Packaged-Food-Rating-App

1. Project Overview

Problem Statement:

Consumers often struggle to assess the healthiness of packaged food products quickly. Nutritional labels can be complex, and different products use varying formats and terminologies for ingredients and nutrients. There is a need for a system that can automatically analyze packaged foods, standardize nutrition data, and provide an easy-to-understand health score.

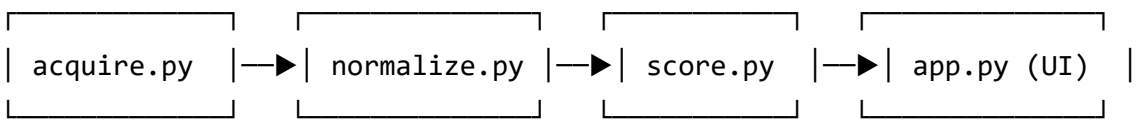
Solution:

This project provides a **Streamlit-based application** that allows users to analyze packaged food products using:

1. Barcode lookup
2. Product name search
3. Image-based OCR (extracting text from packaging)

The system normalizes the nutritional and ingredient data, computes a **Health Score** (0–100) using standardized criteria, and classifies the product into health bands (Healthy, Lightly Healthy, Moderate, Less Healthy, Unhealthy). It also provides a detailed explanation of score drivers and evidence.

2. High-Level Architecture



Module Roles:

- `acquire.py` – Fetches raw product data from OpenFoodFacts API (by barcode or name) or extracts text via OCR from image URLs.

- `normalize.py` – Standardizes ingredient names, nutrient names, and converts them into numerical values suitable for scoring.
- `score.py` – Calculates the health score using negative and positive points based on sugar, fat, salt, fiber, protein, and other optional factors.
- `app.py` – Streamlit frontend that integrates the above modules, provides the user interface, displays results, logs events, and handles session state.

3. Detailed Workflow

Step 1: Input Selection

- Users select one of the three input modes in the sidebar:
    1. Barcode
    2. Product Name
    3. Image URL
- The system prompts for the relevant input (e.g., barcode number, product name, or packaging image URL).

Step 2: Data Acquisition (`acquire.py`)

- **Barcode Lookup:** Fetch product JSON from OpenFoodFacts API.
- **Product Name Search:** Fetch top N product results and allow user selection.
- **Image URL OCR:** Download image, preprocess (grayscale, sharpen, contrast enhancement, thresholding), extract text using Tesseract OCR.

Output:

- Raw product data (JSON) or extracted text.

Step 3: Normalization (`normalize.py`)

- **Ingredients:**
    - Lowercase and remove punctuation.
    - Replace synonyms with standardized terms (e.g., “sucrose” → “sugar”).
    - Split ingredients by commas, semicolons, or newlines.
    - Remove duplicates.

- **Nutrients:**
    - Extract and standardize key nutrients: `energy_kcal`, `sugars_g`, `salt_g`, `fat_g`, `saturated_fat_g`, `fiber_g`, `proteins_g`.
    - Convert values to floats where applicable.

Output:

- Normalized ingredients list.
- Normalized nutrients dictionary.

Step 4: Health Score Calculation (`score.py`)

- **Negative Points:** Penalties for high sugar, fat, saturated fat, sodium, ultra-processed foods.
- **Positive Points:** Bonuses for fiber, protein, fruit/vegetable percentage.
- **Score Normalization:** Map raw points to 0–100 scale.
- **Band Assignment:** Classify into `Healthy`, `Lightly Healthy`, `Moderate`, `Less Healthy`, `Unhealthy`.

Output:

- Health score (`0–100`)
- Grade (`A–E`)
- Band (descriptive)
- Drivers (what contributed to score)
- Evidence (regulatory or threshold references)

Step 5: Display & Interaction (`app.py`)

- Show selected product image, normalized ingredients, and nutrients.
- Display health score in a colored panel.
- Provide expandable explanation panel for drivers and evidence.
- Maintain session state to support multi-step interactions.
- Log all events in `run.log`.

