MVP Plan: AI-Powered Global Food Exchange

Project: AI-Powered Global Food Exchange Purpose: High-Quality MVP for Microsoft Imagine Cup 2026 Objective: To build a functional and visually compelling MVP demonstrating the project's workflow, features, and impact, with a clear path for integration with Azure services.

1. UX/UI & Design Principles (Figma/Canva)

Color Scheme:
Primary Green (#28A745): For calls-to-action, success states, and sustainability branding.
Trustworthy Blue (#007BFF): For links, important information, and trust-building elements like the ledger.
Clarity White (#FFFFFF) & Light Gray (#F8F9FA): For backgrounds to ensure a clean, uncluttered interface.
Design: Create a clean, modern, mobile-first design.
Interactivity: Design interactive buttons with clear hover/click states. Highlight key metrics and notifications with subtle animations or color shifts.

2. Core Features & Technical Implementation

This section details the functionality of each component.

A. Farmer/Restaurant Dashboard

Functionality:
Upload Form: A simple form with fields for Food Type, Quantity (kg), Expiry Date, and Location (auto-detected or manual input).
Photo Upload: An optional feature to upload a photo of the surplus food, increasing trust for NGOs.
Input Validation:
Expiry date must be in the future.
Quantity must be a positive number.
Confirmation: Upon submission, display a notification: "Thank you! Our AI is matching your surplus with a local NGO."
[Insert Mockup Image: Enhanced farmer dashboard with photo upload Here]

B. NGO Dashboard

Functionality:

Donation View: Display available donations as interactive cards.
Filtering: Allow NGOs to filter donations by:
Distance (e.g., < 5km, < 10km)
Food Type (e.g., Produce, Dairy, Grains)
Urgency (based on expiry date)
Claim Process: A one-click "Claim" button on each card.
Confirmation: After claiming, show a confirmation screen with the logistics map.
[Insert Mockup Image: NGO dashboard with filtering options Here]

C. AI Matching Engine (Python Backend)

Logic: A rule-based matching algorithm that prioritizes items based on a calculated score:
Score = (Weight_Distance * 1/Distance) + (Weight_Expiry * 1/Days_to_Expiry).
Process:
When a farmer uploads surplus, the engine identifies the top 3 closest NGOs.
It calculates a match score for each of the 3 NGOs.
The NGO with the highest score is considered the "best match" and can be notified first.
Code: See backend_api.py for a functional Python implementation.

D. Logistics Map (Azure Maps)

Functionality:
Integration: Use the Azure Maps API to display an interactive map.
Route Display: On the claim confirmation screen, render the optimal route from the donor's location to the NGO's location.
Estimated Time: Display the estimated delivery time provided by the Azure Maps API.
[Insert Mockup Image: Map showing route and estimated delivery time Here]

E. Blockchain Ledger Simulation (Python Backend)

Functionality:
Transaction Record: When an NGO claims a donation, a new block is added to a simulated chain.
Hash Generation: Each block contains a Transaction ID (a SHA-256 hash of its contents and the previous block's hash), ensuring integrity.
Data Points: The block stores: Transaction ID, Donor, NGO, Food Type, Quantity, Date.
Future Integration: This simulation can be replaced with Azure Confidential Ledger for a secure, production-ready solution.
Code: See backend_api.py for the simulation logic.

F. Impact Dashboard (Power BI)

Data Source: Connect Power BI to the improved dummy datasets (donors_data.csv, ngos_data.csv, transactions_ledger.csv).
KPI Cards:
Food Saved (kg)
People Fed (Est. kg * 4)
CO_2 Emissions Reduced (Est. kg * 2.5)
Charts:
Pie Chart: Breakdown of Donations by Food Type.
Bar Chart: Top 5 NGOs by Quantity received.
Map: A map visual plotting all donor and NGO locations.
[Insert Screenshot: Professional Power BI dashboard with KPI cards and charts Here]

3. Technical Stack Recommendations

Frontend: React with TailwindCSS (for web app) or Figma (for prototype).
Backend: Python with FastAPI (recommended for its speed and ease of use).
Database: PostgreSQL for production; use the provided CSV files for the MVP.
AI/ML: Python with scikit-learn for future predictive models; the rule-based engine in the provided code is sufficient for the MVP.
Maps: Azure Maps API.
Dashboard: Power BI Desktop.

4. Step-by-Step Integration & Deployment Guide

Design: Create all app screens and visuals in Figma.
Backend: Run the backend_api.py server locally. It serves the API for the frontend to consume.
Frontend: Develop the React frontend, connecting UI components to the local backend API endpoints.
Dashboard: Build the Power BI dashboard by importing the provided CSV data files.
Local Deployment:
Run backend: uvicorn backend_api:app --reload
Run frontend: npm start
Azure Deployment (Future):
Deploy the backend API as an Azure App Service.
Deploy the React frontend as an Azure Static Web App.
Publish the Power BI report to the Power BI Service.

5. Bonus Features (Next Steps)

Azure OpenAI Chatbot: Integrate a chatbot that can answer user questions like "Where can I find vegetable donations?" by querying the backend API.

Automated Notifications: Use Azure Logic Apps to trigger automated email or SMS notifications to NGOs when a high-priority match is found for them.