

April 5, 2024

FlexyCook

Recipe Web Application



<https://unsplash.com/photos/wooden-ladle-and-chopping-board-with-ginger-during-daytime-vuDXJ60mjOA>

Yuna Hu, Monica Gao, Fuka Nagata,
Simran Kaur (left since March 19th)

(CPSC-2350-M01 Group 4)

GitHub Repository: <https://github.com/yunahu/flexyc-ook-api>
<https://github.com/yunahu/FlexyCook>

Table of Contents

 Overview	4
 Achievements	4
 Not Yet Accomplished	4
 SDLC	6
 Features & User Stories	7
 Features	8
 10 Usability Heuristics	14
 Tests	21
 CI / CD Infrastructure	25
 Project Challenges	27
 Lessons Learnt / Project Takeaway	28
 Work Division	29
 Sources	30



Overview

This recipe web application is designed to provide an easy-to-use tool for people who would **like to cook and explore the culinary world**. FlexyCook combines common cooking practices that are usually done in separate applications into a one-click-ready format. The two APIs: Spoonacular and todoist provide some of the most commonly expected features for foodies, including **searching for recipes with different parameters** and **tools to support the cooking process**.



Achievements

- Successfully implemented the planned 2 APIs and 6 features.
- Gained necessary knowledge about software development using the React framework.
- Granted experience in collaborating with other programmers through the use of Git and Github.
- Learned better time managing methods such as work breakdown structure.
- Experienced the whole project management process from requirement gathering to product deployment.



Not Yet Accomplished

- Additional and optional features listed during the planning phase, include theme switching by current weather and time, like/dislike recipe lists, and recommend recipes by geolocation.



Overview

For our SDLC, we have chosen the **Kanban Agile framework**. This took advantage of both agile and Kanban methods, iterating through the development cycle and choosing tasks from the Kanban board. We have had regular weekly meetings to decide what to do for the week, address any blockers hindering progress, and reflect on what has been done and what changes were made during the week. The Kanban board provided visibility into task progression and overall workflow, each team member was left to choose the tasks and their deadlines, allowing them to work at their own pace.

Things that went well

- It enabled team members to choose the right tasks at the pace that suited us, which accelerated the progress of product creation.
- Regular weekly meetings and a clear board of task progress to keep each other accountable, ask questions, and exchange knowledge, allowed each team member to be on the same page at all times, and there were no gaps in communication or knowledge.
- By having a visual representation of task progression, we were able to avoid scenarios where two people worked on the same task and were able to pair programs on any challenging tasks.

Things did not work

- Because of the Kanban system, there was an uneven distribution of work among team members.
- We could not fully use the Kanban system, and we sometimes forgot to change a task's progress (e.g., from "not started" to "in progress" / even when the task is completed, does not move to "Done").

Features & User Stories

Recipe API ([Spoonacular](#))

- Search by Ingredients

 As a person who travels often, I want to find recipes from the ingredients I have so that I can prevent food waste.

- Search by Nutrient

 As an athlete with strict dietary restrictions, I want to find recipes based on nutritional criteria so that I can make dishes that supplement deficient nutrients or do not have specific nutrients that I want to avoid.

- Recipe Recommendation

To-do list API ([Todoist](#))

- Cooking steps

 As a person with mild dyslexia who has difficulty reading long sentences, I want a recipe that is written in brief, separated steps so that even a person with a disability who has difficulty reading can easily process all the steps of the recipe.

- Shopping list

- General todo list

 As a working mom who does not have much time to find recipes, I want to be able to save all the information I need for a recipe with just one click of a button so that I can access it conveniently whenever I need it.

Features

Recipe API ([Spoonacular](#))

Search by Ingredients

- Users can find recipes that include ingredients such as "salmon" or "cream" by entering them into the search bar.

The screenshot shows the Spoonacular search interface. At the top, there's a search bar with the query "Salmon, cream". Below it is an "Advanced Search" dropdown. The main area displays four recipe cards:

- Salmon Confit with Lemongrass Sauce**: A dish of salmon with lemongrass sauce. Details: 865 kcal, 45 minutes, 2 servings. Tags: Gluten free, Lunch, Healthy, Sour. [Check It Out](#)
- Bigoli with smoked salmon**: A dish of pasta with smoked salmon. Details: 579 kcal, 45 minutes, 3 servings. Tags: Pescatarian, Lunch, Fatty. [Check It Out](#)
- water, tomato sauce, beef bouillon cubes,**: A bowl of soup. Details: 4 servings.
- Laksa kuhing**: A bowl of laksa. Details: 734 kcal, 45 minutes, 2 servings. Tags: Dairy free, Dessert, Sweet. [Check It Out](#)

- The user can search from both the navigation bar and the search page.

The screenshot shows the Spoonacular search interface with a dark background. At the top, there's a search bar with the query "chocolate, ice cream, banana". Below it is an "Advanced Search" dropdown. The main area displays two recipe cards:

- Raw Vegan Chocolate and Raspberry Cake**: A chocolate cake topped with raspberries. Details: 565 kcal, 45 minutes, 12 servings. Tags: Dairy free, Dessert, Sweet. [Check It Out](#)
- Mini Banana Splits**: A dessert featuring banana slices, whipped cream, and cherries. Details: 734 kcal, 45 minutes, 2 servings. Tags: Gluten free, Sweet. [Check It Out](#)



Features (cont.)

Search by Nutrients

- The user can set a minimum and maximum amount of nutrients, such as fat, calcium, protein, and carbs, and find recipes that follow their nutrient requirement.
- The user can also set a range for other fields such as calories, alcohol, and caffeine and find recipes that meet those limits.

