Cloud 14

Milestone 2

Sahil Manak (301465247), Ramtin Rezaei (301582747), Srinjay Mitra (301582986), Sukhman Virk (301468282)

December 5th, 2024

Table of Contents:

Providing a Solution

User's needs:

The primary objective of our project was to develop a digital cookbook designed to assist a diverse range of users with their culinary endeavors. Our target user base encompassed individuals seeking guidance and inspiration in the kitchen.

To identify the specific needs of our target audience, we conducted a thorough analysis. This involved drawing upon our personal experiences, as well as consulting with family and friends. Additionally, we studied existing digital cookbooks and recipe websites to gain insights into industry best practices and popular features. Based on this comprehensive research, we identified the following core requirements:

- Recipe Search:
    - By name
    - By difficulty and cooking time
    - By ingredient availability
- Meal Planning:
    - Weekly meal planning functionality
- User Preferences:
    - Saving favorite recipes
- Nutrition calculator:
    - Getting nutritional information by entering the ingredients
- User Interface:
    - Intuitive and user-friendly design accessible to all users

While these features may not cater to every individual's specific needs, our goal is to provide a comprehensive and versatile digital cookbook that can accommodate a wide range of user preferences and cooking styles.

How we solved them:

To address these identified needs, we developed six core features for our website, each designed to fulfill a specific user requirement. In the development process, we leveraged the capabilities of the Edamam and Spoonacular APIs, which are widely recognized as industry-leading tools for recipe search and culinary guidance.

The following features and their corresponding user needs are as follows:

- **Recipe Search:** Enables users to search for recipes based on difficulty and cooking time.

- **Recommendations:** Provides personalized recipe suggestions based on available ingredients.
- **Chatbot:** Offers a user-friendly interface, particularly for users unfamiliar with the website.
- **Meal Plan:** Facilitates weekly meal planning.
- **Nutrition Calculator:** Calculates nutritional information for recipes based on ingredient input.
- **Save Recipe:** Allows users to save their favorite recipes for future reference.

A more detailed exploration of these features and their implementation using the APIs will be presented in the "API & Features" section of this report. To evaluate the effectiveness of these features in meeting user needs, we conducted a peer testing session during the final class lecture.

Feedback:

The feedback collected from the testing session revealed several areas for improvement:

- Navigation: Some users found the website's navigation to be somewhat cumbersome.
- Title Clarity: The titles of certain sections were deemed overly wordy.
- Feature Integration: A suggestion was made to integrate meal planning with one of the search features for a more streamlined user experience.
- User Interface: The overall aesthetic and design of the user interface received positive feedback.

It is important to note that this feedback was collected while our website was still under development, approximately 80% complete. Many of the identified issues, particularly those related to navigation and title clarity, were addressed during the remaining development phase. However, suggestions that would have required significant changes, such as feature integration, were not feasible within our given timeframe.

A key takeaway from the testing session is that users were generally satisfied with the functionality of the website's core features, indicating that their primary needs were met. This positive feedback validated our design choices and provided valuable insights into user behavior. The majority of the remaining issues were related to the website's overall organization and presentation, which were subsequently resolved**.**

Here is the file in which we stored the feedbacks in: The results file

Analysis of Our Software Development Life Cycle Model (Agile Scrum)

Team fourteen employed an agile Scrum software development life cycle (SDLC). Various reasons motivated this choice over other methodologies. One significant advantage of Scrum was its clear role assignments. This approach minimized confusion regarding individual responsibilities, with roles such as product owner, Scrum master, and development team members. The team leader assumed the role of Scrum master, overseeing team management and actions. One team member was designated as the product owner, responsible for providing details on the desired features and functionality of the website. The remaining team members contributed as developers, actively working on the website's construction.

Scrum also introduced an accountability factor. By assigning specific tasks to each team member, individuals were held responsible for completing their designated work. The Scrum framework incorporated daily stand-up meetings, fostering effective communication within the team. Recognizing the crucial role of communication in successful teamwork, as evidenced by the AI failures analyzed, we opted for a methodology that prioritized open dialogue. These meetings facilitated discussions on completed tasks, ongoing work, and future plans, helping to resolve potential conflicts.

The Scrum methodology employs iterative development cycles known as sprints. Before each sprint, the product owner created a backlog, outlining their preferred features and functionalities. The team utilized this backlog to initiate the sprint, breaking down tasks into smaller, more manageable components. This approach aligned well with our goals, as it allowed us to develop and test smaller parts of the website incrementally.

Furthermore, the flexibility of Scrum was a key factor in its selection. If we encountered any issues or errors during a sprint, we could address them and adapt our plans for the next iteration. This flexibility contrasted with other methodologies that required a more rigid, linear approach, making Scrum a more suitable choice for our project.

How consistent we were following this model:

We endeavored to adhere to the agile Scrum methodology, believing it would optimize our project's success. However, several challenges arose that deviated from our intended course. Initially, the designated Scrum master's absence due to a school-related trip necessitated a temporary relinquishment of the role. Consequently, the daily stand-up meetings, a cornerstone of Scrum, were not consistently held. While we did convene for specific assignments to coordinate tasks and deadlines, the overall adherence to the Scrum framework was compromised. Despite these setbacks, we strived to synchronize our schedules and project phases with the agile Scrum SDLC to the best of our ability.

Modification we made along the way:

After recognizing that certain aspects of the agile Scrum SDLC did not align with our group's schedule, we proactively sought alternative approaches to maintain project momentum. Given the challenges associated with the Scrum master role, our team members collectively assumed responsibility for weekly planning and task allocation. Instead of relying on a single individual, we fostered a collaborative approach where everyone contributed to the decision-making process.

We also opted to eliminate daily stand-up meetings, as our established working relationship and effective communication through a dedicated Discord channel provided sufficient visibility into each member's progress and plans. This streamlined approach saved time and further strengthened our team cohesion.

Throughout the project, our team demonstrated resilience in adapting to challenges and maintaining the core principles of the agile Scrum SDLC. By embracing flexibility and a collaborative mindset, we were able to effectively navigate obstacles and achieve our project objectives.

