Final Project Report: Recipedia

Group members: Alan Palayoor Francis, Ajit Chauhan, Alex Guo, Leeann D’Souza

GitHub Repo: https://github.com/CMPT-276-SUMMER-2025/final-project-16-moons

Video Demo: Website Demo

Project Analysis

The goal of the app was to help busy people stay on track with their nutrition goals and make healthy eating easy. During the testing session, users could enter a main ingredient, area or category and the search results would be displayed. The testers' constraints stayed the same: they were busy students and needed to make healthy eating choices. Due to these restrictions, our app needed to be easy to use and deliver the nutrition information in a way that's easy and quick to understand.

When users entered a main ingredient, the search results were displayed and the tester chose a recipe based on their preferences. They were able to see the recipe and nutrition information in 3 clicks, meeting our goal of our app being quick and easy to use. If the tester was unsure about what to eat, the randomly generated recipes suggested options for them. If they were curious about nutrition information from a simple text-based menu, they uploaded the image to the scanner page and nutrition information was generated.

The peer testing session was invaluable to gaining real user feedback on the application. Overall, students found the website minimalistic and easy to navigate. However, some users found that they were unable to use certain features, such as the Scanner. This feature delivers nutrition information based on text images, however due to certain limitations with the API, they could not carry out the task effectively. There was also a lack of certainty around the categories on the search page. However, the features are still being improved upon and overall, the feedback was very constructive. Users remarked that they liked the layout of the website and its compliance with heuristic principles. Additionally, users were able to follow the instructions without any complications, resulting in a smoother user experience.

Analysis of SDLC Model

The Agile SDLC model enabled us to work collaboratively while ensuring flexibility throughout each phase. First, the requirements phase ensured that all the group members knew what the project's agreed upon requirements were and how they were going to be integrated in the app. For example, we agreed on a recipe app with search features and nutrition information, which were non-negotiables. Next, we decided that having a random recipe generator and menu scanner would also be a benefit to the user and made these features part of the requirements. As part of the requirements phase, we agreed to meet once a week and have dedicated sprints to carry out development of the app. The design phase was carried out through Figma, where group members collaborated on the wireframe of the application. The user interface was designed to be minimal and simple to use. We decided on a colour theme and the general layout of the app. At some point, we decided that the buttons on the search page should be explicit and not just a drop-down menu, so we pivoted and executed those changes promptly.

Next, the development phase meant that team members were developing their assigned parts of the app and using Github to compare and add their parts of the code to the existing code base. We used Kanban to organize tasks that were to be carried out and completed, ensuring that requirements of the app were being met and group members were carrying out their assigned tasks. Code was reviewed for approval by other team members before updating the main code base and if changes were to be made, we would implement them before the update. Additionally, each sprint ensured that the development was carried out in manageable parts and completed by a dedicated deadline. The testing phase meant that we needed to review the code and test it in parts to ensure functionality and robustness. The peer feedback session was a great opportunity for us to extend that testing to someone who may not already be familiar with the app and allowed us to see blind spots. This testing session, along with the unit and integration tests allowed us to improve the user's experience.

Finally, a working, tested version of the app was deployed using a CI/CD pipeline. This topic will be explored further later in this report. Overall, continuous collaboration, flexibility and a working application, which are cornerstones of the Agile framework, resulted in an application that was functional and beneficial for the end user.

Features

Additionally, our app utilized six key features that provided benefits to the end user. The features remain consistent with our initial proposal. The first API, TheMealDB, was used for three key features: search for a recipe based on a type of cuisine, search for a recipe based on a main ingredient and generate three random recipes for each meal of the day. The first feature enabled users to search for a recipe based on a type of cuisine or area. Users were shown up to 10 recipes, pulled from TheMealDB database, allowing them to choose recipes they may not have previously known about. The second feature allows users to search for a recipe based on a main ingredient also utilizing TheMealDB recipe database. This feature allows users to make a recipe using ingredients that they already have in their kitchen. Finally, users were able to generate three random recipes, one for each meal. This makes it easier for the user to decide what to eat for each meal, enabling them to find each recipe with one click.