The screenshot shows a search interface with a green header bar. On the left, there's a search input field with placeholder text "Enter ingredients with comma-separated list" and a "Search" button. Below it is an "Advanced Search" dropdown. In the center, there's a "Select filter" section with a "Select Range" button, a "Select Nutrient" dropdown, and an "Amount" input field with a plus sign. To the right, a "Select Nutrient" dropdown menu is open, listing various nutrients: Carbs, Protein, Fat, Fiber, Iron, Sodium, Sugar, Magnesium, Folate, Zinc, FolicAcid, Calcium, Cholesterol, VitaminC, VitaminE, VitaminB12, Caffeine, Alcohol, and Calories. At the bottom of the interface, there are three selected filters: "Min Calories 1000(kcal)", "Min Fiber 20(g)", and "Max Caffeine 100(mg)".

- Users can use the advanced search feature on the search page and navigation bar.
- Users can search by both nutrients and ingredients.

Search Results:

You are searching the recipes by
Nutrients:Min Calories1000kcal



Corn-Crusted Fish Tacos With Jalapeno-Lime Sauce and Spicy Black Beans

1342 kcal | 45 minutes | 4 servings

Mexican | Gluten free | Lunch | Healthy

[Check It Out](#)

You are searching the recipes by
Ingredients: Apple, cream
Nutrients:

Min Calories-100kcal

Max Calories-300kcal



Appetizing Apple and Almond Soup

175 kcal | 45 minutes | 4 servings

Gluten free | Antipasti | Sweet

[Check It Out](#)



Dried Fruit and Ginger Scones

272 kcal | 45 minutes | 8 servings

English | Morning meal | Sweet

[Check It Out](#)



Features (cont.)

Recipe Recommendation

- It displays recommended recipes every time the home page is loaded.
- If the user does not like the recipes, they can refresh the recommendations until they are satisfied, and the recipes they do not like will no longer be displayed.

The screenshot shows the FlexyCook homepage with a green header bar. On the left is the FC FlexyCook logo. To its right is a search bar with placeholder text "Enter Ingredients with comma-separated list" and a "Search" button. Further right are "Theme" and "TODO" buttons. Below the header, there are four recipe cards:

- Gluten Free Onion Rings**: An image of golden-brown onion rings in a white bowl. Below it is a brief description of the dish and its ingredients, including a note that it's "Lacto ovo vegetarian". A "Check It Out" button is at the bottom.
- Chèvre With Sautéed Grapes**: An image of a salad with cheese and grapes. Below it is a brief description of the dish and its ingredients, including a note that it's "Gluten free".
- Garlic Lemon Pepper Shrimp Salad (Clean Eating)**: An image of shrimp on a bed of greens. Below it is a brief description of the dish and its ingredients, including a note that it's "Gluten free".
- Buttery Pull Apart Monkey Bread**: An image of monkey bread. Below it is a brief description of the dish and its ingredients, including a note that it's "Dessert".

A "Refresh Recipes" button is located on the right side of the main content area. At the bottom, there's a partial view of another recipe card for "Easy Chicken with White Wine Sauce".

Search by Tags

- Users can search by dish types, diets, intolerances, and cuisines by clicking tags.

The screenshot shows a search results page for "Gluten free" recipes. At the top left is a "Search Results:" heading. Below it, a message says "You are searching the recipes by Diet: Gluten free". There are two recipe cards shown:

- Cannellini Bean and Asparagus Salad with Mushrooms**: An image of a salad. Below it is a brief description of the dish and its ingredients, including a note that it's "Gluten free". A "Check It Out" button is at the bottom.
- Red Lentil Soup with Chicken and Turnips**: An image of a soup. Below it is a brief description of the dish and its ingredients, including a note that it's "Gluten free". A "Check It Out" button is at the bottom.

To the right of the search results is a sidebar titled "Advanced Search ▾". It features a section titled "Recommended Tags" with several colored buttons:

- Orange: American, No Wheat, No Shellfish
- Purple: Ketogenic, Thai, Greek
- Red: Bread
- Green: Vegan
- Blue: Snack, Fast



Features (cont.)

To-do list API ([Todoist](#))

Cooking steps

- The original recipe can be processed into simple individual cooking steps with a single click of a button.
- The users can easily understand the cooking steps without having to read a long recipe.
- When the user has completed a step, they can check that step off to keep track.
- The user can also save the cooking steps and look back later.

The image shows two screenshots of the Todoist application. The left screenshot displays a recipe card for 'Bibimbab (Korean Rice w Vegetables & Beef)'. The card includes the title, ingredients (2 cups), nutritional information (90 kcal, 72 g protein, 10.87 g fat), and cooking time (45 minutes). Below the card are buttons for 'Gluten free', 'Lunch', and 'Sweet'. The right screenshot shows a list of cooking steps for the same dish. The steps are numbered 1 through 7, describing the preparation of rice, vegetables, and other ingredients. Step 1 is checked off.

General todo list

- This is used for task management as a general to-do list.
- The user can create sub / sub-sub /sub....sub-tasks.
- If the user wants to delete the task, they can delete it by “Delete Task”, or the task will be removed by checking it.

The image shows two screenshots of the Todoist application. The left screenshot shows a general to-do list with tasks like 'New task' and 'Add task'. The right screenshot shows a more detailed view with a main task 'Cleaning' and a sub-task 'Sink' which is checked off.



Features (cont.)

Shopping list

- The user can easily generate a shopping list for a recipe with the click of a button.
- If there are any items that they need to buy, they can add them manually to their shopping list on the spot and refer to it while shopping.

The screenshot displays the FlexyCook application interface. At the top, there is a search bar with placeholder text "Enter ingredients with comma-separated list" and a "Search" button. To the right are "Theme" and "TODO" settings. Below the header, a recipe card for "Bibimbab (Korean Rice w Vegetables & Beef)" is shown, featuring a photo of the dish, a list of ingredients, and nutritional information. A modal window titled "localhost:5173 says Success" is overlaid on the page. In the bottom left, a note says "Cook rice according to package directions." with a list of required ingredients. In the bottom right, a separate "Shopping List" tab is open, showing a list of items with checkboxes and a text input field for adding new items like "Facial tissue".

