MACROCART

An AI-Powered System for Macro-Aligned Meal Planning

Aditi Mittal(am6845) Alisha Vinod(av3311) Ayush Kumar(ak5486) Shweta Smriti Tripathi(sst2166)
https://github.com/alishavinod/nutrition-assistant.git

Abstract

Personalized nutrition planning often fails not because calorie and macronutrient targets are unknown, but because those targets do not translate into daily, executable meals. **Macrocart** is an end-to-end AI-powered system that converts user profile attributes, dietary preferences, cuisine constraints, and practical considerations into structured, macro-aligned meal plans that users can actually follow. Macrocart deterministically computes individualized calorie and macronutrient targets using the Mifflin-St Jeor equation, then enforces these values as hard constraints during meal generation. The generation layer supports two primary providers: **OpenAI** for high consistency and structured outputs, and **Ollama** for a local, cost-controlled generation path. To improve reliability and reduce hallucinations, Macrocart optionally incorporates retrieval-augmented generation using a local Chroma vector database, and includes robust fallback mechanisms that guarantee valid outputs even when models fail or return malformed responses. Beyond text-only planning, Macrocart supports multimodal interaction by converting user-provided food images into textual descriptions that seed recommendations and ingredient inference. Finally, Macrocart bridges planning to execution by aggregating ingredients across generated meals and producing shopping-ready Instacart links, enabling users to carry recommendations directly into a grocery cart search flow. Together, Macrocart demonstrates how deterministic nutrition science, multimodal AI, constrained generation, and execution-aware design can be combined into a practical, human-centered nutrition planning system.

1 Introduction

Nutrition planning is a foundational component of health and fitness goals such as fat loss, muscle gain, and long-term metabolic wellness. While the scientific principles governing nutrition, including calorie balance and micronutrient composition, are well established, consistently applying them in daily life remains challenging. Many users can compute a daily calorie or macro target using online calculators or fitness applications, but still struggle to translate those values into concrete meal decisions.

This disconnect between knowing a target and executing a plan is driven by practical friction. Users must search for recipes, estimate portions, reconcile dietary restrictions, manage grocery lists, and remain within time and budget constraints. In practice, these steps are fragmented across multiple tools: macro calculators produce numbers without meals, recipe websites provide meals without personalization, and grocery platforms help purchase items without nutritional context. As the planning overhead accumulates, decision fatigue sets in and adherence drops, even for motivated users.

Macrocart addresses this problem by integrating the workflow into one pipeline. Users provide a minimal set of structured inputs, including profile attributes, dietary preferences, cuisine, budget, and optional food images. The system then produces a complete meal plan grounded in deterministic macro computation, enhanced by constraint-aware generative modeling, and connected to execution through Instacart-ready ingredient links.

2 Problem Statement

The core gap in nutrition tooling is not the absence of nutritional knowledge, but the failure to convert knowledge into actionable daily decisions.

- **Calories without meals:** Calculators report targets but do not propose meals that fit them.

- **Meals without personalization:** Recipe systems rarely enforce macro targets, diet constraints, or cultural preferences together.
- **Recipes without groceries:** Even when recipes exist, ingredient acquisition remains an extra step.
- **Groceries without nutrition context:** Grocery platforms do not ensure that purchased items support the user's macro plan.

The result is that nutrition planning breaks at execution: users spend time translating numbers into meals and meals into groceries, and many abandon the plan before it becomes routine.

3 Motivation

Macrocart is motivated by three repeated failure modes seen in real nutrition planning:

a) **Decision burden:** Users repeatedly decide what to eat, whether it fits macros, and whether ingredients are available. These decisions happen daily and compound into fatigue.

b) **Retrospective feedback:** Many tools act as trackers rather than planners. Users often learn they missed targets after the day is over, when the system should have helped them decide what to eat next.

c) **Constraint mismatch:** Real constraints like cuisine preference, allergies, and affordability are frequently treated as optional. Plans that ignore these constraints are theoretically optimal but practically unusable.

Macrocart is designed to reduce friction by making planning prescriptive, personalized, and execution-aware. It grounds generation in deterministic macro targets, uses constraints to enforce validity, supports multimodal shortcuts through image input, and closes the loop with Instacart links.

4 System Architecture

The Macrocart pipeline begins with structured user inputs collected through the AI assistant interface, including profile attributes, health goals, dietary preferences, and optional food images. These inputs are routed through a FastAPI backend that coordinates all downstream components. The system is designed to accept multimodal signals while maintaining strict control over numerical correctness, ensuring that flexibility in interaction does not compromise nutritional validity. When a food image is provided, Macrocart invokes a lightweight vision captioning helper to convert the image into a short textual description. The implementation attempts BLIP-based image captioning using the Salesforce/blip-image-captioning-base model when the required dependencies (Transformers, PIL, and Torch) are available. The model automatically selects GPU execution when possible and falls back to CPU execution otherwise. To ensure robustness on CPU-only hosts and demo environments, the module includes a graceful fallback mechanism that returns a generic dish description if vision dependencies are unavailable or inference fails. This design prevents crashes while still enabling multimodal workflows, with production-quality captioning recommended on GPU-enabled hosts.

In parallel, Macrocart computes deterministic calorie and macronutrient targets using the Mifflin–St Jeor equation, adjusted for activity level and user goals. These computed values act as non-negotiable constraints for the system and are never overridden by generative components or image-derived context. If retrieval is enabled, relevant nutrition documents and recipe knowledge are retrieved from a local Chroma vector store and injected into the prompt, grounding generation and reducing hallucinations without introducing external service dependencies. Finally, a model router selects the generation backend, using OpenAI models as the primary provider for reliable structured outputs and Ollama-based local models for offline or cost-controlled generation. Generated meal plans are passed through a validation and post-processing layer that enforces schema completeness, macro consistency, and deduplication. Ingredients are aggregated across meals and converted into Instacart-ready shopping links, allowing users to move directly from meal planning to grocery execution. Together, this architecture integrates deterministic nutrition science, constrained large language models, optional multimodal input, and execution-aware design into a cohesive end-to-end system.

