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/flexy-cook-api
https://github.com/yunahu/FlexyCook

Table of Contents

📊 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.

♻️ SDLC

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

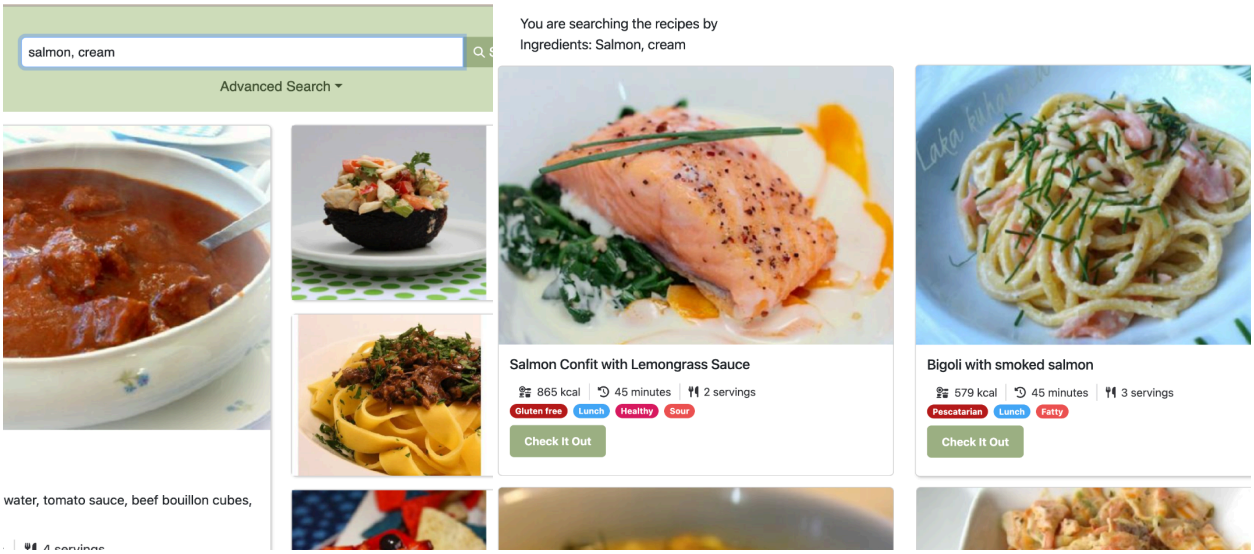
🗣 As a working mom who does not have much time to find recipes, I want to be able to save all the information I need for a recipe with just one click of a button so that I can access it conveniently whenever I need it.

👩‍🍳 Features

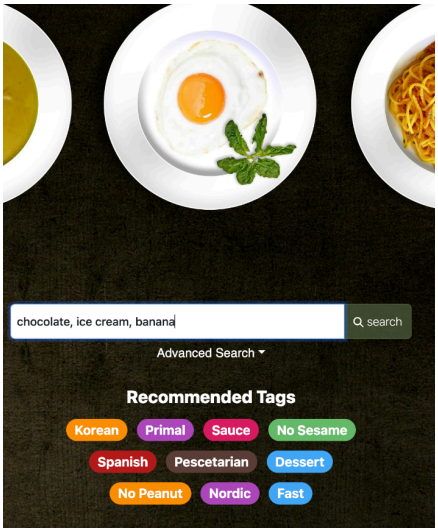
Recipe API (Spoonacular)

