CompareCarts

Save money on online groceries.

Problem

Finding the most affordable prices for grocery delivery from different grocery providers is tedious and requires visits to multiple websites or apps.

Substantial Possible Savings

- Average family of 4 spends $262/week on food
  - $150/week on groceries. Cities about $180/week
- Saving 10% for a month means that the average family can save **$60-72/month = $720-$864/year**
- Savings of 10-15% per item is not uncommon
  - Ie. milk at one store is $2.00 and at another it's $2.30

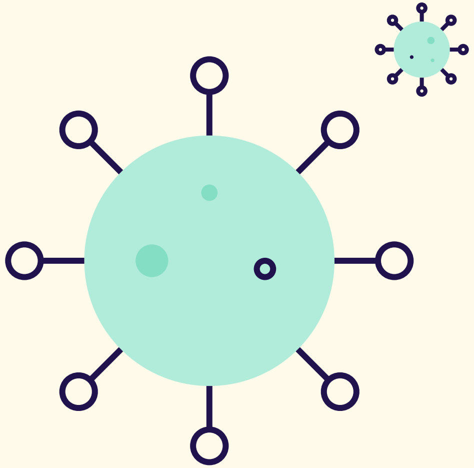
Source: https://www.bls.gov/news.release/cesan.nr0.htm, https://www.bls.gov/cex/tables.htm#MSA

Lots of Manual Work

If that family wanted to compare prices, it’s a lot of work.

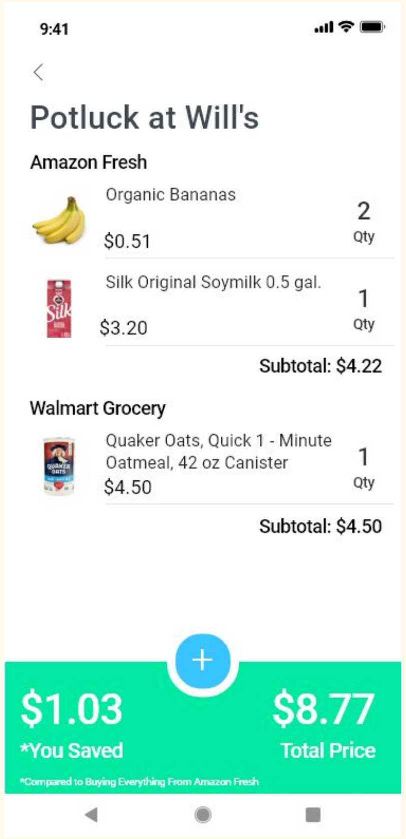
- Assuming 50 grocery items per family (keep in mind average amount spent is $150/week)
- Comparing just three grocery providers would mean they would have to do **150 searches/week**
    - Prices can update weekly
- Once figuring out the prices, there would be substantial calculations to figure out which provider(s) to buy from

Problem Exacerbated



With COVID-19, the problem has been exacerbated with many people turning to buying groceries online but also many people becoming unemployed.