4. Sample Input & Output

Sample Input (Image OCR):


```
{
    "url": "https://c8.alamy.com/comp/BA63M6/nutritional-label-on-food-packaging-for-ready-salted-crisps-BA63M6.jpg"
}
```


Normalized Output:


```
{
  "product_name": "Ready Salted Crisps",
  "nutrients": {
    "energy_kcal": 133.0,
    "fat_g": 8.5,
    "salt_g": 0.24
  },
  "ingredients": ["potato", "vegetable oil", "salt"],
  "score": {
    "value": 50,
    "band": "Moderate",
    "grade": "C"
  },
  "explanation": {
    "drivers": ["High sodium: 24.0 g salt/100g"],
    "evidence": ["Sodium > 900 mg/100g"]
  }
}
```


Sample Trace (`trace.txt`):

```
[INGEST] Barcode lookup: 3017620429484 → Found: Nutella
[NORMALIZE] Product ingredients: ['sugar', 'palm oil', 'milk', 'hazelnuts']
[NORMALIZE] Product nutrients: {'energy_kcal': 539, 'sugars_g': 56.3, 'salt_g': 0.107, 'fat_g': 30.9, 'saturated_fat_g': 10.6, 'fiber_g': 0, 'proteins_g': 6.3}
[SCORE] Calculation started
```

```
[SCORE] Result: 34, Band: Unhealthy, Grade: E
[EXPLAIN] Drivers: ['High sugar content: 56.3 g/100g', 'High saturated fat:
10.6 g/100g']
[EXPLAIN] Evidence: ['Sugars > 45 g/100g', 'Saturated fat > 10 g/100g']
```

5. Installation & Setup

1. **Clone the repository**

   ```
   git clone https://github.com/yourusername/packaged-food-rating-app.git
   cd packaged-food-rating-app
   ```

2. **Create virtual environment & install dependencies**

   ```
   python -m venv venv
   source venv/bin/activate   # Linux/Mac
   venv\Scripts\activate      # Windows
   pip install -r requirements.txt
   ```

3. **Run the application**

   ```
   streamlit run app.py
   ```

6. Output Paths

- Streamlit UI: Health score and explanations appear on the webpage.
- Logs: Saved at `run.log` in the repository root.
- Trace: `trace_example.txt` in `samples/outputs/`.
- Sample JSON outputs: `samples/outputs/`.

Refreshing Cached Records:

- Delete cache if implemented (`cache/` folder).
- Restart app: `streamlit run app.py`.

7. Config Template

`config_template.json`:

```
{
  "tesseract_path": "D:/Tesseract/tesseract.exe",
```

```
  "openfoodfacts_url": "https://world.openfoodfacts.org/api/v0/product",
  "ocr_languages": ["eng"]
}
```

- Copy this template, rename as `config.json`, and update paths as per your environment.

8. Scoring Design (Summary)

- **Score Range:** 0 – 100
- **Bands:**
  - 80–100 → Healthy
  - 65–79 → Lightly Healthy
  - 50–64 → Moderate
  - 35–49 → Less Healthy
  - 0–34 → Unhealthy
- **Negative Points:** Added for energy, sugar, saturated fat, sodium, ultra-processed ingredients.
- **Positive Points:** Added for fiber, protein, fruit/vegetable content.
- **Normalization:** Raw points mapped to 0–100 scale to provide a consumer-friendly health index.

9. References

- World Health Organization. *Guidelines on Sugars Intake for Adults and Children*. 2015.
- FSSAI. *Food Safety and Standards Regulations*. 2023.
- OpenFoodFacts API Documentation. *https://world.openfoodfacts.org/data*

Conclusion

The **Packaged-Food-Rating-App** provides an automated, reliable, and reproducible way to evaluate packaged foods based on standardized nutritional data. Users can obtain a health score and actionable insights without manually interpreting complex labels. The system supports multiple input modes, offers normalized outputs, and provides detailed explanations for transparency, making it a valuable tool for informed dietary choices.