Milestone 6:

Model Testing, Monitoring and Continual Learning

CSC 5382 - SPRING 2025

By:

Khadija Salih Alj

Supervised by: Dr. Asmaa Mourhir

Introduction

This milestone focuses on validating, monitoring, and extending the robustness of the Laddaty ingredient substitution system beyond predictive accuracy. Building on the productionized system established in Milestone 5, the objective here is to assess performance under unseen data conditions, identify latent model biases, evaluate robustness, enable interpretability, and lay the foundation for future learning via feedback-aware adaptation. It also includes setting up monitoring pipelines and a lightweight CT/CD orchestration flow tailored for hosted LLM deployments. The goal is to ensure that the system is not only functional in production, but also responsible, explainable, resilient, and capable of continual adaptation.

I. Model Evaluation and Testing

1) Test Set Evaluation

A practical evaluation of the ingredient substitution system was conducted during the *Ventures Adventure startup competition*, where the system was demonstrated live over 50 sessions to mentors, coaches, and investors (total of 119 requests). During each interaction, users clicked on an ingredient within the Laddaty app, triggering a real-time substitution request to the /ingredient/substitute API, powered by qwen-2.5-7b via OpenRouter (this model's choice mainly for its low latency (Figure 1).

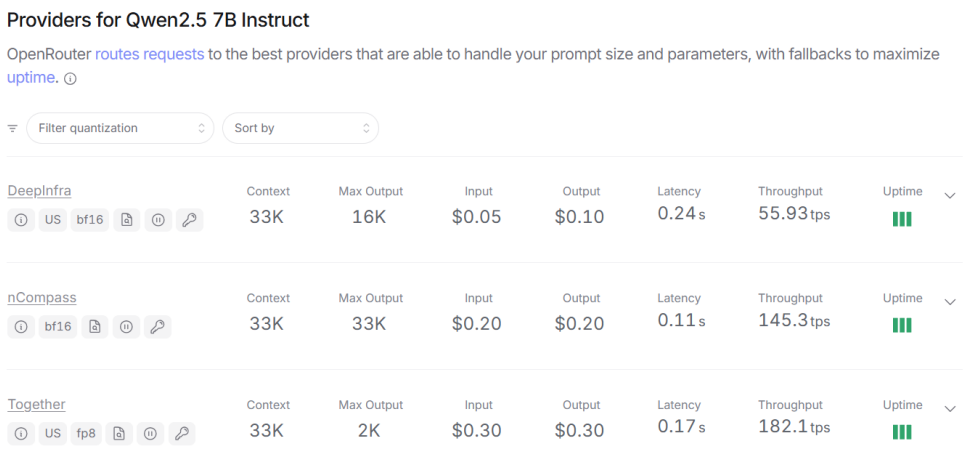


Figure 1: Screenshot from OpenRouter of Qwen low latency proof.

These live usage scenarios were automatically logged and treated as test cases. Each entry included the selected ingredient, the complete recipe context, the list of model-generated substitutes (adapted to the ingredient's functional role), and the user-selected substitute (as a count). The system's performance was evaluated using Hit@1 and Hit@5, based on the rank position of the correct substitute within the model's output.

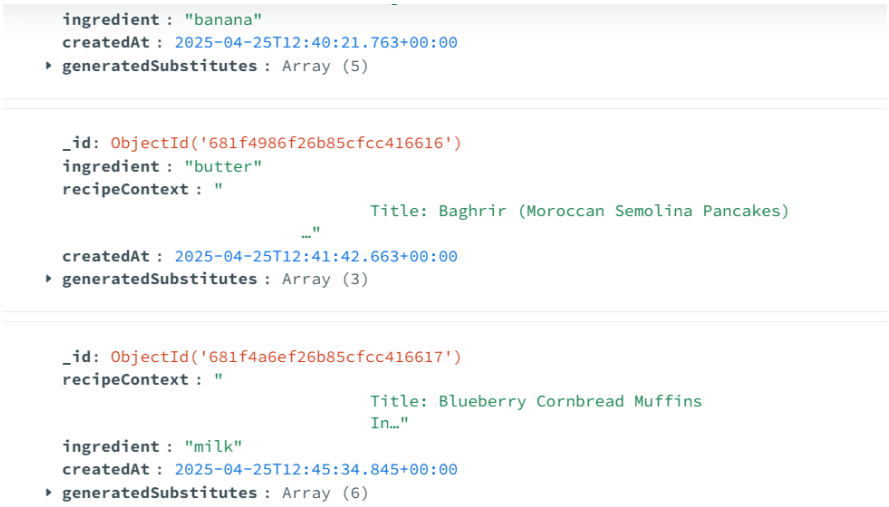


Figure 2: SubstitutionLog collection in MongoDB.