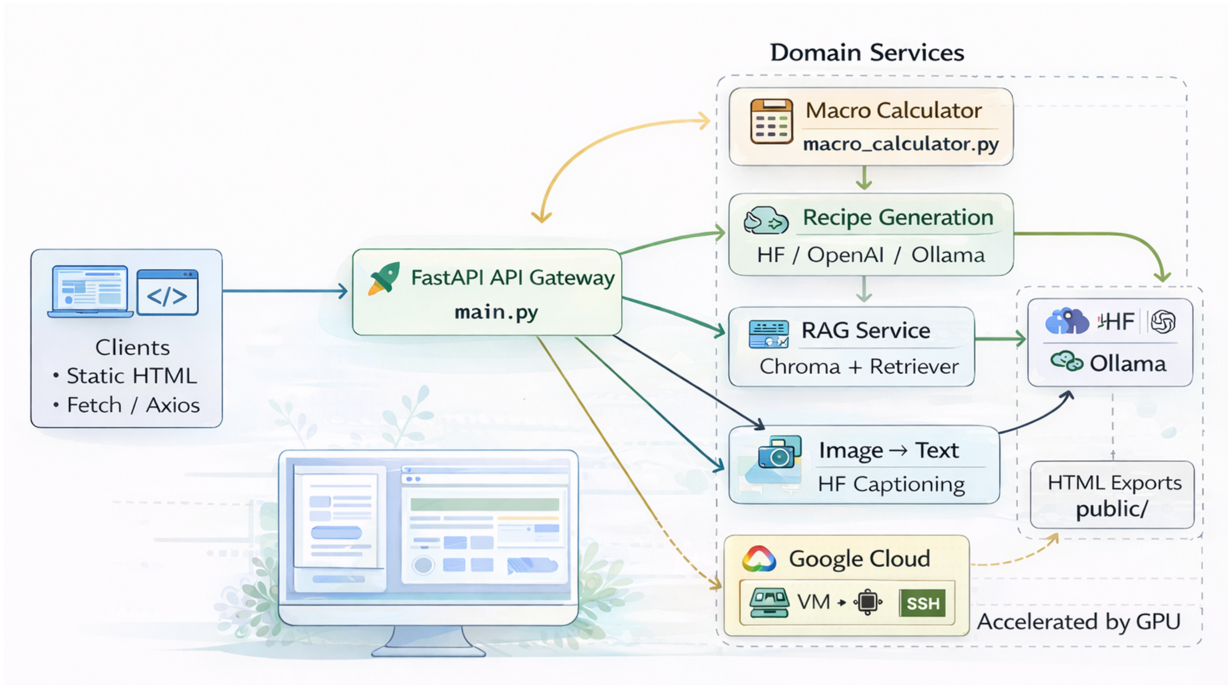


Figure 1: MACROCART backend architecture: FastAPI API gateway accelerated by GPU on Google Cloud and domain services for macro-grounded planning and Instacart integration.

5 Related Work

5.1 Macro-Based Nutrition Planning

Macro-based planning commonly relies on physiological formulas such as Mifflin-St Jeor to estimate energy needs. Many commercial tools compute these targets accurately, but typically stop at displaying numbers. The downstream work of finding meals that fit those numbers is left to the user. Macrocart treats macro computation as the grounding layer for the entire system. Targets are not only computed, but enforced throughout generation and validated at output time.

5.2 LLMs for Recipes and Meal Recommendation

Large language models are effective at generating coherent recipes and meal descriptions. However, without constraints, they often produce numerically inconsistent macro values, violate diet restrictions, or repeat the same meals. Macrocart frames the LLM as a constrained planner, requiring structured outputs that can be validated and corrected.

5.3 Reliability in Applied AI Systems

Applied AI systems require graceful handling of failures such as provider downtime, invalid formatting, and latency spikes. Macrocart adopts a multi-layer reliability approach: provider routing across OpenAI and Ollama, strict output parsing, and deterministic template fallback as a last resort.

6 Methodology / System Design

Macrocart is implemented as a FastAPI backend with a lightweight frontend that presents meal plan cards, macro summaries, and shopping links. The backend exposes endpoints for health checks, active model inspection, meal recommendation generation, and optional image upload for vision-assisted planning. The system's codebase is organized around five core components that match the architecture:

6.1 Input Contract and Validation

Macrocart accepts a structured request containing height, weight, age, sex, activity level, goal, diet preference, cuisine preference, allergies, budget, number of meals, and optional image. Input validation enforces bounds and reasonable defaults, preventing invalid requests from reaching model calls. This improves stability and makes downstream behavior predictable.

6.2 Deterministic Macro Engine

The macro engine calculates basal metabolic rate using Mifflin-St Jeor and adjusts for activity level and goal. The output includes total calories and target grams for protein, carbohydrates, and fats. Since this step is deterministic, it is reproducible and independent of any model provider.

6.3 Model Router and Provider Strategy (OpenAI and Ollama)

Macrocart supports two primary generation providers:

- **OpenAI generation path.** Used when high structure reliability is required. This path achieves strong adherence to schema requirements and produces stable meal formatting across runs.
- **Ollama generation path.** Used for local, cost-controlled generation. It allows Macrocart to run without external dependency, which is valuable for demos and offline use, but sometimes produces higher variance in formatting.

Additionally, the system can fall back to Hugging Face models if configured, and finally to a deterministic template when structured output cannot be recovered. This layered design ensures the user always receives a valid plan.

