Documentation

Index.mjs script is a serverless function (AWS Lambda) written in JavaScript (Node.js) for handling barcode lookup requests. It fetches product information from various APIs, processes the data, and stores it in an AWS DynamoDB database. The function is triggered by HTTP requests, with parameters such as barcode, latitude, longitude, and userId. It responds with product information in JSON format. It also processes the analysis.

# Documentation - Admin

**- Programs Used:**

- AWS Lambda: The serverless function is deployed and executed on AWS Lambda, allowing for scalable and cost-effective execution.

- AWS DynamoDB: The NoSQL database provided by AWS is utilized for storing product data and transaction logs.

- AWS API Gateway: Facilitates the HTTP endpoint for triggering the Lambda function.

- AWS CloudWatch: Used for logging, monitoring, and debugging Lambda function executions.

- axios: A JavaScript library used for making HTTP requests to external APIs.

**- Storage Locations:**

- Lambda Function Code: The Lambda function code is stored within the AWS Lambda service.

- DynamoDB Tables:

- Product Data: Product information is stored in a DynamoDB table named "fsg".

- Transaction Logs: Request details are stored in a separate DynamoDB table named "logs".

**- Transaction Log:**

- The transaction log records details of each request, including barcode, latitude, longitude, userId, timestamp, and timezone.

- It provides an audit trail for all barcode lookup transactions, aiding in troubleshooting and analysis.

- The log data can be accessed and queried directly from the DynamoDB table "logs" or through AWS CloudWatch Logs.

**- Links:**

- AWS Management Console: [AWS Console](https://aws.amazon.com/console/)

- AWS DynamoDB: [DynamoDB Console](https://console.aws.amazon.com/dynamodb/)

- AWS Lambda: [Lambda Console](https://console.aws.amazon.com/lambda/)

- AWS API Gateway: [API Gateway Console](https://console.aws.amazon.com/apigateway/)

- AWS CloudWatch: [CloudWatch Console](https://console.aws.amazon.com/cloudwatch/)

# Documentation - Code

**Overview:**

The code is a Node.js application deployed as an AWS Lambda function. It serves as a barcode lookup service, retrieving product information from various external APIs and storing it in AWS DynamoDB. The architecture includes handling HTTP requests, fetching data from APIs, processing and formatting the data, and interacting with DynamoDB for storage. It also processes the analysis.

**Functionality:**

1. HTTP Request Handling: The Lambda function is triggered by HTTP requests via AWS API Gateway. Request parameters include barcode, latitude, longitude, and userId.

2. Data Retrieval: Product data is fetched from multiple external APIs including Open Food Facts, UPC, Edamam, USDA, and Nutritionix using Axios.

3. Data Processing: Fetched data is processed, merged, and formatted according to predefined criteria. This includes extracting relevant fields, calculating percentages, sorting, and cleaning up data.

4. Database Interaction: Processed data is stored in DynamoDB tables. A separate table is used for transaction logs to maintain an audit trail of requests.

5. Error Handling: The code includes error handling mechanisms for various scenarios, logging errors to AWS CloudWatch, and returning appropriate HTTP response codes.

**Components:**

- Dependencies:

- axios: For making HTTP requests to external APIs.

- aws-sdk: AWS SDK for interacting with DynamoDB.

- geo-tz: For determining timezone based on latitude and longitude.

- Main Functions:

- `handler`: Main Lambda function handling the barcode lookup.

- `getTimezone`: Retrieves timezone based on latitude and longitude.

- `getFromDynamoDB`: Retrieves data from DynamoDB.

- `uploadToDynamoDB`: Uploads data to DynamoDB.

- `uploadToLogsTable`: Uploads request details to the logs DynamoDB table.

- `fetchDataAndProcess`: Fetches product data from external APIs, processes it, and returns formatted data.

- External APIs:

- `getProductByBarcode`: Fetches product data from Open Food Facts API based on barcode.

- `getProductByUPC`: Fetches product data from UPC API based on barcode.

- `getProductByEdamam`, `USDA\_searchFoodByName`, `getProductByNutritionix`: Fetches product data from Edamam, USDA, and Nutritionix APIs respectively.

**Databases and Protocols:**

- DynamoDB: NoSQL database service provided by AWS for storing product data and transaction logs. Utilized for its scalability, flexibility, and low latency.

- HTTP Protocol: HTTP requests are handled via AWS API Gateway, triggering the Lambda function. External API interactions are also performed using HTTP.

**Error Handling:**

- Error handling mechanisms are implemented throughout the code to handle various scenarios such as missing parameters, failed API requests, and database errors.

- Errors are logged to AWS CloudWatch for monitoring and debugging purposes.

- HTTP response codes are returned accordingly to indicate success or failure of the request.

# Documentation - Analysis Logic

**Data Gathering:**

The `processApiResponseToLabels` function is responsible for analyzing the response data from external APIs and extracting meaningful information to categorize products based on various attributes such as allergens, dietary preferences, religious restrictions, sustainability choices, packaging, food ratings, and country of origin.

**Classification Process:**

1. Data Availability Check:

- The function begins by checking if the response data is available. If not, it throws an error indicating missing data.

2. Attribute Analysis:

- The function then proceeds to analyze the product data based on the presence of key attributes such as ingredients, tags, keywords, traces, allergens, and nutritional information.

3. Conditional Classification:

- Based on the presence of specific attributes and their values, the function categorizes products into various labels representing different characteristics.

4. Allergens and Lifestyle Choices:

- It checks for the presence of allergens like celery, gluten-containing cereals, crustaceans, eggs, fish, milk, etc., and categorizes products accordingly.

- Lifestyle choices such as vegan, lacto-vegetarian, ovo-vegetarian, pescatarian, etc., are determined based on the absence or presence of animal-derived ingredients.

5. Religious Restrictions:

- Religious dietary restrictions like halal, kosher, beef, Jain, and onion restrictions are evaluated based on the presence or absence of specific ingredients.

6. Dietary Preferences:

- Dietary choices such as keto, paleo, Mediterranean, and sugar-free are determined based on the nutritional content and ingredient composition.

7. Sustainability and Packaging:

- Sustainability choices like local, organic, and genetically modified, as well as packaging characteristics like fully recycled and part recycled, are assessed based on product descriptions and labels.

8. Food Ratings and Country of Origin:

- Food ratings (e.g., ABCDE ratings) and indicators of high sugar/salt content are extracted from nutritional data.

- The country of origin is determined based on available tags and information.