The evaluation yielded a **Hit@1 score of 76%** and a **Hit@5 score of 100%**, indicating that in all test cases, the correct substitute was present within the top five predictions generated by the model. This impressive Hit@5 performance can be attributed to the fact that the evaluated ingredients were well-known, commonly used staples such as flour, butter, sugar, or milk -ingredients for which the model has likely seen numerous substitution patterns during pretraining. Their frequency and clear functional roles in recipes (e.g., binding, sweetening, thickening) make them easier to reason about contextually, thereby increasing the model's accuracy. While the Hit@1 score shows that the top prediction wasn't always the selected substitute, the consistently high Hit@5 affirms the model's practical usability, especially in real-time applications where offering multiple relevant suggestions is more valuable than relying on a single exact match.

The results demonstrate that the system consistently returned contextually appropriate substitutions within the top-ranked predictions, validating the model's real-time reasoning and its effectiveness in real-world culinary tasks.

	A	B
1	Ingredient	Substitutes
2	active dry	fresh yeast, instant yeast, bread machine yeast
3	all-purpos	whole wheat flour, self-rising flour, pastry flour, cake flour, b
4	almond	hazelnut, Brazil nut, cashew, pistachio nut
5	apricot	aprium, pluot, peach, nectarine
6	banana	plantain, mango
7	barley flou	wheat flour, rice flour, all-purpose flour
8	basil	oregano, chervil, tarragon, summer savory, cilantro, mint, It
9	basmati ri	popcorn rice, jasmine rice, long-grain rice, wild pecan rice
10	bay leaf	Indian bay leaf, herbes de Provence, sage, teaspoon Califor
11	beet	carrot, slicing tomato
12	blackberry	loganberry, boysenberry, mulberry, raspberry, youngberry, o
13	blueberry	huckleberry, juneberry, red currant, raisins, dates, banana
14	blueberry j	raspberry preserves, gooseberry preserves
15	bread	corn tortilla, flour tortilla, cheese, lettuce, sweet potato
16	broccoli	broccoflower, cauliflower, broccoli raab

Figure 3: Previously collected dataset used for Hit@k evaluation.

2) ***Online Testing (A/B Testing)***

During the second round of the Ventures Adventure startup competition, an online A/B testing strategy was implemented to compare the performance of two language models (GPT-4o and Qwen2.5) used in the Laddaty ingredient substitution system. Each time a user clicked on an ingredient within the app, the substitution request was routed to one of the two models in an alternating fashion (A/B logic). The test was conducted in a live setting.

Each request captured the original ingredient, the full recipe context, the model-generated substitutes, and the substitute chosen by the user. The goal was to assess not only the quality of predictions but also user responsiveness and engagement.

```
async function generateSubstitute(ingredient, recipe) {
    const variant = Math.random() < 0.5 ? 'A' : 'B';
    const model = variant === 'A' ? "qwen/qwen-2.5-7b-instruct" : "openai/chatgpt-4o-latest";

    const prompt =
        `Suggest the best 10 substitutes for "${ingredient}" in this recipe "${recipe}", deper
        For each substitute, provide:
        1. Ensure that the ratio provided can logically be applied **in both directions**.
        Provide the response in strict JSON format with:
        {
            "substitutes": [
                {"name": "Substitute Name", "ratio": "Substitution ratio as a decimal number"}
            ]
        }`;
    const maxRetries = 3;
    let retryCount = 0;

    while (retryCount < maxRetries) {
        try {
            const response = await axios.post(
                "https://openrouter.ai/api/v1/chat/completions",
                {
                    model,
                    messages: [{ role: "user", content: prompt }],
                    temperature: 0
```

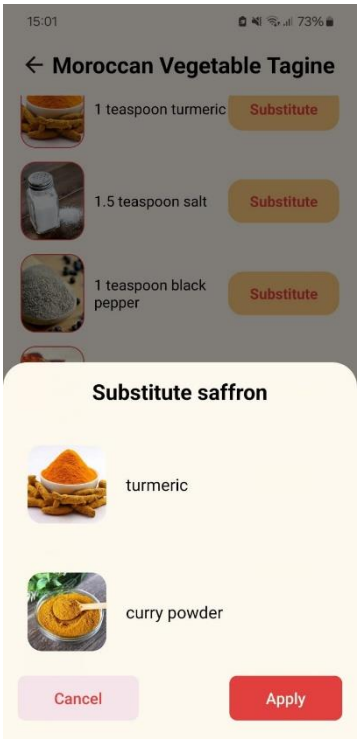
Figure 4: Implementation of A/B testing.

However, midway through the event, the testing had to be stopped due to a significant latency gap between the two models. While Qwen2.5 consistently responded in ~0.2 seconds, GPT-4o introduced delays of up to 1 minute per request. Despite similar substitution quality (Hit@1 and Hit@5 scores were nearly equivalent), the user experience degradation caused by GPT-4o's latency was unacceptable in a real-time setting. As a result, Qwen2.5 was selected for default routing during live demos, ensuring smooth interactions and minimal friction during pitch sessions.