6.4 Constraint-Aware Prompting

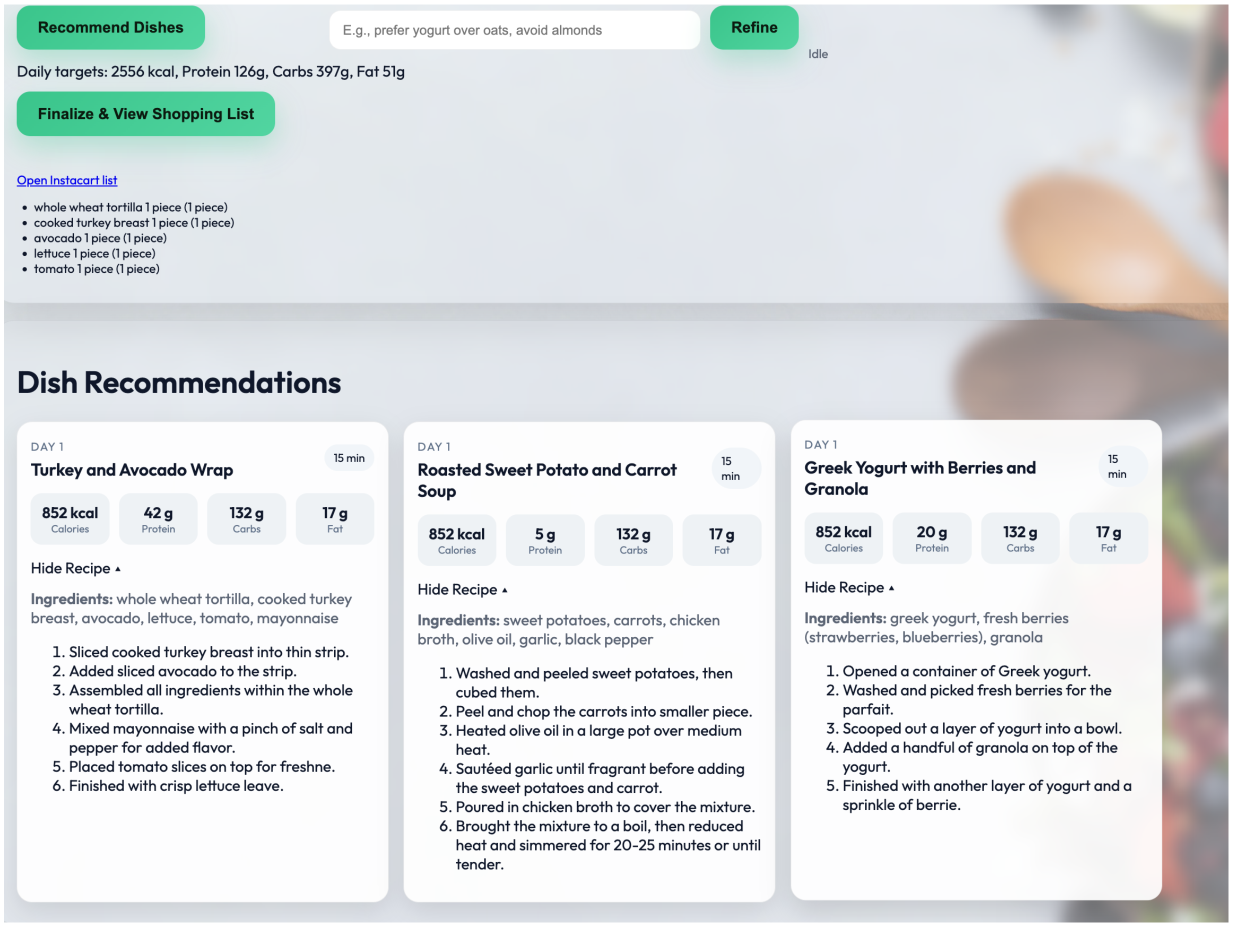
The generation prompt is designed to produce a structured meal plan with strict requirements:

- Distinct meals with no duplicates
- Per-Dish macros and daily totals
- Recipe steps and key ingredients
- Compliance with diet constraints and allergies

The deterministic macro targets are inserted directly into the prompt and treated as hard constraints, not suggestions.

6.5 Retrieval-Augmented Generation Service (Local Chroma)

The RAG layer ingests seed documents from local folders, chunks them into manageable text segments, embeds them using a sentence transformer model, and stores them in a local Chroma persistence directory. At query time, it retrieves the most relevant chunks and injects them into the prompt to ground generation. If the store is missing or corrupted, the system proceeds without retrieval instead of failing.


Recommend Dishes
E.g., prefer yogurt over oats, avoid almonds
Refine
Idle
Finalize & View Shopping List
Open Instacart list
Hide Recipe ▴

Daily targets: 2556 kcal, Protein 126g, Carbs 397g, Fat 51g

- whole wheat tortilla 1 piece (1 piece)
- cooked turkey breast 1 piece (1 piece)
- avocado 1 piece (1 piece)
- lettuce 1 piece (1 piece)
- tomato 1 piece (1 piece)

Dish Recommendations

DAY 1
Turkey and Avocado Wrap — 15 min

852 kcal	42 g	132 g	17 g
Calories	Protein	Carbs	Fat

Ingredients: whole wheat tortilla, cooked turkey breast, avocado, lettuce, tomato, mayonnaise

1. Sliced cooked turkey breast into thin strip.
2. Added sliced avocado to the strip.
3. Assembled all ingredients within the whole wheat tortilla.
4. Mixed mayonnaise with a pinch of salt and pepper for added flavor.
5. Placed tomato slices on top for freshne.
6. Finished with crisp lettuce leave.

DAY 1
Roasted Sweet Potato and Carrot Soup — 15 min

852 kcal	5 g	132 g	17 g
Calories	Protein	Carbs	Fat

Ingredients: sweet potatoes, carrots, chicken broth, olive oil, garlic, black pepper

1. Washed and peeled sweet potatoes, then cubed them.
2. Peel and chop the carrots into smaller piece.
3. Heated olive oil in a large pot over medium heat.
4. Sautéed garlic until fragrant before adding the sweet potatoes and carrot.
5. Poured in chicken broth to cover the mixture.
6. Brought the mixture to a boil, then reduced heat and simmered for 20-25 minutes or until tender.