The second API, CalorieNinjas, was used to provide nutrition information based on a search query. It also provides nutrition information based on a text-based image and a way to search for recipes based on name. After searching for a recipe, the API displays nutrition information, such as protein, carbohydrates and fats, based on the ingredients for each recipe. This enables the user to know the nutritional facts of the food they make. The scanner feature allows users to upload a text-based image, such as a menu, and the API is called to display the nutritional profile for the food items. This helps users decide on better choices when eating at restaurants. Finally, the API was also used for its search feature. Users could enter a recipe and the database would display all the recipes related to the search query. This helps the user broaden their search for a recipe when deciding what to eat.

Overview of CI/CD Pipeline

The project's CI/CD pipeline is designed to automate the process of building, testing, and deploying the application. The Continuous Integration (CI) phase is triggered by Push and Pull request events on the main branch, ensuring that all code changes are automatically validated. The CI.yml file establishes our Continuous Integration pipeline. It manages processes such as ensuring the utilization of Node.js version 18, installing necessary dependencies "via npm ci", executing lint checks with "npm run lint", and conducting unit tests using "npm test". This automates the process ensuring the new code integrates correctly with the existing codebase.

Testing Strategy

The project's Testing Strategy was to employ both unit tests and Integration tests for all feature pages. We used Vitest as our testing framework.

Unit Testing : Unit tests focus on isolated components to verify their functionality

1. SearchPage Unit Test : Unit test for Search mocks the fetch API to simulate successful searches, scenarios with no results, and network failures.This verifies that the component correctly renders search results or appropriate error messages
2. IndecisivePage Unit Test : Tests for indecisive Page validate that clicking the "Surprise Me!" button triggers three mock API calls and renders the expected recipes. It also confirms that an error message is displayed if a fetch fails.
3. ScannerPage Unit Test : Unit tests for the Scanner Page check input validation by simulating the selection of files that are too large or have unsupported formats. It also ensures that the "Get Nutrition!" button remains disabled and an error message is shown in these cases.
4. SearchResult component Unit Test : The SearchResult component is tested to ensure that when a modal is opened, it initiates an API call to fetch nutrition data for each ingredient listed in the recipe and displays this information correctly.

Integration Testing : Integration Test ensures that components work together as intended.

1. Search Page Integration Test : This test verifies the functionality of Search Page and its interaction with APIs. For testing Search by name it confirms a single call to the API-Ninjas endpoint. For an "ingredient" search, it verifies that two separate calls are made to TheMealDB API: one to filter by ingredient and a second to retrieve detailed recipe information. Test also checks if error message is displayed on failed fetch

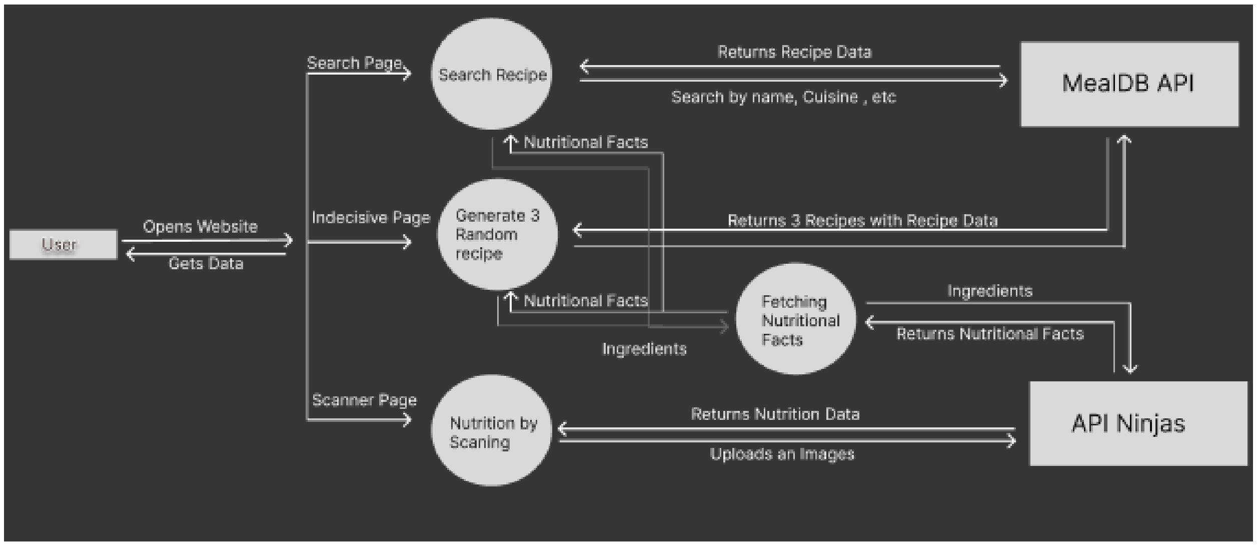
2. Indecisive Page Integration Test : This test confirms that clicking "Surprise me!" triggers three fetch calls . Resulting in three SearchResult components that are also verified if it displayed correct data from the mock responses. Test also checks if error message is displayed on failed fetch

