FreshAI Implementation Guide using OpenAI

This document outlines the step-by-step process to implement FreshAI — an AI-powered sustainability app — using OpenAI.

Overview

FreshAI helps users reduce food waste by: - Tracking groceries (manually or via receipt scanning) - Notifying users when items are expiring - Generating creative, personalized recipes based on soon-to-expire ingredients

The OpenAI API is used to: - Generate recipe suggestions - Tailor recipes to dietary preferences - Optionally generate storage tips or advice

---

Step-by-Step Guide

1. Set up your Environment

- Install Python and Jupyter Notebook.
- Install required libraries:

```
pip install openai pytesseract pandas
```

- Obtain an OpenAI API key and set it in your environment or code.

---

2. Parse and Store Grocery Data

- Use OCR libraries (like `pytesseract`) to extract grocery items from scanned receipts.
- Alternatively, allow users to enter items manually.
- Store grocery data in a structured format (e.g., a pandas DataFrame) with:
- Item name
- Quantity (optional)
- Expiration date (user-provided or estimated)
- Timestamp of entry

---

3. Track Expiration Dates

- Estimate shelf life for items without an explicit expiration date.
- Write a Python function to check which items are nearing expiration.

- Schedule regular checks to notify the user about expiring items.

---

4. Integrate OpenAI API

- Use the OpenAI API to generate recipes tailored to the ingredients and user preferences.
- Sample Python code:

```
import openai

openai.api_key = "your_api_key"

def generate_recipe(ingredients, preferences=None):
    prompt = f"Give me a creative recipe using these ingredients: {',
'.join(ingredients)}."
    if preferences:
        prompt += f" Please consider these preferences: {preferences}."
    prompt += " Provide ingredients and step-by-step instructions."

    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        max_tokens=300,
        temperature=0.7
    )

    return response.choices[0].text.strip()
```

- Example usage:

```
generate_recipe(['tomatoes', 'spinach'], preferences='gluten-free,
Italian')
```

---

5. Enhance Suggestions

- Optionally use OpenAI to:
- Generate storage tips: “Best way to store avocados?”
- Offer shopping or meal planning advice.

---

6. Testing & Iteration

- Test the app to ensure:
- Accurate receipt parsing

- Proper expiration tracking and notifications
- High-quality, relevant recipe suggestions
- Respect for user preferences (dietary and cuisine)
- Adjust the prompt and parameters (e.g., `temperature`, `max_tokens`) to improve OpenAI outputs.

Future Development

- Transition from Jupyter Notebook to a mobile app.
- Build backend APIs and cloud storage.
- Implement real push notifications and account management.
- Add sustainability metrics and partnerships with grocery/recipe services.

Summary Table of OpenAI Integration

Task	How OpenAI Helps
Recipe Suggestions	GPT generates creative recipes
Tailored to Preferences	Include allergies, diets, cuisines in prompt
Storage Tips	GPT suggests proper storage methods
User Interaction	GPT answers questions and provides advice

For assistance with code, prompt crafting, or mobile app workflows — feel free to ask.