

ShopSmart

A more efficient shopping experience



Our Team

Team Name: Always Blue

Contributors:

- Matthew Biller
- Adam Cavins
- Joo Young Han
- Kalen Wiley
- Travis Williams



What is ShopSmart?



What is ShopSmart?

- ShopSmart is a smart shopping assistant that helps users effortlessly manage and organize their grocery lists across multiple stores
- Users begin by creating a profile, then adding one or more stores to their homepage. From there, they can enter items they need for upcoming shopping trips
- Newly added items appear in an unassigned list, where users can sort them by assigning each item to a specific store. Once assigned, the item automatically appears under that store's list
- Once an item has been purchased, users can mark the item as Complete using a checkbox. They can also remove an item from a store's list
- This design is intended to keep shopping experiences simple, organized, and efficient



Tools Used

Tools Used – Project Tracker

- As a team we used Trello to track our progress throughout development
- The Trello board served as a centralized location where team members could pick up issues that needed to be worked and move them to their own pipeline
- We monitored progress by tracking the Backlog and Completed issues as development progressed

trello board: <https://trello.com/b/hBYltvo3/shopsmart-project-tracking>

Tools Used – Version Control

- We used GitHub to monitor version control during development
- Each team members was responsible for branching the ShopSmart repository
- When changes were merged, we used Pull Requests to verify that any work done would not break or regress the app
- We verified each other's PRs to ensure code validity and to gain experience with PRs



Programming Languages

Programming Languages

- For this project, we used the following programming languages:
 - React – Frontend
 - Allowed us to develop re-usable components necessary for our application UI
 - Compiles to HTML, CSS, and JavaScript in the deployed environment
 - Python - Backend
 - Flask
 - sqlite3
 - SQLite – Database
 - Utilized multiple CREATE and INSERT statements to generate tables for users, stores, and items



Deployment Environment

Deployment Environment

- Application was deployed with Render:
 - Api – Web Service configured to receive requests at the deployed URL
 - Python3 runtime
 - Automatic deployment on repository commit
 - Web – Static
 - React project is compiled to static HTML, JavaScript, and CSS
 - Fetch requests make calls to the deployed API using environment variables configured in Render
 - Automatic deployment on repository commit



Challenges

Challenges

- **Life** - We had several 'life events' happen during the semester
 - Jury duty, overseas travel, family events
 - This occasionally made meeting and working around these events difficult
- **Sprint Planning / Dev Work** – Throughout the project we started planning with our own implementations, only to be reeled back by the week's material, finding there were easier ways to do certain tasks
 - Ultimately this was helpful, but it caused a few roadblocks as we had to rethink an active implementation
- **Time Commitment** - All of us had to split our time between the project and other responsibilities
 - Additional courses
 - Work
 - Family



Demo

ShopSmart Deployed Application

Add

Unassigned

Gift for Steve

>

Wall Decor

>

Costco

Cheese

>

Eggs

>

Butter

>

Blueberries

>

Lidl

Ketchup

>

Mustard

>

Lowes

Nails

>

3/4" Wrench

>

Target

Loofah

>

Body Wash

>

Walmart

Camping Supplies

>

