Recipedia

Final Project Milestone 1

Alex Guo, Ajit Chauhan, Alan Palayoor Francis, Leeann D'Souza

Github link: https://github.com/CMPT-276-SUMMER-2025/final-project-16-moons

API Selection:

1. TheMealDB:
    a. TheMealDB is a crowd-sourced database with hundreds of recipes that include ingredients, instructions, pictures, and more. TheMealDB offers free access to select API calls, along with a considerable amount of API calls that allow users to find recipes using filters like random, main ingredient, or cuisine. TheMealDB will be used in our project as one of the primary APIs, in which we will create 3 features from.
2. CalorieNinjas:
    a. CalorieNinjas is a robust nutritional data API that allows users to get important information about their food. CalorieNinjas also offers free access to select API calls, along with a considerable amount of API calls that allow users to generate nutritional data from text and images, as well as searching for recipes by name. CalorieNinjas will be used in our project as the other primary API, in which we will create 3 features from.

API Features:

1. TheMealDB:
    a. Feature 1: Generate 3 random recipes for each meal of the day
        i. This API provides an endpoint to generate 1 random meal from its database. We will be using this to generate 3 random recipes for each meal of the day, by calling the endpoint 3 times. The user can use this feature by clicking a button on the “Indecisive Page” of our website and then it will display a random recipe for breakfast, lunch, and dinner. This feature benefits users by helping them decide on what to eat in a day, which can help them diversify their food palette, and also provides them with detailed recipes.
    b. Feature 2: Search for recipes using 1 main ingredient
        i. This API provides an endpoint to find all the recipes in its database that contain the 1 main ingredient being searched up. The user can use this feature by selecting the main ingredient filter on the “Search Page” of our website, then entering the main ingredient into the search bar, and finally it will display all the recipes in the database that contain this main ingredient. An example of a main ingredient could be “Chicken Breast.” This feature benefits users by helping them decide on what to eat based on 1 main ingredient, which can be useful when they only have certain ingredients, and also provides them with detailed recipes.
    c. Feature 3: Search for recipes using 1 cuisine
        i. This API provides 2 endpoints to find all the recipes in its database that either contain the area or category being searched up. Although searching by area or category are 2 different endpoints, we are considering them to be 1 feature because they both fall under the term “cuisine.” The user can use this feature by selecting either the area or category filter on the “Search Page” of our website, then entering either

the area or category into the search bar, and finally it will display all the recipes in the database that contain the area or category. An example of an area could be “Italian.” An example of a category could be “Seafood”. This feature benefits users by helping them decide on what to eat based on 1 area or category, which can help them discover new recipes from a cuisine they already like, and also provides them with detailed recipes.

2. CalorieNinjas:
    a. Feature 1: Generate a nutritional analysis from text
        i. This API provides an endpoint to generate a detailed nutritional analysis based on a text-query with a maximum of 1500 characters. The nutritional analysis contains important information such as calories, protein, sugar, saturated fat, and more. The user can use this feature by searching up a recipe on the “Search Page” of the website, and then it will display a nutrition analysis (along with the recipe) without any further user-interaction. No further user-interaction is needed because the analysis is generated from the ingredients of the recipe they searched up. An example of a text-query could be “10oz onion and a tomato.” This feature benefits users by providing valuable nutritional information based on the ingredients of a recipe, which can be useful for people who track their meals or are curious about the nutritional facts.
    b. Feature 2: Generate a nutritional analysis from image
        i. This API provides an endpoint to generate a detailed nutritional analysis based on an image with food-based text. The nutritional analysis generates the same important information as the text-based one. The user can use this feature by uploading an image to the “Scanner Page” of the website, and then it will display a nutritional analysis. An example of an image could be a menu, recipe, or food journal entry. This feature benefits users by providing valuable nutritional information based on an image, which can be useful for when they’re at a restaurant or are just curious about an image with food-based text in it.
    c. Feature 3: Search for recipes using 1 name
        i. This API provides an endpoint to find up to 10 recipes in its database that contain the 1 name being searched up. The user can use this feature by selecting the name filter on the “Search Page” of our website, then entering the name into the search bar, and finally it will display up to 10 recipes in the database that contain the name. An example of a name could be “Mushroom Risotto.” This feature benefits users by helping them decide on what to eat based on 1 name, which can be useful when they don’t remember the full name of a recipe, and also provides them with detailed recipes.

