)

→ Fitt food :

3 Dashboards :

1. Patients

↳ Live vitals /

$$\begin{tikzcd} \text{upload Prescr.} \arrow[d, "\text{OCR}"] \\ \text{Data} \arrow[d] \\ \text{RAG + Recommender} \arrow[d] \\ \text{Recommended meal Plan} \end{tikzcd}$$

+ Early warning model

2. NGOs

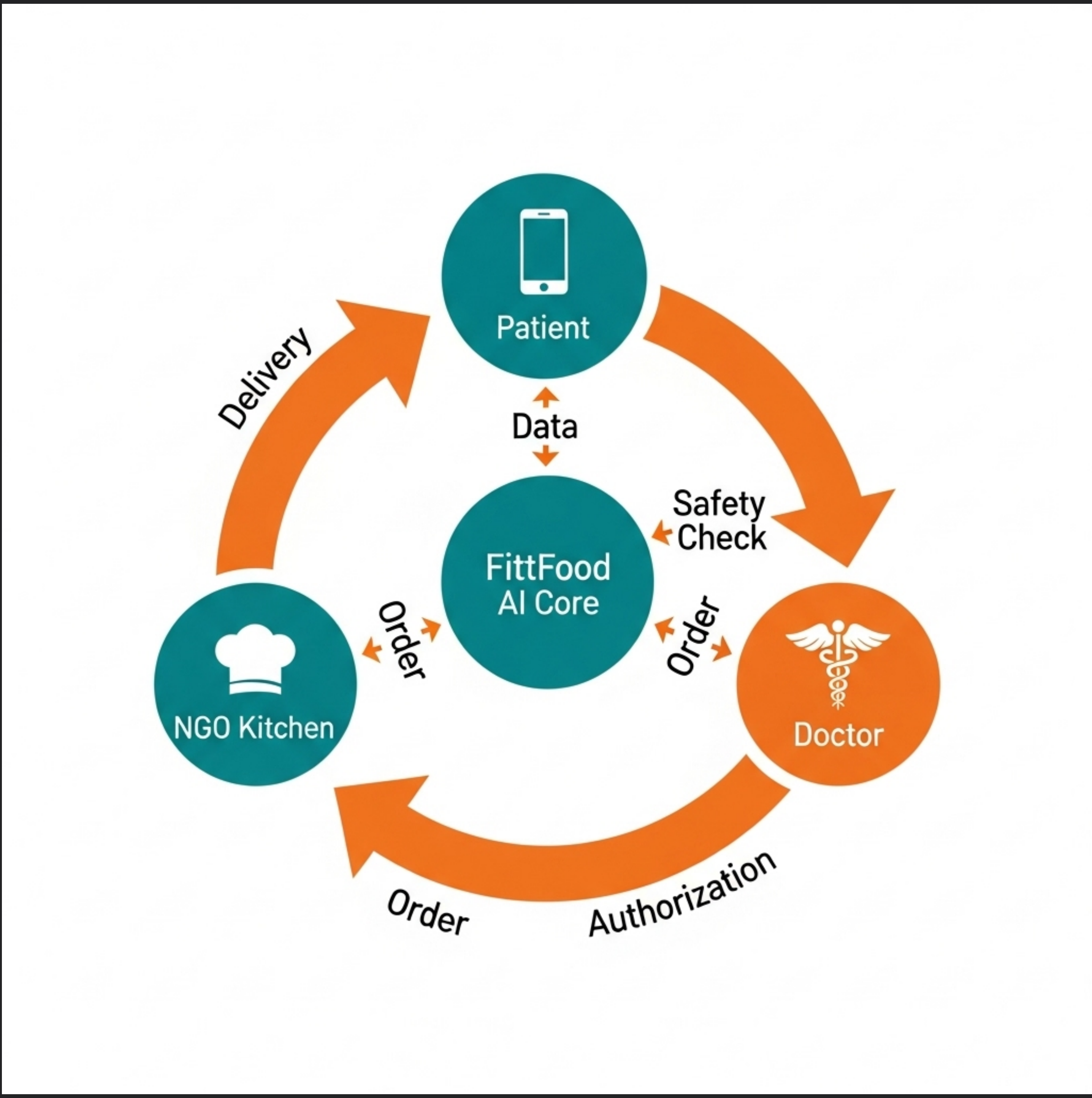
↳ Audio AI assistant for help
+
video of meals from the storage on o/p of recommender system

3. Doctor

↳ Early warning sys notification after threshold crossed

↳ If less confidence score in recommendⁿ, call for supervision & alert

→ Once recommendatⁿ sys gives recommended meal plan, you can either →
i) order directly from NGOs
ii) cook it urself.