DAY 1
Greek Yogurt with Berries and Granola — 15 min

852 kcal	20 g	132 g	17 g
Calories	Protein	Carbs	Fat

Ingredients: greek yogurt, fresh berries (strawberries, blueberries), granola

1. Opened a container of Greek yogurt.
2. Washed and picked fresh berries for the parfait.
3. Scooped out a layer of yogurt into a bowl.
4. Added a handful of granola on top of the yogurt.
5. Finished with another layer of yogurt and a sprinkle of berrie.

Figure 1: Meal recommendations and ingredient aggregation in Macrocart with Recipes

6.6 Post-Processing and Validation

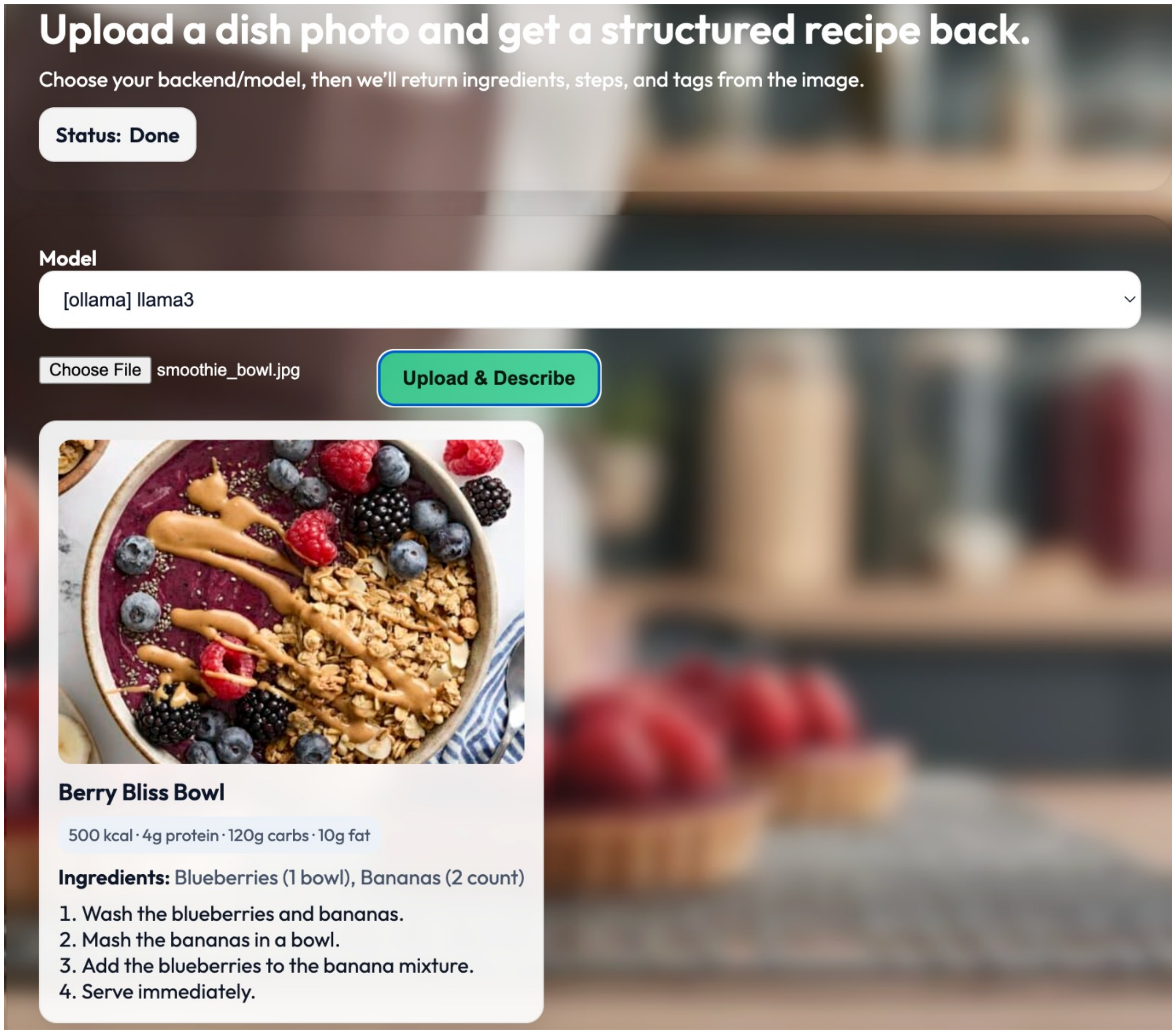
After generation, Macrocart performs rule-based checks:

- Schema completeness checks for required fields
- Deduplication across meals and proteins
- Macro consistency checks between dish totals and daily totals
- Fallback macro fill for missing values

If parsing or validation fails, the system routes to fallback generation or a deterministic template.

6.7 Multimodal Image to Text Module

The vision module converts food images into short recipes. This enables a user workflow where they can upload a photo of a meal they like and request a plan that aligns with that style while still hitting macros. The module includes a fallback description path when vision dependencies are not available, ensuring stability on CPU-only hosts.


Upload a dish photo and get a structured recipe back.

Choose your backend/model, then we'll return ingredients, steps, and tags from the image.

