Custom Web Application MealCards

Author:

Name: Tri Trung Ton

Student ID: 103808977

Tutor, **Lecturer, Convenor**: Grace Tao.

Swinburne University of Technology

COS30043 - Interface Design and Development, Semester 1, March 2024

Table of Contents

Table of Contents 1
1. Introduction 2
2. Project Overview 2
3. Requirements 3
4. Design Architecture 4
5. Technical Aspects 7
6. Innovative Features 8
7. Project Development 8
8. Challenges and Solution 8
9. Technology Used 9
10. Project File Structure 9
11. Appendix 12
12. Screenshot of the web app 12
Reference 14

1. Introduction

The MealCards project is an HD quality customised web application for COS30043 - Interface Design and Development that enables users to find, explore, and organise their favourite meals. This program lets users explore an extensive database of meals, see specifics about each one, and store their favourites for easy access using Vue.js. Users may access a variety of meal choices along with ingredients and preparation instructions thanks to the connection with the TheMealDB API.

2. Project Overview

Comprehensive features of the MealCards software improve the user's meal discovery experience. The project is constructed using Vue.js, a progressive JavaScript framework renowned for its component-based design and simplicity of interaction with other technologies. Principal elements of the software include

User Authentication

- Registration: Users can sign up using their email and password, which are securely stored in the database.
- Login: Users can access personalised features like viewing and managing their favourite meals.

Meal Discovery

- Search: Users can search for meals by name, category, or ingredients.
- Meal Details: Detailed information about each meal includes ingredients, preparation steps, and nutritional information.
- Integration with TheMealDB API: The app fetches meal data from the TheMealDB API, ensuring a comprehensive and up-to-date database of meals.

Favourite Meals Management

- Add to Favourites: Users can add meals to their favourites list for quick access.
- View Favourites: A dedicated section where users can view and manage their favourite meals.
- Remove from Favourites: Users can easily remove meals from their favourites list.

Responsive Design

- Responsiveness: The programme is designed to work flawlessly on PCs, tablets, and mobile phones.
- Utility-First CSS: Tailwind CSS's utility-first design methodology guarantees quick user interface development.

API Integration: The app fetches Meal data via the TheMealDB API, guaranteeing an extensive and current database.

Socialise Function

- Like and Dislike: Users can like or dislike meals, contributing to the community's ratings and helping others discover popular meals.

3. Requirements

1. **Functional Requirements**
    a. User Registration and Authentication: Users must be able to register for an account, log in, and log out.
    b. Meal Search: Users must be able to search for meals by name, ingredient, and category.
    c. Meal Details: Users must be able to view detailed information about each meal, including ingredients and preparation instructions.
    d. Favourite Meals Management: Users must be able to add meals to their favourite list, view their favourite meals, and remove meals from their favourite list.
    e. Responsive Interface: The application must provide a responsive interface that works seamlessly across desktops, tablets, and mobile devices.
    f. Socialise Function: The social function allows authenticated users to like or dislike meals, view and manage their liked or disliked meals, and see the total like and dislike counts for each meal, enhancing user interaction and personalisation.
2. **Non-Functional Requirements**
    a. Performance: The application should load quickly and provide a smooth user experience, even on slower networks.
    b. Usability: The application should be intuitive and easy to use, with a clean and user-friendly interface.
    c. Maintainability: The codebase should be well-organized and documented to facilitate easy maintenance and future development.

4. Design Architecture

1. **Front-end architecture**

MealCards' front end is constructed on the progressive JavaScript framework Vue.js. Component-based design of the front end encourages maintainability and reusability. Important elements of the front-end design consist of:

- Component Structure:
    - App.vue: The root component of the application, serving as a container for other components and providing the basic layout.
    - Navbar.vue: The navigation bar component links to different application parts.
    - MealItem.vue: Component for displaying individual meal items in a card format.
    - FavoriteMeals.vue: Component for managing and displaying the user's favourite meals.
    - Login.vue: Component for user login functionality.
    - Register.vue: Component for user registration functionality.
    - Notification.vue: Component for displaying messages to the user.
    - PaginateControl.vue: Component for pagination controls.
    - Footer.vue: Footer component for the application.
    - DefaultLayout.vue: Layout component for authenticated users.
    - UnauthenticatedLayout.vue: Layout component for unauthenticated users.
    - YoutubeButton.vue: Component for linking to YouTube videos related to meals.
- Views:
    - Home.vue: The landing page of the application displaying a welcome message and featured meals.
    - Ingredients.vue: Displays a list of ingredients available in the database.
    - MealDetails.vue: Shows detailed information about a selected meal, including ingredients and preparation instructions.
    - SearchByIngredient.vue: Allows users to search for meals by ingredient.
    - SearchByLetter.vue: Allows users to search for meals by the first letter.
    - SearchByName.vue: Allows users to search for meals by name.
- Routing:
    - router/index.js: Manages navigation within the app, enabling smooth transitions between views such as Home, Meal Details, Login, and Favorites.
- State Management:
    - Vuex: Employed for state management, particularly for handling user authentication status and favourite meals.

    - store/index.js: Initializes the Vuex store and combines various modules like actions, getters, mutations, and state management for the application.
- API Integration:
    - axiosClient.js: Manages API requests to the TheMealDB API and backend services.

2. **Back-End Architecture**

With MySQL as the database management system, the MealCards application's back end is developed in PHP. The back-end architecture is made to manage user authentication, data storage, and API endpoints for meal management, user preferences, and socialising.

API Endpoints

- User Management:
    - api_register.php: Handles user registration by creating new user accounts in the database.
    - api_user.php: Manages user authentication and retrieves user information from the database.
- Meals Management:
    - favorites.php: Manages user favourite meals, allowing users to add, view, and remove meals from their favourites.
    - vote.php: Manages user interactions with meals, allowing users to like or dislike meals and view the total counts of likes and dislikes.

Database

- MySQL:
    - Stores user data, meal data, user favourites, and social interaction data (likes/dislikes).
    - Tables include users, meals, favorites, and likes.
- Database Connection:
    - db.php: Manages the connection to the MySQL database using PDO, ensuring secure and efficient communication.

