Grocery Sustainability Scoring API – Integration Guide

1. Setup & Requirements

- Python Packages Needed:
    - `pandas`
    - `scikit-learn`
    - `google-generativeai`
- Data File:
    - Place `foodemissions.xlsx` in the project root.
    - Ensure the sheet is named `ES` and columns include:
        - `Food product`
        - `Total kg CO2-eq/kg`
        - `Agriculture`, `iLUC`, `Food processing`, `Packaging`, `Transport`, `Retail`
- API Key:
    - Set your Gemini API key as an environment variable:
    - bash

```
export GEMINI_API_KEY="your_actual_key_here"
```

-

2. How to Use the Function

Function:
`get_sustainability_score(food_item: str) -> dict`
Input:

- `food_item` (string): Name of the food product to check.

Output:
A dictionary with:

- `score`: `"High"`, `"Medium"`, or `"Low"`
- `rationale`: Short bullet-point explanation (from Gemini)
- `components`: Dict of normalized emission components
- `total_emissions`: Raw CO2-eq/kg value
- `error`: (optional) Only present if item not found

Example Call:
python

```
result = get_sustainability_score("Tomato")
print(result)
```

Example Output:
json

```
{
  "score": "High",
  "rationale": "- Tomatoes have low agricultural emissions
compared to animal products.\n- Minimal packaging and transport
emissions.\n- Total CO2-eq/kg is well below the average for
common foods.",
  "components": {
    "Agriculture": 0.12,
    "iLUC": 0.05,
    "Food processing": 0.08,
    "Packaging": 0.07,
    "Transport": 0.10,
    "Retail": 0.03
  },
  "total_emissions": 1.2
}
```

3. Error Handling

- If the food item is not found:
- json

```
{ "error": "Item not found" }
```

-

4. Performance & Caching

- First request for a new food: ~2–3 seconds (Gemini API call)
- Repeated requests: Instant (cache)
- Rate limit: 1 request every 2 seconds (due to API)