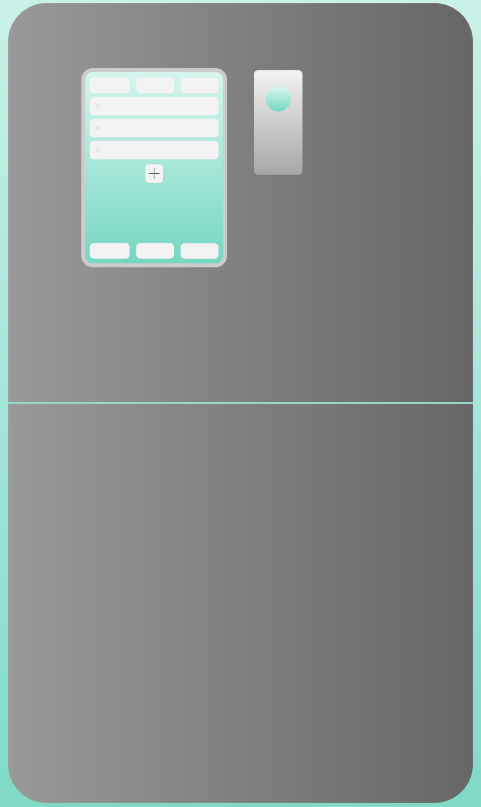


FridgeBuddy

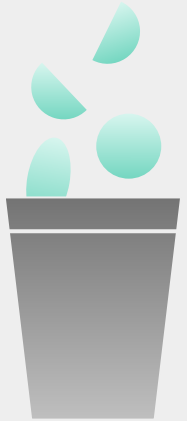
Your Waste-Reducing Smart Fridge Companion

Jeffrey Chen | Andres Garcia | Cole Wentzel | Trevor Chan

Introduction

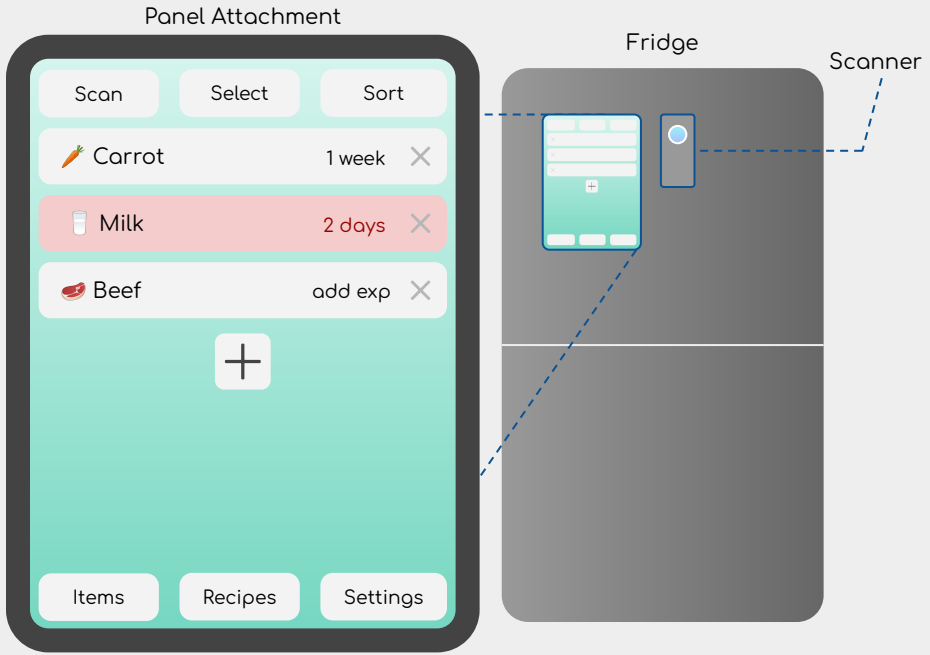
Problem statement

- America wastes approximately 2.5 billion tons of food every year (~40% of food supply)
- The average person discards $53.81 worth of spoiled food weekly
- Excess food production also contributes to 11% of greenhouse gas emissions