Bibimbab (Korean Rice w Vegetables & Beef)

Ingredients:

- Soy sauce: 1 Tbsp
- Spinach: 1 bunch
- Sandwich steaks: 1 lb

Nutrients:

- Calories: 1090.48 kcal
- Fat: 32.72 g
- Saturated Fat: 10.87 g

1090 kcal | 45 minutes | 4 servings

Korean Gluten free Lunch Sweet

Cook rice according to package directions.

Equipments: none
 Ingredients: rice

New task

soy sauce - 1 Tbsp

gochujang - 4 Tbps

carrots - 2

spinach - 1 bunch

garlic clove - 1

cucumber - 1

rice - 632 grams

olive oil - 1 Tbsp

sesame oil - 4 teaspoons

sprinkle gochugaru - 4 servings

sesame seeds - 1 teaspoon

sandwich steaks - 453.592 grams

eggs - 6

Facial tissue



Features (cont.)

All these lists can be synced to the Todoist app.

The screenshot shows the Todoist mobile application interface. At the top, there's a navigation bar with icons for Apple, Share, View, and more. Below the bar, the main screen displays a project titled "FlexyCook". Under this project, there are three sections: "Shopping List" (29 items), "Memos" (3 items), and "Tortellini in Brodo" (2 items). The "Shopping List" section contains items like "spinach - 1 bunch", "cucumber - 1", "rice - 632 grams", and "garlic clove - 1". The "Memos" section includes tasks for "Cleaning" (Bathroom, Sink) and "Tortellini in Brodo" (Heat stock to boil, Serve immediately). The "Tortellini in Brodo" section provides a detailed recipe with numbered steps. At the bottom of the screen, there's a footer with project navigation and a search bar.

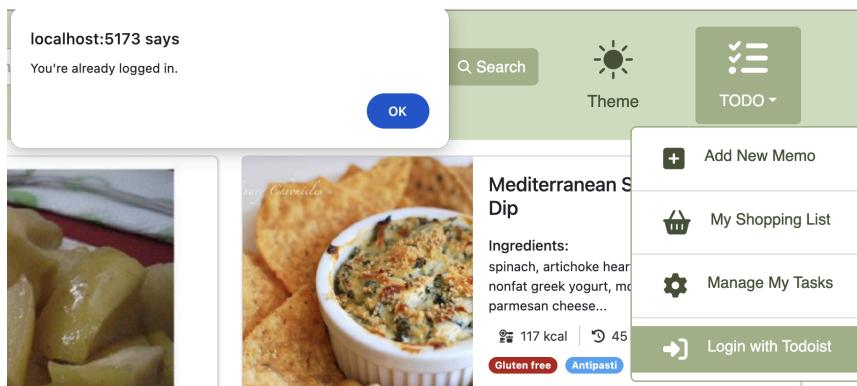
This screenshot shows the same Todoist mobile application interface as the previous one, but with a different focus. A task in the "Bathroom" section of the "Memos" list is being edited. A modal dialog box is open, prompting the user to enter a "Description" (e.g., "Sink") and providing options for "Due date", "Priority", "Reminders", and "Upgrade". The "Cancel" and "Save" buttons are visible at the bottom of the dialog. The rest of the screen shows the same project structure and lists as the first screenshot.



10 Usability Heuristics

1. Visibility of System Status

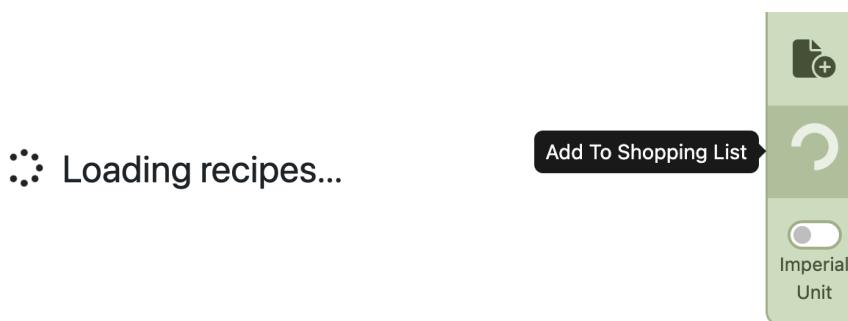
- Show the user has logged in to todoist



- Show the user is clicking the button



- Show recipes are loading / the list is creating



- Show lists were successfully created

localhost:5173 says

✓ Successfully created a cooking step list

OK

localhost:5173 says

✓ Successfully added to the shopping list

OK



10 Usability Heuristics (cont.)

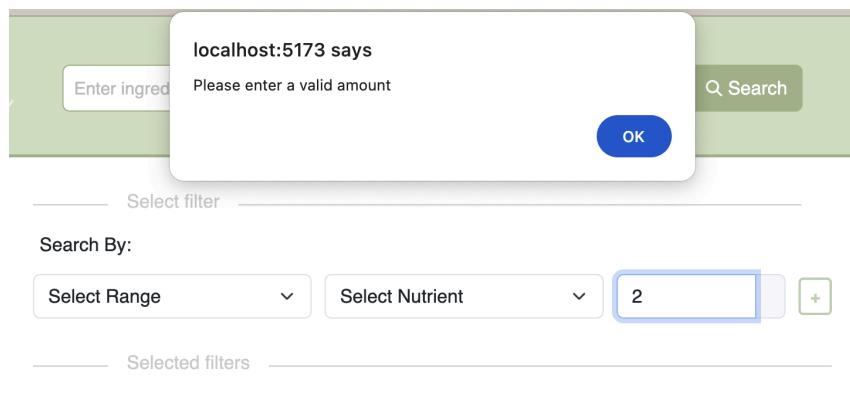
2. Match Between the System and the Real World