Patient
Data
Delivery
Safety Check
FittFood AI Core
Order
Order
NGO Kitchen
Doctor
Order
Authorization

API Gateway:

↳ This is where each request comes.

↳ ye requests ko further direct karega to specific components.

eg: login req to Auth
photo upload to OCR

services:

1. Auth:

→ Firebase auth

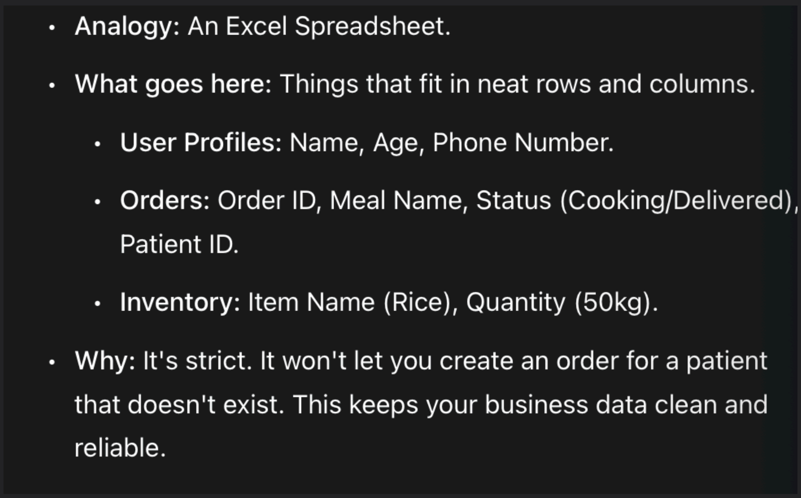
→ verifies who is logging in to which dashboard to maintain privacy.

2. Notificat^n:

→ socket.io

Storage system:

1. PostgreSQL (Relational D.B)



- **Analogy:** An Excel Spreadsheet.
- **What goes here:** Things that fit in neat rows and columns.
    - **User Profiles:** Name, Age, Phone Number.
    - **Orders:** Order ID, Meal Name, Status (Cooking/Delivered), Patient ID.
    - **Inventory:** Item Name (Rice), Quantity (50kg).
- **Why:** It's strict. It won't let you create an order for a patient that doesn't exist. This keeps your business data clean and reliable.

B. For Medical Logic: Neo4j (Graph Database)

- **Analogy:** A Mind Map or Spider Web.
- **What goes here:** Connections and Rules.
  - Nodes: "Drug: Rifampicin", "Side Effect: Liver Stress", "Food: Spinach".
  - Lines: Rifampicin --> Causes --> Liver Stress.
- **Why:** Medical data is about relationships. If you ask "What food helps with Rifampicin?", a standard spreadsheet is too slow. A Graph Database can "hop" from Drug to Side Effect to Food instantly to give the recommendation.

C. For Messy Health Logs: MongoDB (NoSQL Database)

- **Analogy:** A boundless box where you can throw documents.
- **What goes here:** Fast, messy data from smartwatches.
  - Heart rate logs (every second), SpO2 readings, Step counts.
- **Why:** Watch data is huge and unstructured. Sometimes it sends heart rate, sometimes sleep data. MongoDB doesn't care about the structure; it just catches it all quickly.

D. For Big Files: AWS S3 (Object Storage)

- **Analogy:** A Warehouse or Google Drive.
- **What goes here:** Actual files.
  - The JPG image of the prescription the patient scanned.
  - The MP4 video file of the cooking tutorial.
- **Why:** You never store big files inside a database (it slows it down). You store the file in the warehouse (S3) and just keep the address (URL) in the database.

Summary Checklist for Implementation

1. **Backend:** Node.js (Traffic Controller) + Socket.io (Alerts).
2. **Structured DB:** PostgreSQL (Users/Orders).
3. **Graph DB:** Neo4j (Medical Rules).
4. **Log DB:** MongoDB (Watch Data).
5. **File Store:** S3 (Images/Videos).

