Contributions:

Member Name	Contributions	Percentage %
Tyler Niemonen	Created the GANTT/Flow charts, assisted with the project schedule, and helped decide what languages we will use.	25%
Ryan Wegner	Assisted outlining project schedule and details, and decided on Python/Node.js and SQLite	25%
Carson Carmody	Created the project details & risk management sections. Assisted with the project schedule.	25%
Jeff Pautrat	Helped with formatting, editing and flow, provided insight to the schedule and things with risk management section, gave insight to topics on the document	25%