- Shows what the user is searching recipes by

Search Results:

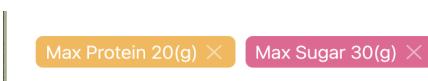
You are searching the recipes by
Ingredients: Salmon, cream
Nutrients:Max Fiber2g

- Let the users know when they've entered a invalid amount, and not saying "you've entered illegal input"

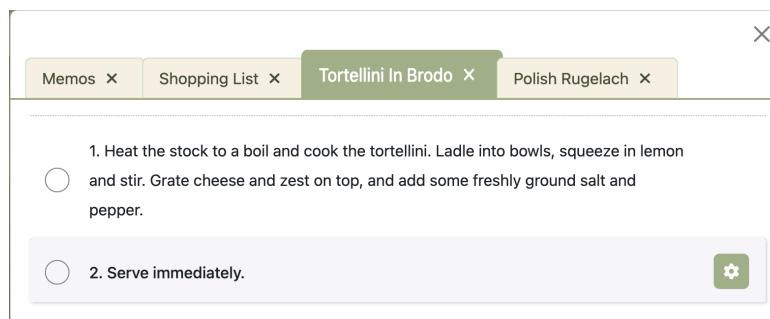


3. User Control and Freedom

- If the users mistakenly added a tag, they can delete them by clicking the close button



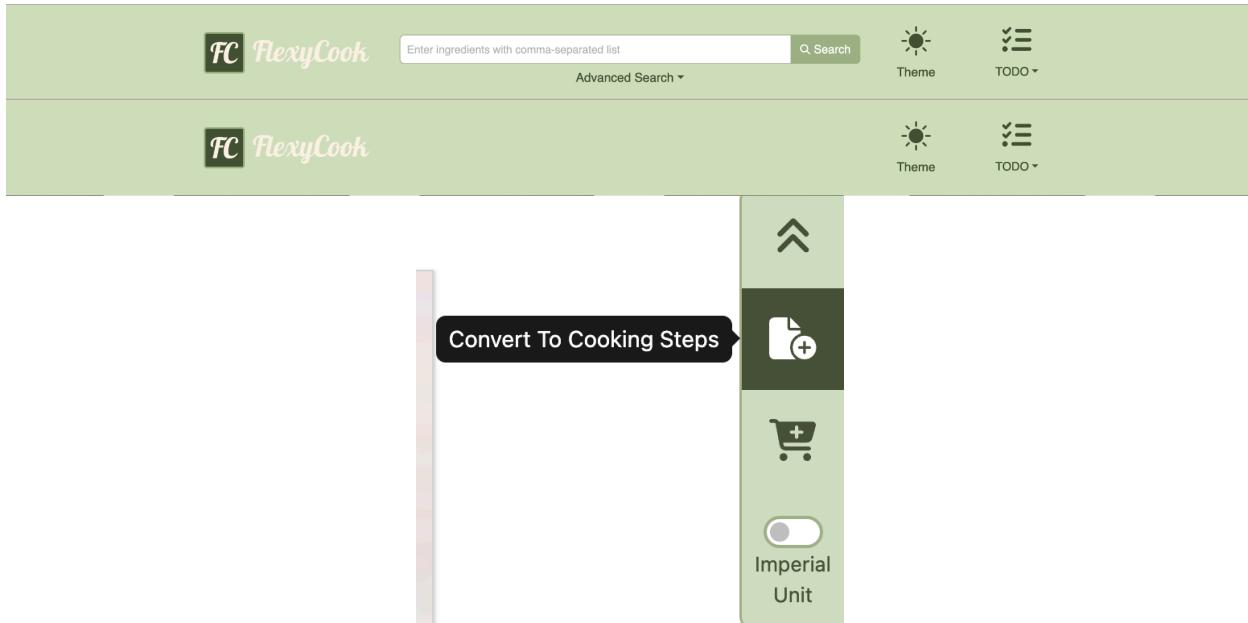
- Users can delete tasks through the settings, which are placed right side of the task



💡 10 Usability Heuristics (cont.)

4. Consistency and Standards

- Consist design and alignment



- Cards have the same structure

The image displays three cards from the FlexyCook app, each representing a different recipe:

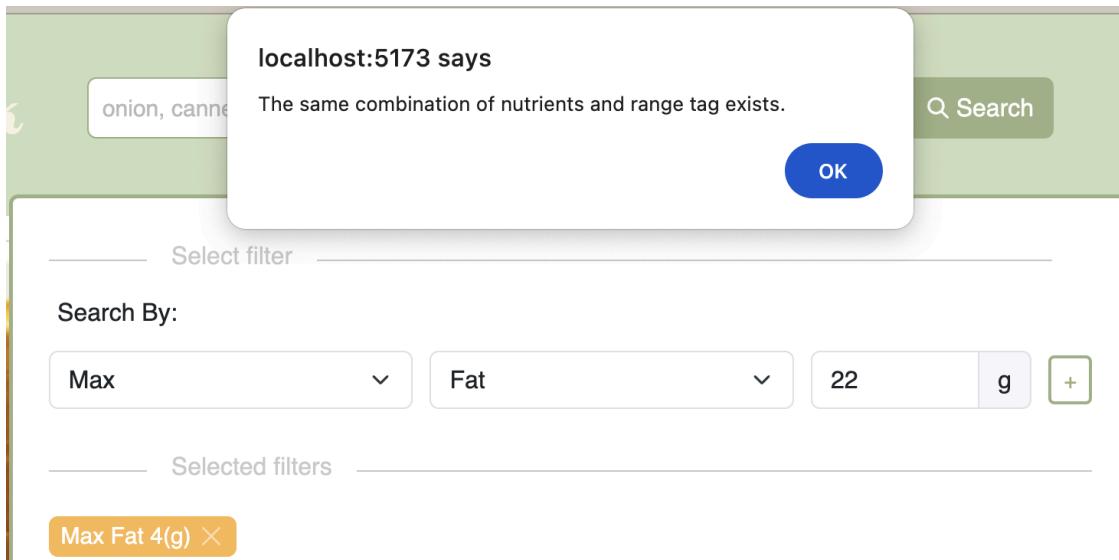
- Mango Fried Rice**: A black square bowl filled with fried rice containing mango pieces. The card includes the title, ingredients (chicken stock, seasoning cubes, mango, rice, vegetables), nutritional information (546 kcal, 45 minutes, 2 servings), and tags (Chinese, Gluten free, Side dish, Salty). A 'Check It Out' button is at the bottom.
- Rosemary Thyme Marinated Mushrooms**: A close-up photo of marinated mushrooms. The card includes the title, ingredients (water, balsamic vinegar, olive oil, garlic, rosemary, thyme), nutritional information (56 kcal, 45 minutes, 6 servings), and tags (Gluten free, Side dish, Salty).
- Brussels Sprouts with Bacon and Shallots**: A close-up photo of Brussels sprouts with bacon. The card includes the title, ingredients (bacon, brussels sprouts, lemon juice, salt and pepper, shallot), nutritional information (124 kcal, 45 minutes, 4 servings), and tags (Gluten free, Side dish, Salty).
- Wild Blueberry Lemon Muffins**: A close-up photo of a muffin with blueberries. The card includes the title, ingredients (all purpose flour, baking powder, butter, buttermilk, egg, lemon juice), nutritional information (147 kcal, 45 minutes, 14 servings), and tags (Lacto ovo vegetarian, Morning meal, Sweet).



10 Usability Heuristics (cont.)

5. Error Prevention

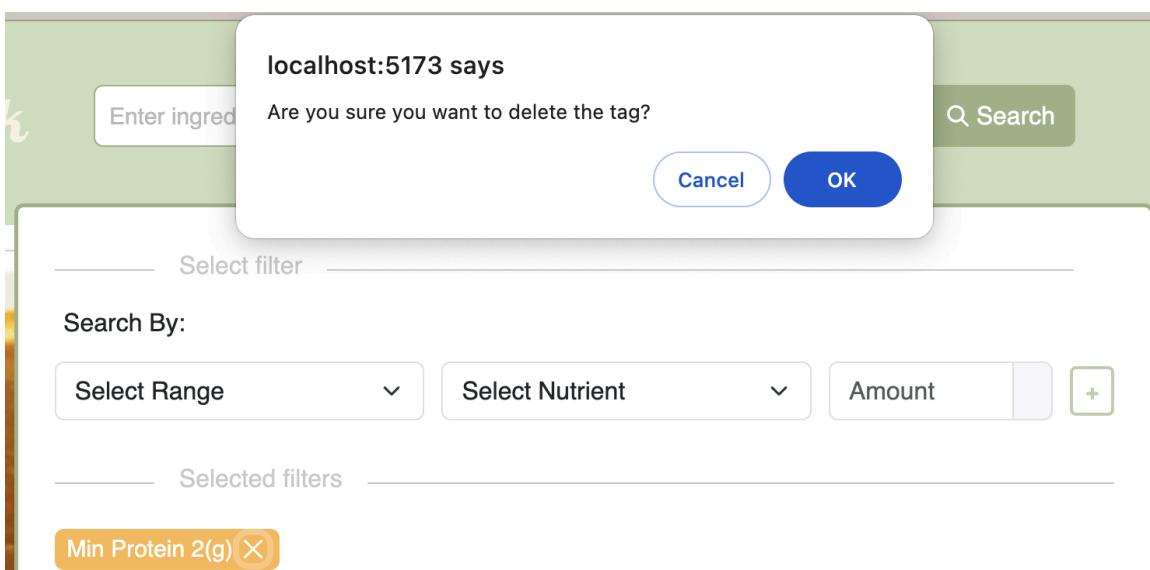
- Disable adding a tag that has the same nutrient and range



- Disable the button when there are not enough parameters



- Show confirmation when the users try to delete the tag



💡 10 Usability Heuristics (cont.)

6. Recognition Rather than Recall

- Depending on the nutrient, change the scale. Recall nutrients

Folate 22 µg

VitaminE 22 mg

Carbs 22 g



- Enable to see ingredients in both imperial and metric scale

Ingredients:

Pecans: 99 g
Powdered sugar: 240 g
Cocoa powder: 2 Tbsps

Nutrients:

Sodium: 12.66 mg

Ingredients:

Pecans: 1 cup
Powdered sugar: 2 cups
Cocoa powder: 2 Tbsps

Nutrients:

Sodium: 12.66 mg



- Show equipments with the image to recall how it looks

Remove from heat and let cool for about 10 minutes, season with a bit of salt, and remove the flesh from the eggplant. If there is too much water, drain in a strainer. Set aside.