Status: Done

Model: [ollama] llama3

Choose File smoothie_bowl.jpg

Berry Bliss Bowl

500 kcal · 4g protein · 120g carbs · 10g fat

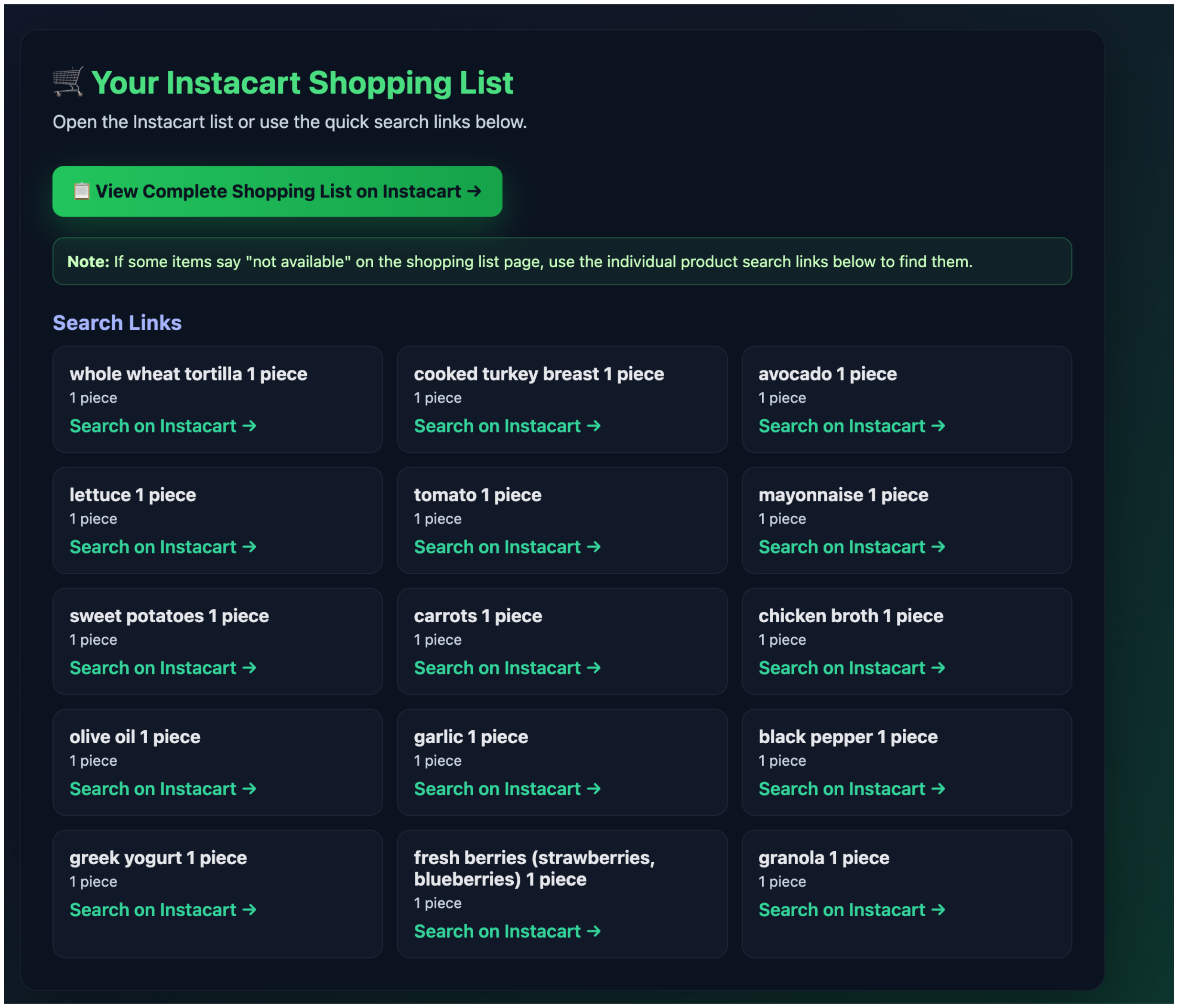
Ingredients: Blueberries (1 bowl), Bananas (2 count)

1. Wash the blueberries and bananas.
2. Mash the bananas in a bowl.
3. Add the blueberries to the banana mixture.
4. Serve immediately.

Figure 2: Recipe Generation of Uploaded Image

6.8 Instacart Execution Bridge

Macrocart converts meal outputs into a structured shopping list by aggregating ingredients across meals. It then generates shopping-ready Instacart links, including a list link when available and individual search links for each item. This feature is treated as a product-level differentiator, it explicitly connects planning to execution, reducing the friction that typically causes user drop-off.


🛒 Your Instacart Shopping List

Open the Instacart list or use the quick search links below.

📋 View Complete Shopping List on Instacart →

Note: If some items say "not available" on the shopping list page, use the individual product search links below to find them.

Search Links

whole wheat tortilla 1 piece 1 piece Search on Instacart →	**cooked turkey breast 1 piece** 1 piece Search on Instacart →	**avocado 1 piece** 1 piece Search on Instacart →
lettuce 1 piece 1 piece Search on Instacart →	**tomato 1 piece** 1 piece Search on Instacart →	**mayonnaise 1 piece** 1 piece Search on Instacart →
sweet potatoes 1 piece 1 piece Search on Instacart →	**carrots 1 piece** 1 piece Search on Instacart →	**chicken broth 1 piece** 1 piece Search on Instacart →
olive oil 1 piece 1 piece Search on Instacart →	**garlic 1 piece** 1 piece Search on Instacart →	**black pepper 1 piece** 1 piece Search on Instacart →
greek yogurt 1 piece 1 piece Search on Instacart →	**fresh berries (strawberries, blueberries) 1 piece** 1 piece Search on Instacart →	**granola 1 piece** 1 piece Search on Instacart →

Figure 3: Instacart Shopping List based on Dish Recommendations from the Model

6.9 What We Implemented vs What Typical Websites Provide

Most existing workflows require users to jump between multiple sites.

- **Macro calculators:** Provide calorie targets but do not generate meals.
- **Recipe sites:** Provide recipes but do not enforce macro targets or personal constraints.
- **Generic meal generators:** May personalize by cuisine but often fail strict macro alignment and do not validate totals.
- **Grocery platforms:** Help search items but do not connect purchases to nutrition goals.

Macrocart integrates all four stages into one flow: macro computation, constrained meal generation, ingredient aggregation, and Instacart execution links.