APIs Features

To realize the full potential of our digital cookbook, we integrated two powerful APIs: Edamam and Spoonacular. These APIs provide a robust foundation for our core features, enhancing user experience and functionality.

Edamam: A Versatile Culinary Tool

Edamam, a popular API for food-related applications, offers a range of features that align seamlessly with our project goals. Its free plan, which provides 400 monthly requests, is sufficient to support our needs.

Key Features and User Benefits:

- **Recipe Search:** Users can discover recipes tailored to their specific preferences, filtering by ingredients, cooking time, and difficulty level. This feature saves time and effort, allowing users to quickly find suitable recipes.
    - Changes:We removed the dietary needs field and replaced it with ingredients as we thought it would suit the feature better.
- **Nutrition Calculator:** By inputting ingredients, users can obtain detailed nutritional information, including calorie counts, and macronutrient breakdowns. This empowers users to make informed dietary choices.

    - Changes: We removed the cost from our response as we could not find a good way to add it to our feature

Spoonacular: A Comprehensive Culinary Solution

Spoonacular, another widely-used API, offers a diverse set of features that complement Edamam's capabilities. Its free plan, with 3000 monthly requests, provides ample capacity for our project.

Key Features and User Benefits:

- **Meal Planning:** Users can generate personalized meal plans by searching their own recipes and calorie goals. Users can plan each day of the week
    - Changes: We removed making a shopping list for users.
- **Recipe Recommendations:** By analyzing input ingredients, Spoonacular suggests relevant recipes, helping users discover new culinary delights.
    - Changes: We removed the recommended foods based on user history, because we did want to make a backend for our website. This was also used to be a feature of Edamam API, but we changed it because the Spoonacular feature was easier to use.
- **Recipe Saving:** Users can save their favorite recipes for easy access and future reference.
    - Changes: We made no changes to this feature.
- **Chatbot:** A chatbot interface enables users to interact with the website in a conversational manner, receiving real-time assistance and culinary tips.
    - Changes: This feature is completely new, and it replaced the difficulty estimator feature.

By strategically utilizing these APIs, we have been able to develop a feature-rich digital cookbook that offers a seamless and enjoyable user experience.

Description & Overview of The CI/CD pipeline & How It Was Used

The CI/CD pipeline for our project is defined in the workflow/ci_cd_pipeline.yml file.

Pipeline Configuration: Our pipeline is configured to trigger on two events, push and pull requests.

Push Events: Whenever code is pushed to the main branch, the pipeline is activated.

Pull Request Events: The pipeline also runs when a pull request is made to the main branch.

This setup ensures that any changes to the main codebase are automatically tested and verified before being merged or deployed.

Jobs and Steps

The pipeline consists of a single job named build-and-test, which runs on the latest Ubuntu environment. However this job includes several key steps:

The first step uses the actions/checkout@v2 action to automatically clone the most up-to-date repository's code into the github runner environment. This is essential for accessing the codebase so other workflow steps can access the files.

Set Up Node.js: Secondly, the pipeline sets up Node.js version 22.11.0 using the actions/setup-node@v2 action. This ensures that the environment is correctly configured to run Node.js applications and scripts. This saves our team time and avoids inconsistencies of different versions

Install Dependencies: Third, the npm install command is executed to install all necessary project dependencies. This step is needed to ensure that the application has all of the required packages that are needed for it to be run and tested. Otherwise our team would have to manually run npm install each time we pull new changes, potentially receiving missing or outdated dependencies if we don't.

Run Tests: Lastly, the npm test command is executed to automatically run the project's test suite. This step verifies that the code changes do not break existing functionality and that all tests pass successfully. Running all of our tests manually would be time consuming and would result in a higher chance of missing a test. Also, it provides immediate feedback on github for whether the code changes pass all tests. This is helpful for when our team is making pull requests and is checking to make sure it won't mess up our main branch. Without this we would need to get it into our local machine before running tests.

Overall, the pipeline automates the testing process, which reduces the need for manual intervention and minimizes any human error our team might cause. Also by running tests on every push and pull request, the pipeline ensures that the codebase remains stable and consistent. Furthermore, automated tests help in identifying issues early in the development cycle, allowing for quicker fixes and reducing the risk of deploying faulty code. Lastly once tests pass, the code is ready for deployment, streamlining the release process and enabling faster delivery of features and fixes. Overall, our CI/CD pipeline resulted in a faster, easier and more consistent development process for our team.

‘

Testing Strategy & Implementation

Testing Strategy for Cloud-14-Project Our project’s testing strategy included an automated integration test and automated Unit tests for each feature. These automated tests were made using Jest. We also had manual real user testing for each feature, through the help of our classmates. Our unit tests validated the functionality of individual components, one at a time. Our project uses Jest to mock dependencies and test specific functions. For example: In our Nutrition-Calculator feature, we tested functions like addIngredient, removeIngredient, and calculateNutrition to ensure they perform their tasks correctly, such as managing ingredient lists and calculating nutritional information. Also for our Chatbox feature, methods like sendMessage, displayMessage, and toggleChat were tested to verify message handling and chatbox visibility.

Integration Testing: Our integration test was used to verify that different modules work together as intended. Our integration.test.js file was used to achieve this and we tested interactions between recipe recommendations and difficulty estimation. By mocking the recommendation.js module, the tests ensure that displayRecipes and findInfo functions integrate seamlessly, providing accurate recipe recommendations based on difficulty levels. Between all of the automated tests, we have 19 in total and they all pass.

Code Coverage: We use Istanbul to generate code coverage reports, which help us identify untested parts of our code. After we identified untested parts we created more tests to ensure that our code coverage gets to 100%. By aiming for high code coverage, we enhance the reliability of the application and thoroughness of our testing. We finished with 100% code coverage on all of our tested components.

Continuous Integration : Our CI/CD pipeline, defined in our ci_cd_pipeline.yml file, automates the testing process by triggering on every push or pull request to the main branch and running the build and test jobs on a consistent Ubuntu environment. This setup ensures stable and automatic execution of all tests, conveniently providing immediate feedback on code changes in github and maintaining code quality.