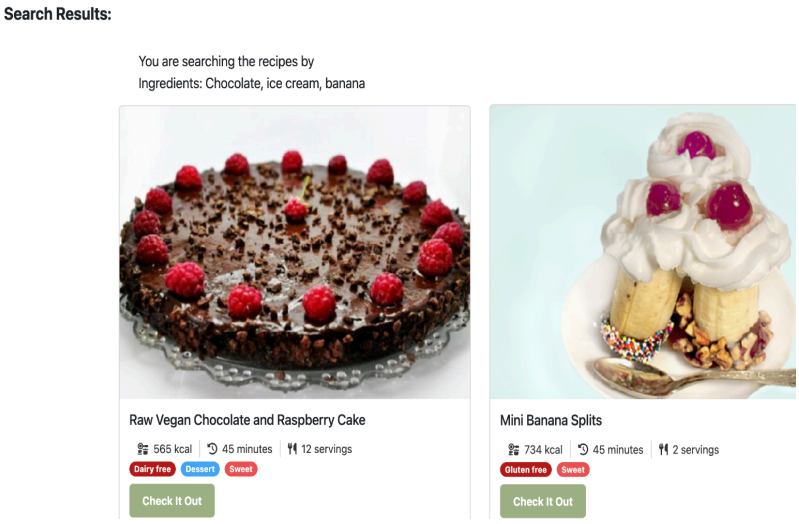
Search by Ingredients

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



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

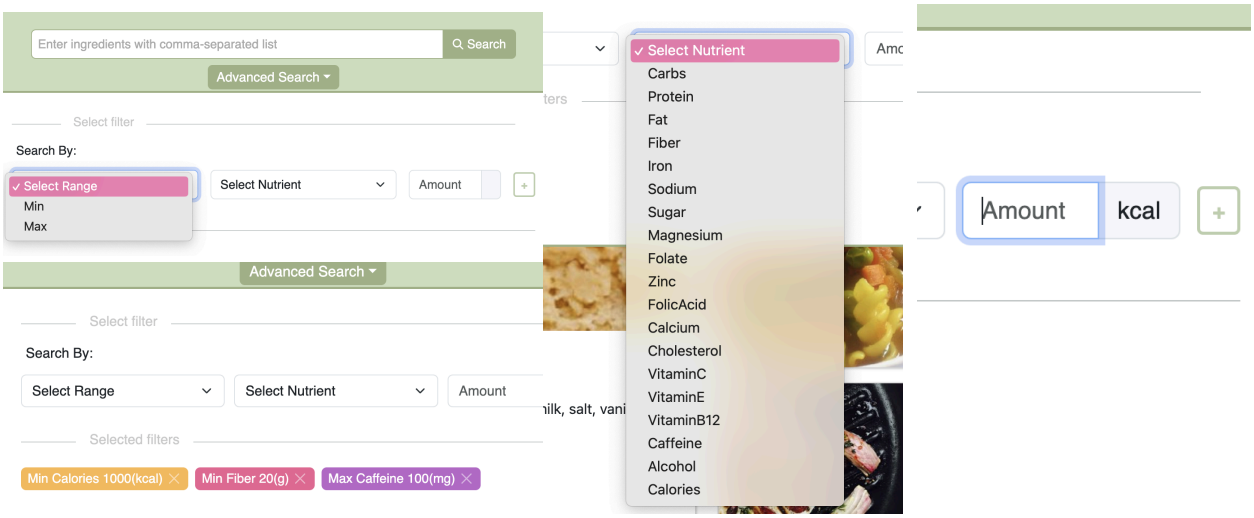




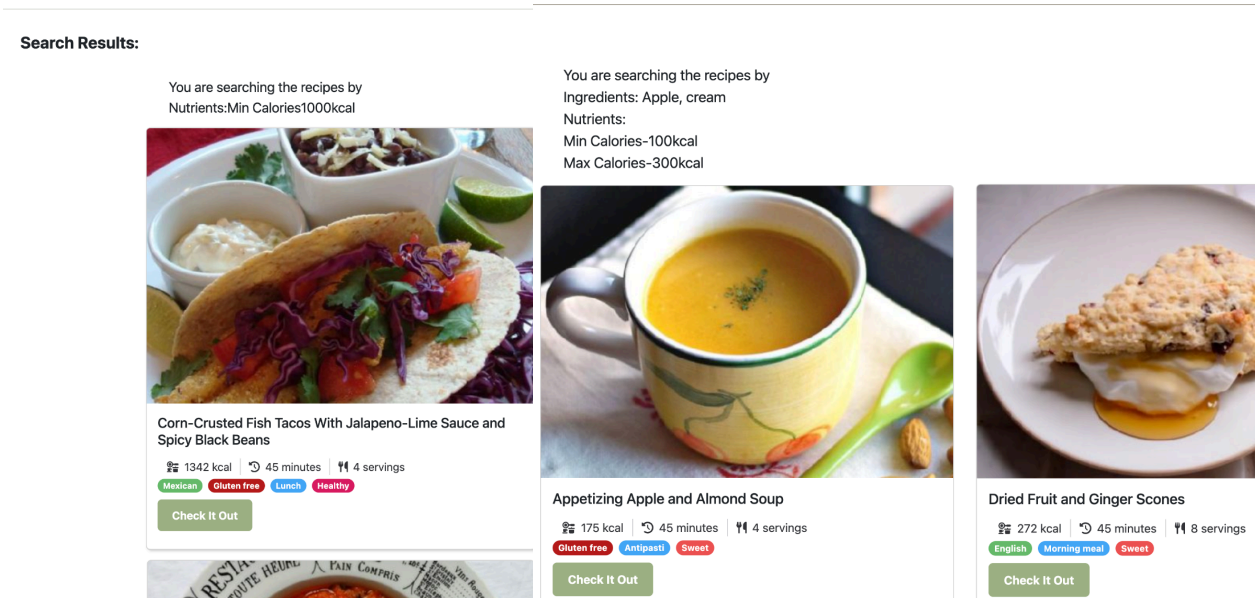
👨‍🍳 Features (cont.)

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.



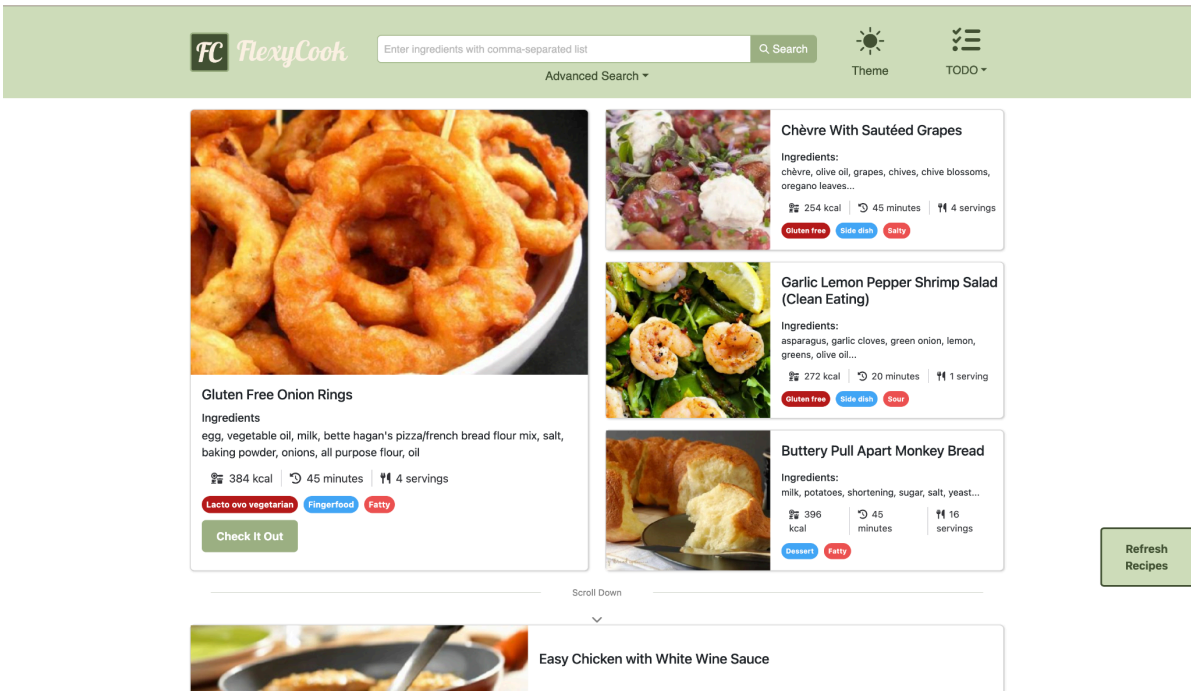
- Users can use the advanced search feature on the search page and navigation bar.
- Users can search by both nutrients and ingredients.



👨‍🍳 Features (cont.)

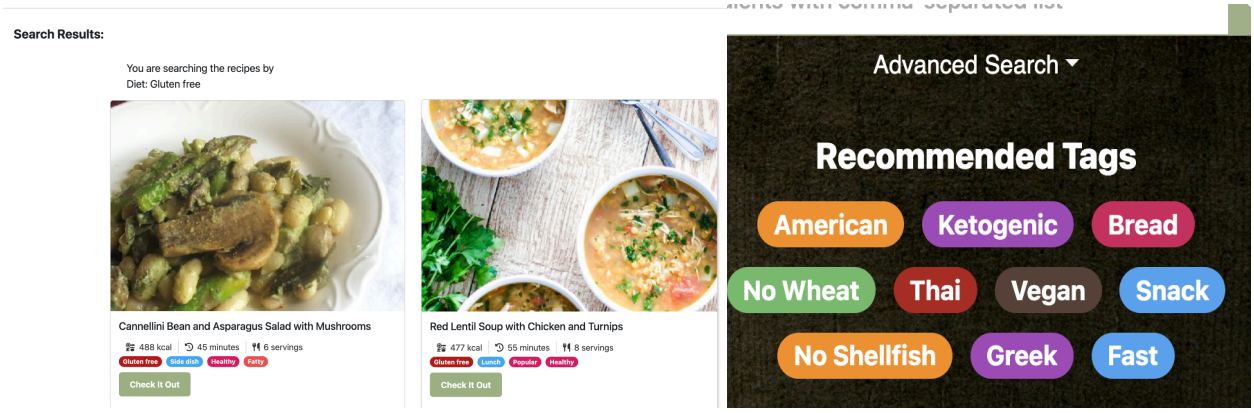
Recipe Recommendation

- 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.