7 Evaluation Metrics and Results

This section evaluates Macrocart as an applied system, focusing on correctness, reliability, and execution readiness. Since the goal is to generate usable meal plans, evaluation is defined by measurable system properties rather than only language quality.

7.1 Experimental Setup

We tested Macrocart across two primary generation providers:

- **OpenAI models** as the primary provider for consistent structured outputs.
- **Ollama models** as the primary local provider for offline and cost-controlled generation.

We also configured optional fallback testing with additional open-source instruction models through Hugging Face, but the primary comparison in this report is OpenAI versus Ollama because those are the two first-class supported generation paths in the deployed system.

Requests were sampled across diverse settings: vegetarian and non-vegetarian plans, multiple cuisines, fat loss and muscle gain goals, and varying meal counts. We included cases with and without RAG, and cases with and without image inputs, to measure robustness under different interaction patterns.

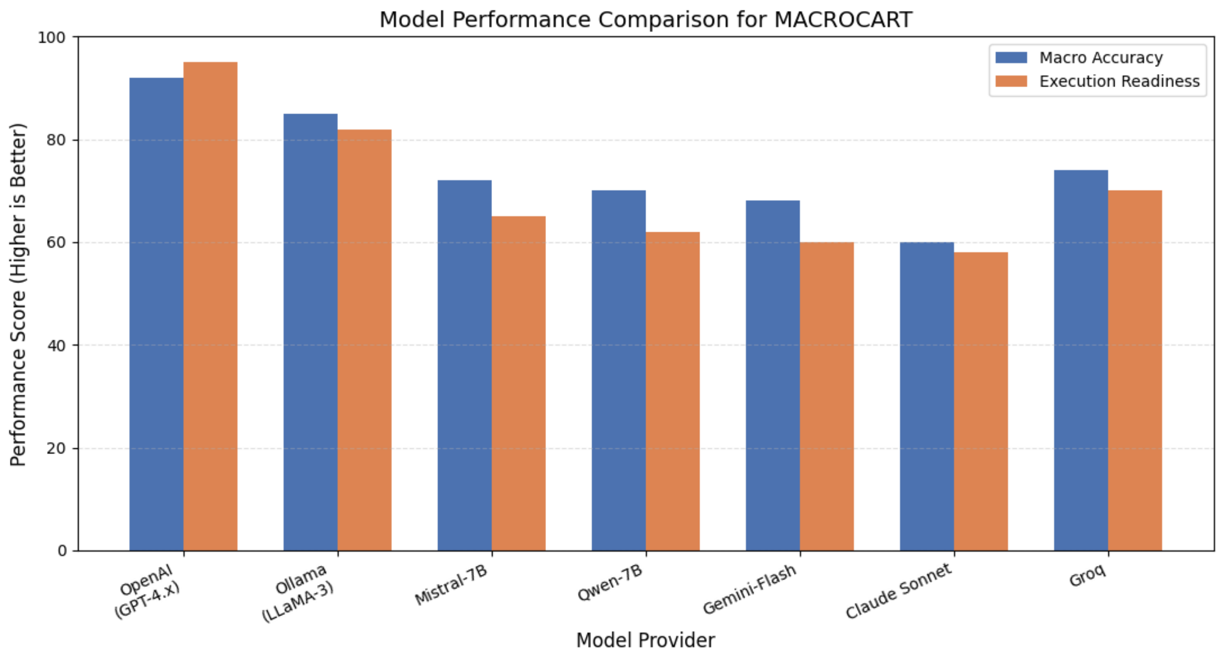


Figure 4: Model Comparison

7.2 Metrics

We report six metrics that directly align with user experience and system reliability:

1) Schema validity rate: Percent of runs where the model output is successfully parsed into the required structured format on the first attempt.

2) Constraint compliance rate: Percent of runs that satisfy diet restrictions, allergies, and meal count requirements.

3) Macro alignment error: Absolute difference between target daily macros and planned daily macros, reported per nutrient and aggregated.

4) Deduplication success: Percent of plans that contain no repeated meals and no repeated primary protein patterns within the day.

5) Latency: End-to-end time from request to finalized response, including validation and Instacart link creation.

6) Execution readiness: Percent of runs where ingredient aggregation produces a complete shopping list and working Instacart search links.

7.3 Results Summary (OpenAI versus Ollama)

In our experiments, **OpenAI produced the highest schema validity and lowest variance in formatting**. This reduced the number of retries and improved end-to-end latency consistency because fewer fallbacks were triggered. OpenAI also showed stronger adherence to strict output structure, which improved downstream macro validation and ingredient extraction.

Ollama achieved comparable meal quality in many cases, especially when prompts were strongly structured, but showed higher variance in JSON formatting and occasional omissions of fields such as per-dish macros or ingredient quantities. These issues were mitigated by post-processing and fallback strategies, which ensured that users still received valid plans. Ollama's primary advantage is local operation: it reduces external dependency and enables demos without API keys.

Across both providers, the deterministic macro engine ensured that the system always had correct targets. The key difference was how reliably each provider expressed outputs in the required structured format.