Key Takeaways

Throughout this project, we faced many challenges and learned to overcome them. For example, one challenge was time management. All of our group members were full-time students. Thus, we always had other homework and assignments to complete. However, we overcame this by starting our work early. Our team had meetings every Friday to discuss the next sprint. Everyone understood their roles and tasks. Knowing we had other classes, we all started working on our parts of the project as soon as possible. Another challenge we faced was programming bugs. Nothing always went as smoothly as we thought. Thus, we used outside sources such as Chat GPT and YouTube videos to help us understand how to resolve some issues. We also collaborated, as some group members volunteered to help fix issues others were facing. Another challenge faced was GitHub merge conflicts. However, thanks to this class, we learned how to solve them through the labs given in class. Another issue we faced was creating an excellent visual interface. Our teaching assistant told us that our design could be better. Yet, many of us were not experienced user interface designers. However, we took time to learn ways to improve our interface. We add animations such as preloads and moving icons. The animations made our website much more engaging. We know we improved our design because our classmates gave us many compliments. The testing sessions helped bring new perspectives to the table. Many of our users had issues locating some of our features. They suggested adding titles. Manual testing helped us overcome the challenge of meeting visibility heuristics such as system status. As the developers, we already knew how everything operated. Therefore, it was already easy for us to use. However, we added loading signs to help people understand their requests were processed. Another issue we faced was implementing the CI/CD pipeline. The pipelines were new to all of us. We did not know how to implement them. Thus, we were lost and did not know where to begin. The syntax was confusing, and the team was confused. Thankfully, there were online resources that helped explain the pipeline in more detail. Over time, we got used to the syntax and were successfully able to develop a functional pipeline. Also, we had trouble with API integration. Again, many of us had no idea how APIs worked and how to implement them. Also, we had issues with our API because sometimes it would stop working. The reason for this was we went over the limit of our API. We had no other option, as we needed to make a request to test our API. To overcome this issue, we had to regenerate our keys multiple times throughout our journey to complete the project. We also had the challenge of assigning roles. However, because we chose the Scrum Agile life cycle, everyone had their roles. We also had stand-up meetings, and everyone got to voice their concerns. Picking the correct life cycle was crucial. Choosing the wrong one could have made things more difficult for our group. Also, because we created a new page for each feature, we needed more user interface designs. However, it was tough to visualize and go off scratch. Thus, we created a prototype, which helped provide a plan on how to style them visually. However, we did change the styling as needed. We were confused about how our website would work. So, we used data flow diagrams to explain its usage. The diagrams helped the implementation of the features become effortless. Although we faced many challenges, we overcame them.

Future improvements

At the beginning of the project, we wanted to implement a login section. However, due to the strict deadline and lack of knowledge, it's not a part of our website. However, this is something we look forward to adding in the future. One benefit this option provides is privacy. Everyone using the computer cannot access personal information. Right now, our recipes are on the device's local memory. The login features need a backend to handle processing with token-based authentication. We would also need a database to store emails and passwords. For the front end, we would need a login section with the correct styling. Also, we would need to link API from the backend to the front end. The last step would be to test the feature. During manual testing, we saw many people had a meal planning website. We got to see our peers have various features. Some of them gave us inspiration. We saw teams were using AI. Although we did use AI from Spoonacular for our chatbot, we could have used it in more features. One feature we could have improved is our recommendation feature. Instead of having people enter ingredients, it would have been nice to see AI automatically generate recommendations based on the foods the user has eaten over the week. The AI could be developed through the behaviors of the users. We would need to train AI using Frameworks such as Pytorch. Then, display our results on the recommendations page. Another area of future improvement is making the website a little better on mobile. Right now, our website works fine on a computer. It also works fine on the mobile. However, it looks less appealing on mobile. Therefore, adding better styling for mobile could be implemented. Mobile styling can be done by using CSS and targeting various screen sizes. Other ways to improve the website would be to add tutorials. We know that many of our users will be beginners. Although we did give difficult ratings for people to access their levels, giving them tutorials would be helpful. We can implement this by adding links to YouTube videos. Adding the videos can be done through HTML or Javascript. Another feature we could add is a cost estimator. We know many university students or people with a strict budget would use our website. The cost estimator could help individuals stay within their budget. It could also help them save time, as they would know what to get. This feature could be implemented using an API, which has the cost of certain items. Users could enter various foods, and the API would add the cost of the meals or ingredients entered. Another section we could have added is an about us section. We worked very hard on this project. Yet, there isn't any information about us. The About Us section could help people understand who we are and our accolades. The implementation would be simple. The section would need HTML. We would also need CSS to style the page. However, we did not want to crowd the homepage.