Search by Tags

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

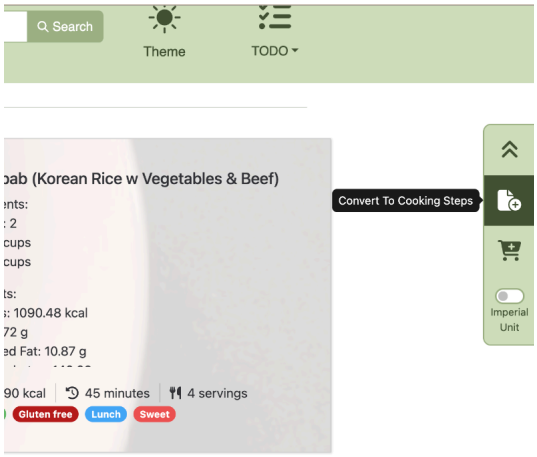


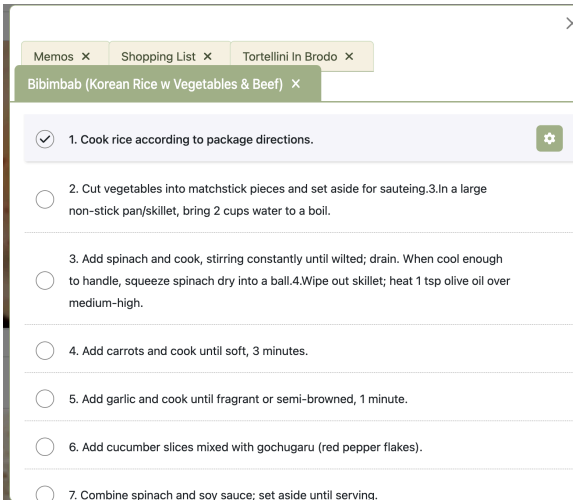
👩‍🍳 Features (cont.)

To-do list API (Todoist)

Cooking steps

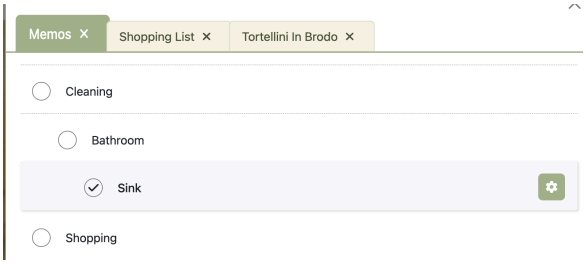
- The original recipe can be processed into simple individual cooking steps with a single click of a button.
- The users can easily understand the cooking steps without having to read a long recipe.
- When the user has completed a step, they can check that step off to keep track.
- The user can also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





General todo list

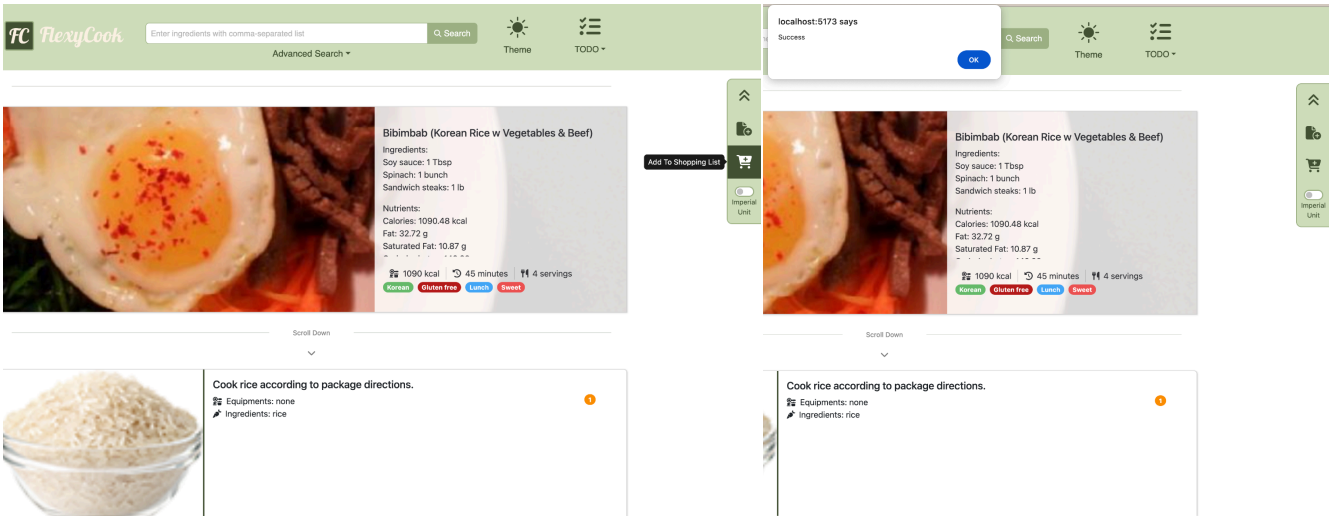
- This is used for task management as a general to-do list.
- The user can create sub / sub-sub /sub….sub-tasks.
- If the user wants to delete the task, they can delete it by “Delete Task”, or the task will be removed by checking it.