Recommendation system:

Step 1: OCR Output ("Rifampcn")

↓

Step 2: Vector Match → Identified Drug: **"Rifampicin"**

↓

Step 3: Neo4j Lookup → Needs: **"Vitamin B6"**

↓

Step 4: Neo4j Food Search → Candidates: **["Spinach", "Chicken"]**

↓

Step 5: User Filter (Veg) → Final: **"Spinach"**

↓

Step 6: Safety Check (SpO2 > 90?) → **Safe to Recommend.**

Summary of the "From Scratch" Tech Stack

1. Embedding Model: BioBERT (for text understanding).
2. Recommender Model: TransE (Knowledge Graph Embedding). Trained by you on OpenFDA/ICMR data.
3. Language: Python (PyTorch + PyKeen).
4. Database: Neo4j (to visualize the graph) and PostgreSQL (to store user logs).

Why this removes "RAG"?

- RAG = Retrieve Text + LLM Generates Answer.
- Your Solution = Retrieve Embeddings + Math Calculate Score + Template.

There is no "Generation." There is only Prediction. This is safer, lighter, and strictly scientific. This is exactly what a high-tech medical startup would build.

To implement the Early Warning System (EWS) for detecting critical health deterioration (like Sepsis or Cardiac Arrest) using Multimodal Data (OCR Clinical Context + Live Vitals), you need a specific set of Time-Series and Classification models.

Here is the Technical Stack & Methodology for building the "FittFood Early Warning Engine."

1. The Core AI Models (The "Prediction Engines")

Since you are dealing with two types of data (Static History vs. Dynamic Vitals), you need a Hybrid Architecture.

A. For Time-Series Analysis (The Watch Data)

- Model: LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Units).
- Why: Standard models (like Random Forest) only look at a "snapshot." LSTM remembers the history. It knows that "Heart Rate 100" is fine if you were exercising, but dangerous if it has been creeping up slowly for 6 hours while resting.
- Framework: PyTorch or TensorFlow.

B. For Risk Classification (The "Fusion" Logic)

- Model: XGBoost (Extreme Gradient Boosting).
- Why: It is incredibly fast and interpretable. It excels at combining "categorical data" (e.g., Disease = TB) with "numerical data" (e.g., SpO2 = 88%).
- Framework: XGBoost library (Python).

2. The Implementation Workflow (Step-by-Step)

Step 1: Feature Engineering (Preparing the Data)

You don't feed raw numbers to the AI. You feed it "Features."

- From OCR (Context Features):
- has_tb: 1 or 0
- has_diabetes: 1 or 0
- medication_risk_level: High/Medium/Low (derived from the drug list).
- From Watch (Vital Features):
- spO2_rolling_avg_1hr: The average oxygen over the last hour.
- hr_variability (HRV): The standard deviation of heartbeats (Low HRV = Stress/Sepsis).
- shock_index: Heart Rate divided by Systolic BP (if available).