CompareCarts User Stories


9:41

Potluck at Will's

Amazon Fresh

Organic Bananas
$0.51
2
Qty

Silk Original Soymilk 0.5 gal.
$3.20
1
Qty

Subtotal: $4.22

Walmart Grocery

Quaker Oats, Quick 1 - Minute Oatmeal, 42 oz Canister
$4.50
1
Qty

Subtotal: $4.50

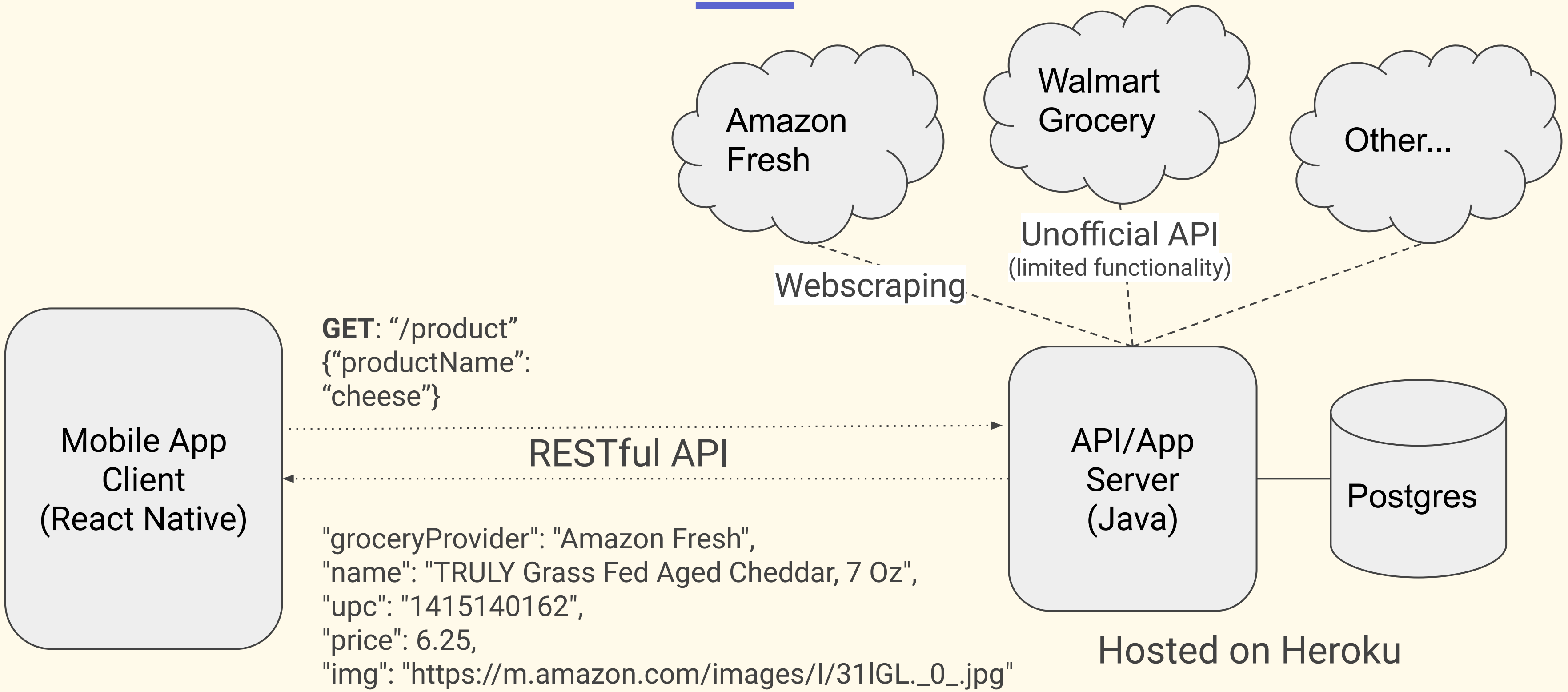
$1.03
*You Saved

$8.77
Total Price

*Compared to Buying Everything From Amazon Fresh

- A grocery shopper needs the ability to **create a list** of groceries and **see previous lists** that they created.
- A grocery shopper needs the ability to **search** for an item from **multiple grocery providers**.
- A grocery shopper needs the ability to **add and remove items** from the list in order to keep track of their order and allow them to change their mind.
- A grocery shopper wants to see the **optimal combination of items** in order to save the most money.
- A grocery shopper wants the ability to **share a list** with their family, friends, or roommates.