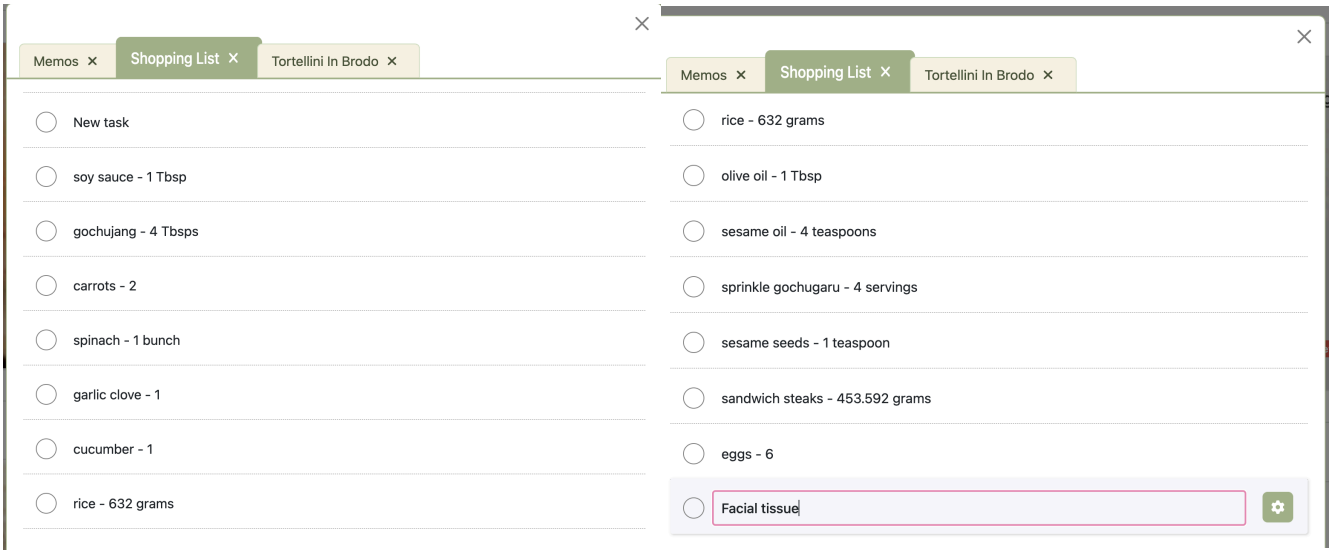


👩‍🍳 Features (cont.)

Shopping list

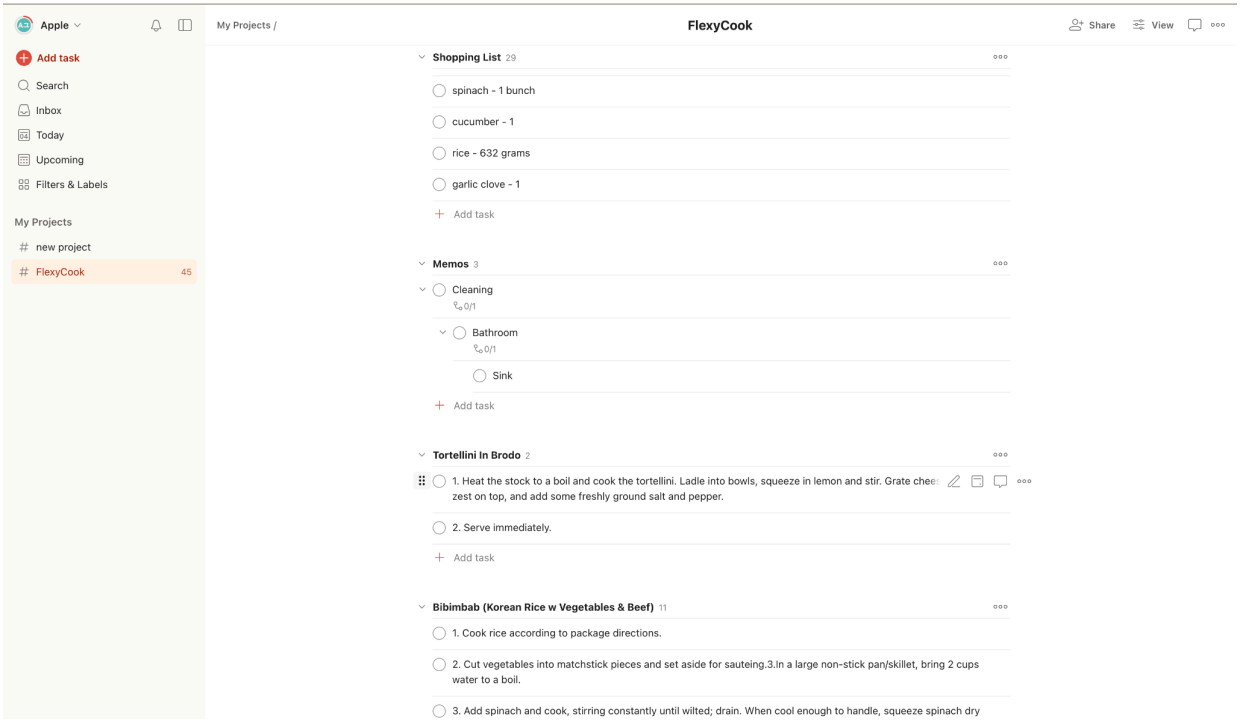
- The user can easily generate a shopping list for a recipe with the click of a button.
- If there are any items that they need to buy, they can add them manually to their shopping list on the spot and refer to it while shopping.