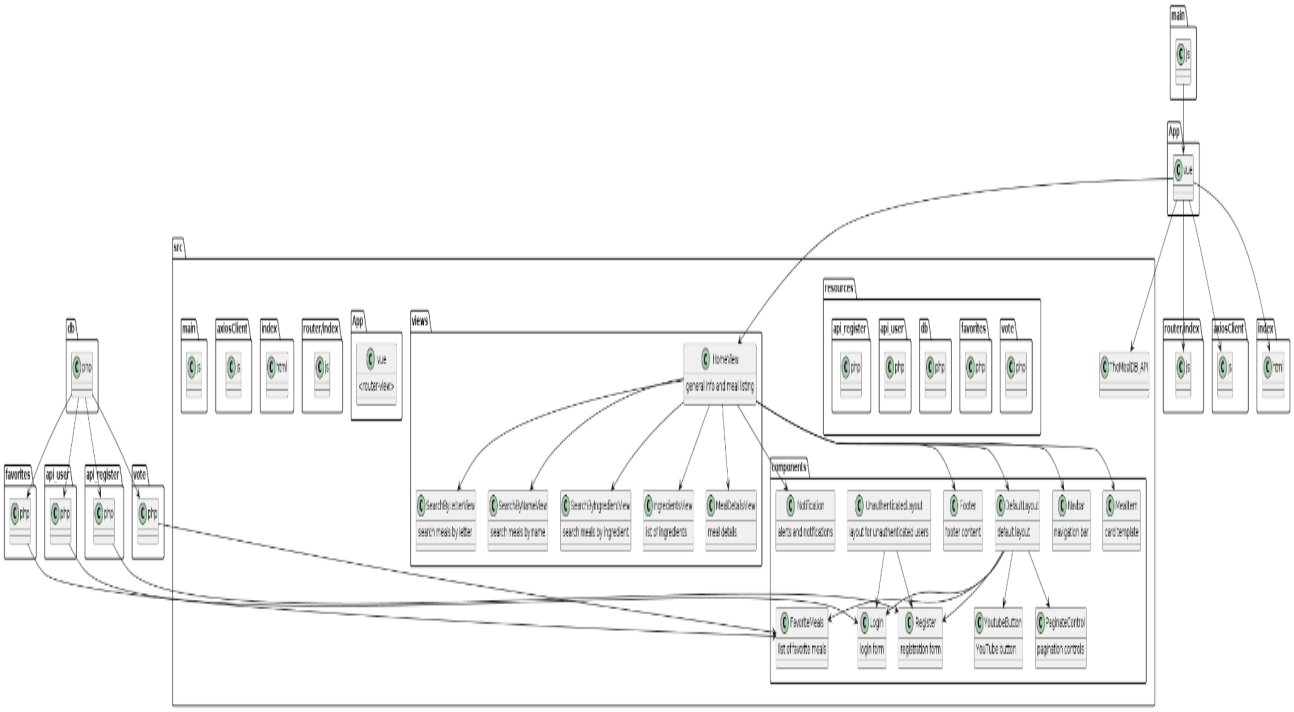
Server Configuration

- Environment Configuration:

- .env: File for environment-specific variables and configuration settings (e.g., database credentials).

- Server-Side Validation:
  - Custom validation scripts within each PHP file ensure data integrity and security by validating user inputs before processing requests.

Design Diagram



https://github.com/Walts03/cos30043/blob/main/6.2D-HD/ArchitectureDiagram.png

5. Technical Aspects

The MealCards application uses a component-based architecture, promoting code reusability and maintainability. The main technical aspects of the project include:

- Routing: Vue Router manages navigation within the app, enabling smooth transitions between views such as home, meal details, and user login.
- State Management: Vuex is employed for state management, particularly for handling user authentication status and favourite meals.
- API Integration: The MealDB API is integrated to fetch meal data. Axios is used to make HTTP requests to the API.
- Form Validation: User input forms are validated using Vue's built-in validation directives to ensure data integrity and provide user feedback.

6. Innovative Features

The MealCards project incorporates several innovative features designed to enhance the user experience:

- Advanced Search and Filtering: Users can search for meals by various criteria such as name, ingredient, and category, providing a highly customisable search experience.
- Dynamic Data Handling: Arrays and computed properties are utilised to dynamically handle and display data, ensuring a seamless and interactive user experience.
- Accessibility and Responsiveness: The application adheres to a mobile-first approach, ensuring accessibility and usability across various devices. Accessibility features are incorporated to cater to users with disabilities.
- Favourite Function: Users can add meals to their favourites for quick access. This feature allows users to manage their list of favourite meals, including adding, viewing, and removing meals from their favourite list.
- Socialise Function: The socialise function allows users to like or dislike meals. This adds an interactive element to the application, enabling users to express their preferences and see the popularity of different meals through total like and dislike counts.

7. Project Development

The development process involved several stages: planning, design, implementation, and testing. The primary challenge was integrating the TheMealDB API and handling asynchronous data fetching efficiently. This was addressed by leveraging Vuex for state management and implementing loading indicators to enhance user experience.

8. Challenges and Solution

- API Rate Limiting: Handling API rate limits from the TheMealDB API posed a challenge. To mitigate this, caching mechanisms were implemented to store frequently accessed meal data, thereby reducing the number of API requests and preventing rate limit issues.
- Authentication: Managing user authentication securely was critical. The solution involved using PHP sessions and tokens to handle user sessions securely. Authentication was validated on each API request to maintain security.

- Responsive Design: Ensuring the application was responsive and worked on various devices was a significant challenge. This was achieved by utilising CSS frameworks like Bootstrap and Tailwind CSS, which facilitated the creation of a responsive layout that adapted to different screen sizes.

9. Technology Used

1. Vue.js: The core JavaScript framework used for building the user interface.
2. Vue Router: This is used to manage navigation and routing within the application.
3. Vuex: State management library used to manage the application's state.
4. Axios: A promise-based HTTPs client is used to make requests to the TheMealDB API.
5. Bootstrap: CSS framework used for responsive design and styling.
6. JavaScript: The primary programming language used for development.
7. PHP: Backend language used to manage user authentication and interaction with the MySQL database.
8. MySQL: Database management system used to store user data and favourite meals.