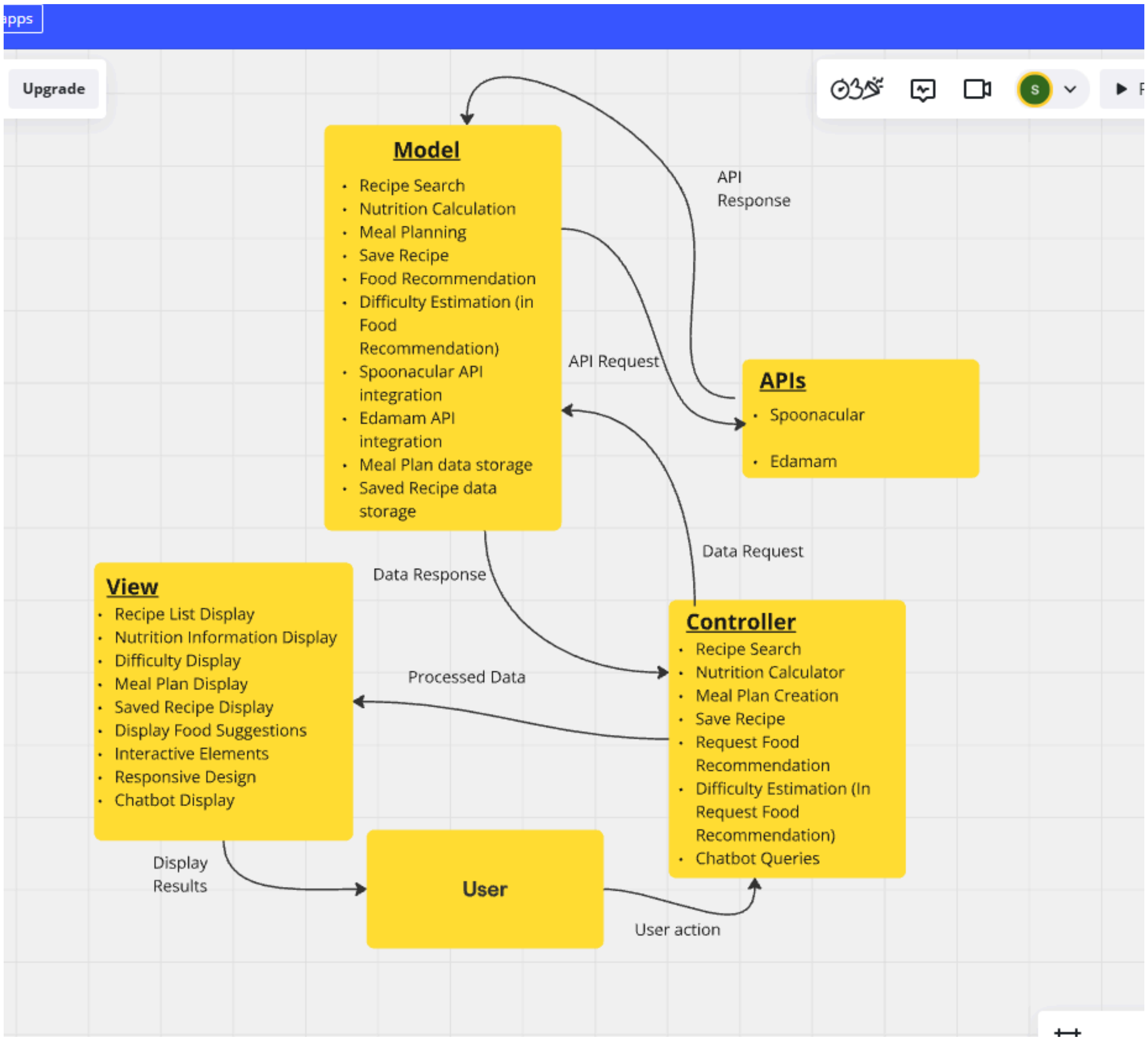
Bugs

Bug Report: No Error Message for Invalid Search Queries **Description** When users input random or invalid queries into the search bar (e.g., "asdf123" or "!!!"), the system fails to display an error message or feedback. Instead, the search results remain empty without any explanation, leaving users confused about whether the search is still processing or if no results were found. This lack of feedback reduces the clarity and usability of the search feature.	**Steps to Reproduce** 1. Navigate to the search bar on the homepage. 2. Input a random, nonsensical query (e.g., "asdf123" or "!@#$%^"). 3. Submit the search by pressing "Enter" or clicking the search button. 4. Observe that no results are displayed and no error message appears to inform the user about the invalid query.	**Severity Level** **Medium**: This bug does not break the functionality of the search feature but negatively impacts the user experience. Users may become frustrated or confused due to the lack of feedback when their queries do not yield results. Adding a clear error message or a "No results found" notification would significantly enhance usability Search Recipe Error Message · Issue #51 · Ramtin-Mandom/Cloud-14-Project.
Footer issue Meal PLan **Description** The footer of the meal planning website displays incorrect styling, causing it to resemble two headers stacked together. This inconsistency disrupts the design hierarchy and creates confusion, as the footer should visually differ from the header and have a more subdued style. Despite attempts to resolve this using AI tools and YouTube tutorials, the issue persists.	**Steps to Reproduce** 1.Open the meal planning website in any browser. 2. Navigate to the bottom of the page where the footer is located. 3. Observe the footer styling, which appears similar to a header:	**Severity:** **Medium**: While the bug does not break website functionality, it significantly affects the user experience by disrupting the visual hierarchy and design consistency. while the bug does not prevent users from accessing core features or completing essential tasks, leaving it unresolved could have broader implications. It detracts from the overall user