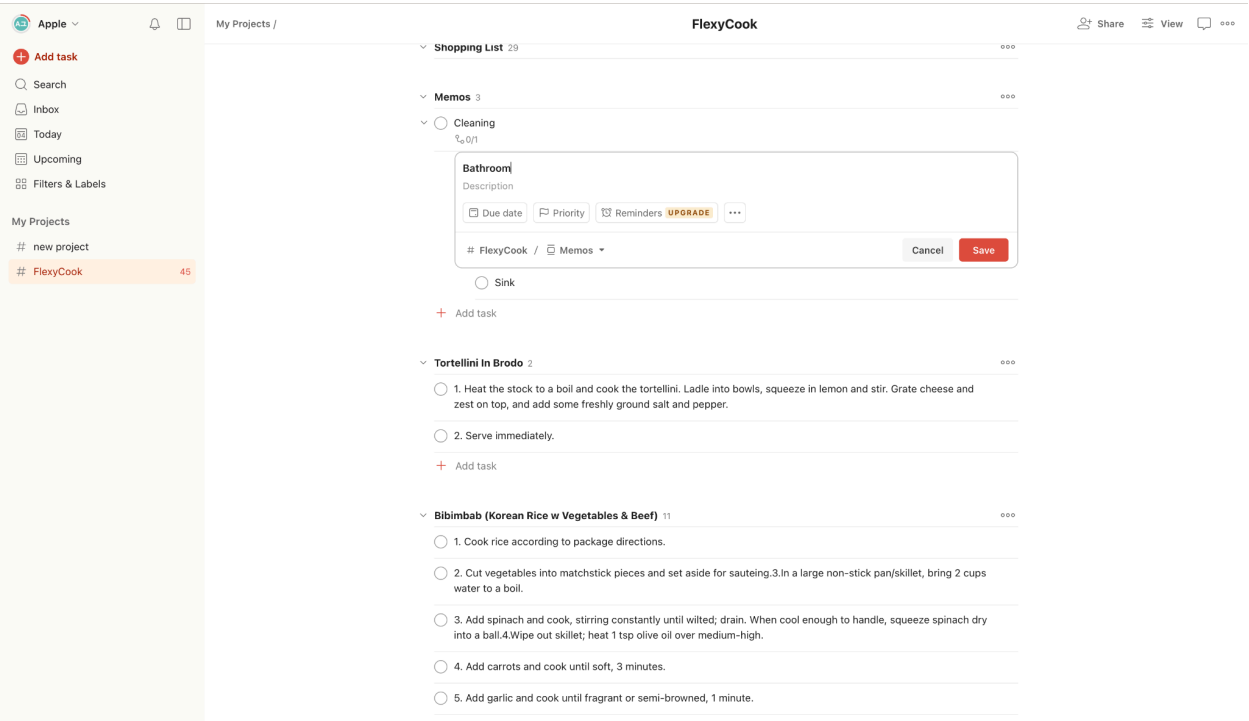




👩‍🍳 Features (cont.)

All these lists can be synced to the Todoist app.

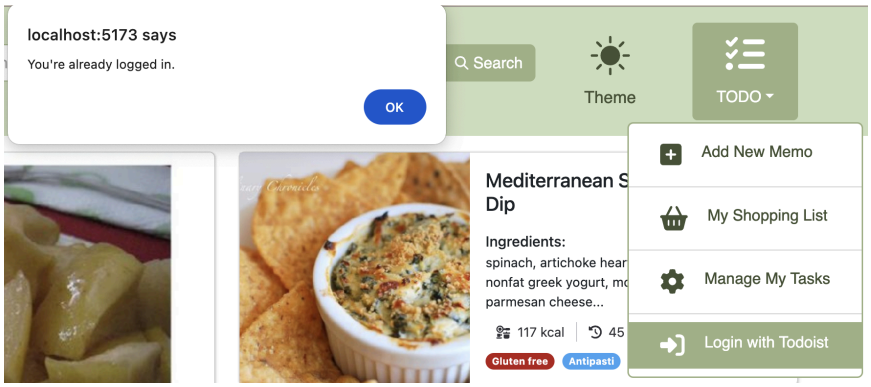




🚦 10 Usability Heuristics

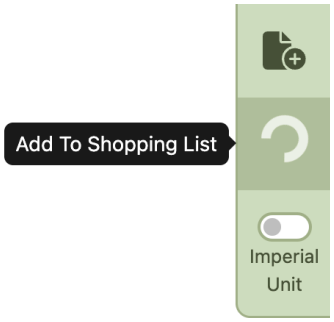
1. Visibility of System Status

- Show the user has logged in to todoist

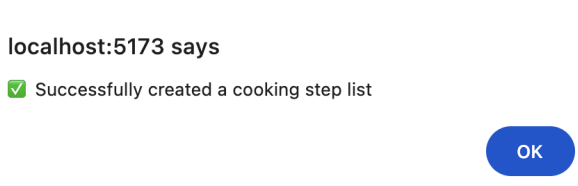


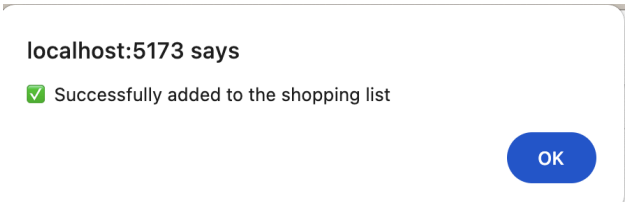
- Show the user is clicking the button

- Show recipes are loading / the list is creating



- Show lists were successfully created

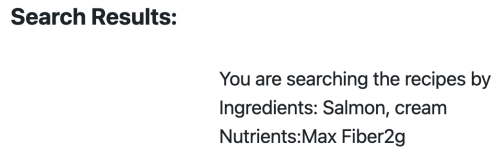




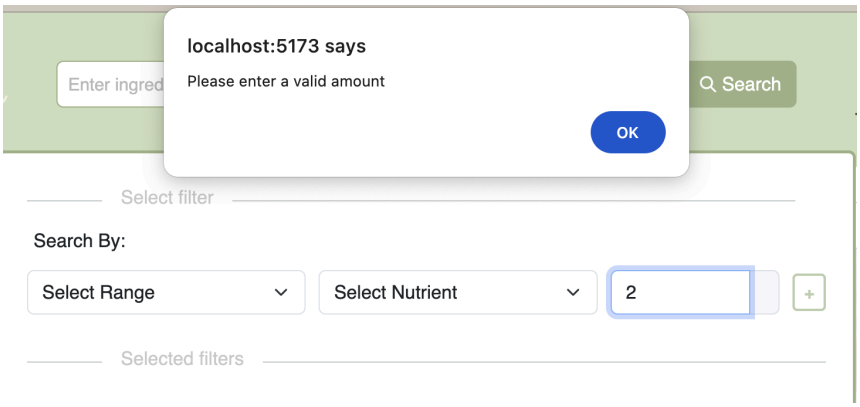
🚦 10 Usability Heuristics (cont.)

2. Match Between the System and the Real World

- Shows what the user is searching recipes by

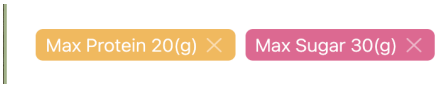


- Let the users know when they've entered a invalid amount, and not saying "you've entered illegal input"

