Step-by-Step Guide to Building a Working Prototype of the AI-Driven Demand Forecasting & Order Optimization System

This guide provides **clear steps** for Kashmira to build an initial working prototype of her AI-driven food waste reduction system. It includes **data collection, model development, optimization, a simple dashboard, and deployment** using a structured **tech stack**.

🛠️ Tech Stack for the Prototype

Component	Technology
Programming Language	Python
Data Handling & Processing	Pandas, NumPy
Machine Learning/AI	scikit-learn, TensorFlow/Keras (if needed for deep learning)
Time Series Forecasting	Prophet (by Facebook), ARIMA, XGBoost
Optimization Algorithms	SciPy, PuLP (for linear programming-based inventory optimization)
Database	PostgreSQL or SQLite (for lightweight prototyping)
Backend API	Flask or FastAPI (for serving model predictions)
Frontend (Dashboard/UI)	Streamlit (for quick prototyping) or React (for more scalability)
Visualization	Matplotlib, Plotly (for charts/graphs)
Cloud Hosting	AWS (EC2 for app hosting, RDS for database) or Google Cloud/App Engine
Version Control	GitHub (for code tracking and collaboration)

📌 Step 1: Define Objectives & Collect Data

1.1. Define the Core Features of the Prototype

The MVP (Minimum Viable Product) should include:
✅ Forecasting Model: Predict meal demand based on past data.
✅ Order Optimization: Suggest optimal ordering amounts.
✅ Visualization Dashboard: Display forecasts and optimization results.
✅ Simulation Tool: Allow users to adjust input variables and see projected waste/cost savings.

1.2. Gather Initial Data

- **Request historical meal sales/orders from Hockaday's cafeteria (or use open-source food waste datasets initially).**

- **External Data to Collect for Better Forecasting:**

    - School schedule (holidays, exams, special events affecting meal counts).
    - Weather data (hot/cold days impact meal preferences).
    - Foot traffic (if available) or overall school attendance per day.
    - Shelf life of ingredients (for optimization).
- If real data is not available yet, **simulate a dataset** using Python:

```python
import pandas as pd
import numpy as np

# Generate sample meal sales data
np.random.seed(42)
dates = pd.date_range(start="2023-01-01", periods=180, freq="D")
sales = np.random.randint(150, 300, size=len(dates))  # Example: 150-300 meals per day
data = pd.DataFrame({"date": dates, "meal_sales": sales})
data.to_csv("synthetic_meal_sales.csv", index=False)
```

📌 Step 2: Build the Forecasting Model

2.1. Choose a Baseline Forecasting Model

- Start simple: **Prophet** (by Facebook) or **ARIMA** (AutoRegressive Integrated Moving Average) for time series predictions.

- Later, consider **XGBoost** or **LSTMs (Long Short-Term Memory models)** for more complex patterns.

Train a Prophet Model

```python
from fbprophet import Prophet
import pandas as pd

# Load data
df = pd.read_csv("synthetic_meal_sales.csv")
df.rename(columns={"date": "ds", "meal_sales": "y"},
inplace=True)

# Train model
model = Prophet()
model.fit(df)

# Forecast next 30 days
future = model.make_future_dataframe(periods=30)
forecast = model.predict(future)

# Visualize results
import matplotlib.pyplot as plt
model.plot(forecast)
plt.show()
```

✅ **Goal:** Predict meal demand based on past trends.

📌 Step 3: Implement Order Optimization Algorithm

- Use **linear programming (LP)** to **optimize order quantities** (e.g., minimize waste while ensuring enough food is available).
- Constraints include:
    - Shelf life of ingredients.
    - Storage capacity.
    - Budget limits.

Example: Linear Programming for Order Optimization

```python
from scipy.optimize import linprog

# Example: minimize cost while meeting demand
c = [2, 3, 1.5]  # Cost per unit for ingredients A, B, and C
A = [[1, 1, 0], [0, 2, 1]]  # Constraints (e.g., minimum quantity of A & B needed)
b = [100, 150]  # Minimum demand requirement

res = linprog(c, A_ub=A, b_ub=b, method='highs')
print(res)
```

✅ **Goal:** Recommend daily ingredient orders based on predicted meal demand.

📌 Step 4: Build a Simple Dashboard (UI)

4.1. Fastest Option: Use Streamlit for Quick Prototyping

```python
import streamlit as st
import pandas as pd

st.title("AI-Driven Meal Forecasting & Optimization")

# Load forecast data
df = pd.read_csv("synthetic_meal_sales.csv")
st.line_chart(df.set_index("date"))

st.write("Predicted meals for next week:")
st.table(df.tail(7))
```

✅ **Goal:** Display forecasts and order recommendations in an interactive way.

📌 Step 5: Deploy the Prototype

5.1. Deploy Model as an API Using Flask or FastAPI

- Create an **API endpoint** that serves model predictions:

```python
from flask import Flask, jsonify
import pandas as pd

app = Flask(__name__)

@app.route("/predict", methods=["GET"])
def predict():
    forecast = pd.read_csv("forecasted_meal_sales.csv")
    return jsonify(forecast.tail(7).to_dict())

if __name__ == "__main__":
    app.run(debug=True)
```

5.2. Host on Cloud

- Use **AWS EC2**, **Google Cloud App Engine**, or **Heroku**.
- Deploy using Docker if scaling is needed.

```
# Deploy to Heroku (example)
heroku login
heroku create meal-forecasting-app
git push heroku main
```

✅ **Goal:** Make the forecasting model accessible via a web API.

📌 Step 6: Test & Iterate

- Run **test cases** to verify the accuracy of forecasts and optimization.

- Collect feedback from users (e.g., school kitchen staff).
- **Improve model performance** by adding **more data sources** (weather, menu trends, etc.).
- Implement **real-time adjustments** (e.g., modify forecasts mid-day if attendance changes).

📌 Future Enhancements & Scaling

Next Steps	Enhancements
Phase 2: Improve Model	Add real-time updates, deep learning (LSTMs) for better predictions.
Phase 3: Expand UI	Move from Streamlit to React for a more scalable dashboard.
Phase 4: Multi-Sector Scaling	Adapt system for restaurants and grocery retailers.

🎯 Summary of Steps

1. **Collect historical meal data & external variables.**
2. **Train a demand forecasting model (Prophet/ARIMA/XGBoost).**
3. **Develop an optimization algorithm (linear programming) to adjust orders.**
4. **Build a simple UI dashboard (Streamlit for prototype, React later).**
5. **Deploy the model as an API (Flask/FastAPI) and host on AWS/Heroku.**
6. **Test, iterate, and refine with real-world data from Hockaday & The Stewpot.**

🎯 Expected Timeline

Week	Task
Week 1-2	Collect & clean initial dataset
Week 3-4	Build & test forecasting model
Week 5-6	Develop optimization algorithm
Week 7	Create dashboard prototype
Week 8-9	Deploy prototype & gather feedback

Week 10+	Improve accuracy, refine UI, expand datasets

🚀 Final Thought

By following these structured steps, Kashmira can **rapidly build a functional prototype** that proves her AI-driven approach works. Once validated in a school setting, she can **scale it to restaurants and retailers**, creating a **high-impact AI solution** for food waste reduction. 🚀