aDMe Assignment

October 2024

This doc consists of 1 assignment question and the instructions to do and submit it.

How to Submit

Once done :

- Create a GitHub repository (don't use **aDMe, Advertyzement, Truflect, Zaka Network** in the repo name or anywhere in repo).
- Share the link with us
- This repository should contain a readme file describing the method you have used to solve the problem.
- Time taken to complete this assignment.

Let us know if you have any doubts regarding the assignment.

Note :

- These assignments are designed purely for the evaluations purpose and any part of your submission won't be used in the actual product we are working on.
- Using the advertyzement, Adme, truflect, Zaka Network or any related name in the repo or in the code will lead to direct rejection.

Assignment

Food Product Explorer

Objective:

Develop a scalable and performant web application that allows users to search, filter, and view detailed information about food products using the **OpenFoodFacts API**. The project should include advanced features like efficient API handling, state management, and modern UI components.

🚀 Scope & Features

1️⃣ Homepage - Product Listing

- Fetch and display food products from the **OpenFoodFacts API**.
- Each product should show:
    - Product Name
    - Image
    - Category
    - Ingredients (if available)
    - Nutrition Grade (A, B, C, D, E)
- Implement **pagination** using infinite scroll or a "Load More" button.
- Implement **skeleton loaders** for a smooth UI experience.

2️⃣ Search Functionality

- Add a **search bar** to filter food products by name.

- Implement **debounced API requests** to prevent excessive API calls.

3️⃣ Barcode Search

- Users should be able to search for products by barcode.
- Implement **error handling** for invalid or unknown barcodes.

4️⃣ Advanced Filtering & Sorting

- **Filter by category** (e.g., Beverages, Dairy, Snacks).
- Allow **multi-category selection** instead of a single dropdown.
- Implement **range-based filtering** (e.g., filter products with sugar < 10g per serving).
- Allow sorting by:
    - Product Name (A-Z, Z-A)
    - Nutrition Grade (Ascending/Descending)
    - Lowest/Highest Calories

5️⃣ Product Detail Page

When a user clicks on a product, navigate to a detailed view showing:

- **Product Image**
- **Complete list of Ingredients**
- **Nutritional Values** (Energy, Fat, Carbs, Proteins, etc.)
- **Labels & Tags** (Vegan, Gluten-Free, Organic, etc.)

6️⃣ UI/UX & Design

- Fully **responsive** and mobile-friendly UI.

- Implement **dark mode support**.
- Use **Material UI, TailwindCSS, or ShadCN** for modern styling.

7️⃣ Performance & API Optimization

- Use **React Query** or **Redux Toolkit Query** for API state management.
- Implement **caching and lazy loading**.
- Ensure **error handling** for API failures.

8️⃣ Deployment & DevOps

- Deploy the project using **Vercel, Netlify, or AWS**.
- Set up **CI/CD with GitHub Actions** to automate testing and deployment.

📌 API Documentation

🔗 **Base URL:** `https://world.openfoodfacts.org/`

Get Products by Category

📌 `https://world.openfoodfacts.org/category/{category}.json`

Search Products by Name

📌 `https://world.openfoodfacts.org/cgi/search.pl?search_terms={name}&json=true`

Get Product by Barcode

📌 `https://world.openfoodfacts.org/api/v0/product/{barcode}.json`

Example Query:
📌 https://world.openfoodfacts.org/api/v0/product/737628064502.json

Note : https://world.openfoodfacts.org/ is an external website maintained by open food facts (an french non profit organization), if server not responding then wait for some time before trying again

---

💡 Evaluation Criteria

📍 **Code Quality**
✅ Clean, modular, and readable code
✅ Component-based architecture in React

📍 **API Integration**
✅ Efficient API usage with proper error handling
✅ Optimized API calls with caching and debouncing

📍 **UI/UX**
✅ A professional, user-friendly, and responsive design

📍 **Functionality**
✅ Correct implementation of search, filtering, sorting
✅ Pagination should be smooth and performant

📍 **Performance**
✅ Efficient state management using **Redux Toolkit / React Query**
✅ API request optimization using **lazy loading & caching**

📍 **Bonus**
✅ Deployment with **CI/CD pipelines**