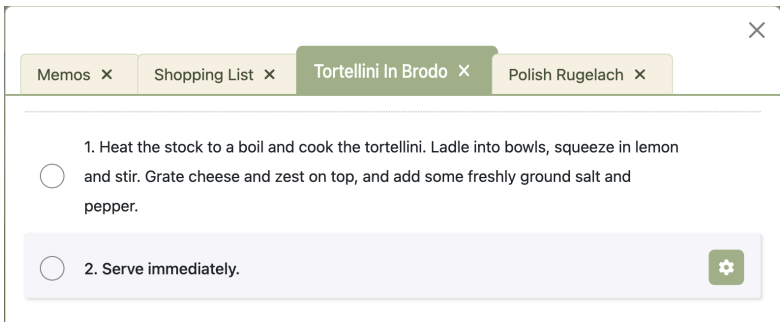


3. User Control and Freedom

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



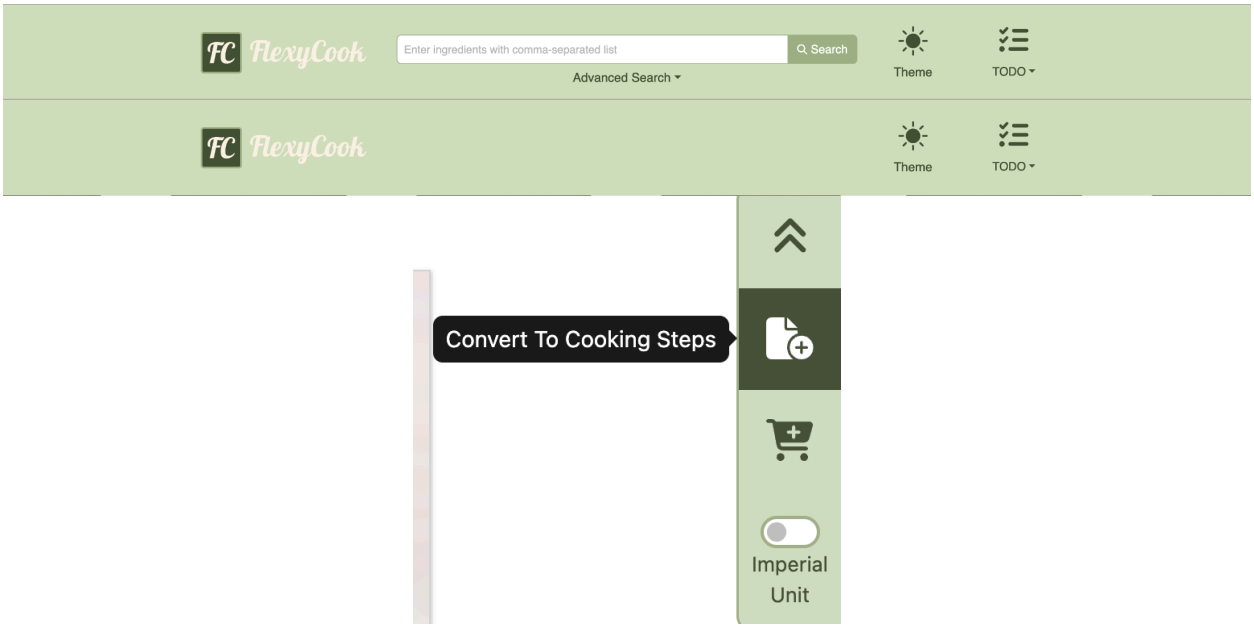
- Users can delete tasks through the settings, which are placed right side of the task



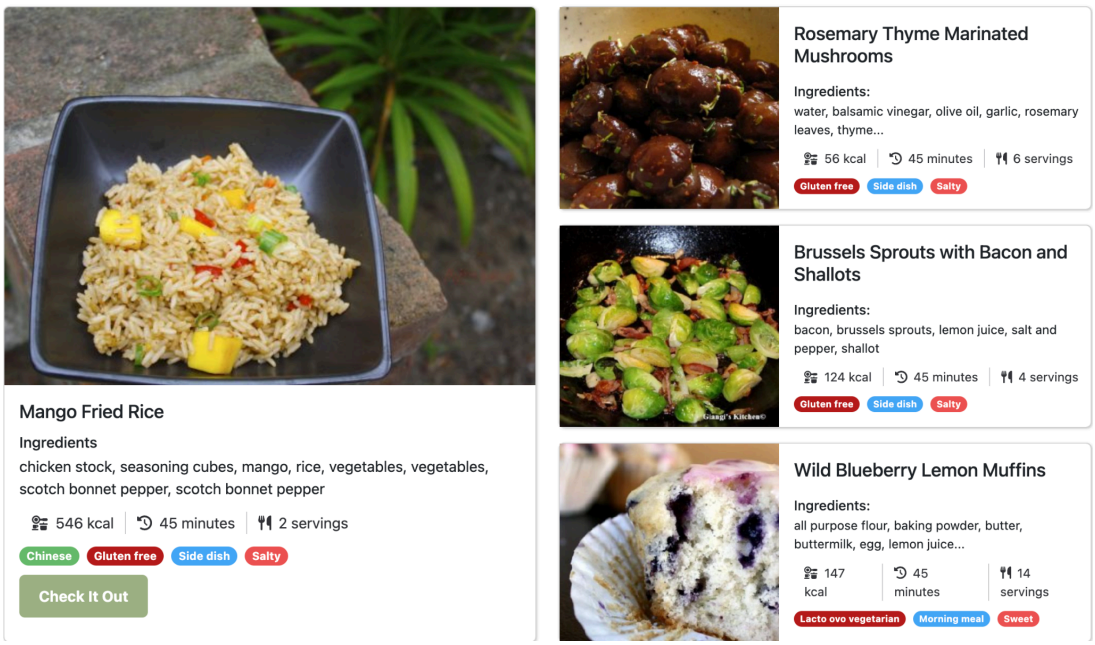
🚦 10 Usability Heuristics (cont.)

4. Consistency and Standards

- Consist design and alignment



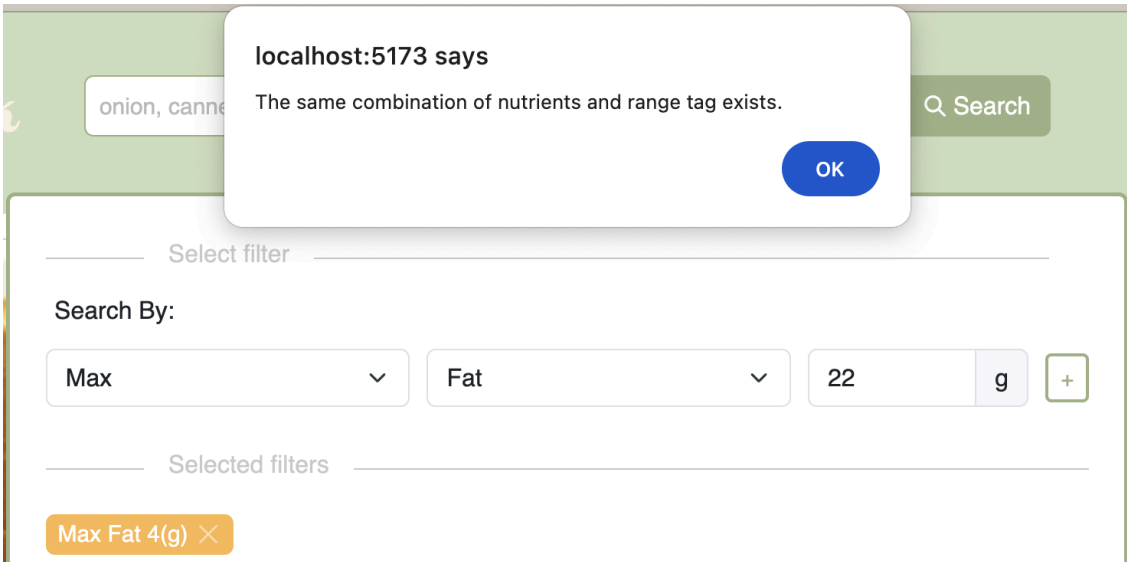
- Cards have the same structure



🚦 10 Usability Heuristics (cont.)