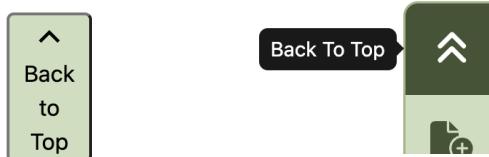
Equipments: sieve
 Ingredients: eggplant, water, salt



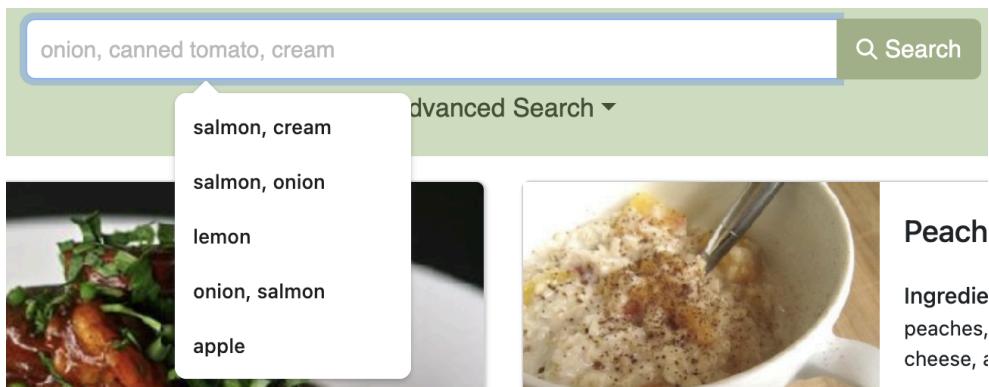
10 Usability Heuristics (cont.)

7. Flexibility and Efficiency of Use

- Enable back to top



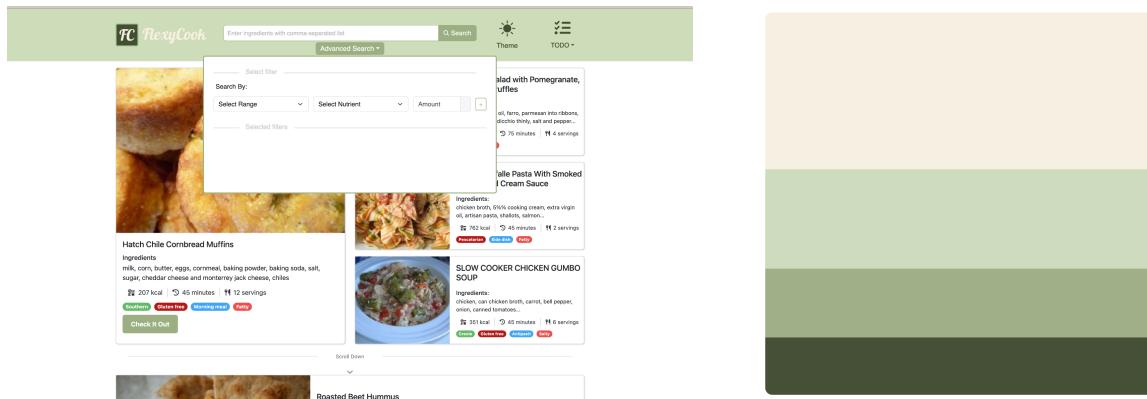
- Search History



- Keyboard shortcut:
 - Create tags & submit a search by clicking enter key
 - Escape from the toggle menu with the ESC key

8. Aesthetic and Minimalist Design

- Used few fonts & colours, colorblind-friendly colours





10 Usability Heuristics (cont.)

9. Help Users Recognize, Diagnose, and Recover from Errors

- If the recipe is not found, or there is an internet/server error, offer solutions



What You Can Do:

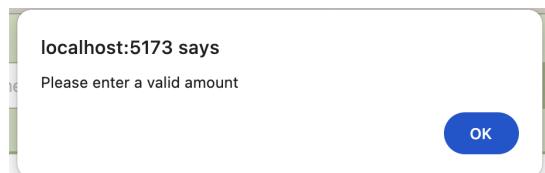
- Check your spelling
- Use ingredients as searching keywords
- Double check the filter setting in Advanced Search panel



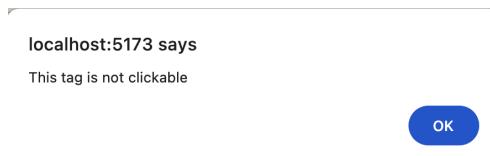
Try these:

- Check internet connection
- Reload page
- Check back in a few minutes

- If the user entered an invalid input(characters, negative amount), let the user know to enter a valid number



- If the user clicks the unclickable tag, show the tag is unclickable

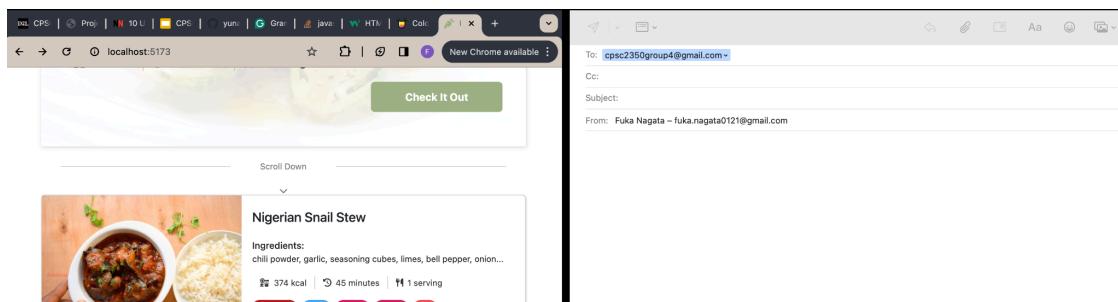


10. Help and Documentation

- Navigate users to the FlexyCook GitHub repository page by clicking "Help" to see the documentation



- Users can contact/ask for help to Group 4 by clicking Contact; It navigates to email





Tests

We received approval from Professor Rajabi on March 20 that the number of test requirements was reduced to 15 instead of 24 due to one team member having quit.

Overview

We used the [Vitest](#) and [React testing library](#) to test.

[Vitest](#):

Our application goes well with Vitest because Vite has been used to create an environment for this application. Vitest is almost the same as Jest, but Jest for Vite.

[React testing library](#):

This application is coded with React, and testing uses Vitest, so using the React testing library is almost necessary.