Step 2: Building the "Fusion" Model Architecture

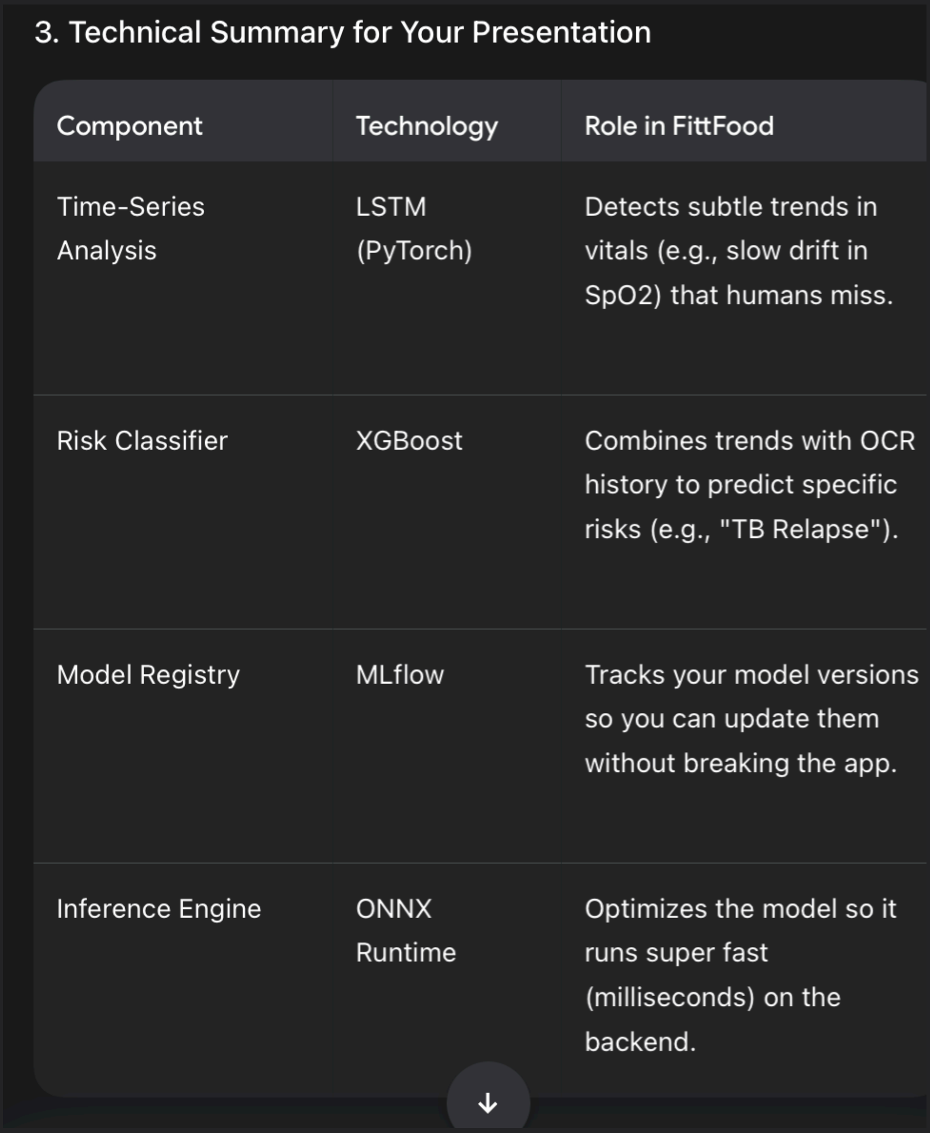
You will build a Two-Stage Pipeline:

1. Stage 1 (The Trend Detector):
- Input: Last 6 hours of Heart Rate & SpO2.
- Model: LSTM.
- Output: A "Deterioration Score" (0 to 1).

2. Stage 2 (The Risk Calculator):
- Input: Deterioration Score (from Stage 1) + Clinical Context (from OCR).
- Model: XGBoost.
- Logic: If Deterioration Score is high AND has_tb is True \rightarrow Prediction: "Sepsis Risk High".