3. Scanner Page Integration Test : Upon uploading a valid image, a test confirms the following sequence: an initial mock call to an image-to-text API, followed by a second mock call to a nutrition API, culminating in the display of comprehensive nutritional data

on the screen. Test also checks if error message is displayed on failed fetch or incorrect file size.

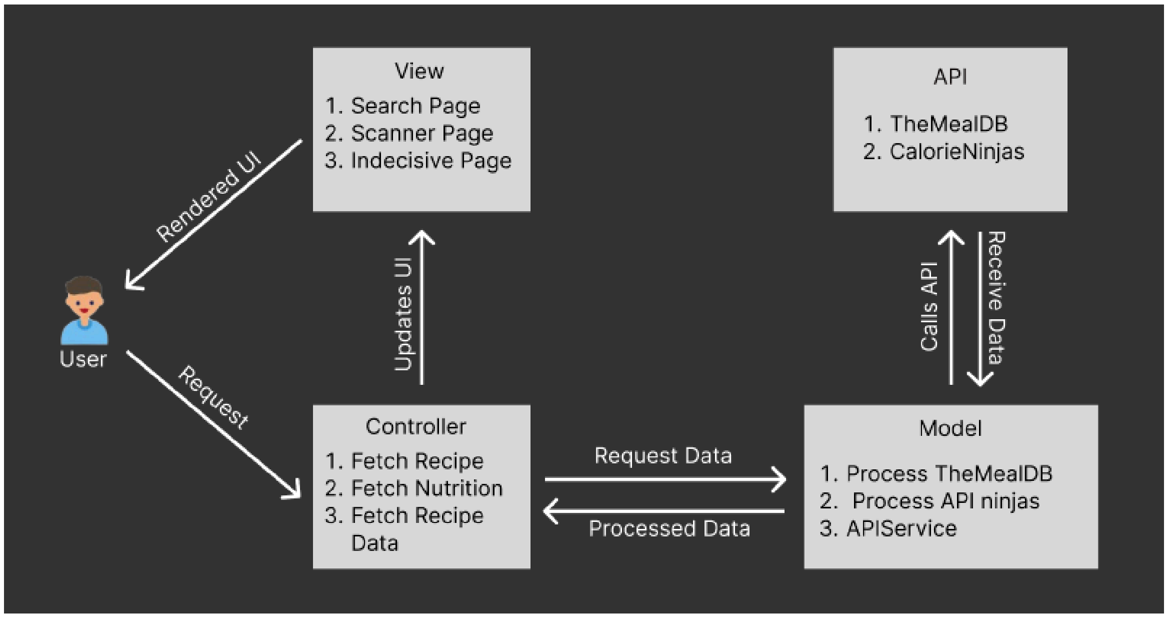
As part of the CD stage, we used Render to continuously deploy the app through the server. After CI build which includes linting and passing all the tests , the code is automatically deployed to the server. This ensures that the application is always up-to-date with the main branch.

