ISSUE 1: Data Understanding, Dataset Preparation & Basic Recommendation System (FEB 12TH – FEB 17TH)

🎯 Objective

Set up data structure, prepare sample dataset, and build a rule-based food recommendation system.

Task 1: Understand Platform Data Requirements

Consumers

- `user_id`
- `budget`
- `location`
- `household_size`
- `food_preferences`

🏹 Food Items

- `food_id`
- `name`
- `category`
- `price`
- `quantity_available`
- `nutrition_tag`
- `seller_id`
- `location`

⚾ Farmers / Sellers

- `seller_id`
- `name`
- `location`
- `product_type`

🎁 Deliverable

- Document listing all required fields

Task 2: Define Food Categories & Items

Carbohydrates (Energy Foods)

- Rice
- Maize
- Garri
- Yam
- Sweet potato
- Irish potato
- Bread
- Pasta
- Plantain
- Millet
- Cassava
- Oats

Category value: `"carb"`

Proteins (Body-Building Foods)

- Beans
- Lentils
- Chickpeas
- Groundnuts
- Eggs
- Chicken
- Turkey
- Beef
- Goat meat
- Fish (Tilapia)
- Dried fish
- Milk
- Yogurt
- Tofu

Category value: `"protein"`

Vegetables

- Spinach
- Kale
- Ugu
- Cabbage
- Carrots

- Tomatoes
- Onions
- Okra
- Green pepper
- Lettuce
- Garden eggs
- Broccoli

Category value: `"vegetable"`

Fruits

- Banana
- Apple
- Orange
- Mango
- Pawpaw
- Pineapple
- Watermelon
- Avocado
- Guava

Category value: `"fruit"`

Healthy Fats

- Palm oil
- Vegetable oil
- Olive oil
- Groundnut oil
- Nuts
- Avocado

Category value: `"fat"`

📷 Recommended Dataset Size

- 10–12 carbs
- 12–15 proteins
- 12 vegetables
- 8 fruits
- 5 fats

Total ≈ 45–55 food items

Task 3: Create Mock Dataset (CSV Files)

Create: - `food_data.csv` - `sales_data.csv`

Minimum Requirements: - 50 food items - 10 sellers - 30–50 sample sales records

---

Task 4: Data Cleaning & Preprocessing

- Handle missing values
- Convert prices to numeric
- Encode food categories if necessary
- Format dates properly

🎁 Deliverables

- Preprocessing Python script
- Clean dataset

Task 5: Build Basic Recommendation System (Rule-Based)

Logic

1. Input: `user budget + location`
2. Filter food items within budget
3. Ensure basket contains:
4. At least 1 carb
5. At least 1 protein
6. At least 1 vegetable
7. Add fruit if budget allows
8. Total price must not exceed budget

Example Output

User Budget: ₦5000

```
{
  "total_budget": 5000,
  "total_cost": 4750,
  "items": [
    {"name": "Rice", "category": "carb", "price": 1500},
    {"name": "Beans", "category": "protein", "price": 1200},
    {"name": "Spinach", "category": "vegetable", "price": 800},
```

```
    {"name": "Banana", "category": "fruit", "price": 1250}
  ]
}
```

🎁 Deliverables

- Python recommendation function
- Jupyter notebook with test examples

ISSUE 2: Demand Prediction Model & Backend Integration Support (FEB 18TH – FEB 22ND)

🎯 Objective

Build a simple demand forecasting model and prepare AI services for backend integration.

Task 1: Prepare Time-Series Sales Data

- Load `sales_data.csv`
- Convert date column to proper datetime format
- Group sales by date (daily or weekly)
- Aggregate total quantity sold per food item
- Sort data by date

🎁 Deliverable

- Clean time-series dataset ready for modeling

Task 2: Build Simple Demand Prediction Model

Choose ONE approach:

Option A (Recommended)

- Implement Moving Average model

Option B

- Implement Linear Regression model

Goal

Predict demand for the next 7 days for a selected food item.

🎁 Deliverables

- `demand_prediction.py`
- Model evaluation using:
- MAE (Mean Absolute Error)
- OR MSE (Mean Squared Error)

Task 3: Visualize Results

- Plot historical demand
- Plot predicted demand
- Label axes clearly (Date vs Quantity)

🎁 Deliverable

- Demand forecast graph (Matplotlib)

Task 4: Prepare Backend API Structure

Example Input (JSON)

```
{
  "food_id": 101,
  "location": "Nairobi"
}
```

Example Output (JSON)

```
{
  "predicted_demand_next_week": 120
}
```

- Define function that accepts `food_id`
- Ensure output is JSON serializable
- Document expected request & response format

🎁 Deliverable

- API structure documentation

Task 5: Documentation

Create `AI_README.md` explaining: - How recommendation system works - How demand prediction works - Assumptions made - Limitations of the prototype - Future improvements (real-time data, advanced ML, etc.)

🎺 Suggested Timeline (FEB 18TH – FEB 22ND)

FEB 18TH: Prepare time-series data
FEB 19TH: Build prediction model
FEB 20TH: Evaluate + visualize results
FEB 21ST: Prepare API structure
FEB 22ND: Documentation + cleanup

Tools & Environment

- Python
- Pandas
- Scikit-learn (if using regression)
- Matplotlib
- Google Colab (for development, experimentation, and notebook execution)

Important Notes

- Keep it simple
- No deep learning
- Focus on a working prototype
- Ensure backend team can integrate outputs easily
- Push code daily