The test result

```
v tests/IntegrationTest/Spoonacular/SearchByNutrients/AdvancedSearch.test.jsx (11) 887ms
v tests/IntegrationTest/Spoonacular/Recommendation/Home.test.jsx (4)
v tests/IntegrationTest/Spoonacular/SearchByIngridients/Search.test.jsx (11) 515ms
v tests/IntegrationTest/Todoist/CookingStepsList/ToolBar.test.jsx (2)
v tests/UnitTest/Spoonacular/Recommendation/LargeSquareCard.test.jsx (4)
v tests/IntegrationTest/Todoist/ShoppingList/ToolBar.test.jsx (2)
v tests/UnitTest/Spoonacular/SearchByIngridients/Tags.test.jsx (4)
v tests/IntegrationTest/Spoonacular/Recommendation/Recipe.test.jsx (6)
v tests/IntegrationTest/Todoist/GeneralTodoList/Navbar.test.jsx (2)
v tests/UnitTest/Todoist/CookingStepsList/TodoList.test.jsx (2)
v tests/UnitTest/Spoonacular/SearchByIngridients/SearchBar.test.jsx (6)
v tests/IntegrationTest/Spoonacular/SearchByNutrients/RecipeBanner.test.jsx (6)
v tests/IntegrationTest/Spoonacular/SearchByIngridients/StickyButton.test.jsx (4)
v tests/UnitTest/Todoist/ShoppingList/TodoList.test.jsx (2)
v tests/UnitTest/Spoonacular/SearchByNutrients/ModifiedButton.test.jsx (4)
v tests/UnitTest/Todoist/GeneralTodoList/TodoList.test.jsx (2)
v tests/UnitTest/Spoonacular/Recommendation/CookingInfo.test.jsx (4)
v tests/UnitTest/Spoonacular/SearchByNutrients/DeleteableTag.test.jsx (2)

Test Files 18 passed (18)
Tests 78 passed (78)
Start at 15:53:04
Duration 4.57s (transform 893ms, setup 2.47s, collect 4.35s, tests 3.53s, environment 3.83s, prepare 918ms)
```



Tests (cont.)

Spoonacular

Unit Tests

Unit testing was primarily for smaller, less logical components to test whether the component was doing its job. For example, check if they display the necessary information on the screen, have the appropriate roles, such as button or input, can be clicked or typed information, and if given props, process it correctly.

Search by ingredients

- SearchBar: A search bar where users enter the ingredients to search
- Tags: Clickable tags that show dish type/cuisine / Diet... of recipes

Search by nutrients

- DeletableTag: A deletable tag that shows the nutrient the users want to search by
- ModifiedButton: A button that modified the Bootstrap button to make it more flexible

Recipe recommendation

- CookingInfo: A bar that shows cooking info(cooking time, serving portion, calories)
- LargeSquareCard: A card that displays recipe info(cooking info, image, tags, title..)

Integration Tests

Integration testing was mainly performed on components that were logic-heavy, complex in structure, and made up of a combination of components while using API response mocks. For example, if the pages render correctly, they can accommodate different user-user inputs, clicking a button navigates to another page, and the API calls properly and returns the correct response, and the data flows and is processed work correctly.



Tests (cont.)

Search by ingredients

- StickyButton: The position-fixed button that navigates to the top
- Search: The search page that lets the users search the recipes and display the result

Search by nutrients

- AdvancedSearch: The toggle menu that users choose range, nutrients, and amount of nutrients to search recipes by nutrients
- RecipeBanner: A banner on the recipe page that shows detailed info about the recipe

Recipe recommendation

- Home: The home page that shows recommendation recipes
- Recipe: The recipe page shows detailed steps and information about the recipe

Todoist

Unit Tests

Shopping list

- TodoList:
 - It should render the “TodoList” component from the FlexyCook shopping list
 - The “Delete” button should trigger the deletion of the task.

Cooking steps

- TodoList:
 - It should render the “TodoList” component from a cooking steps list
 - The “Add Subtask” button should trigger the addition of a new subtask.



Tests (cont.)

General to-do list

- TodoList:
 - It should render the “TodoList” component from the FlexyCook “memos” list
 - The “Add Task” button should trigger the addition of a new task.

Integration Tests

Shopping list

- ToolBar:
 - It should render the “ToolBar” component.
 - It should render the “Add to shopping list” button on the “ToolBar” component.
 - “Add to shopping list” button should trigger the addition of the ingredients to the shopping list.

Cooking steps

- ToolBart:
 - It should render the “ToolBar” component.
 - It should render the “Convert to cooking steps” button on the “ToolBar” component.
 - “Convert to cooking steps” button should create a new list and name it as the recipe name, and add all the cooking steps to the created list.

General to-do list

- NavBar:
 - It should render the “Navbar” component
 - “Add New Memo” button of “TODO” button dropdown of “Navbar” should trigger rendering of the TodoListModal

CI / CD Infrastructure

 Git branching strategy: *development → staging → production*

CI/CD system from Feature A into deployment.

A new feature, 'Feature A' is fully implemented in the 'feature-a' branch.

1. A pull request is submitted for the development branch.
2. CI: Automatically builds projects, and unit & integration tests are run.
3. Pull request is approved after a code revision from a teammate and the 'feature-a' gets merged into the development branch.
4. Pushing into the development branch triggers automatic build and deployment on the development website (website for internal use).
5. Developers agree that a version is ready to be deployed and push it to the staging branch. (If this project gets bigger, A separate staging website can be created and shared with all non-programming stakeholders of the project before moving these versions into the production, and further tests can be done)
6. Developers push this version into the production branch to make it public.
7. CD: This triggers the automatic build and deployment of our website to our production website.

CI / CD Infrastructure (cont.)