10. Project File Structure

Root Directory

- .env: Environment variables for configuration.
- eslint.config.js: Configuration file for ESLint.
- index.html: The main HTML file.
- package.json: Contains metadata about the project, including dependencies, scripts, and configuration details.
- package-lock.json: Automatically generated file that describes the exact tree that was generated.
- postcss.config.js: Configuration file for PostCSS.
- tailwind.config.js: Configuration file for Tailwind CSS.
- vite.config.js: Configuration file for Vite.

public Directory

- logo.png: Static assets.

src Directory

This is the leading directory for the application's source code.

- App.vue: The root component of the application, serving as a container for other components and providing the basic layout.
- axiosClient.js: Axios configuration for making API calls.
- main.js: The application's entry point where the Vue instance is created, and the main App component is mounted to the DOM.
- style.css: Global styles.

assets Directory

- bg.svg: Background image.
- logo.png: Logo image.

components Directory

Contains reusable Vue components that are used throughout the application.

- DefaultLayout.vue: The default layout component.
- FavoriteMeals.vue: Manages and displays the user's favourite meals.
- Footer.vue: Contains footer content and links.
- Login.vue: Handles user login functionality.
- MealItem.vue: Displays individual meal items.
- Meals.vue: Displays a list of meals fetched from the TheMealDB API.
- Navbar.vue: Defines the navigation bar and branding for the application.
- Notification.vue: Displays notifications to the user.
- PaginateControl.vue: Pagination control component.
- Register.vue: Handles user registration functionality.

- UnauthenticatedLayout.vue: Layout for unauthenticated users.
- YoutubeButton.vue: Button component for linking to YouTube.

resources Directory

Contains PHP files for backend functionality.

- api_register.php: API endpoint for user registration.
- api_user.php: API endpoint for user information.
- favorites.php: API endpoint for managing favourite meals.
- vote.php: API endpoint for managing social interactions, allowing users to like or dislike meals and view the total counts of likes and dislikes.

router Directory

Contains the configuration for Vue Router.

- index.js: Defines the routes for the application, mapping paths to components

store Directory

Houses the Vuex store files for state management.

- actions.js: Vuex actions for dispatching mutations.
- favoriteService.js: Service for handling favorite meals in the Vuex store.
- getters.js: Vuex getters for retrieving state.
- index.js: Initializes the Vuex store and combines various modules.
- mutations.js: Vuex mutations for modifying state.
- socialService.js: Service for handing social interactions in the Vuex store

views Directory

Includes higher-level components representing different views or pages in the application.