Updated Level 1 DFD



Data Flow Diagram Level 1

Updated MVC model



MVC model

List of Known Bugs/Issues & Limitations

- There are currently no known bugs/issues.
- You might see: “Uncaught (in promise) Error: A listener indicated an asynchronous response by returning true, but the message channel closed before a response was received” as an error in your console, but this is due to the browser extensions that you have installed.
- You might also see some console errors that appear after you encounter an error (i.e., uploading an image with unclear text) on the website. This is intentional, and is for debugging purposes.

Known Limitations	Severity (Fibonacci scale of 1-8)
API Ninja’s text to nutrition endpoint may go down without warning	**8**
The recipe search by name feature isn’t as good as it can be	**8**
The website isn’t responsive	**5**
API Ninja’s image to text and text to nutrition endpoints sometimes don’t work properly	**3**
There is a limit of 10,000 API calls to API Ninjas per month	**1**
The white boxes on the Search & Indecisive pages don’t have animations	**1**

Future Work and Potential Improvements

- Make the website responsive by adding extra styling to the components and elements
- Implement a better recipe search by name feature by using TheMealDB’s endpoint instead of API Ninja’s endpoint (this can already be accomplished by commenting out and uncommenting some things in Search.jsx)
- Implement a text to nutrition feature for the Scanner page by using API Ninja’s endpoint
- Implement the “Recipedia” page that has more info on the project/website
- Implement “The Team” page that has more info on the team
- Implement animations for the white boxes on the Search & Indecisive pages
- Implement a better/more engaging website design/layout using Figma and inspiration from existing websites

Lessons Learned and Project Takeaways

Throughout the development of the Recipedia, the team encountered various challenges that tested our technical skills, teamwork, and adaptability. These experiences provided valuable learning opportunities and helped us grow as a team.

1. API Limitations
   TheMealDB and API ninjas are quite beginner friendly APIs to work with, but they also introduced points of failure. For instance, CalorieNinjas' text-to-nutrition and image-to-text endpoints were occasionally unreliable. Additionally, the monthly request limit of 10,000 calls imposed a constraint that we had to plan around.
   Takeaway : We learned the importance of implementing fallback mechanisms, handling API failures gracefully in the UI . In future projects we would consider caching responses or using more reliable alternatives if available.

2. Collaborative Development with GitHub
   Managing a shared codebase via GitHub was a new experience for team members. We encountered issues such as merge conflicts, inconsistent commit messages, and redundant code during the early phase of the Project.
   Takeaway : Effective Communication, consistent Git workflow and regular code reviews helped us overcome these challenges.

3. Agile Workflow and Team Coordination
   The adoption of the Agile SDLC model proved advantageous. However, coordinating sprints, managing tasks, and conducting regular check-ins necessitated discipline. Initially, we encountered difficulties in aligning schedules and maintaining sprint momentum
   Takeaway: Weekly meetings and a shared Kanban board helped us stay on track.

4. User Feedback
   The peer testing session revealed issues we had not previously considered, such as confusion around category labels and occasional non-functional components
   Takeway: Real user testing is crucial. We learned that even minimal feedback can lead to significant improvements.

Appendix

Group member contributions:

Alex-
-Worked on group contract with team
-Joined team GitHub
-Brainstormed project ideas
-Researched API's for project ideas
-Created a low-fidelity storyboard in Figma
-Revised group contract with group members
-Revised project proposal
-Setup tech stack in the repo
-Got familiar with tech stack
-Learnt about React hooks and TailwindCSS classes
-Converted the low-fidelity storyboard to mid-fidelity
-Finalized on our 2 primary API's
-Finalized 6 features of the websites
-Revised and confirmed Milestone 1 report with team
-Setup CD using Render
-Setup page routing
-Added skeleton pages/components to website
-Added custom DaisyUI colour theme to project
-Worked on Milestone 1 video presentation
-Implemented Section 1 for Home Page
-Completed peer review
-Implemented functionality for the search bar and search results
-Implemented Indecisive and Contact pages.
-Worked on Milestone 2 report