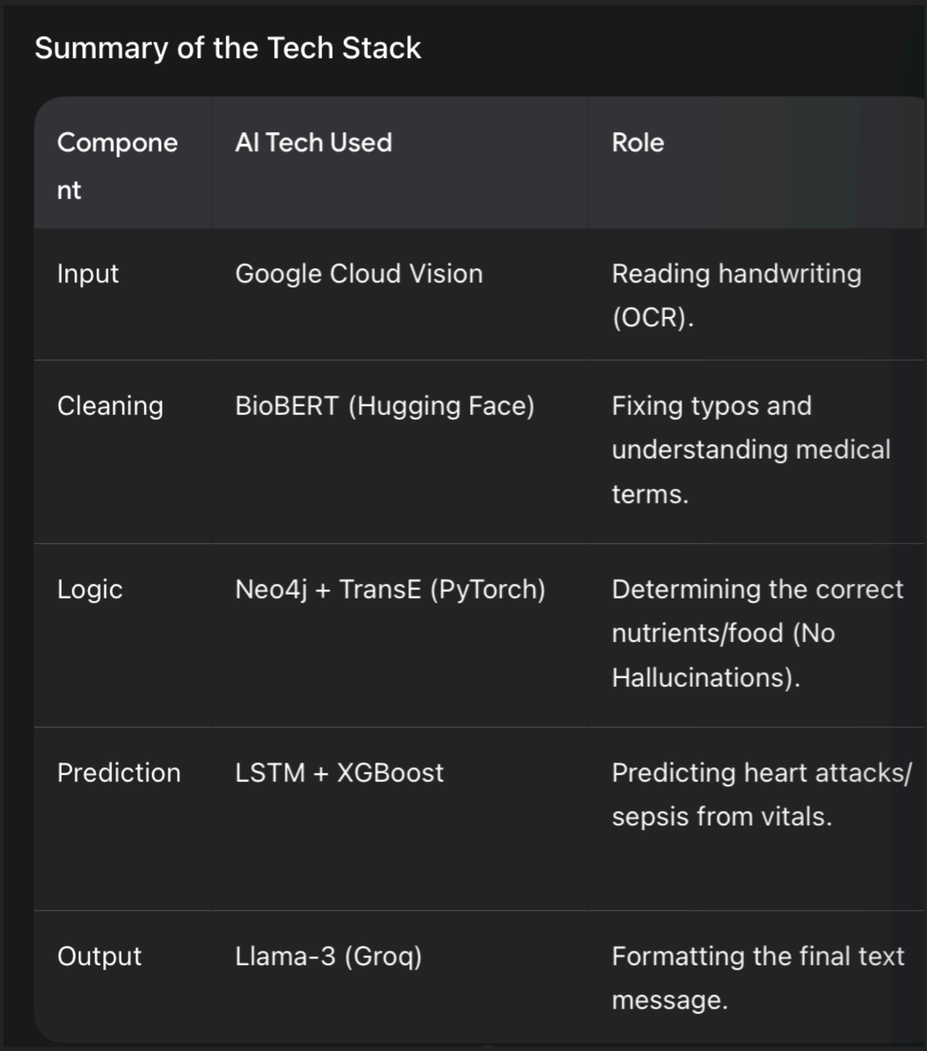
Step 3: Training the Model

- Dataset: MIMIC-IV (ICU Dataset).
- Filter for patients with Infectious Diseases (proxy for TB).
- Extract their vitals 6 hours before they went into septic shock.
- Label this data as Target = 1 (Sepsis).
- Label stable patients as Target = 0.
- Action: Train XGBoost to differentiate between the two groups.



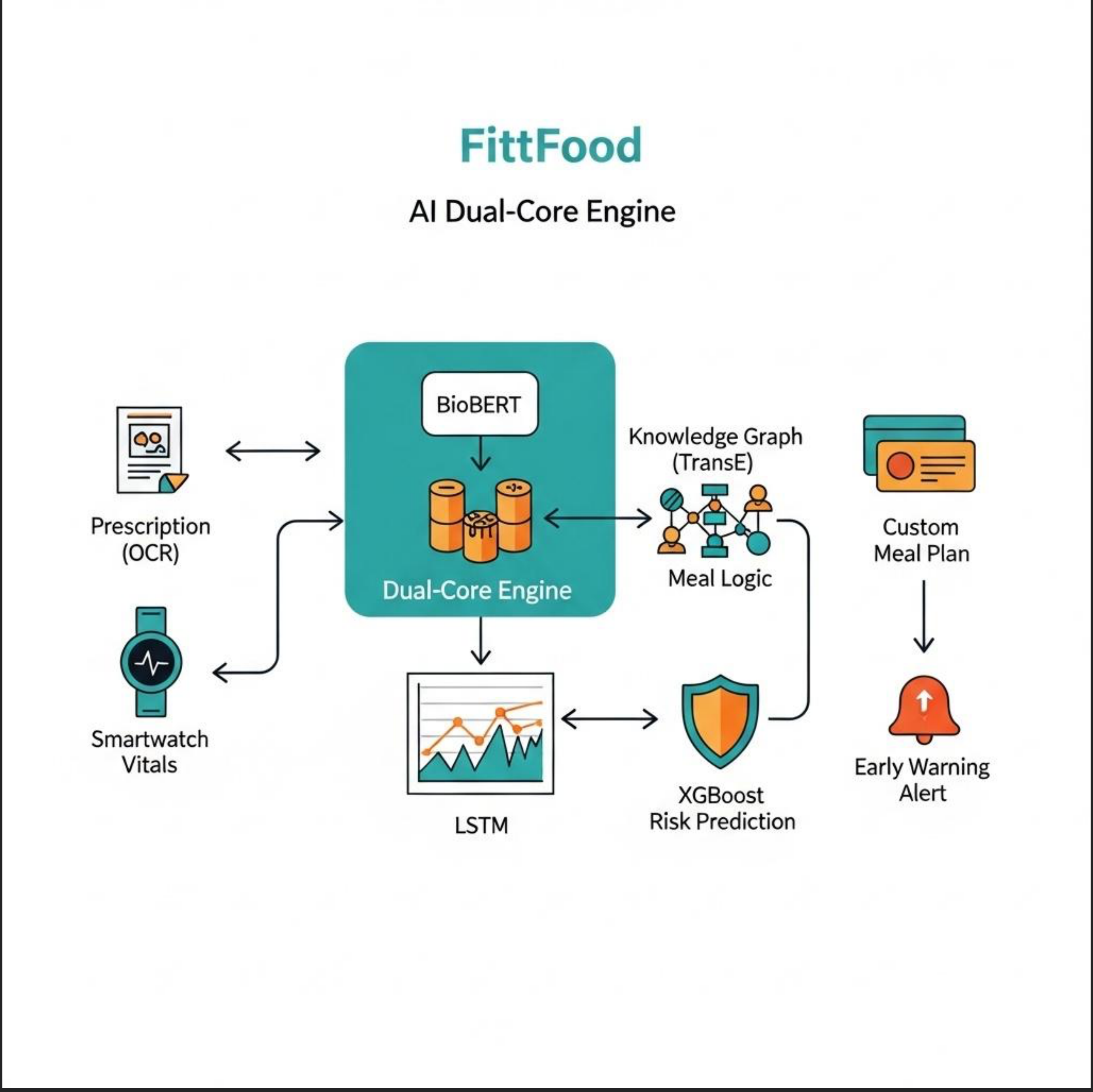
3. Technical Summary for Your Presentation