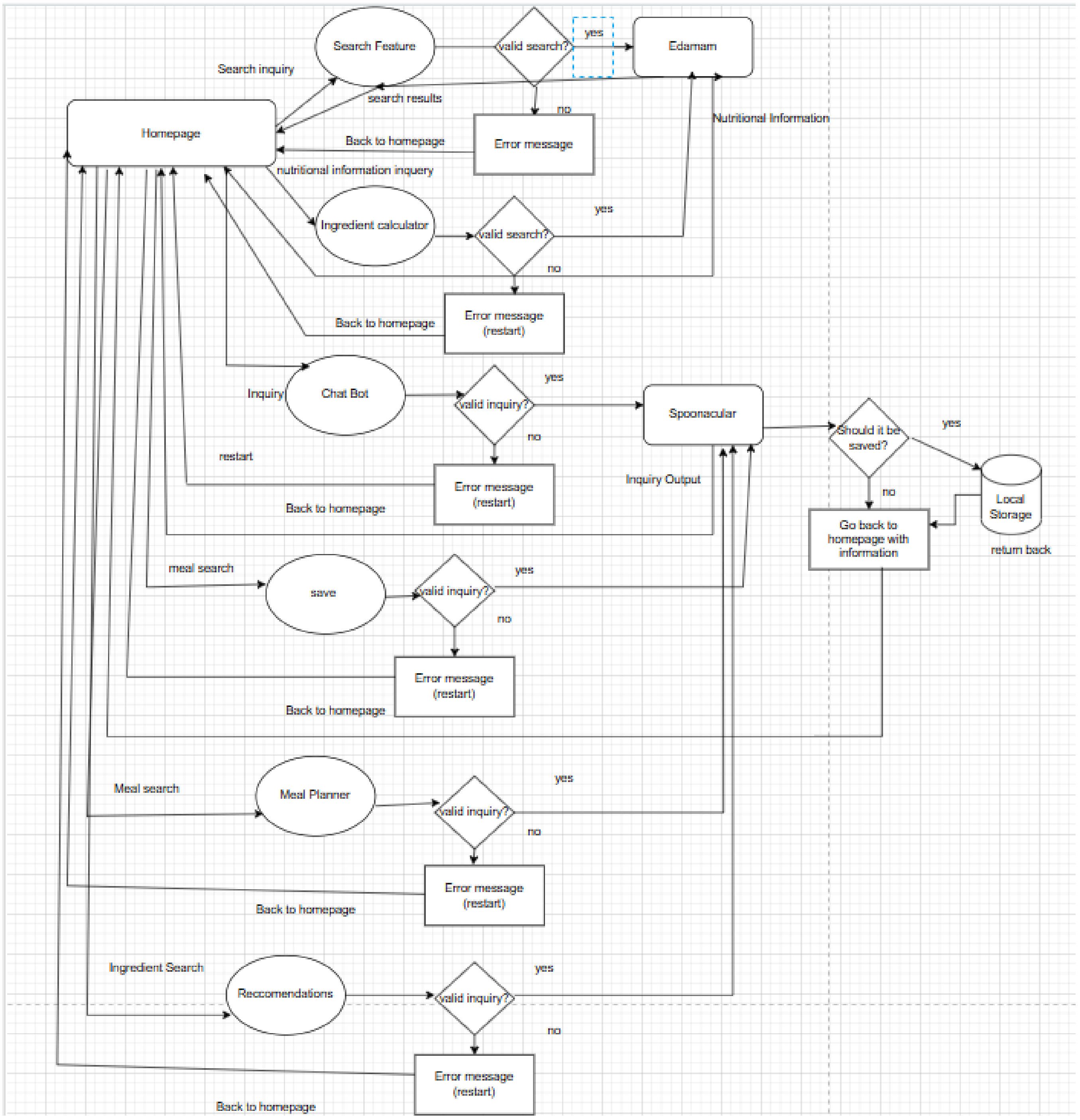
		experience, potentially reducing engagement, and introduces readability challenges, especially for secondary information that should not demand attention like a header. https://github.com/Ramtin-Mandom/Cloud-14-Project/issues/47
Chatbot issue The chatbot sometimes fails to give back the content that it is supposed to give. It will give you a response like: "Here are your recipes" and give you back nothing. This issue is from the API as sometimes as mentioned gives back empty arrays. Also the chatbot cannot handle questions other than asking for recipes. This is an issue from our side as our display function does not know how to handle anything other than recipes from the API.	**Steps to Reproduce** 1.Go to chatbot section of the website 2.In the input box enter: "give me recipes with 1000 calories or less" 3.Press "Answer" button 4.You will see the malfunction	**Severity Level** **High**: This bug makes the usage of this feature inconsistent, and it will leave the user with frustrations. The chatbot is experiencing significant functional limitations. It frequently fails to provide the expected content, such as recipes, returning empty responses. This issue stems from the underlying API, which sometimes delivers empty data arrays. Additionally, the chatbot's capabilities are restricted to recipe-related queries. Our display function is unable to process and present information beyond recipes, further limiting the chatbot's usefulness. This bug severely impacts user experience, rendering the

		feature inconsistent and frustrating. Link: Issue#48

Updated MVC



https://miro.com/welcomeonboard/eFc2SythVm4ybEd4SDFLMERObTRuYnVnMVNNOGkvL1o4VHRFdXRyb0tETThiU2xBODJvWm9QTXRPY0xPelkzZzBtS0M4WDZrTXFmL2l6WkNra016S205TUpQa1dNd1ZpR2MyamwreGRsZDBtcEhKcFBjVjVlY0QyU3VnSFRiZHIhZQ==?share_link_id=67089862192



Link to the Data Flow Diagram can be found here:https://drive.google.com/file/d/1OQZarcWfa2nFXx03nGn2ErIYx4R50vAi/view?usp=sharing

Appendix:

Report Workload:

Ramtin:
Analysis of the project's success in meeting the user needs and solving the problem This should include feedback from real users (testing session with classmates) Analysis of the project's SDLC model and how it was used in the project Include any changes made to the model during the project

Sahil:
Detailed description and overview of the CI/CD pipeline and how it was used in the project Testing, deployment, and monitoring should be included in this description Description of the project's testing strategy and how it was implemented

Sukhman:
Description of the project's future work and potential improvements This should include any features that were not implemented and how they could be implemented in the future Lessons learned and project takeaways Include challenges faced during the project and how they were overcome

Srinjay:
Detailed diagram of the project's architecture An updated (since M1) level 1 DFD of the application's structure and how data flows within the application An updated (since M1) MVC model of the application

Everyone that noticed bugs:

List of known bugs and issues with the project and their severity (table format)
Any bug identified and missing from this list will impact the project grade It is recommended to use a bug tracking tool (e.g., Github Issues) to track these bugs Bug reports should include a description of the bug, steps to reproduce, and severity level and link to Github Issue

Project Workload:

Ramtin (Project Manager): Two features implemented by Ramtin including The chatbot using Spoonacular and the Meal planner.

The chatbot provides users with a simple way to interact with the application for tasks like finding recipes, checking nutrition details, or creating meal plans. Users type queries such as "What can I cook with chicken?" or "How many calories are in rice?" and receive clear, actionable responses. Recipe suggestions come with options to view, save, or modify, while nutritional details are presented in a concise format. The chatbot enhances the user experience by simplifying interactions and providing instant, helpful responses.

The meal planner allows users to create personalized meal schedules based on their preferences and dietary needs. Users input details like meal types, specific ingredients, or calorie limits, and the planner generates a tailored plan with recipes and serving sizes. The plan is easy to follow and includes preparation instructions, with options to save or edit it for future use. Additionally, users can generate a shopping list based on the plan, making meal preparation streamlined and efficient. Also helped Srinjay implement the rest of the features.