Mid Fidelity Prototype:

- Mid-Fidelity Interactive Prototype Link
- Mid-Fidelity Non-interactive Wireframe Link
- Professor and TA's were also invited by email for both links.

SDLC Model: Agile (Kanban)

The SDLC model we have chosen for our project is the "Agile" model. The reason we have chosen Agile is because it works well for small scale projects that are run by self-governed developer groups like our group. The name "Agile" comes from the fact that this SDLC model can be seen as more agile than other models like waterfall or V-shaped, because unlike those models, Agile is more suited for repeated changes in functional requirements and design and allows developers to be more receptive to feedback in the development part of the SDLC. This is useful for us as our projects will be getting feedback after every milestone, as well it leaves an avenue open for us to change things in our project as we see fit. Our SDLC model also uses a kanban board, which makes us use a shared board with multiple lanes for issues which are based on their status of completion (backlog, picked up, in progress, completed), so that all members can quickly see what is being done and what needs to be done next. We have already implemented a GitHub kanban board and have used it for milestone 0. A kanban board is useful for us as well because it allows us to efficiently order tasks that need to be done.

WBS:

Research (Done)

APIs

MealDB & API Calls:

Recipe generation for each meal of day

Recipe using user-specified main ingredient search

Recipe generation based on cuisine

CalorieNinjas & API Calls:

Recipe search by name

Nutritional info from image

Nutritional data from entered ingredients

Chomp & API Calls (Backup API):

- Recipe search by name
- Search recipes using specific ingredients
- Nutritional data from barcode

Spoonacular & API Calls (Backup API):

- Generating recipe from desired nutrition or ingredients
- Price breakdown of ingredients
- Meal plan for day or week
- Wine pairing suggestions
- Generation random popular recipe

Tech Stack

React

DaisyUI

Vite

TailwindCSS

Implementation

Footer

“Features”

“About”

Navbar

Selected page is underlined

“Home, Search, Scanner, Indecisive, Contact”

Home Page – Section 1

- Intro for website

Home Page – Section 2

- "What We Offer"
- 3 squares for our main 3 features

Search Page

- Search bar
- List of search results
- Dropdown button for search options

Search Page – Recipe Popup

- Recipe details
    - Picture, name, ingredients, instructions, nutritional data
- Buttons to save as pdf and close recipe

Scanner Page

- Buttons to upload image and get nutrition
- Nutritional analysis displayed
    - Option to save as pdf

Indecisive Page

- Button to generate recipes
- List of recipes for 3 meals of the day

Contact Page

- Email form to contact us.

Colour scheme

Choose colour scheme / theme

Apply to elements of website

Testing

Test API Calls

Test Basic UI

Test Final Implementation

Test Deployed Website

Deployment

Uploading final product to webhost (Vercel)

Project Schedule:

Starting 8 July

Sprint 1 : UI Skeleton (July 8 - July 14)

Milestone :Completing Core page layouts and deciding color scheme

1. Build Navbar and footer
2. Create HomePage (Intro for website, What We Offer)
3. Choose and apply final color scheme

Buffer : 14-15 July

Deadline : 15 July

Sprint 2 : Search and Scanner Core Features (July 16 – July 22)

Milestone : Search and Scanner Page Completed

1. Implement Search Page
2. Implement Scanner Page

Buffer : 22–23 July

Deadline : 23 July

Sprint 3 : Functionality and Styling (July 24 – July 30)

Milestone : All pages functional and styled

1. Implement Indecisive Page
2. Build Contact Page
3. Add Spoonacular advanced features (Optional)

Buffer : 30–31 July

Deadline : 31 July

Sprint 4 : Testing and Deployment Prep (August 1 – August 6)

Milestone: Fully tested app ready for deployment

1. Test API calls
2. Test UI and feature functionality
3. Test “Save as PDF”
4. Prepare deployment

Buffer : 6-7 August

Deadline : 7 August

Sprint 5: Deployment (Also Buffer Week for entire project) (August 8 – August 10)