Component	Technology	Role in FittFood
Time-Series Analysis	LSTM (PyTorch)	Detects subtle trends in vitals (e.g., slow drift in SpO2) that humans miss.
Risk Classifier	XGBoost	Combines trends with OCR history to predict specific risks (e.g., "TB Relapse").
Model Registry	MLflow	Tracks your model versions so you can update them without breaking the app.
Inference Engine	ONNX Runtime	Optimizes the model so it runs super fast (milliseconds) on the backend.



Summary of the Tech Stack

Component	AI Tech Used	Role
Input	Google Cloud Vision	Reading handwriting (OCR).
Cleaning	BioBERT (Hugging Face)	Fixing typos and understanding medical terms.
Logic	Neo4j + TransE (PyTorch)	Determining the correct nutrients/food (No Hallucinations).
Prediction	LSTM + XGBoost	Predicting heart attacks/ sepsis from vitals.
Output	Llama-3 (Groq)	Formatting the final text message.


FittFood
AI Dual-Core Engine
Prescription (OCR)
Smartwatch Vitals
BioBERT
Dual-Core Engine
Knowledge Graph (TransE)
Meal Logic
LSTM
XGBoost Risk Prediction
Custom Meal Plan
Early Warning Alert

Modern VLMs

$\downarrow$

Biobert
(creates vectors)

$\downarrow$

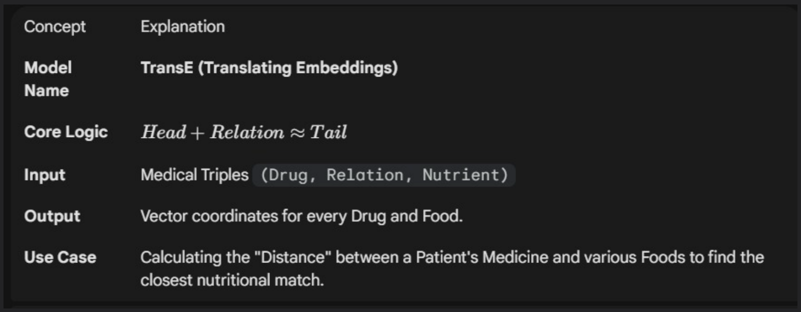
Recommender Engine
(Neo4J + Translatn embeddings)

$\downarrow$

Early Warning
(XGBOOST + LSTM)

