NUTRI-CRAFT

PERSONALIZED FOOD AND DIET RECOMMENDATION SYSTEM

Team members

Rajeev R (70)
Nidha Shameer T S (45)
Brighty Jobin (26)
Adeela Farshana (7)

Guide

Maanasa N A S

CONTENTS

1.Current Status

2.Working Demo

3.Challenges and Solutions

4.Project Roadmap

CURRENT STATUS

FEATURES COMPLETED

Frontend

- Adjustable Recommendations: User-specified number of recommendations.
- Custom Nutritional Values: User-defined nutrient levels.
- Ingredient Filtering: Include specific ingredients
- Automated Image Retrieval: Web scraping for recipe images.
- Recipe Overview Panel: At-a-glance nutritional summary.
- Detailed Recipe Information: Displays cooking instructions and prep time

Backend

- (main.py)Pydantic model to validate the list.
- (model.py)Recommendation System.

WORK IN PROGRESS

- Diet Oriented Recommendation System

FEATURES PENDING

- Docker Deployment and Integration

WORKING DEMO

https://github.com/Rajeev-08/Nutri-Caftt

CHALLENGES AND SOLUTIONS

- **Choosing the Right Distance Metric (model.py - NearestNeighbors)**

  Challenge : k-NN relies on distance calculations, and Euclidean distance was ineffective for comparing nutritional profiles.

  Our Approach : We switched to cosine distance, which measures vector angles

- **Constructing an Efficient Recommendation Pipeline(model.py - sklearn.pipeline)**

  Challenge : Managing data filtering, scaling, and k-NN prediction separately was complex and prone to errors.

  Our Approach : We implemented scikit-learn's Pipeline to streamline these steps

- **Race Conditions in Streamlit (Food_Recommendation.py)**

  Challenge : Streamlit's rerun-on-every-interaction behavior risked race conditions, triggering multiple recommendation requests if users interacted rapidly.

  Our Approach : We used Streamlit's session state with a "generating" flag to block new requests until the current one was complete

- **Requests Timeout (Generator.py)**

  Challenge : The frontend sometimes failed to receive backend responses due to network issues or downtime.

  Our Approach : We added a timeout to the requests.post call and implemented retry logic

- **Type Errors in FastAPI (main.py - Request Handling)**

  Challenge : Ensuring the backend handled requests with the correct data format was challenging

  Our Approach : We used Pydantic models to enforce proper input types and formats

Completed

Pending

PROJECT ROADMAP