High-Level Architecture

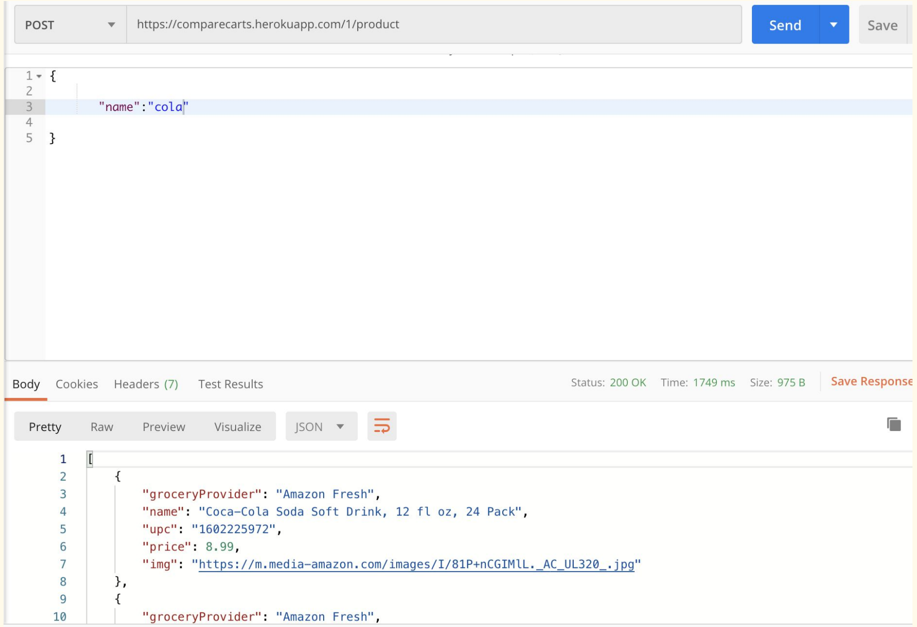


GET: "/product"
{"productName": "cheese"}

```
"groceryProvider": "Amazon Fresh",
"name": "TRULY Grass Fed Aged Cheddar, 7 Oz",
"upc": "1415140162",
"price": 6.25,
"img": "https://m.amazon.com/images/I/31lGL._0_.jpg"
```

REST API

getItemForName



POST https://comparecarts.herokuapp.com/1/product

Send | Save

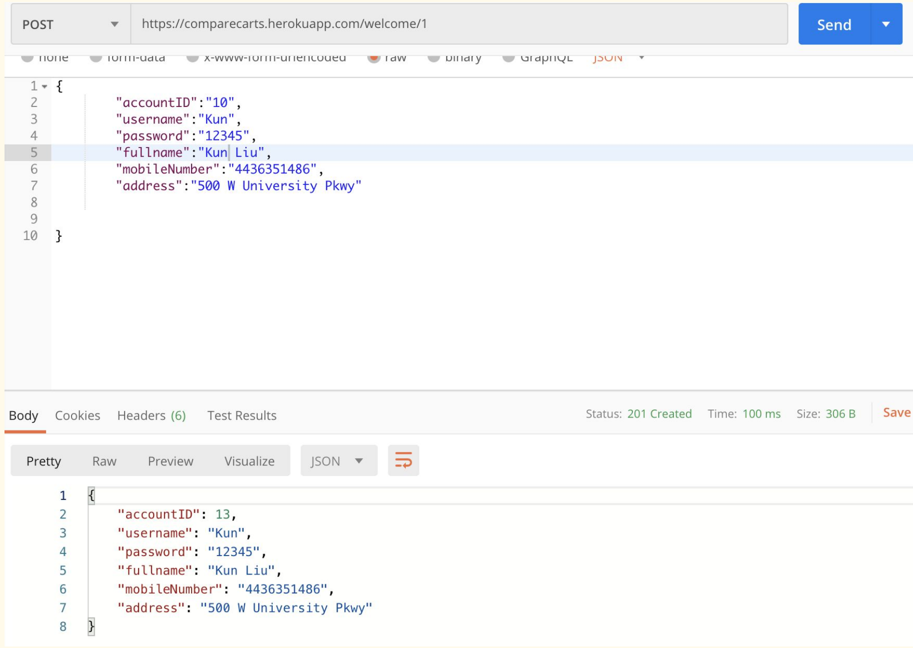
```
{

    "name":"cola"

}
```

Body | Cookies | Headers (7) | Test Results   Status: 200 OK Time: 1749 ms Size: 975 B Save Response

