Next up: fetch saved items, useLastID (to not fetch all), start from lastID on fetch

Connect Cart context + api

Connect Form to create item

Connect account details

Add styles

Connect browsable products

Cart analyzing

Add more APIs spoonacular

Search for nutrition facts

Add barcode scanner

Add image searching

<https://www.grocery.com/open-grocery-database-project/>

<https://www.figma.com/file/EQUOebmuGSgCrOpD1qeC8b/QuickCart?node-id=13%3A1869>

Visit creating quickart doc in articles

Get any images from projects/QuickCart

* Link GitHub page
* Link Demo page

Refer Grocery app birth idea doc

Create backstory of QuickCart article (why I needed to build it)

Tweets: <https://docs.google.com/document/d/1R-S4NMvEPArv28lCH9XpASt8NlffSDhLaF6NeVt6MSY/edit>

Checkout old commits

The Creation of QuickCart:

During my coding journey, QuickCart has been a long-standing project and learning experience. I applied many newfound React concepts to learn about frontend development. I worked on the backend with MongoDB and Node.js before migrating to Firebase as well. As the codebase grew, it became increasingly dependent on organization and maintainability. This led me to refactor several times, after learning new best practices and thinking about scalability.

At the beginning, I was completely new to Test Driven Development. It seemed like a waste of time at first, but that didn’t last long. I was able to utilize Jest + Enzyme to verify each new functionality. That provided a much more robust foundation of modules to build off of, and there was no more need to worry about breaking everything I wrote earlier.

I ended up exploring many new technologies and getting valuable experience. Here I document my journey of creating QuickCart to cover everything I’ve learned.

**What is the goal of this app?**

What it does ~

Where the idea came from ~

Show off finished product ~

**Where to start?**

Since I was learning ReactJS when I started QuickCart, I would build the frontend client with that framework. I had a rough idea of the functionality of the app, but I wasn’t sure how to design it. Mainly, it was going to allow me to interact quickly with data previously stored on spreadsheets. This idea evolved overtime, and eventually I began to add features that were mobile-friendly.

**Working on the design:**

Before I could start writing any code, I first needed to plan out what the final product would look like with a wireframe. To create this, I used an in-browser tool called Figma. Here is what the app will look like broken into different components:

[INSERT WIREFRAME IMAGE]

[INSERT WIREFRAME IMAGE]

[INSERT WIREFRAME IMAGE]

**How will this data be stored?**

That all provides a solid foundation and blueprint for the frontend, but what about the actual data being used? Storing one list of products wouldn’t be very useful. This app would need to store data for each individual user, and read and write to any of the information they store. There are many ways to store data depending on the use-case, but were was my approach:

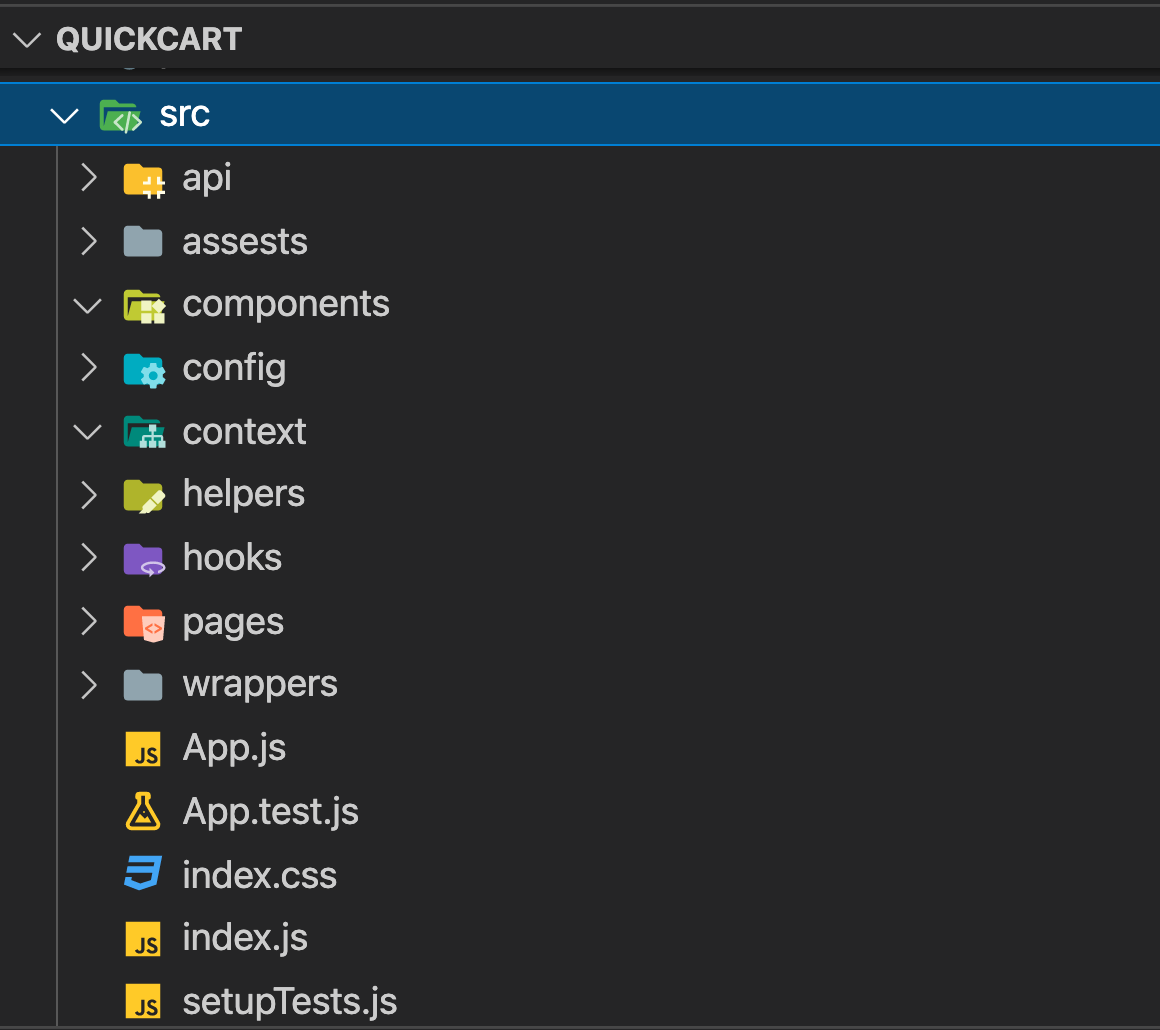
* Users will create an account
* Users can: add items directly to cart
* Users can: upload custom products from the form
* User can: save items to their collection to edit details
* Store list for each user:
  + User doc (email, name, createdDate, totalGrocery count)
  + Saved items doc
    - Category Collection (e.g. Dairy)
      * Grocery Item DOC (grocery schema, id, info)
  + Cart logs doc
    - (Receipt) Cart purchase doc
      * purchase Item DOC (date, item details, price)
        + Item (doc)
        + Receipt date
        + Cart total
        + Cart Breakdown

**Create-react-app:**

With a solid plan and layout of the site, it was time to set up the codebase. This project would use the frontend JavaScript framework React. Everything in the root directory will be compiled into static files to display.

Setting up a template project and installing the dependencies:

* run ```npx create-react-app quickcart```
* Install dependencies
* Add configuration files
* Use pre-commit hooks
* Create folder layout



This project will use the Firebase API, (HERE any other APIS) We use this to make requests the the endpoint for getting data. Brief explanation of what they are, and how thye tie into our app

Setting up the linter, husky and lint-staged

Folder structure of the codebase:

* src/
  + /components

- …

Create absolute imports for components - set up jsconfig.js

Translating the wireframe into folders/components

Setting up redux + react-redux:

* How will we store this data in the app?
* We need to fetch the ID of grocery items and give them to a card
  + When the card enters the screen, it will request to load the actual card details

Here is a breakdown of the redux

* userDetails:
  + currentUser (from @firebase/auth) // uid, displayname, profile, email
  + accountDetails (from firebase doc), // name, email, createdOn, totalSaved

- Cart Items:

- Items stored in the cart

Writing basic tests for each file

Adding static data to showcase app

Half-MVC! See the pages connect and data being displayed

Adding hooks (clickOutSide, isAuthPage, scrollReveal)

Adding data in context (passing sample data through grocery context)

Configuring tests to conform with Context providers

Writing more tests

Writing API functions

Connect the API to Context

Connect the Context to components

Write STATE in components

Create a system for recommendations

Everything is connected, from front end to back end

Add sample images and demos

Add sample grocery data stored under public firebase endpoint

Improve grocery data APIs for even better data

Improve Recommendation algs