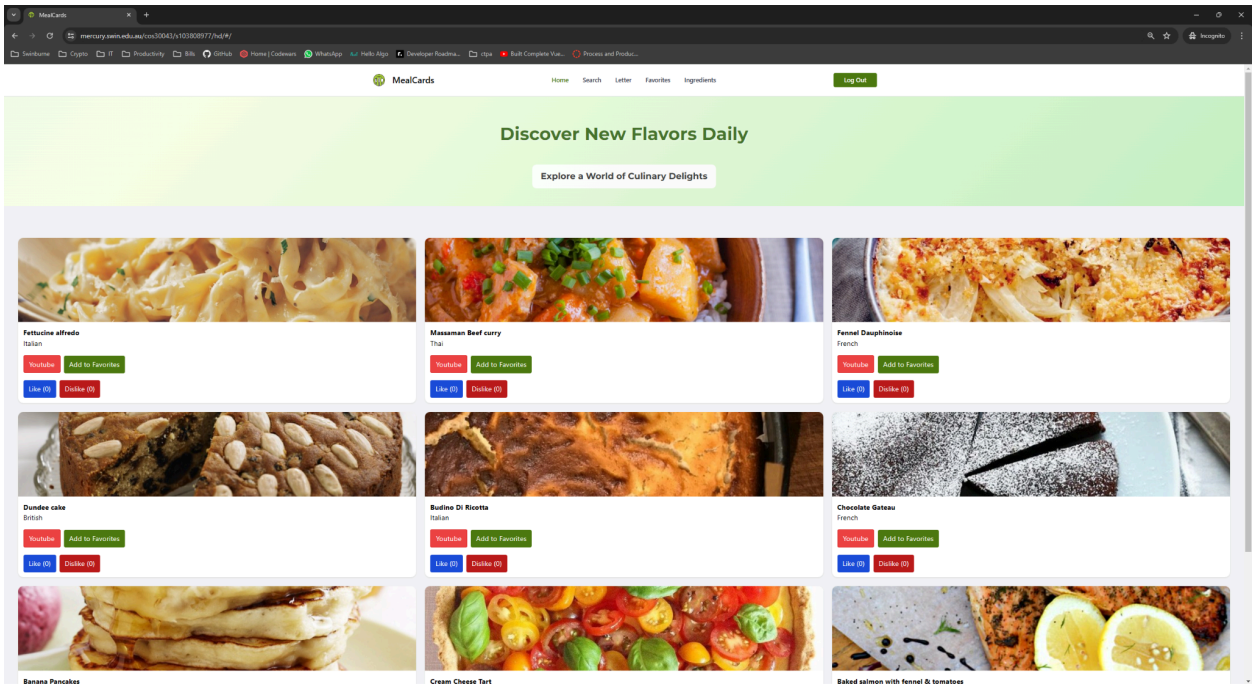
- Home.vue: The landing page of the application displaying a welcome message and featured meals.
- Ingredients.vue: Displays a list of ingredients.

- MealDetails.vue: Shows detailed information about a selected meal, including ingredients and preparation instructions.
- SearchByIngredient.vue: Allows users to search meals by ingredient.
- SearchByLetter.vue: Allows users to search meals by the first letter.
- SearchByName.vue: Allows users to search meals by name.