Pretty | Raw | Preview | Visualize | JSON

```
[
    {
        "groceryProvider": "Amazon Fresh",
        "name": "Coca-Cola Soda Soft Drink, 12 fl oz, 24 Pack",
        "upc": "1602225972",
        "price": 8.99,
        "img": "https://m.media-amazon.com/images/I/81P+nCGIMlL._AC_UL320_.jpg"
    },
    {
        "groceryProvider": "Amazon Fresh",
```

addNewUser



POST https://comparecarts.herokuapp.com/welcome/1

Send

none | form-data | x-www-form-urlencoded | raw | binary | GraphQL | JSON

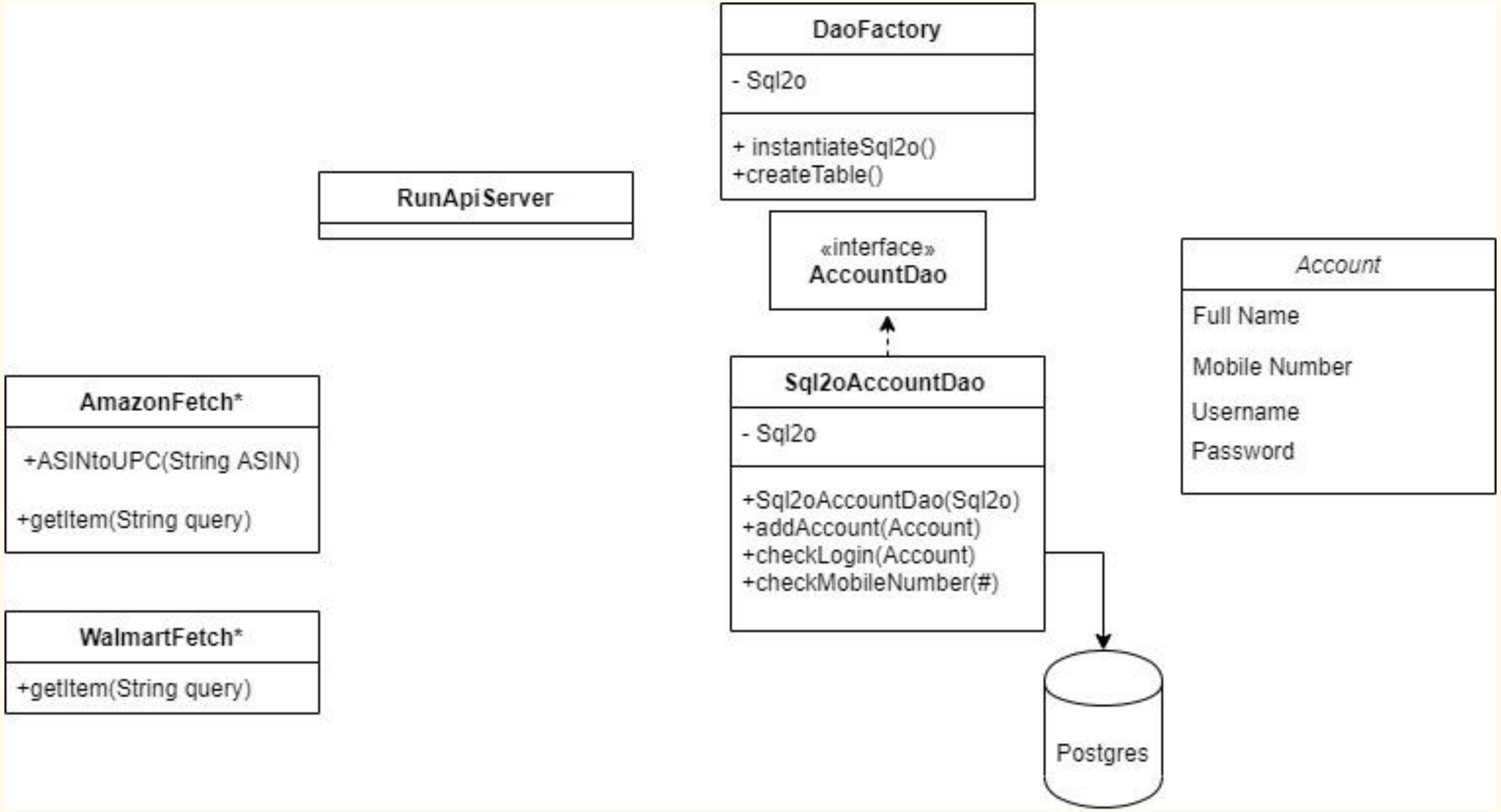
```
{
        "accountID":"10",
        "username":"Kun",
        "password":"12345",
        "fullname":"Kun Liu",
        "mobileNumber":"4436351486",
        "address":"500 W University Pkwy"


}
```