Ajit-
-Worked on group contract with the team
-Created GitHub repo
-Finalized tech stack with the team
-Researched API's and brainstormed potential project ideas
-Brainstormed 3 potential features for Nutritionix API
-Completed assigned work for Project proposal
-Got familiar with tech stack
-Revised and resubmitted Milestone 0 according to TA feedback
-Made MVC model diagram with detailed description of each part
-Created project schedule
-Drafted appendix for Milestone 1

-Setup initial CI to check if app is building and syntax is correct
-Worked on Milestone 1 video presentation
-Completed Section 2 of app Home Page
-Created integration tests for All the features
-Implemented recipe pop-up functionality on search page
-Created unit Tests for All the Features

Leeann-
-Worked on group contract with the team
-Joined GitHub with team
-Researched API's and brainstormed potential project ideas
-Completed cover page of project proposal
-Researched possible users for proposal personas
-Got familiar with tech stack
-Looked at potential theme colours for the app
-Learned about API's
-Conducted risk assessment
-Came up with 15 risks and solutions
-Integrated risks and solutions into report
-Created data flow diagram for Milestone 1
-Worked on Milestone 1 video presentation
-Edited M1 video
-Finalized colour scheme for website
-Implemented NavBar for website
-Completed peer evaluation
-Implemented UI for search page and search bar
-Worked on report for Milestone 2

Alan-
-Worked on group contract with the team
-Joined group GitHub
-Brainstormed project ideas
-Shortlisted API's for our best ideas
-Came up with 3 features for assigned API
-Completed assigned task in project proposal
-Revised group contract with team
-Got familiar with tech stack
-Made Work Breakdown Structure for M1
-Chose SDLC model with justification
-Created GitHub issues for WBS tasks
-Finished Milestone 1 report with the team

-Completed M1 video presentation tasks
-Chose colour scheme for website
-Implemented footer for app
-Completed peer review
-Implemented scanner page
-Made peer testing tasks and feedback survey
-Tested app for any missed bugs/issues
-Worked on report for Milestone 2

Revisions since last milestone:

In the last milestone our features were split between the two API's with 4 features coming from MealDB and only 2 from API Ninjas. Since then we have revised our website to evenly split the features between both MealDB and API Ninjas.
Since our last milestone we have also added a Contact page and Indecisive page, which was not originally included as we didn't know whether to add or not due to concerns about our time limit.

Peer testing survey & feedback: Link to responses

Q1 - What did you think of the layout of our website?
-I thought the layout of the website was clean/minimalistic and well designed.
-I like the UI. It follow the heuristic principles
-well structured

Q2 - What did you think of the colour scheme of our website?
-I like the orange but maybe the website uses too much of a grey colour scheme
-It is very nice. Can easily tell the difference between background and text.
-a bit bland could be more colorful

Q3 - How easy was it to navigate our website?
-No problems
-Very easy
-yes

Q4 - Which page element, if any, felt confusing?
-No problems
-I think in the homepage the introduction to the video can be reduced to show the features maybe.
-the search feature was slightly confusing

Q5 - Did any page fail to load or break? What technical issue, if any, did you notice?

-nope
-the nutrition scanner failed to load any results

Search Page Questions

Q1 - How easy was it to follow the steps of your task? (Task 2 and 3)

-easy
-very easy
-pretty easy to understand and follow

Q2 - What do you think of the layout of our search page?

-very good
-abit bland but otherwise well structured

Q3 - What do you think of the usability of our search page? Did it work as expected?

-maybe make it more obvious you can scroll when you click a result from search
-yes. works perfectly
-kinda, some of the categories arent really well defined likely an api issue

Q4 - What quality would you rate the information provided? What additional details would you like to see?

-7.7
-Already mentioned about the home page layout
-8/10

Scanner Page Questions

Q1 - How easy was it to follow the steps of your task? (Task 4)

-very easy
-very simple

Q2 - What do you think of the layout of our scanner page?

-very good
-easy to use

Q3 - What do you think of the usability of our scanner page? Did it work as expected?

-yes. work as expected
-no, failed to work

Q4 - What quality would you rate the information provided? What additional details would you like to see?

-n/a

-n/a - didnt work