GitHub workflows:

(.github/workflows/ci.yml)

(.github/workflows/github-pages.yml)

```
on:
  push:
    branches: [ "development", "staging", "production" ]
  pull_request:
    branches: [ "development", "staging", "production" ]
  - run: npm ci
  - run: npm run build --if-present
  - run: npm test
  - name: Setup Pages
    uses: actions/configure-pages@v4
  - name: Upload artifact
    uses: actions/upload-pages-artifact@v3
  with:
    # Upload dist folder
    path: './dist'
  - name: Deploy to GitHub Pages
    id: deployment
    uses: actions/deploy-pages@v4
```

Github branch protection:



- Our development website for internal use: <https://yunahu.github.io/FlexyCook/>
 - Automatic deployment on every push to the development branch (.github/workflows/github-pages.yml)
 - For now, it requires flexy-cook-api to be running on the developer's local computer.
- Our production website for our users:
<https://production.d3ajroqntq3q3i.amplifyapp.com/>
 - API - AWS Elastic Beanstalk
 - AWS automatic-build & deploys from our production branch

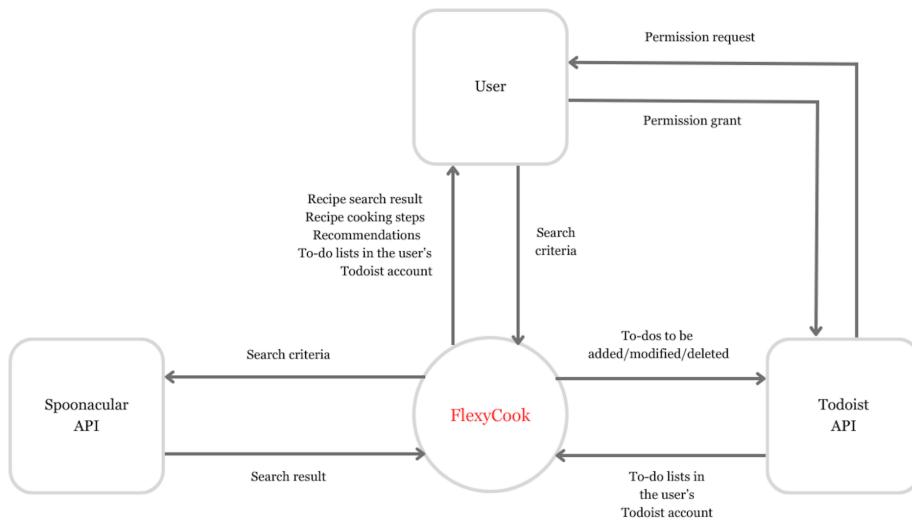
The screenshot shows the 'Repository settings' section in Amplify. It includes:

- Source repository**: <https://github.com/yunahu/FlexyCook/tree/production>
- Production branch**: production
- Branches** table:

Branch name	URL prefix	Auto-build Info
production	production	Enabled



High-level Data Flow Diagram - Level 0



FlexyCook is a recipe provider system with special to-do list functionalities. When the user searches for a recipe, they input the search criteria, and we return various recipes that are the most relevant to the received input.

When a user generates a shopping list, a cooking steps list, or a general to-do list, FlexyCook redirects the user to an authorization page where they can grant permission for our system to access their Todoist account information and add the generated to-do lists.



Project Challenges

- During the planning phase, it was hard for the team to precisely determine the scope of the project. It took three weekly meetings to reach a decision that every team member agreed on.
- Communication is much more time-consuming than our team initially thought.
- Learning a new framework (React framework) appears to be the major factor limiting our team's productivity. Most of the team members did not have previous knowledge of developing with React.
- The total workload of this project was not easily predictable, so each team member ended up spending many more working hours on it, which made this project more challenging for some team members.



Lessons Learnt / Project Takeaway

Teamwork

- For most of us, this was our first team project, and it taught us how collaborating on a project is different from individual work.
- We could learn from team members to enhance each other and improve the product from different perspectives.

Communication

- We communicated online almost every day, improving and stimulating each other, asking questions about what we didn't understand, and correcting each other when we needed to fix things. Without this, the completion of the project would have been significantly different.
- We deeply felt that communication is key for team projects.
 - After the change of progress report from only once a week to three times a week, progress was faster, and work conflicts and information discrepancies were greatly decreased.

Coding

- Most team members did not have much experience with React, Vitest, or other tech stacks used in this project, but we could learn them through the project while actually using those tech stacks.
- We understand that coding is just one part of the development process. There is so much more than coding to making a product.

Work Division

Front-End Team

Monica Gao & Yuna Hu & Fuka Nagata & Simran Kaur (left since March 19th)

Back-End Team

Yuna Hu & Fuka Nagata

Monica

feature ideas, wireframe design, prototype building, website structure constructing, applying styling on website components, creating demo video, finalizing project presentation PPT, finalizing project report, communicating with the professor;

Simran Kaur

prototype building, website structure construction;

Yuna Hu

feature ideas, choosing API, providing user security-enhancing API, website structure constructing, API implementation for Todoist API, developing CI/CD infrastructure, testing for Todoist API-related features, website deployment, creating presentation PPT, finalizing project report;

Fuka Nagata

feature ideas, choosing API, prototype building, API implementation for Spoonacular API, testing for Spoonacular API-related features, applying styling on website components, finalizing project presentation PPT, completing project report;

Sources

APIs

<https://spoonacular.com/food-api>

<https://developer.todoist.com/guides/#developing-with-todoist>

Technology stacks

<https://react.dev/>

<https://react-bootstrap.netlify.app/>

<https://aws.amazon.com/>

<https://vitest.dev/>

<https://testing-library.com/docs/react-testing-library/intro/>

Nielson's 10 usability heuristics

<https://www.nngroup.com/articles/ten-usability-heuristics/>

Cover page image

<https://unsplash.com/photos/wooden-ladle-and-chopping-board-with-ginger-during-daytime-vuDXj60mIOA>