littleFridge

Mei Chen (meic2) | Moderator: Nitish Natarajan (nitishn2)

This is a mobile app about meal planner according to what you have in the refridgerator for CS242.

Google Score sheet link: https://docs.google.com/spreadsheets/d/193Ed_OLXtBZB7jpehPz8dhzlgu3SEUEHjLR2GWkMmAY/edit?usp=sharing

Abstract

Project Purpose

Create an app that can help the user to find out what they can cook for their meal according to what they have in the fridges.

Project Motivation

When we cook in our home, the leftovers of the ingredients are often awakwardly placed in the fridges when we finish one big meal. We don't know how to deal with it sufficiently, and that's why we need an app to help the user cook the meal via those leftovers.

Technical Specification

- API: BigOven, Edamam Recipe Search & Diet, Zestful Recipe & Ingredient Analysis
- Platform: Cross-platform app (React Native)
- Programming Languages: JavaScript Backend: Python for Flask
- Stylistic Conventions: Airbnb JavaScript Style Guide and Python style guide
- SDK: Facebook SDK for React Native (possible for social login)
- IDE: Visual Studio Code, Deco (IDE for React Native, acquired by Airbnb)
- Tools/Interfaces: Mobile devices
- Target Audience: Broad-range audience
- Backend: Flask, MongoDB

Functional Specification

Features

- allow users to store your own fridge into a digital database (potentially with visualization)
- when user cooked some dishes, they can synchronize and remove/add meal from the fridge database.
- can update the expiration/how many portion has consumed based on the user update.
- can mark the ingredients based on the expiration date
- recommend the meal based on your priority of consuming ingredients, or based on the mark of the user (he/she want to consume this ingredients right now).
- if the user use the meal recommendation, the corresponding ingredients will disappear as well.

Scope of the project

- Limitations include that the recommendation system might not meet the user's personal taste.
- Assumptions include that the ingredients should only be in the scope of the API database.

Brief Timeline

Week 1:

- set up the MongoDB database for users' own fridge database (ingredients + meals)
- set up flask server for fetching the API
- create your own API to create corresponding digital library category
- have basic inputs/outputs to update/delete from the database
- implement the query in the backend for filtering on the priority/ specific ingredients/ most popular meal/ most healthy meal etc.

Week 2:

- set up React App
- log-in, fridge, ingredient page set up

Week 3:

- meal database for user set up
- meal seperate page set up

Week 4:

- meal plan page set up

Rubrics

Week 1

Category	Total Score Allocated	Detailed Rubrics
set up the MongoDB database	2	0: Didn't implement anything 1: +1 pt: System can write from a database without errors 2: System connected to a database and can read from a database without errors
Flask Server	3	0: Didn't implement anything 1: implemented a fetch towards the corresponding API 2: set up the meal database with the specific type required by the API (standardize the database) 3: set up the ingredient database with the specific type required by the API (standardize the database)

Category	Total Score Allocated	Detailed Rubrics
API	6	0: Didn't implement anything 2: implemented GET for each database 4: implemented UPDATE for each database 5: implemented POST for each database 6: implemented DELETE for each database -0.5 pt for each wrong return types (200, 400, 415, 404) for each route -0.5 pt for not reporting errors for each route
externalAPI	4	0: Didn't implement anything 2: implement a basic get from the API 4. implement multiple filters/get from the external API
unit test	10	0: Didn't implement tests 1: for every 2 unit tests, gain 1 point

Week 2

Category	Total Score Allocated	Detailed Rubrics
Viewer layer	7	0: Didn't implement anything 2: Barcode view are implemented 2: Fridge view are implemented 3: Navigation between screens 4. Clean separation between model and view 5. Loading views for both barcode profile view and Fridge view
Model layer	3	-3 pts if JSON object used directly as a model -1 pt if no clean separation of parsing request into the data class -1 pt if no handling of errors
layout design	3	+1 pt: Elements resize with viewport changes and without horizontal overflow +1 pt: Accommodate for specified screen sizes
Manual test plan	5	0 pt: No manual tests; for every 2 manual test, gain 1 point
unit test	5	0: Didn't implement tests for every 2 unit test, gain 1 point

Week 3

Category	Total Score Allocated	Detailed Rubrics `

Category	Total Score Allocated	Detailed Rubrics `
Layout Design	1	Complete the design for each screens
meal memo page	4	0: Didn't implement anything 2: implemented meal memo page 4: completed functionailty of updating the database
individual meal page	4	0: Didn't implement anything 2: implemented individual meal page 4: completed functionailty of updating the database
navigate between screens	3	+1 pt: able to navigate between screens +2 pt: able to go back to the previous page by pressing the back button (in individual meal page) +2 pt: Navigation from individual meal page / meal memo page
individual ingredient page	2	0: Didn't implement anything 2: implemented individual ingredient page 4: completed functionailty of updating the database
Manual test plan	5	0 pt: No manual tests; for every 2 manual test, gain 1 point
snapshot test	5	0: Didn't implement tests for every 2 manual test, gain 1 point

Week 4

Category	Total Score Allocated	Detailed Rubrics
wrap up on previous assignments	3	0: Didn't implement anything 3: Finish the grading based on criteria
meal plan page	4	0: Didn't implement anything 2: implemented meal memo page and fetch the data according to the query 4: also dynamically filter based on the drop down
navigate between screens	5	+1 pt: able to navigate between screens +2 pt: able to go back to the previous page by pressing the back button (from meal search to individual meal page <-> meal page back to the meal planner page) +2 pt: meal plan's submission will dynamically update in the recipe page
search bar functionality	3	+1 pt for each screen being able to search and dynamically change the view accordingly

Category	Total Score Allocated	Detailed Rubrics
Manual test plan	5	0 pt: No manual tests; for every 2 manual test, gain 1 point
unit test	5	0: Didn't implement tests for every 2 manual test, gain 1 point