Sahil: Made the automated test along with the CI/CD pipelines. Furthermore, Sahil implemented one feature. Sahil implemented the ingredient calculator. The testing and pipeline information is above.

The ingredient calculator helps users determine the nutritional value and portion sizes of specific ingredients. Users can input an ingredient, such as "chicken breast" or "quinoa," along with the desired quantity or serving size. The calculator provides detailed nutritional information, including calories, protein, carbohydrates, fats, and other nutrients. This feature is especially helpful for users tracking their diet or creating recipes, as it allows them to adjust ingredient quantities to meet their nutritional goals efficiently.

Sukhman: Responsible for UI interface designs for all pages. 6 pages in total. Also helped with implementations with certain features such as meal planner and reccomendations.

The user interface (UI) of the meal planning application is designed to be intuitive, visually appealing, and highly functional. The homepage provides easy navigation to key features like recipe search, meal planner, chatbot, and ingredient calculator, with clearly labeled buttons and sections. The interface uses responsive design to ensure a seamless experience across devices, whether on desktop, tablet, or mobile. Interactive elements, such as dropdown menus, filters, and clickable recipe cards, make it easy for users to engage with the application and customize their experience. The color scheme and typography are chosen to create a clean, professional look, while animations and transitions add subtle polish to enhance the overall usability.

Srinjay: Implemented three features with the help of other members. These include Search, Recommendations, and save.

The save feature allows users to store their favorite recipes and meal plans for future reference. With a single click, users can save a recipe to their personal collection, which is accessible anytime through a dedicated "Saved Recipes" section. This ensures users can quickly revisit dishes they liked without needing to search again. Saved meal plans are also stored, allowing users to reuse or adjust them as needed.

The search feature enables users to find recipes based on ingredients, dietary preferences, or specific keywords. Users can input terms like "chicken dinner" or "vegan desserts" and filter results by calorie count, preparation time, or cuisine. The search results are displayed in a clean layout with recipe images, names, and quick links to save or view details. This functionality ensures users can quickly find recipes that match their exact needs and preferences.

The recommendation feature suggests recipes based solely on the ingredients provided by the user. Users input one or more ingredients, such as "chicken" and "broccoli," and the system generates recipes that include those items. This is especially useful for users trying to create meals with what they already have at home. The recommendations are displayed in an easy-to-browse format, allowing users to quickly view, save, or modify their chosen recipes.

Video:

Sukhman: An overview of the project and the problem it aims to solve Project takeaways and lessons learned

Ramitin: The chosen APIs and the features you implemented Use personas and user stories to explain how these features benefit the user

Sahil Overview of the CI/CD pipeline and how it was used in the project Future work and potential improvements

Sammy: Project demo video (1-2 minutes)

Video edited by Ramtin.

Manual testing questions:

1.Is the website easy to navigate?
(rating question from 1-5 stars)

2. Does the website follow the heuristics we learned?
(Paragraph answer)

3.What are some design changes you would make to the website?
(paragraph answer)

4. Rate how easy the task was to complete 5 = easy, 1 = difficult
(rating question 1-5) stars

5.What would make it easier to complete the tasks?
(paragraph answer)

6. Do you have any recommendations for improving the feature?
(paragraph answer)

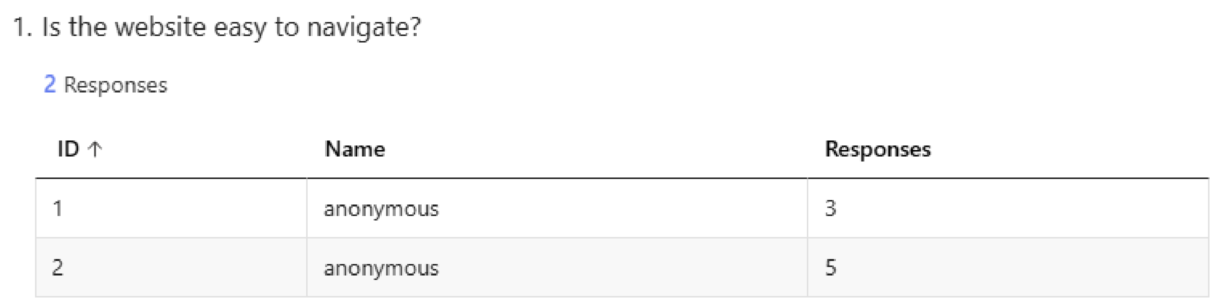
7. Do you have any suggestions for improving navigation or layout?
(paragraph answer)

8.Was there anything about the design or functionality that felt unintuitive?
(paragraph answer)

9.What are your final thoughts?
(paragraph answer)

Answers

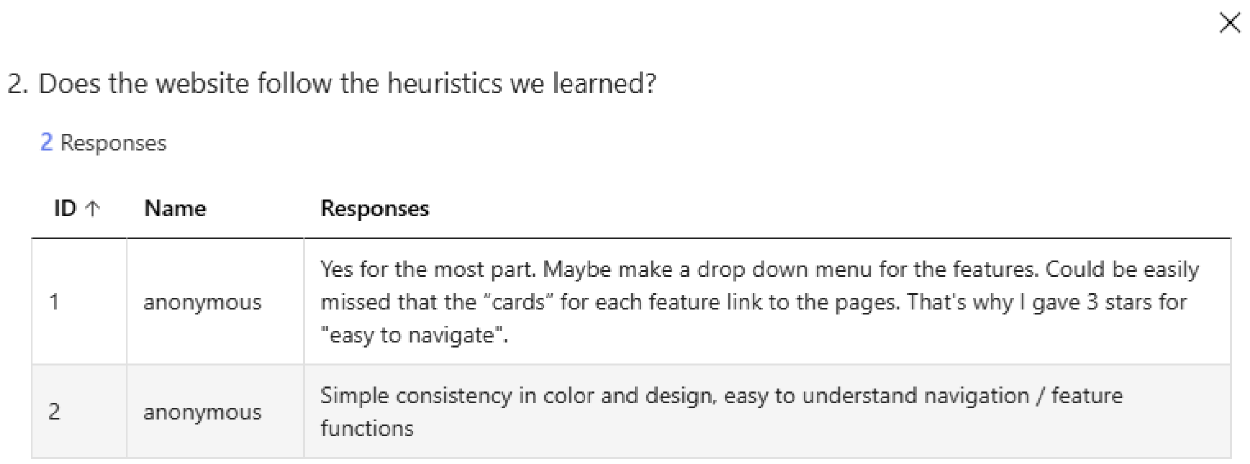
We have received two replies on our testing day.