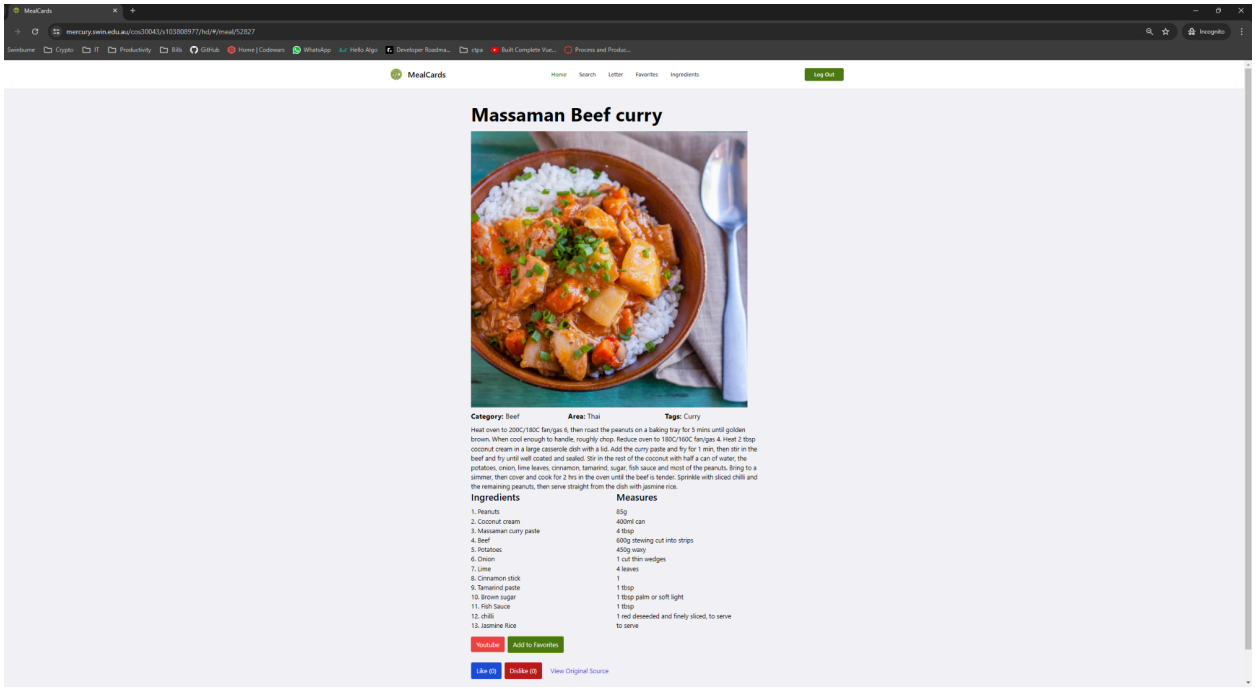
11. Appendix

- Github: https://github.com/Walts03/cos30043/tree/main/6.2D-HD
- Link to website (mercury): https://mercury.swin.edu.au/cos30043/s103808977/hd/
- Videos: https://youtu.be/rO6EcH7CinQ
- Project Draft: Project Draft/Plan for MealCards