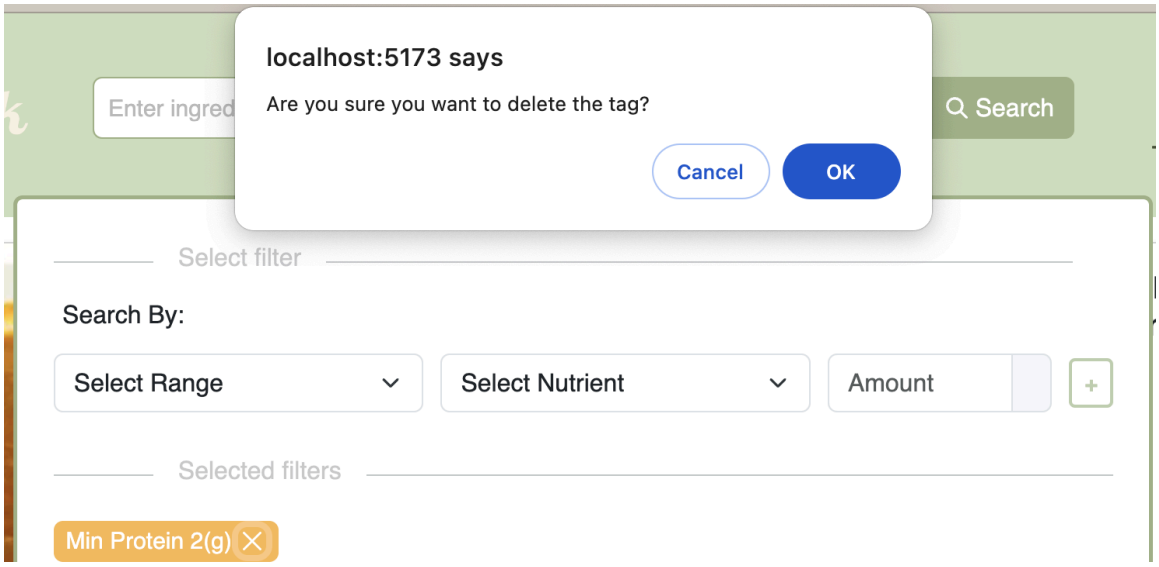
5. Error Prevention

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



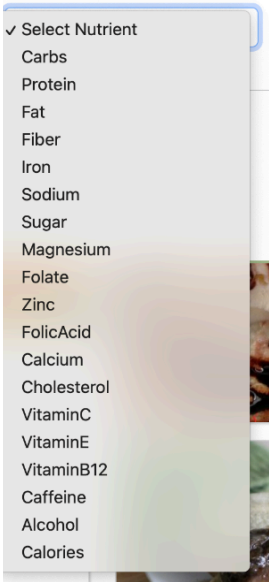
- Disable the button when there are not enough parameters

- Show confirmation when the users try to delete the tag

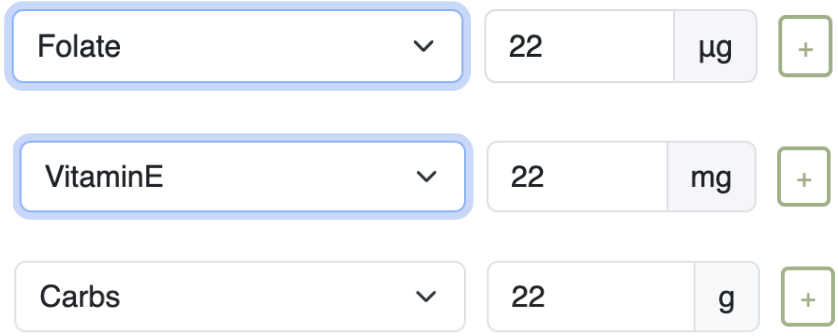


🚦 10 Usability Heuristics (cont.)

6. Recognition Rather than Recall

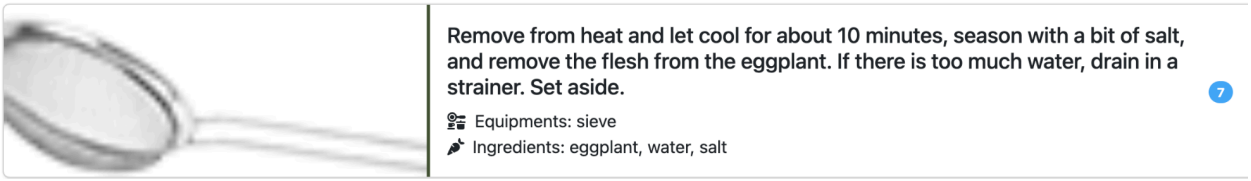


- Depending on the nutrient, change the scale. Recall nutrients



- Enable to see ingredients in both imperial and metric scale

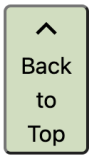
- Show equipments with the image to recall how it looks



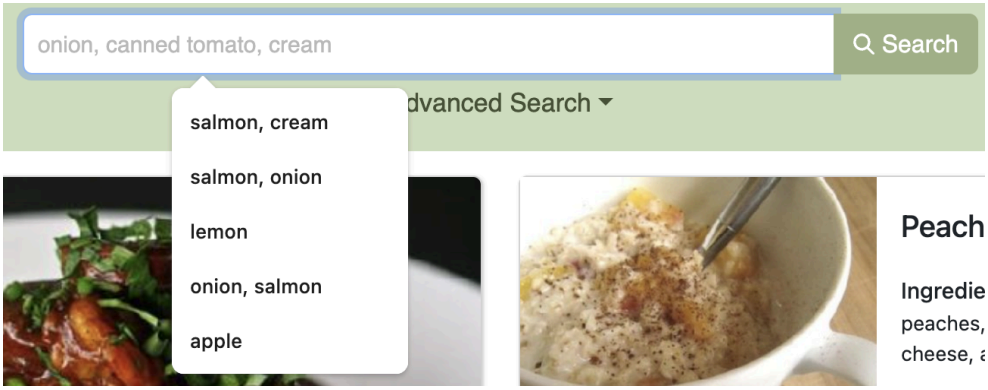
🚦 10 Usability Heuristics (cont.)