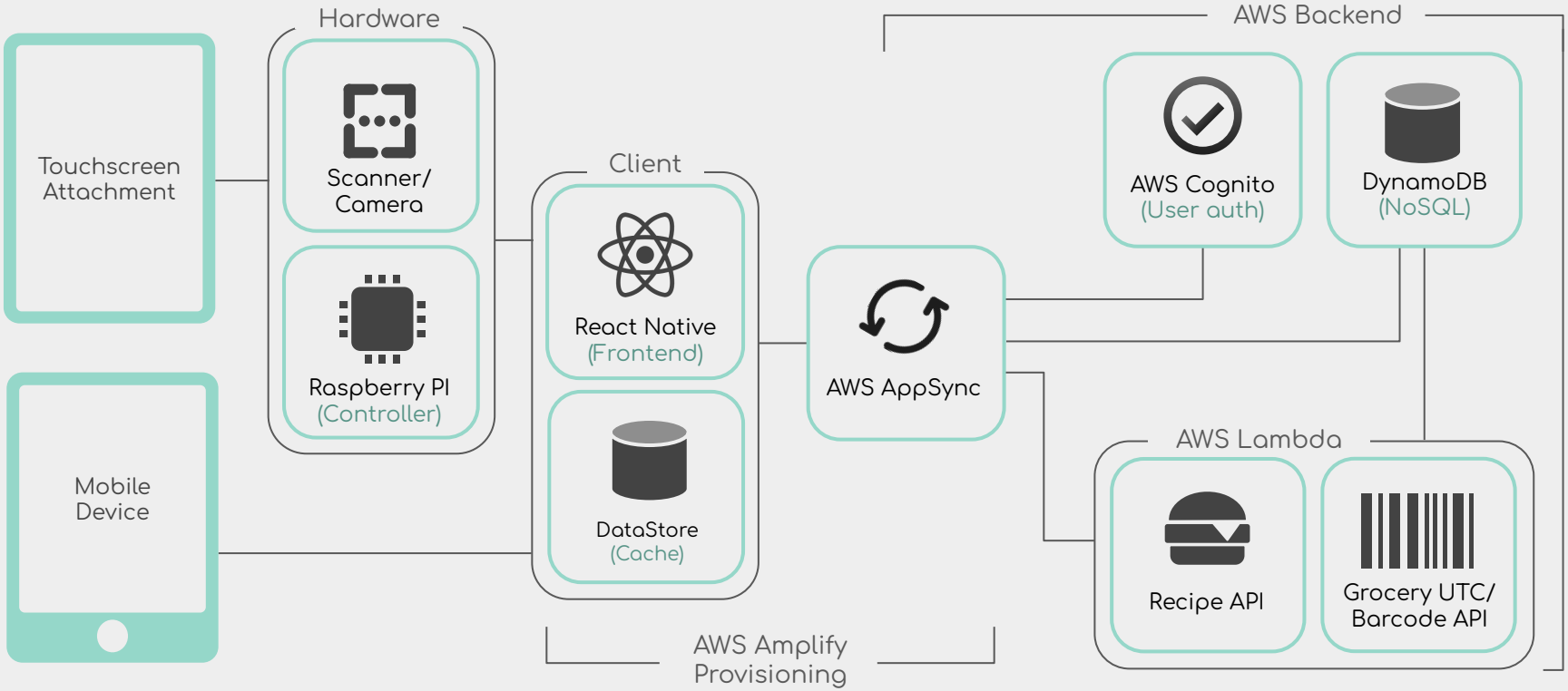


System Visualization

- Touchscreen unit that attaches to fridge via magnetic or adhesive brackets
- Displays and alerts item expiration dates
- Integrated scanner to scan barcodes, receipts and expirations
- Phone app to sync data



System Block Diagram



Requirements

Touchscreen Display

- Easy to set up
- Display current user items, quantity, and expiration date
- Input items via barcode scanner or manual input

Phone App

- Display and input items synced with cloud backend
- Push notifications for soon-to-expire items

API Use

- Fetch information about products
- Generate recipes with current ingredients

Reach Goals

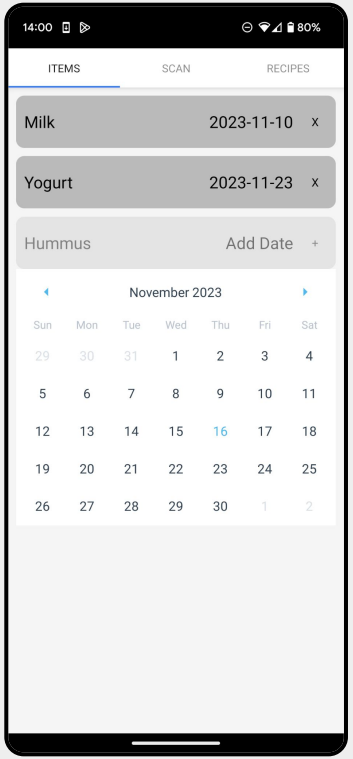
- Text recognition for expirations and receipts
- Low-profile enclosure for panel and controller