Body | Cookies | Headers (6) | Test Results   Status: 201 Created Time: 100 ms Size: 306 B Save

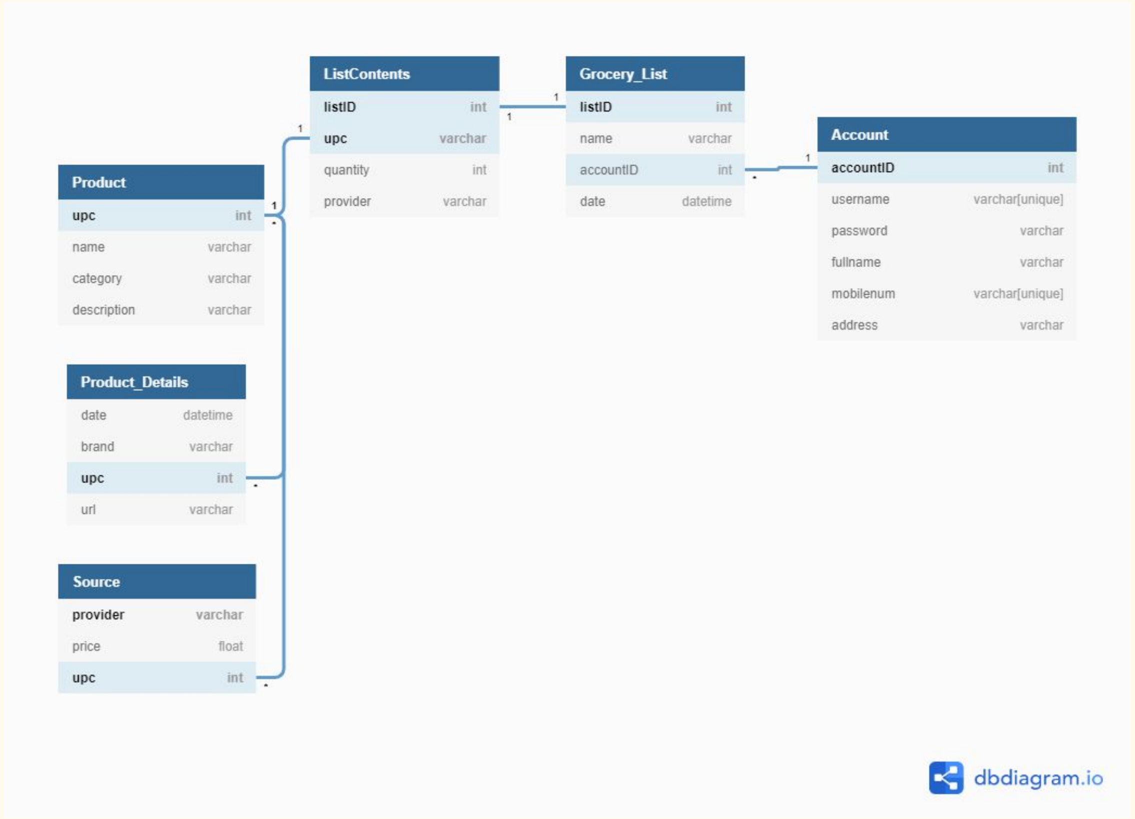
Pretty | Raw | Preview | Visualize | JSON

```
{
    "accountID": 13,
    "username": "Kun",
    "password": "12345",
    "fullname": "Kun Liu",
    "mobileNumber": "4436351486",
    "address": "500 W University Pkwy"
}
```