7. Flexibility and Efficiency of Use

- Enable back to top



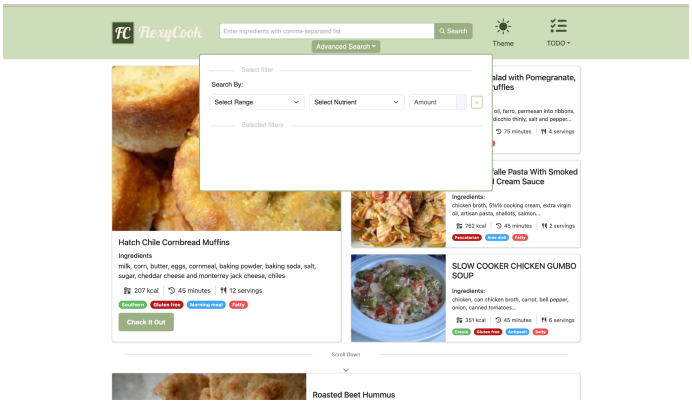
- Search History



- 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

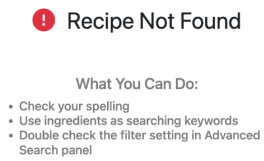
- Used few fonts & colours, colorblind-friendly colours

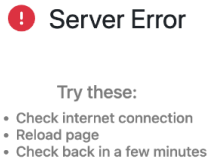


🚦 10 Usability Heuristics (cont.)

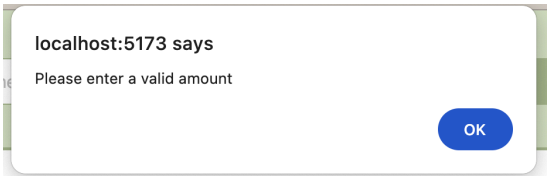
9. Help Users Recognize, Diagnose, and Recover from Errors

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

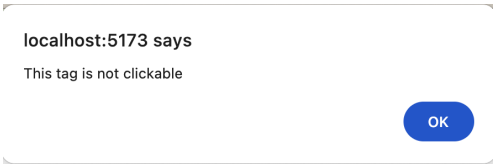




- If the user entered an invalid input(characters, negative amount), let the user know to enter a valid number



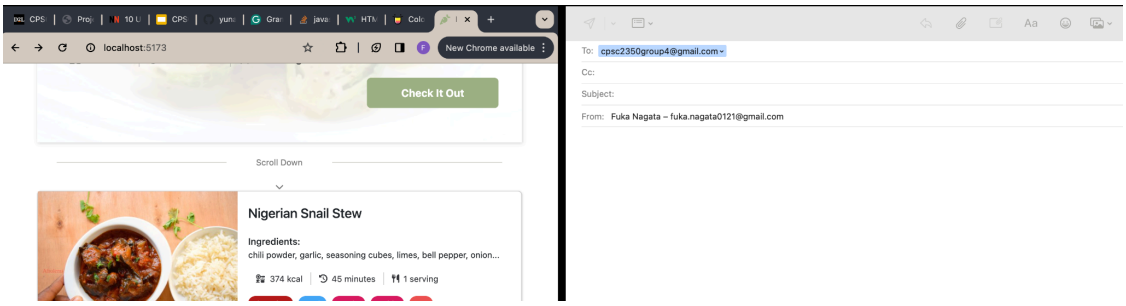
- If the user clicks the unclickable tag, show the tag is unclickable



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

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

```yaml
on:
  push:
    branches: [ "development", "staging", "production" ]
  pull_request:
    branches: [ "development", "staging", "production" ]
```

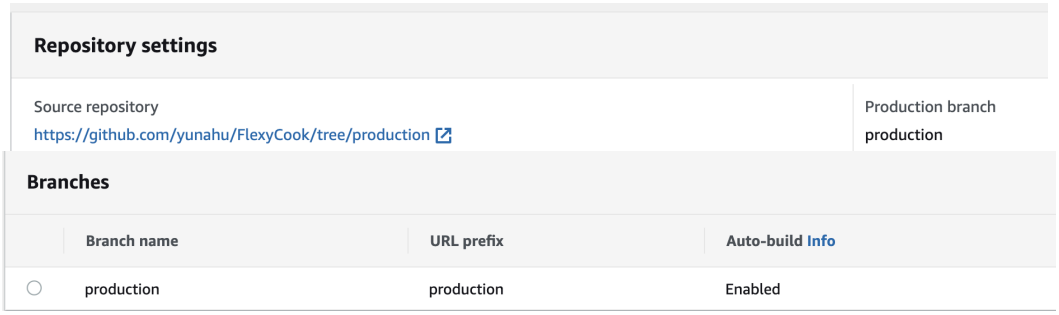
```yaml
    - run: npm ci
    - run: npm run build --if-present
    - run: npm test
```

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

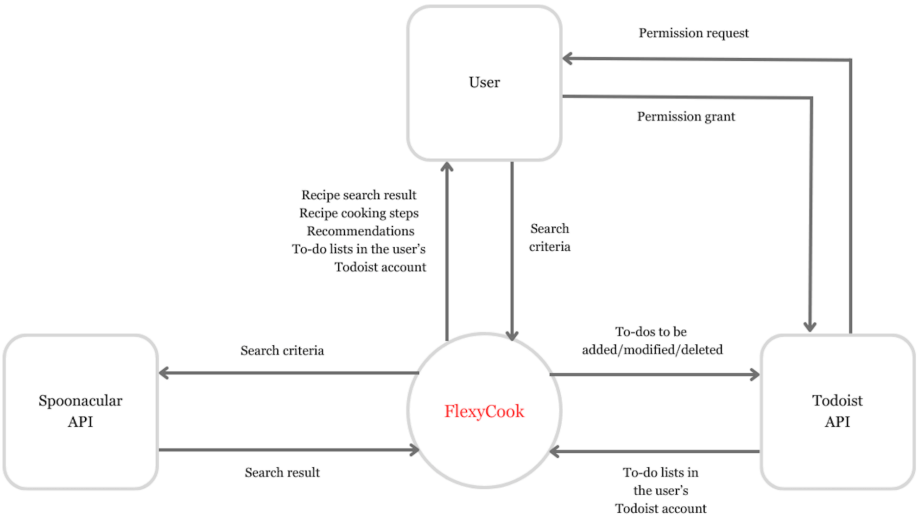
```yaml
    - name: Setup Pages
      uses: actions/configure-pages@v4
    - name: Upload artifact
      uses: actions/upload-pages-artifact@v3
      with:
        # Upload dist folder
        path: './dist'
    - name: Deploy to GitHub Pages
      id: deployment
      uses: actions/deploy-pages@v4
```

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



📈 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-vuDXJ60mJOA