Technical Design: Touchscreen

- Prototype with Raspberry Pi
- On-board wifi and 4gb-ram option
- 10.1-inch touchscreen display with 3d-printed enclosure
- Attach to fridge via removable magnetic or adhesive brackets
- Wired power connection with unobtrusive wires

Technical Design: React Native

- Cross-platform for Android, iOS, and web
  - Single codebase for both mobile app and fridge attachment interface
- Displays inventory item
  - functionality to add/remove items and expiration dates
  - Manage camera/barcode input for input automation
  - Generate recipes from available ingredients
- Handle API calls to AWS backend



Technical Design: AWS Amplify

- Provision React Native application with AWS
  - Provide tools and services for full-stack development in React Native
- Manage GraphQL queries to subscribe to DynamoDB instance
  - Handle real-time updates from database
  - Define specific schema for requested data
- Utilize AWS DataStore to cache data locally
  - Provide faster access to data
  - Limit cost by reducing read/write throughput for remote database
  - Mitigate performance issues due to availability or network latency
- Authenticate users with AWS Cognito
  - Allow authentication through Oauth 2.0 or email/password

Technical Design: AWS DynamoDB

- Database system storing user and item information
- Amazon Web Service's NoSQL Database
  - Non-relational database—no defined schema
  - Can store multiple entity types in a single table
  - All read / write requests based on single unique key indexing (can be broken up into partition and sort key)
- Why NoSQL?
  - Better horizontal scalability than relational database systems
  - Simplifies gathering all user items and user information into one read request

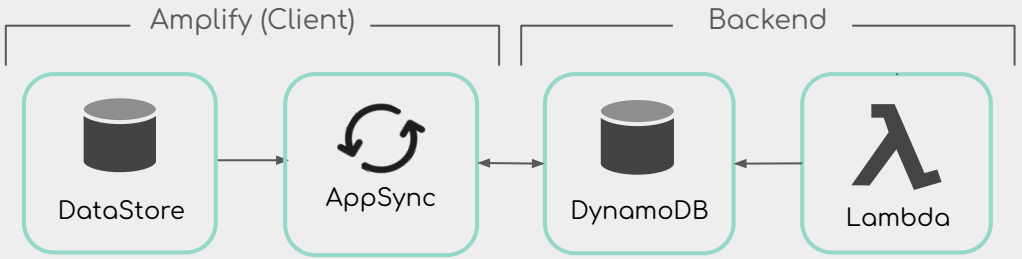
Technical Design: AWS DynamoDB

Partition Key (User ID)	Sort Key (Entity ID)							
UID1	UID1	name	email	username	password			
		Trevor	..@gmail	trevdawg	********			
	IT1	name	UPC	category	calories	img_url	quantity	exp_date
		Beans	085239325742	Canned vegetable	35.0	…	4,000	1699518602
	IT2	name	UPC	category	calories	img_url	quantity	exp_date
		Chips	028400421737	Chips & snacks	571.0	…	2	1701247942
UID2	UID2	name	email	username	password			
		Jeff	@bu.edu	jeffdawg	*******			

Technical Design: AWS DynamoDB

User Stories:

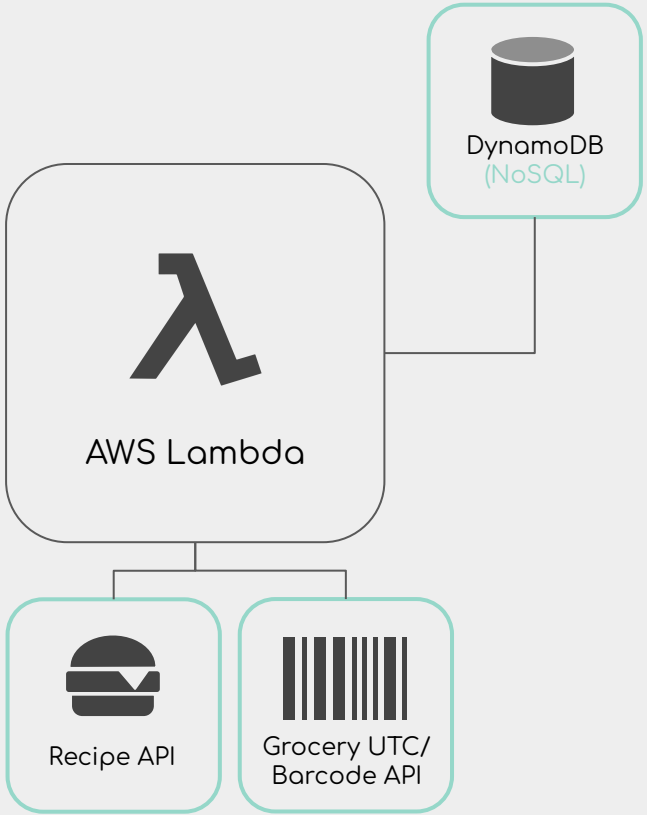
- **Add Item via Barcode**: AWS Lambda inserts item information
- **Local Item Changes (e.g. change quantity)**: Syncs changes via AWS Amplify DataStore and AWS AppSync
- **View User Items**: Query user items, sync changes via DataStore and AppSync



Technical Design: AWS Lambda

User Stories:

- Insert Item Using UPC Code: Query FoodCentral API for item information, insert into user's items
- Request Recipes: Query Spoonacular Recipe API for suggested recipes given items user has,