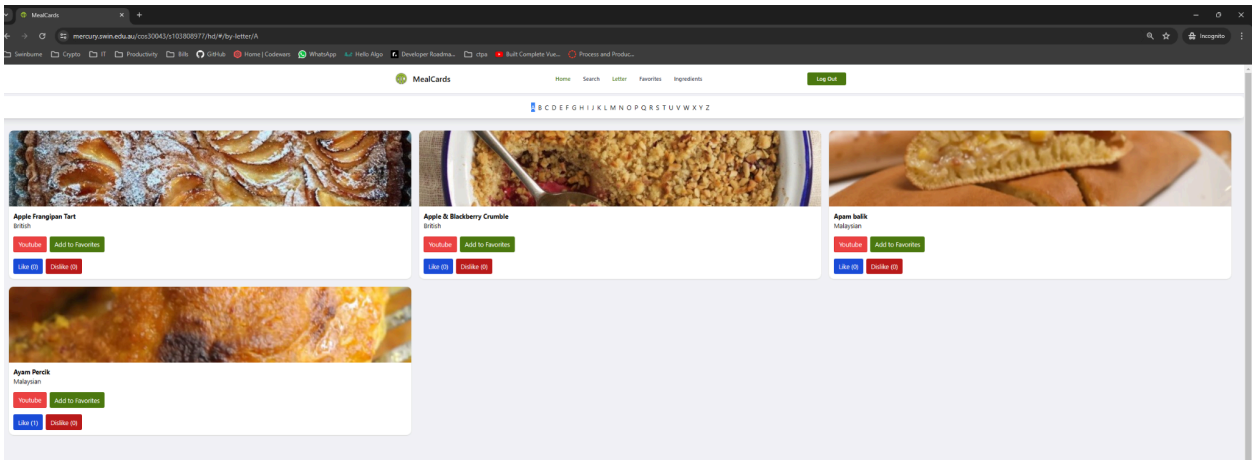
12. Screenshot of the web app



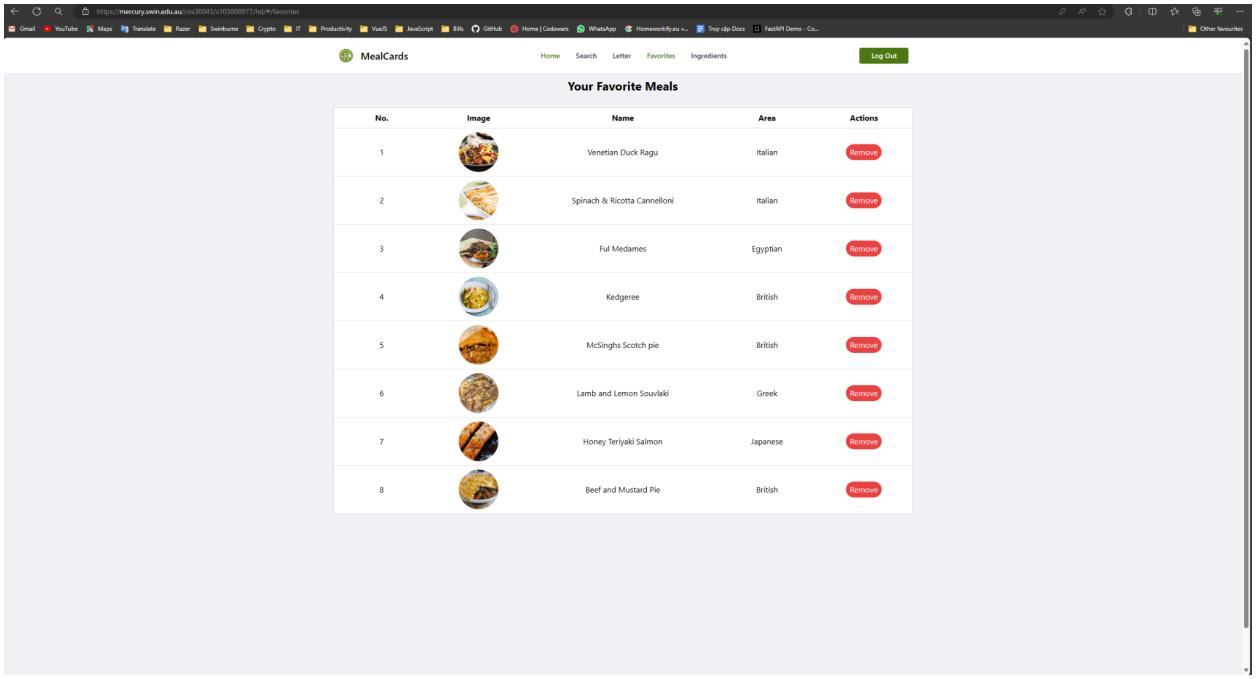
Homepage



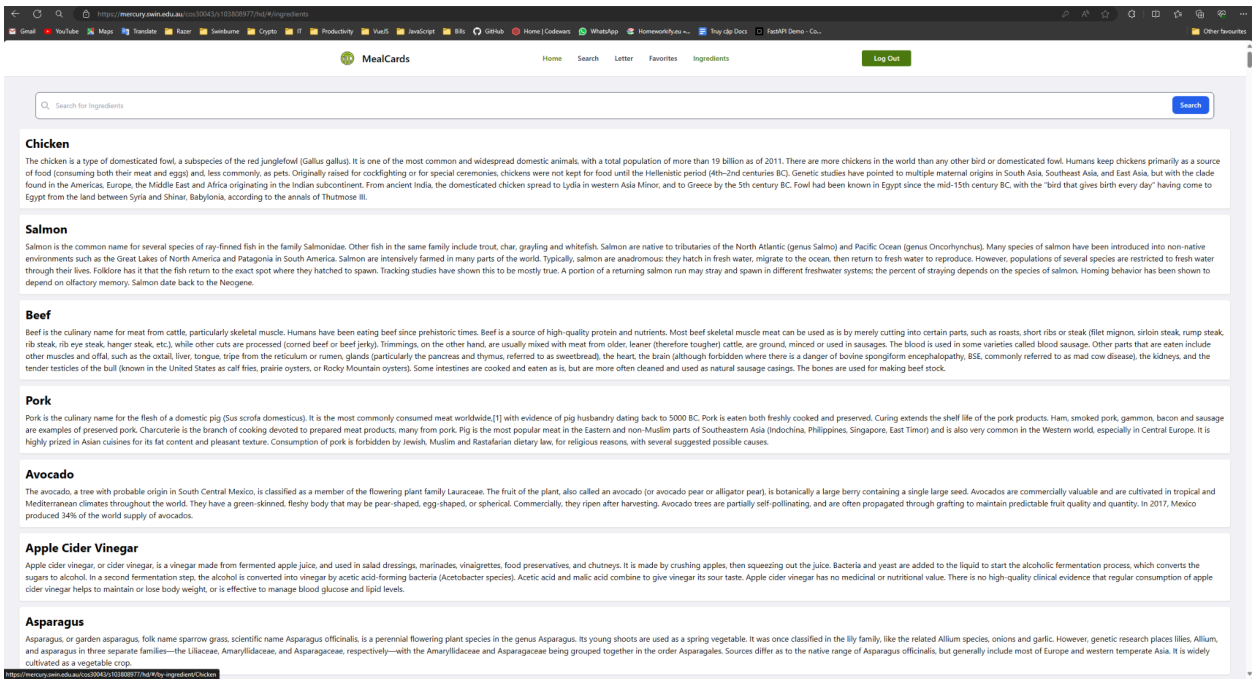
Meal Detail page



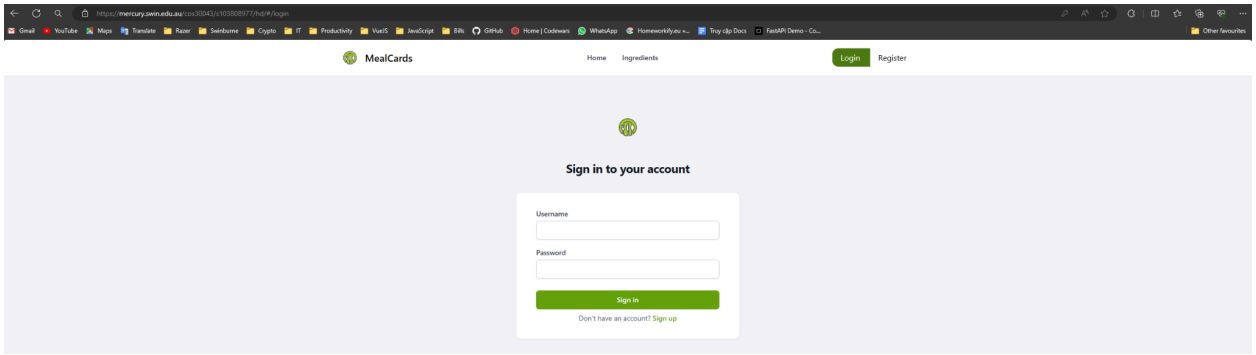
Search Meal by Letter

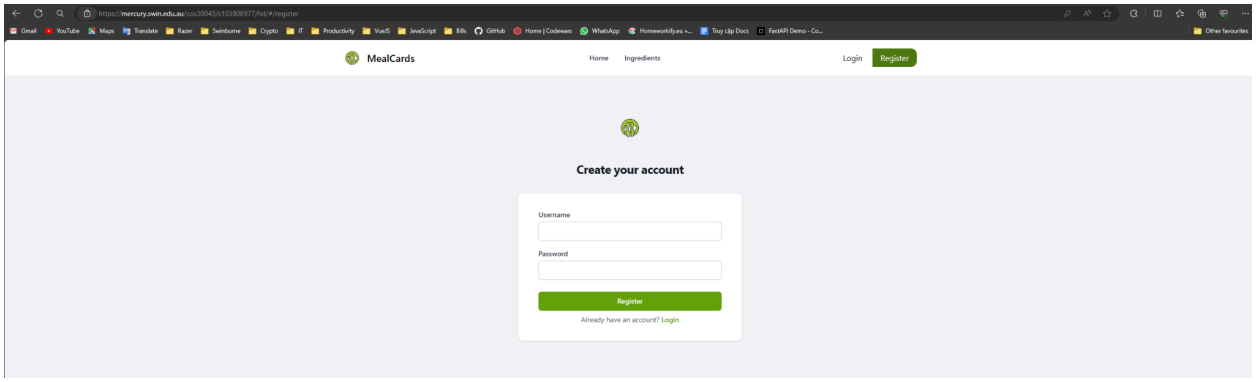


Store user's favourite meals



Search by Ingredients





Login/Register pages

Reference

- Axios 2023, 'Getting Started | Axios Docs', *axios-http.com*, viewed <https://axios-http.com/docs/intro>.
- 'Free Meal API | TheMealDB.com' n.d., *www.themealdb.com*, viewed <https://www.themealdb.com/api.php>.
- 'Getting Started | Vue Router' n.d., *router.vuejs.org*, viewed <https://router.vuejs.org/guide/>.
- Vue 2022, 'Introduction | Vue.js', *vuejs.org*, viewed <https://vuejs.org/guide/introduction.html>.
- 'What is Vuex? | Vuex' n.d., *vuex.vuejs.org*, viewed <https://vuex.vuejs.org/>.