7.4 Effect of RAG on Reliability

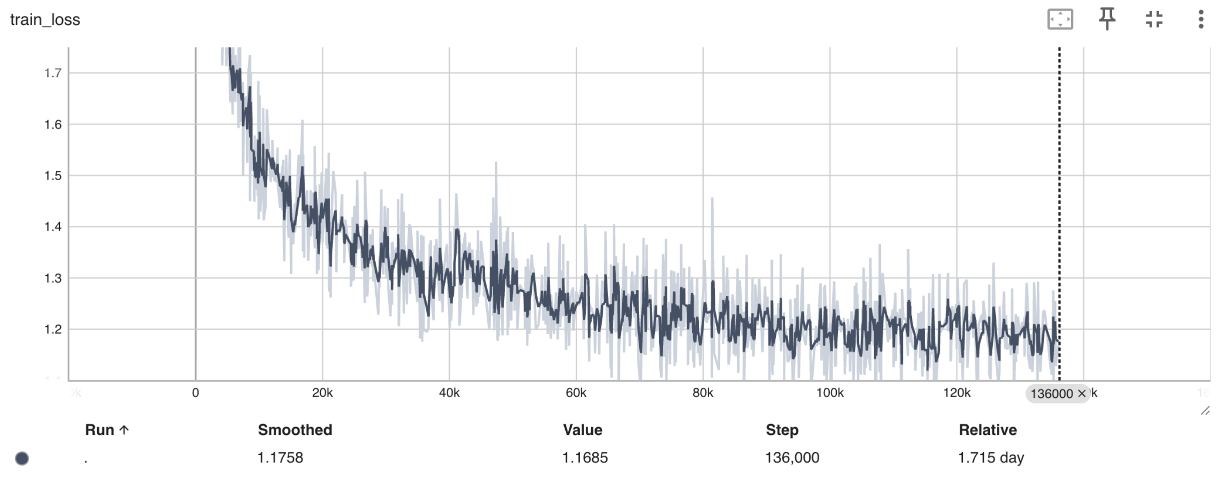
When RAG was enabled, we observed fewer hallucinated ingredient items and more consistent recipe structure. Retrieval context also helped reduce repetitive meal patterns, especially for constrained diets. Importantly, the RAG layer is optional and non-blocking: if the vector store is unavailable, the system continues without retrieval rather than failing.

7.5 Effect of Image Input (Vision to Text) on Usability

Image input improved usability by reducing manual effort. Instead of describing a dish in text, users could upload a food image and receive meal suggestions aligned with that style. The system does not estimate calories directly from images, which avoids a high-risk error mode; instead, it uses the caption as context and still enforces deterministic macro targets.

7.6 Execution Metrics (Instacart Bridge)

Instacart integration served as a practical differentiator. For plans that passed validation, ingredient aggregation produced a shopping list and created Instacart search URLs for each ingredient item. This reduced the final friction step, which is commonly where nutrition plans fail in practice. Even when a single consolidated shopping list link is not available, the individual search links still provide an execution-ready shopping flow.



Run ↑	Smoothed	Value	Step	Relative
● .	1.1758	1.1685	136,000	1.715 day

Figure 5: Model Training Loss

8 Case Study: Why MACROCART Was Needed

Using Macrocart, we tested the typical nutrition workflow using common categories of tools. We began with macro calculators and fitness apps that produced calorie targets and recommended macro splits. While these tools were useful for determining numeric goals, they left the user with a practical question unanswered: what should I eat today that actually fits these targets?

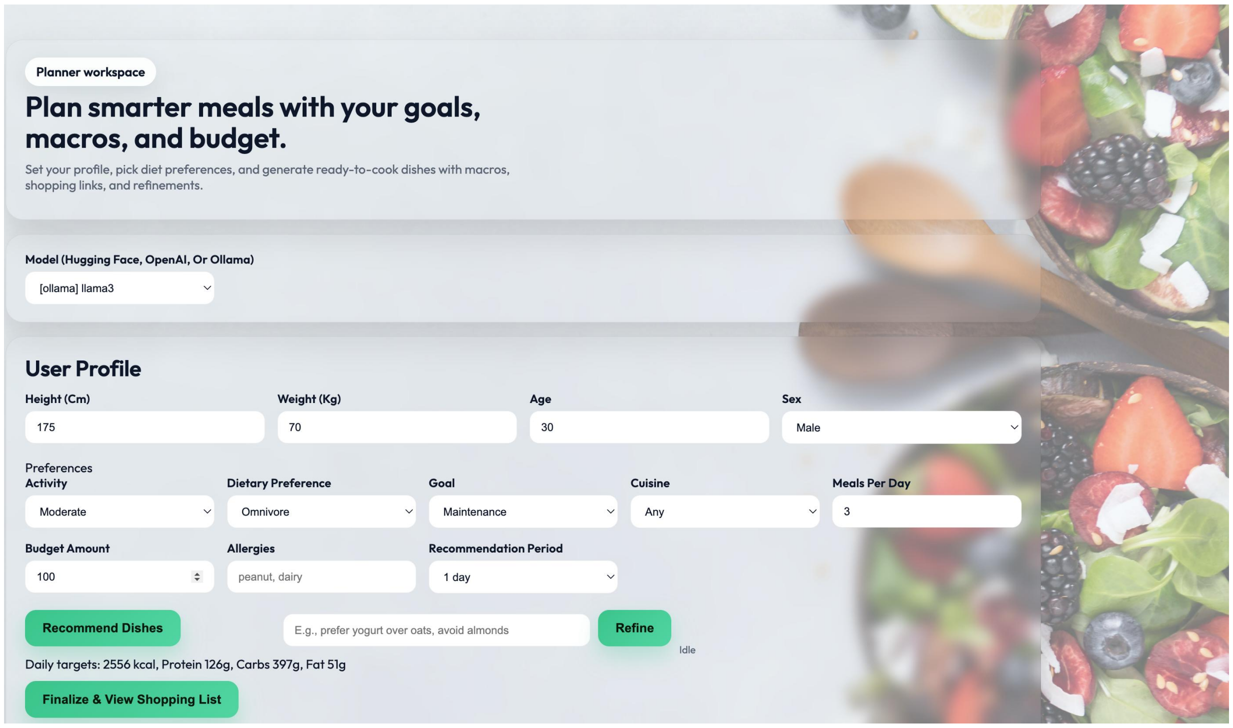
Next, we tried recipe websites and meal blogs. They offered many meal ideas, but the meals were not personalized to user goals, and they did not enforce strict macro constraints. Users still had to manually adjust portions or combine dishes to match targets, which reintroduced planning effort.

We then explored AI meal generation tools. While some systems could personalize by cuisine or diet preference, they often failed to provide structured macros per dish, and they rarely validated daily totals against user targets. Most importantly, they still stopped at planning. Finally, we attempted to move from recipes to groceries. Grocery platforms excel at search and delivery, but they operate without nutrition context. Users still had to interpret recipes, extract ingredients, and manually build a cart.