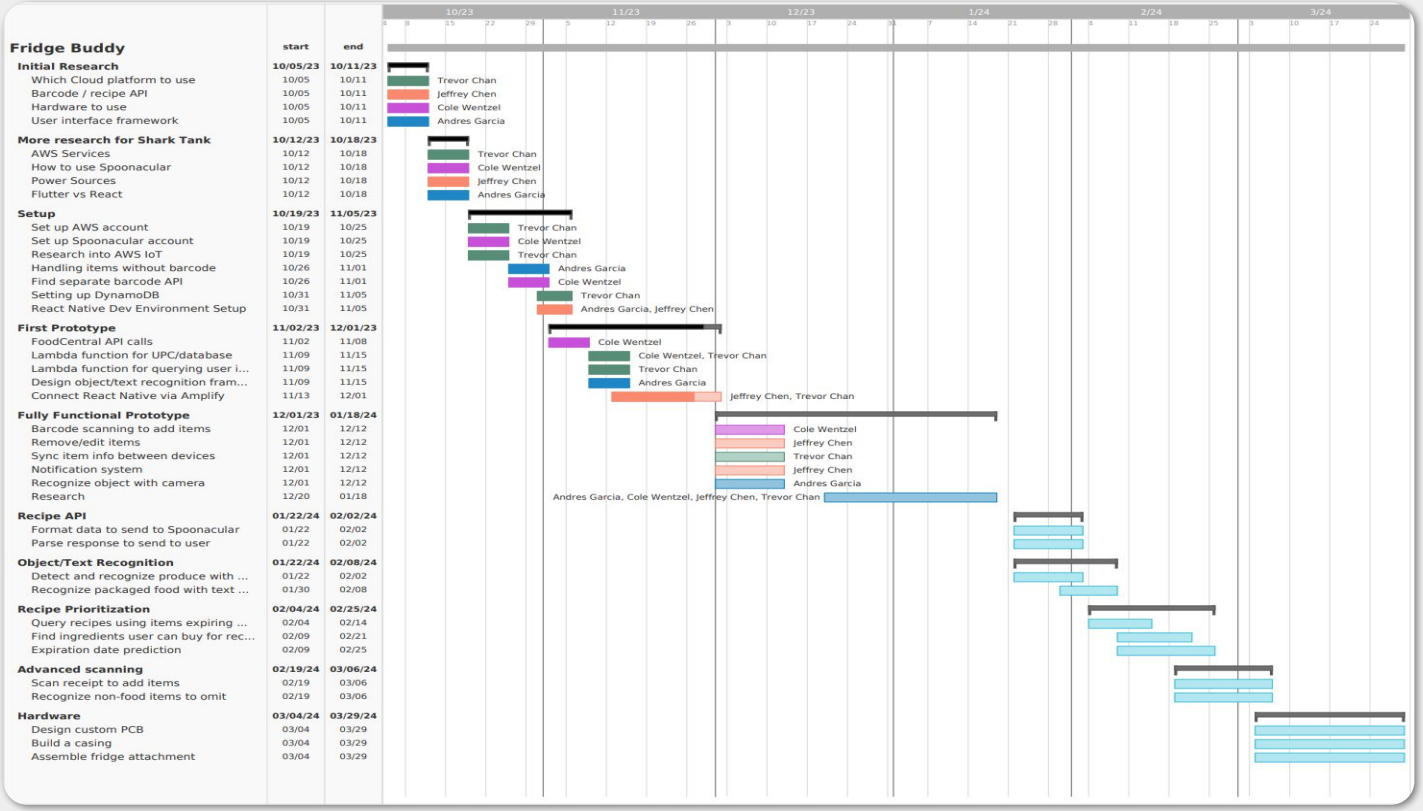
Technical Design: FoodCentral API

- Maintained by the US Department of Agriculture
- Contains only product codes related to food
- Over 450,000 products in database
- Generic information for products that may not have barcodes (ex. fruits, deli products)

Technical Design: Object/Text Recognition

- Using TensorFlow library within React Native app
- Pretrained model for identifying and classifying food offline
- First, object recognition attempts to identify food item
- If object recognition is not enough, text recognition is employed to identify item from packaging labels
- Produce recognized with object recognition, packaged food recognized with object + text recognition
- Available on mobile app and fridge attachment

Gantt Chart / Timeline



Fridge Buddy	start	end
Initial Research	**10/05/23**	**10/11/23**
Which Cloud platform to use	10/05	10/11
Barcode / recipe API	10/05	10/11
Hardware to use	10/05	10/11
User interface framework	10/05	10/11
More research for Shark Tank	**10/12/23**	**10/18/23**
AWS Services	10/12	10/18
How to use Spoonacular	10/12	10/18
Power Sources	10/12	10/18
Flutter vs React	10/12	10/18
Setup	**10/19/23**	**11/05/23**
Set up AWS account	10/19	10/25
Set up Spoonacular account	10/19	10/25
Research into AWS IoT	10/19	10/25
Handling items without barcode	10/26	11/01
Find separate barcode API	10/26	11/01
Setting up DynamoDB	10/31	11/05
React Native Dev Environment Setup	10/31	11/05
First Prototype	**11/02/23**	**12/01/23**
FoodCentral API calls	11/02	11/08
Lambda function for UPC/database	11/09	11/15
Lambda function for querying user i...	11/09	11/15
Design object/text recognition fram...	11/09	11/15
Connect React Native via Amplify	11/13	12/01
Fully Functional Prototype	**12/01/23**	**01/18/24**
Barcode scanning to add items	12/01	12/12
Remove/edit items	12/01	12/12
Sync item info between devices	12/01	12/12
Notification system	12/01	12/12
Recognize object with camera	12/01	12/12
Research	12/20	01/18
Recipe API	**01/22/24**	**02/02/24**
Format data to send to Spoonacular	01/22	02/02
Parse response to send to user	01/22	02/02
Object/Text Recognition	**01/22/24**	**02/08/24**
Detect and recognize produce with ...	01/22	02/02
Recognize packaged food with text ...	01/30	02/08
Recipe Prioritization	**02/04/24**	**02/25/24**
Query recipes using items expiring ...	02/04	02/14
Find ingredients user can buy for rec...	02/09	02/21
Expiration date prediction	02/09	02/25
Advanced scanning	**02/19/24**	**03/06/24**
Scan receipt to add items	02/19	03/06
Recognize non-food items to omit	02/19	03/06
Hardware	**03/04/24**	**03/29/24**
Design custom PCB	03/04	03/29
Build a casing	03/04	03/29
Assemble fridge attachment	03/04	03/29

Future This Semester / Next Semester

- End of fall semester:
  - Barcode image recognition to fetch UPC and add items
  - Sync data for multiple devices logged into the same account
  - Implement notifications for expiration dates
  - Create attachment prototype with Raspberry PI
- Spring semester:
  - Design a custom PCB
    - Shape to fit thinner form factor opposed to Raspberry Pi
    - Fine tune specs to suit our use case
    - Eliminate bulk/power draw from unused components
  - Text recognition
    - Scan a receipt to add all food items quickly
    - Recognize non-food items listed on receipt and omit them

Thank you

Questions?