**1. 数据来源 (Data Sources)**

* **RecipeNLG (full\_dataset.csv)**
  + 来源: [https://recipenlg.cs.put.poznan.pl/](https://recipenlg.cs.put.poznan.pl/?utm_source=chatgpt.com)
  + 内容: ~2M 条食谱，包含字段 title, ingredients, directions, source, link, NER。
* **OpenNutrition (opennutrition\_foods.tsv)**
  + 来源: [https://www.opennutrition.app](https://www.opennutrition.app?utm_source=chatgpt.com)
  + 内容: ~300k 食品营养数据，包含字段 id, name, alternate\_names, description, nutrition\_100g, ingredients, labels。

**2. 数据预处理 (Pre-processing)**

* 将 opennutrition\_foods.tsv 转换为 CSV 格式，方便批量上传。
* 在 full\_dataset.csv 前新增 recipe\_id 字段（从 0 递增），保证每条食谱有唯一主键。
* 统一使用 UTF-8 编码，避免 DynamoDB 导入时出现解析错误。

**3. 存储位置 (Storage)**

* **Amazon S3 (Simple Storage Service)**
  + Bucket 名称: tp33-data-recipe
  + 用途: 存放原始数据文件（CSV/TSV），作为 DynamoDB 导入的来源，同时作为数据备份。

**4. 数据库架构 (Database Architecture)**

**4.1 DynamoDB 表 – Recipes\_v2**

* **主键 (Primary Key)**: recipe\_id (String)
* **二级索引 (GSI)**: gsi\_title\_prefix
  + PK: title\_lc\_first1（标题首字母）
  + SK: title\_lc（标题小写）
  + 用途: 实现按标题前缀的搜索
* **字段**:
  + recipe\_id, title, title\_lc, title\_lc\_first1, ingredients, directions, link, source, NER
* **用途**: 存储并查询所有菜谱信息

**4.2 DynamoDB 表 – Foods**

* **主键 (Primary Key)**: id (String)
* **二级索引 (GSI)**: gsi\_name\_prefix
  + PK: name\_lc\_first1
  + SK: name\_lc
  + 用途: 按食材前缀进行搜索
* **字段**:
  + id, name, name\_lc, name\_lc\_first1, alternate\_names, description, nutrition\_100g, ingredients, labels, serving, source
* **用途**: 存储食材与对应的营养信息

**5. 数据导入 (Data Ingestion)**

* 使用 AWS Console 的 **DynamoDB → Import from S3** 功能，将 S3 中的 CSV 文件直接导入 DynamoDB。
* Recipes 数据量较大（~2M 条），分批导入并验证数据完整性。
* Foods 数据量 (~300k 条)，单次导入即可。
* 成功导入后，通过 DynamoDB Console & AWS CLI 验证：

aws dynamodb get-item --table-name Recipes\_v2 --key '{"recipe\_id":{"S":"0"}}'

**6. 数据访问 (Data Access via APIs)**

* **AWS Lambda**
  + 创建函数：recipes-api, foods-api, nutrition-match-api
  + 运行时环境：Node.js 22.x
  + 配置 IAM Role（如 tp33-recipes-api-role），附加权限：
    - AmazonDynamoDBReadOnlyAccess
    - AWSLambdaBasicExecutionRole
* **API Gateway**
  + 为每个 Lambda 创建对应的 RESTful API：
    - GET /recipes?recipe\_id=... → 查询单个菜谱
    - GET /recipes?title\_prefix=... → 按标题前缀搜索
    - GET /foods?id=... → 查询单个食材营养
    - GET /foods?name\_prefix=... → 食材前缀搜索
    - POST /match → 根据输入的食材数组，匹配 Foods 表并汇总营养

**7. 数据增强 (Data Enhancement – Iteration 1)**

* 在 Lambda (recipes-api) 中增加标签逻辑：
  + **Habits (饮食习惯)**: vegan, vegetarian, gluten\_free, dairy\_free, nut\_free, low\_sugar
  + **Categories (菜谱类型)**: breakfast, lunch, dinner, snack, dessert, soup, salad, drink
* 这些标签通过关键词匹配规则动态生成，不修改 DynamoDB 表结构。
* API 支持参数 habit 和 category，在 Lambda 内存中对结果过滤。

**8. 数据安全与访问控制 (Security & Access Control)**

* IAM 用户: tp33-admin，用于管理 DynamoDB & S3
* IAM 角色: tp33-recipes-api-role，用于 Lambda 函数访问 DynamoDB
* Access Key 与 Secret Key 保存在本地 ~/.aws/credentials，不硬编码进代码仓库
* API Gateway 开启 **CORS**，允许前端 Vue 应用访问

**9. 数据备份与恢复 (Backup & Recovery)**

* 原始 CSV 文件存放在 S3，可随时重新导入
* DynamoDB 自带 Point-in-Time Recovery (PITR) 功能，可恢复到最近 35 天内任意时间点（可选启用）

**10. 数据量控制 (Data Volume Management)**

* 由于原始 Recipes 数据集 (~2M) 对目标用户（60–70 岁老人）过大，不便前端交互：
  + Iteration 1 保留全部数据供演示
  + 后续 Iteration 将使用 Python 脚本在导入前打标签并抽样，生成小规模数据集 (~50k–100k)