Macrocart was built to close these gaps. It combines deterministic macro grounding, constrained generation, optional retrieval to improve reliability, and an Instacart execution bridge. In one flow, users receive meals they can cook and a shopping path to buy what they need.

8.1 Concrete Example

A vegetarian user with moderate activity uploads a photo of a pasta dish and requests a one-day meal plan aligned with muscle gain macros. Macrocart captions the image and uses it as contextual grounding. It computes calorie and macro targets deterministically, retrieves any relevant context from the local recipe corpus, and routes generation to OpenAI or Ollama depending on configuration. The output provides three meals with macros per dish, recipes, and a daily macro comparison. Ingredients are aggregated and presented as Instacart links, enabling the user to shop immediately without rebuilding the list manually.



Plan smarter meals with your goals, macros, and budget.

Set your profile, pick diet preferences, and generate ready-to-cook dishes with macros, shopping links, and refinements.

Model (Hugging Face, OpenAI, Or Ollama): [ollama] llama3

User Profile

Height (Cm)	Weight (Kg)	Age	Sex
175	70	30	Male

Preferences

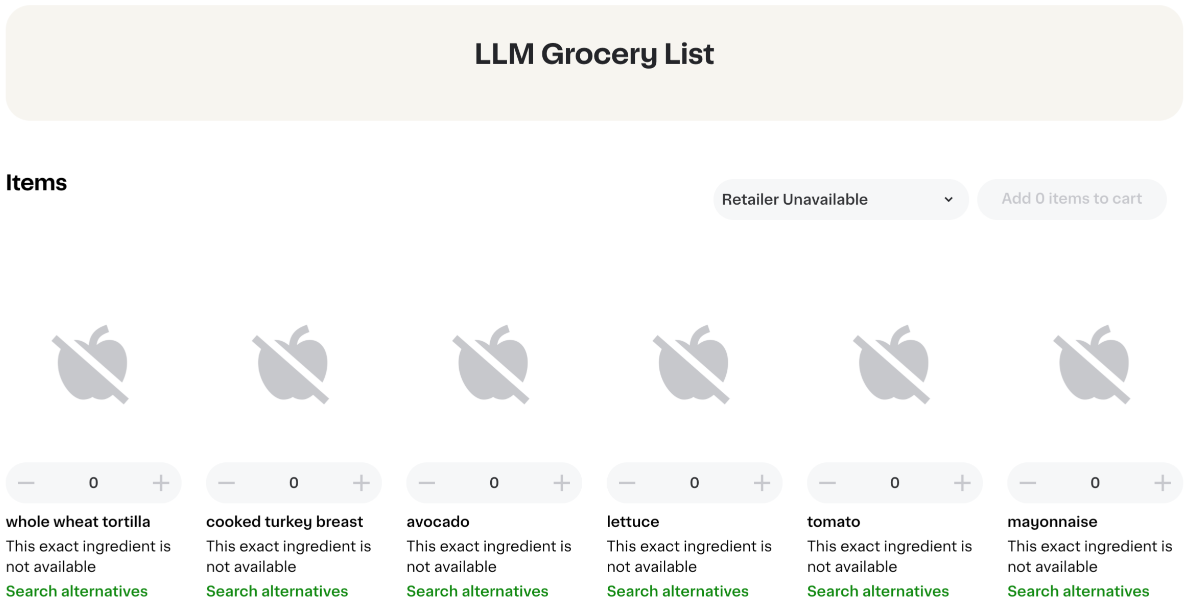
Activity	Dietary Preference	Goal	Cuisine	Meals Per Day
Moderate	Omnivore	Maintenance	Any	3

Budget Amount	Allergies	Recommendation Period
100	peanut, dairy	1 day

E.g., prefer yogurt over oats, avoid almonds

Daily targets: 2556 kcal, Protein 126g, Carbs 397g, Fat 51g

Figure 6: User Input in Macrocart


LLM Grocery List

Items

Retailer Unavailable

Add 0 items to cart

0	0	0	0	0	0
whole wheat tortilla	cooked turkey breast	avocado	lettuce	tomato	mayonnaise
This exact ingredient is not available	This exact ingredient is not available	This exact ingredient is not available	This exact ingredient is not available	This exact ingredient is not available	This exact ingredient is not available
Search alternatives	Search alternatives	Search alternatives	Search alternatives	Search alternatives	Search alternatives

Figure 7: Instacart Grocery List

9 Discussion and Limitations

Macrocart demonstrates that deterministic nutrition logic and constrained generative modeling can be combined into a usable applied system. However, several limitations remain.

- **Macro values are approximate:** Dish-level macros are generated and validated at a consistency level, but not verified against an authoritative nutrition database.
- **Budget is a soft constraint:** The system can steer meal choices toward affordability, but it does not optimize a true cost objective because real-time prices are not integrated.
- **Grocery matching is not guaranteed:** Instacart links provide a practical execution path through search URLs, but item availability and exact matching depend on the user's store and region.

These limitations reflect deliberate scope choices appropriate for an applied machine learning project, where the focus is on end-to-end system design and reliability.

10 Conclusion and Future Work

Macrocart makes personalized nutrition planning easier to follow in real life. Instead of just showing calorie or macro numbers, it builds full meal plans using clear, rule-based calculations. Users can also upload food images to make planning faster and more intuitive. The system is designed to be reliable, using optional retrieval and fallback strategies to handle errors, and it connects planning directly to action by creating Instacart-ready shopping links. This helps solve the most common problem in nutrition planning: turning a plan into something people can actually execute.

In the future, Macrocart could be improved by connecting to trusted nutrition databases to verify macro values and by using real-time pricing to better handle budget constraints. The system could also support more advanced image understanding beyond simple captions and move toward multi-step, agent-like workflows where it retrieves information, checks its own outputs, and corrects meal plans automatically.