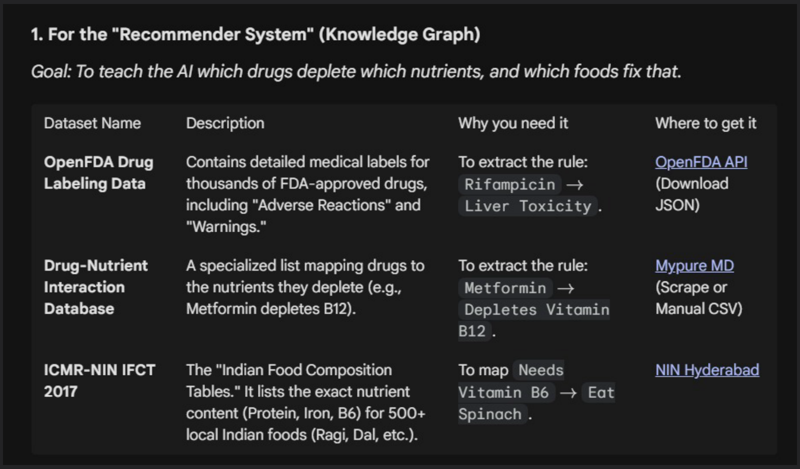
$\downarrow$

AI assistant



Concept	Explanation
Model Name	**TransE (Translating Embeddings)**
Core Logic	$Head + Relation \approx Tail$
Input	Medical Triples `(Drug, Relation, Nutrient)`
Output	Vector coordinates for every Drug and Food.
Use Case	Calculating the "Distance" between a Patient's Medicine and various Foods to find the closest nutritional match.

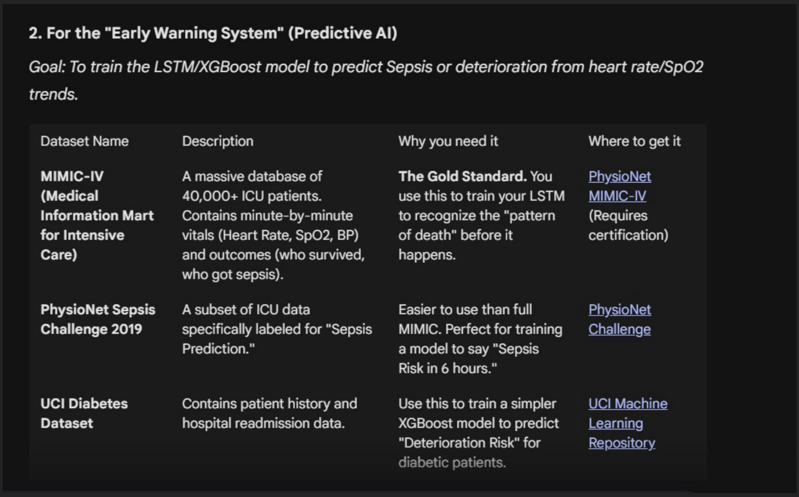
Datasets :



1. For the "Recommender System" (Knowledge Graph)

Goal: To teach the AI which drugs deplete which nutrients, and which foods fix that.

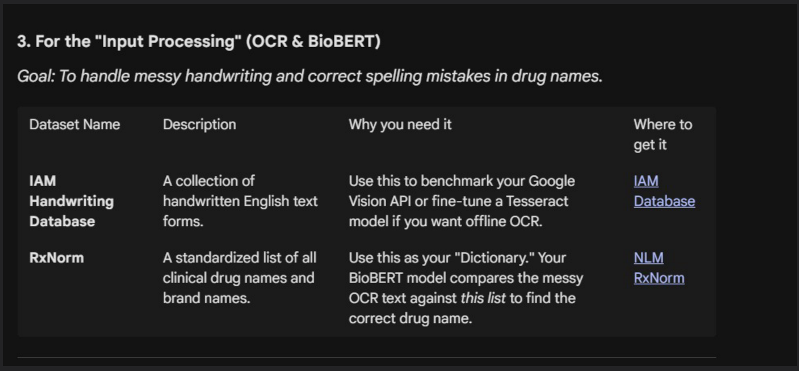
Dataset Name	Description	Why you need it	Where to get it
OpenFDA Drug Labeling Data	Contains detailed medical labels for thousands of FDA-approved drugs, including "Adverse Reactions" and "Warnings."	To extract the rule: `Rifampicin` → `Liver Toxicity`.	OpenFDA API (Download JSON)
Drug-Nutrient Interaction Database	A specialized list mapping drugs to the nutrients they deplete (e.g., Metformin depletes B12).	To extract the rule: `Metformin` → `Depletes Vitamin B12`.	Mypure MD (Scrape or Manual CSV)
ICMR-NIN IFCT 2017	The "Indian Food Composition Tables." It lists the exact nutrient content (Protein, Iron, B6) for 500+ local Indian foods (Ragi, Dal, etc.).	To map `Needs Vitamin B6` → `Eat Spinach`.	NIN Hyderabad



2. For the "Early Warning System" (Predictive AI)

Goal: To train the LSTM/XGBoost model to predict Sepsis or deterioration from heart rate/SpO2 trends.