Milestone: App deployed

1. Upload app to web host
2. Team review of deployed app
3. Final adjustments (if needed)

Deadline : 10 August

Risk Assessment

Low

1. The UI doesn't look cohesive or aesthetically pleasing throughout the app
   Solution: Choose a simple theme that's easy to integrate into the app's interface and stay consistent with it.
   2.Misunderstanding of features and functionalities
   Solution: Keep features simple, clarify if necessary and refer back to the project proposal.
   3.Typos, small bugs and errors are present throughout the app
   Solution: Edit the code and copy, have different people review it
   4.User error when using the app
   Solution: Make sure the app is simple and straightforward to use, make inputting and reading data easy for the user
   5. Users don't find the app necessary
   Solution: Highlight the benefits of the app and the improvement it could make to individuals' lives when presenting it to potential users

Medium

1. The API doesn't work or is very slow to use
   Solution: Revisit the project details and use our backup API instead.
2. There's difficulty implementing certain features or some feature is not feasible
   Solution: Implement features that aren't too complex, ask for help from other group members if necessary, stick to the core functionality of the app.
3. The application doesn't follow usability heuristics
   Solution: Map out the heuristics that aren't being followed and make the necessary changes so that the app complies with them.
4. Inaccurate nutrition information is presented by the API
   Solution: Verify the accuracy of the data provided during implementation. Change the API to one that provides accurate information.
5. The application is slow and difficult to use
   Solution: Keep code clean and functionalities simple. Avoid memory leaks.

High

1. Team members not being able to finish the project by the deadline
   Solution: Everyone keeps a structured plan and follows through with it to prevent a late submission. Other group members will help with the rest of the work.
2. The server is down and affects deployment
   Solution: Change server from Vercel to Netlify
3. Not testing the app sufficiently, thus resulting in unexpected bugs at various stages
   Solution: Perform sufficient unit testing so the application runs smoothly before deployment.
4. The core features don't work as expected
   Solution: Fix bugs that prevent major functionalities and features from working properly, implement alternative features if necessary.
5. The API doesn't provide the necessary information or is behind a paywall
   Solution: Use our backup API's or find another API that provides the same functionality, after confirming the validity with the TA.

Data Flow Diagram :

Figma link:
https://www.figma.com/design/IQORowZr4y1i5r0mcXLba7/Untitled?node-id=0-1&p=f&t=Xqj4z5RQJn8IYVZK-0

MVC model :

Figma link :
https://www.figma.com/design/KR7CTdR5wnBQrP2AzjX9QG/Untitled?node-id=0-1&t=rQd6TKXk6rR3Z1rv-1

MVC Model Overview :

Users can search for meals, recipes, and get nutrition details of those recipes. The Controller handles user input and coordinates between View and Model, the Model interacts with external APIs (TheMealDB and CalorieNinjas) to fetch and process data, and the View displays the appropriate UI based on the user's interactions.

Component Breakdown:

1. **Model -** Handles data operations and API communication

Recipe Repository : Fetches recipe data from TheMealDB

Nutrition Repository : Gets nutrition facts from CalorieNinjas

API Service : Manages all external API requests

2. **View -** Renders UI based on controller and user interactions.

   Pages : Search , Nutrition , Recipe Detail , Nutrition Details are different Component

3. **Controller -** Processes user interactions and co-ordinates between Model and View.

   Functions: RecipeFetch , Nutrition and Search

Appendix :

Team Contribution :

Name	Block	Contribution Details
Alex Guo	Design Block	1. Created mid-fidelity interactive prototype with screens, links, and basic user flow. 2. Wrote short descriptions for design 3. Describe how 2 APIs and 6 Features will be implemented
Alan Palayoor Francis	Process Block	1. Wrote a detailed justification on why we chose Agile (Kanban) 2. Created a WBS for our project tasks 3. Created GitHub issues for WBS tasks
Leeann D'Souza	Analysis Block	1. Conducted a Risk Assessment (5 low, 5 medium, 5 high risks) and came up with clear mitigation strategies for each 2. Created Data Flow Diagrams (Level 0 and Level 1) 3. Used DFD to show data flow

Name	Block	Contribution Details
		between API and application
Ajitsingh Arjuinsingh Chauhan	Architecture Block	1. Designed the MVC Model Diagram with detailed descriptions 2. Created the Project Schedule with milestones, internal deadlines, and buffer days for planning, design, development, testing 3. Drafted the Appendix including group members' contributions and changelog of milestone revisions.

Changelog Table:

No changes were made to the project since Milestone 0.

Additional

- Low-fidelity Storyboard Screenshots Link