1. Is the website easy to navigate?

2 Responses

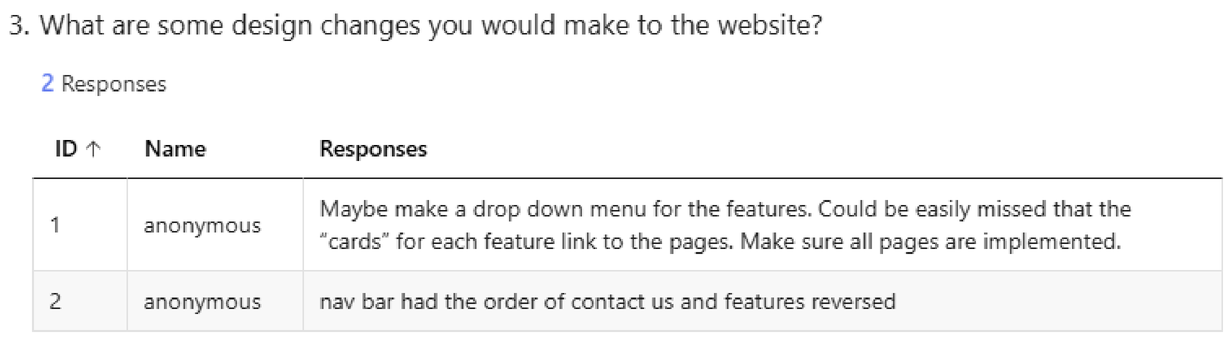
ID ↑	Name	Responses
1	anonymous	3
2	anonymous	5



2. Does the website follow the heuristics we learned?

2 Responses

ID ↑	Name	Responses
1	anonymous	Yes for the most part. Maybe make a drop down menu for the features. Could be easily missed that the “cards” for each feature link to the pages. That's why I gave 3 stars for "easy to navigate".
2	anonymous	Simple consistency in color and design, easy to understand navigation / feature functions



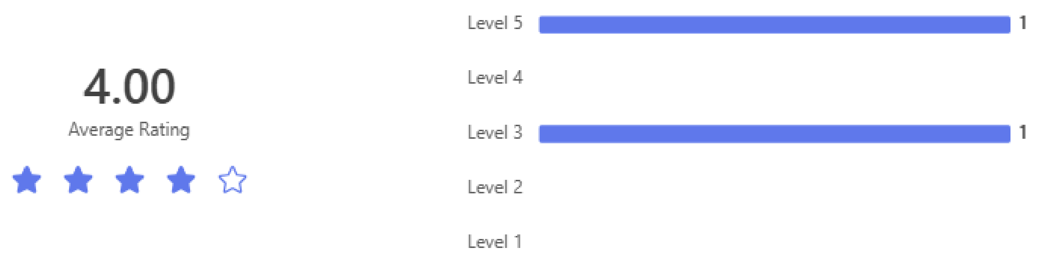
3. What are some design changes you would make to the website?

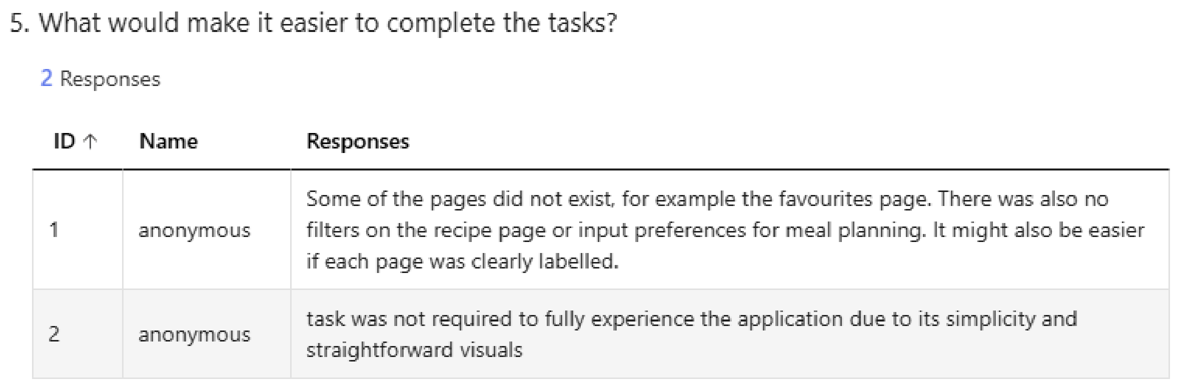
2 Responses

ID ↑	Name	Responses
1	anonymous	Maybe make a drop down menu for the features. Could be easily missed that the “cards” for each feature link to the pages. Make sure all pages are implemented.
2	anonymous	nav bar had the order of contact us and features reversed