Dataset Name	Description	Why you need it	Where to get it
MIMIC-IV (Medical Information Mart for Intensive Care)	A massive database of 40,000+ ICU patients. Contains minute-by-minute vitals (Heart Rate, SpO2, BP) and outcomes (who survived, who got sepsis).	**The Gold Standard.** You use this to train your LSTM to recognize the "pattern of death" before it happens.	PhysioNet MIMIC-IV (Requires certification)
PhysioNet Sepsis Challenge 2019	A subset of ICU data specifically labeled for "Sepsis Prediction."	Easier to use than full MIMIC. Perfect for training a model to say "Sepsis Risk in 6 hours."	PhysioNet Challenge
UCI Diabetes Dataset	Contains patient history and hospital readmission data.	Use this to train a simpler XGBoost model to predict "Deterioration Risk" for diabetic patients.	UCI Machine Learning Repository



3. For the "Input Processing" (OCR & BioBERT)

Goal: To handle messy handwriting and correct spelling mistakes in drug names.

Dataset Name	Description	Why you need it	Where to get it
IAM Handwriting Database	A collection of handwritten English text forms.	Use this to benchmark your Google Vision API or fine-tune a Tesseract model if you want offline OCR.	IAM Database
RxNorm	A standardized list of all clinical drug names and brand names.	Use this as your "Dictionary." Your BioBERT model compares the messy OCR text against *this list* to find the correct drug name.	NLM RxNorm

Here is the explanation of the entire AI Framework for FittFood. This connects every piece of technology we discussed—from reading the prescription to predicting heart attacks—into one cohesive system.

We call this the "Dual-Brain Architecture":

1. The Clinical Brain: Reads history and prescribes food.
2. The Predictive Brain: Watches live vitals and predicts danger.

Phase 1: The Perception Layer (Seeing & Reading)

Before the AI can think, it must read.

• Technology: Google Cloud Vision API (OCR).

• Function: It acts as the "Eyes." It scans the handwritten prescription image.

• The AI "Cleaner" (BioBERT):

• What it is: A Deep Learning model from Hugging Face (dmis-lab/biobert).

• Job: OCR often makes typos (e.g., "Rifampcn"). BioBERT reads the messy text and converts i into a Vector (Number). It compares this number to known drug names and finds the correct match ("Rifampicin").

• Why: This ensures we don't recommend food for the wrong drug due to a spelling error.

Phase 2: The Recommender Engine (The Logic Brain)

Determines the Meal Plan based on Medical Rules.

• Technology: Neo4j (Knowledge Graph) + PyTorch (TransE Model).

• Methodology: Knowledge Graph Embeddings.

• How it works (Step-by-Step):

1. Graph Lookup: The system looks for the "Drug Node" (Rifampicin) in Neo4j.
2. Vector Calculation: The TransE Model (which you trained) calculates the relationship vector: Drug + Depletes ≈ Nutrient.
3. Search: It finds all "Food Nodes" (e.g., Spinach, Chickpeas) that are mathematically close to that Nutrient vector.
4. Filtering: It applies hard rules (e.g., "User is Vegetarian") to filter the list.

• Result: A medically accurate list of ingredients (e.g., "Spinach, Dal").

Phase 3: The Early Warning System (The Intuition Brain)

Predicts "Critical Events" (like Sepsis) before they happen.

• Technology: LSTM (Deep Learning) + XGBoost (Machine Learning).

• Methodology: Multimodal Fusion (Combining History + Live Data).

• How it works:

1. Trend Detection (LSTM): This model looks at the last 6 hours of smartwatch data. It notices subtle trends (e.g., "Heart rate is creeping up slowly") that a human would miss.
2. Context Fusion (XGBoost): This model combines the Trend (from LSTM) with the History (from OCR).

• Logic: "Rising Heart Rate" (Trend) + "Patient has TB" (Context) = High Sepsis Risk.

3. Prediction: It outputs a "Risk Score" (0 to 1). If > 0.8, it triggers the Red Flag.

Phase 4: The Generation Layer (The Interface)

Turns data into human language.

• Technology: Llama-3 via Groq (Optional but recommended for UX).

• Role: The "Formatter."

• Job: It takes the raw list from Phase 2 (['Spinach', 'Dal']) and the risk from Phase 3 (Safe) and writes a polite message:

• "Namaste Ravi. Based on your medication, we have prepared a Spinach Dal for you today to boost your Vitamin B6."

• Safety: It is NOT allowed to decide the ingredients. It is only allowed to write the sentence using the ingredients provided by the Graph.