UML


DaoFactory
- Sql2o
+ instantiateSql2o()
+createTable()
RunApiServer
«interface»
AccountDao
Account
Full Name
Mobile Number
Username
Password
AmazonFetch*
+ASINtoUPC(String ASIN)
+getItem(String query)
Sql2oAccountDao
- Sql2o
+Sql2oAccountDao(Sql2o)
+addAccount(Account)
+checkLogin(Account)
+checkMobileNumber(#)
WalmartFetch*
+getItem(String query)
Postgres

Database



ListContents	
listID	int
upc	varchar
quantity	int
provider	varchar

Grocery_List	
listID	int
name	varchar
accountID	int
date	datetime

Account	
accountID	int
username	varchar[unique]
password	varchar
fullname	varchar
mobilenum	varchar[unique]
address	varchar

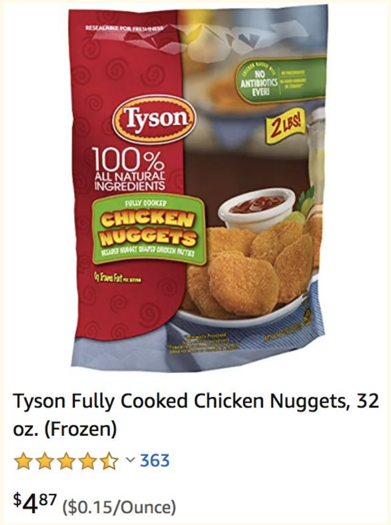
Product	
upc	int
name	varchar
category	varchar
description	varchar

Product_Details	
date	datetime
brand	varchar
upc	int
url	varchar

Source	
provider	varchar
price	float
upc	int

Webscraping

- Commercially interesting problem
  - Expensive APIs, limited data
- Provider APIs
  - JSON parsing
- HTML web scraping
  - Jsoup selector syntax



Tyson Fully Cooked Chicken Nuggets, 32 oz. (Frozen)

363

$4.87 ($0.15/Ounce)

```
<span class="a-price-whole">4</span>
<span class="a-price-fraction">87</span>
```

Price Comparison Algorithm

- How do we find the cheapest option?
- Imagine buying breakfast:
  - Eggs
  - Bacon
  - Orange Juice
- Consider item costs and delivery fees

Price Comparison Algorithm

From Walmart:
Eggs - $2.50
Bacon - $5.00
Orange Juice - $6.50
Delivery Fee - $4.00

From Amazon:
Eggs - $2.70
Bacon - $5.50
Orange Juice - $6.15
Delivery Fee - $6.00

Price Comparison Algorithm

From Walmart:
Eggs - $2.50
Bacon - $5.00
Orange Juice - $6.50
Delivery Fee - $4.00

From Amazon:
Eggs - $2.70
Bacon - $5.50
Orange Juice - $6.15
Delivery Fee - $6.00

- In this example, the algorithm buys ALL items from Walmart
- Only split carts when the savings exceed the delivery fee

Limitations

- Vulnerable to changes in webpage structure
- Limitations with optimization algorithm
- Only 2 grocery providers
- Webscraper is not very robust
- Cannot make orders through the app

Future Plans

- We all have jobs post-graduating (luckily)
- Will make it a public GitHub project for others to get inspiration

Take-Aways

- Learned agile practices we can bring to our software engineering jobs
- Learned about client-server architecture
- Learned about mobile app development
- Planning fallacies