4. Rate how easy the task was to complete 5 = easy, 1 = difficult

More details

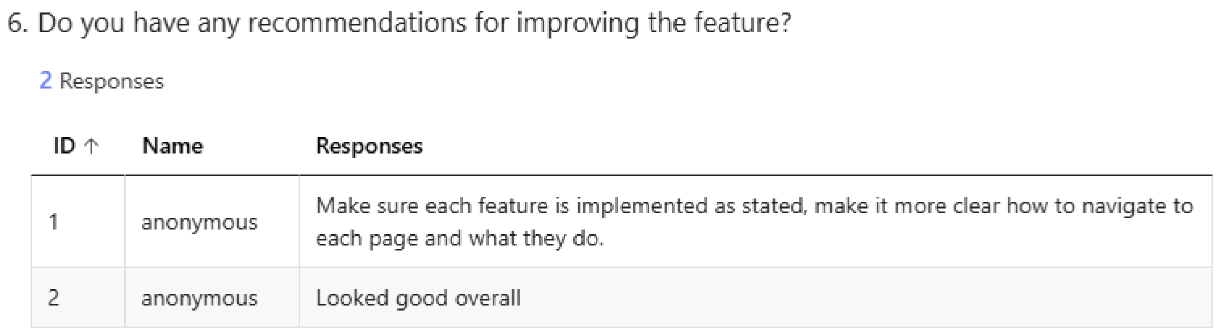




5. What would make it easier to complete the tasks?

2 Responses

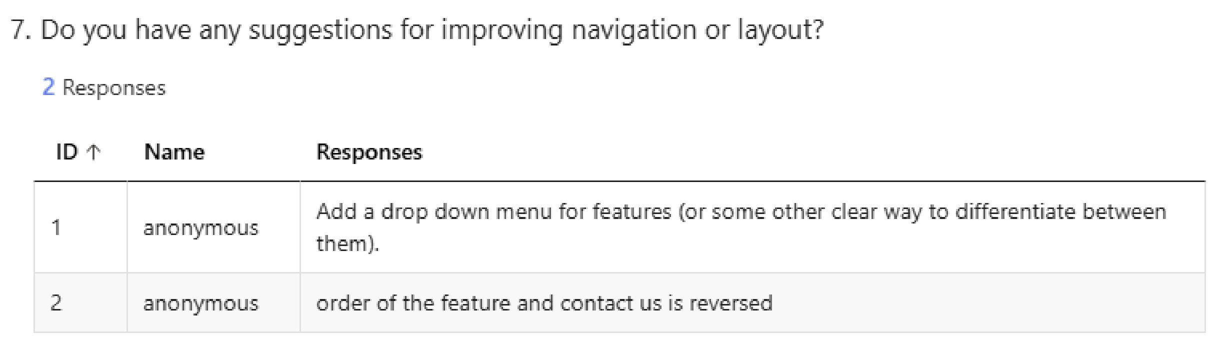
ID ↑	Name	Responses
1	anonymous	Some of the pages did not exist, for example the favourites page. There was also no filters on the recipe page or input preferences for meal planning. It might also be easier if each page was clearly labelled.
2	anonymous	task was not required to fully experience the application due to its simplicity and straightforward visuals



6. Do you have any recommendations for improving the feature?

2 Responses

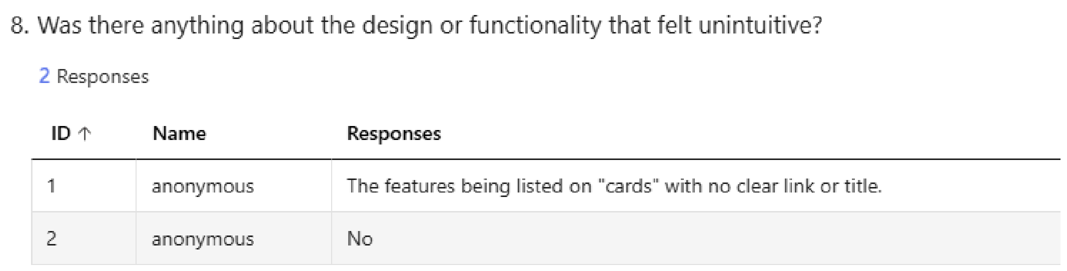
ID ↑	Name	Responses
1	anonymous	Make sure each feature is implemented as stated, make it more clear how to navigate to each page and what they do.
2	anonymous	Looked good overall



7. Do you have any suggestions for improving navigation or layout?

2 Responses

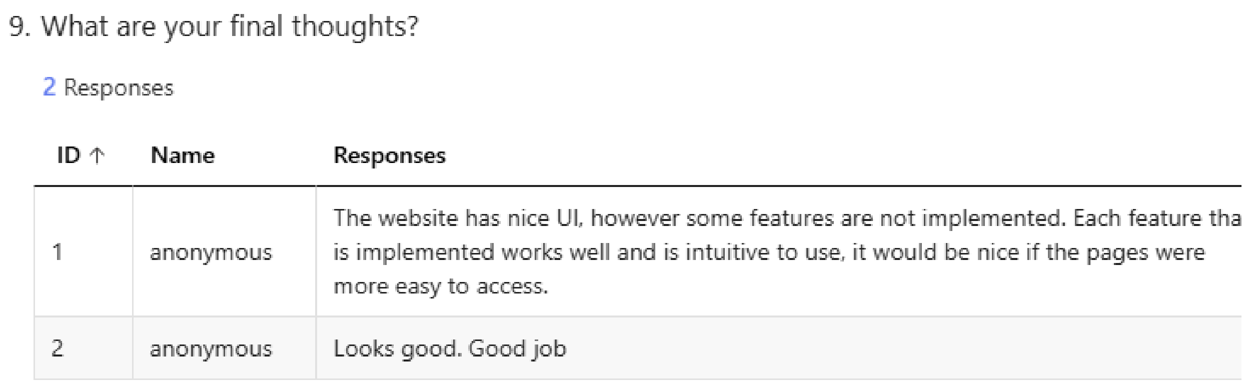
ID ↑	Name	Responses
1	anonymous	Add a drop down menu for features (or some other clear way to differentiate between them).
2	anonymous	order of the feature and contact us is reversed



8. Was there anything about the design or functionality that felt unintuitive?

2 Responses

ID ↑	Name	Responses
1	anonymous	The features being listed on "cards" with no clear link or title.
2	anonymous	No



9. What are your final thoughts?

2 Responses

ID ↑	Name	Responses
1	anonymous	The website has nice UI, however some features are not implemented. Each feature tha is implemented works well and is intuitive to use, it would be nice if the pages were more easy to access.
2	anonymous	Looks good. Good job

Link to survey:
https://forms.office.com/Pages/ResponsePage.aspx?id=fmfoBInJuUeGGdg9Wl9sZ_3j2F-S1BxHrVDGQ6X2NQNUMzk4NjQxUzJYTFA2RUNQNUNUVFNJV1I0Vi4u