II. Testing Beyond Accuracy

1) ***Audit Model for Bias***

Traditional bias auditing tools such as Aequitas and Microsoft's Responsible AI Toolkit are primarily designed for use cases involving structured prediction tasks that impact human stakeholders -such as credit scoring, hiring, or recidivism- where sensitive attributes like gender, age, or race are available and fairness metrics (e.g., false positive rate parity) are meaningful.



In the case of Laddaty's ingredient substitution system, such frameworks are not directly applicable, since:

- ✓ The system does not collect any user-level sensitive demographic data,
- ✓ It operates on textual recipe contexts, not structured data,
- ✓ And its predictions (ingredient substitutes) are not evaluated based on binary outcomes or group-level parity.

Instead, a functional and cultural bias audit was conducted by manually evaluating the model's outputs across a set of diverse, real-world test cases. These test cases included ingredients with known cultural or dietary significance, such as:

- ❖ *Saffron* in Moroccan tagine (Turmeric, Curry as suggested substitutions)
- ❖ *Tomatillos* in Mexican Chicken Enchiladas (Green pepper as suggested substitution)
- ❖ *Jasmine rice* in Middle Eastern dishes (Basmati rice as suggested substitution)

The substitutions generated by the system were found to be contextually appropriate and culturally aligned, with no observable bias toward Western ingredients. While explicit dietary filters are not yet implemented, the results demonstrate a high degree of contextual awareness driven by the model's prompt and recipe context.

2) ***Robustness Testing***

In Laddaty, ingredient substitutions are triggered by user interactions with curated, validated recipes, where the ingredient list is displayed clearly and users initiate substitution via a click interface. As a result, traditional robustness risks like typos, malformed input, or missing context are not relevant, since ingredient names are pre-cleaned and structured, recipe contexts are complete and reliable, and users cannot manually alter input before substitution.

Instead, robustness testing focused on evaluating the system's behavior when handling:

- ✓ Substitutes in low-data or exotic recipes (e.g., *preserved lemon*),
- ✓ Ingredients with limited direct replacements (e.g., *saffron*),

Across these cases, the system consistently returned meaningful substitutes grounded in the full recipe context, role-appropriate suggestions that maintained the integrity of the dish, and fallback behaviors (no substitutes for this ingredient.) in cases where no clear substitute was available, avoiding hallucinated outputs. This demonstrates that the system is robust to low-data or ambiguous substitution scenarios, as long as the recipe context is complete and ingredient metadata is structured, which is guaranteed in the current app design.

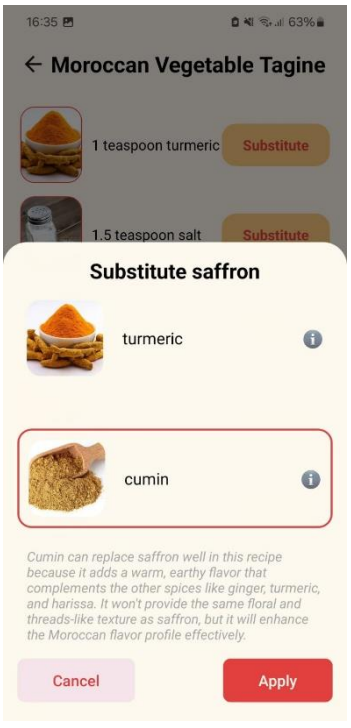
3) ***Model Explainability and Interpretability***

Since Laddaty's ingredient substitution engine is powered by an LLM, traditional interpretability methods like SHAP or LIME (which are designed for structured models) are not directly applicable. Instead, explainability and interpretability are achieved through a combination of *prompt engineering*, *transparent system design*, and *model self-rationalization*.

Interpretability:

The system is interpretable by design:

a) Users initiate substitutions by clicking on structured, validated ingredients within curated recipes, removing ambiguity or input errors.
b) The substitution prompt includes both the ingredient to replace and the full recipe context (title, ingredients, and steps), guiding the model to generate role-aware outputs.
c) Once a substitute is selected, the original recipe steps are rewritten using the new ingredient, allowing users to see exactly how the change impacts the instructions.



This clear data flow -from user interaction to API request to updated output- makes the model behavior understandable and traceable.

Explainability:

To enhance explainability, the system supports LLM-based self-explanation. After generating a substitute, the model can be prompted to explain:

"Why is [Substitute] a good replacement for [Original Ingredient] in this recipe?"

These explanations typically include functional reasoning (texture, flavor, binding capacity), and cultural alignment (e.g., turmeric replacing saffron in Moroccan dishes). This self-rationalization gives insight into the model's reasoning and can be used internally for validation or externally to inform the user. Together, these features ensure that the system is not only effective but also transparent, understandable, and trustworthy, aligning with the principles of Responsible AI.

III. Model Monitoring and Continual Learning

1) Model Performance Monitoring

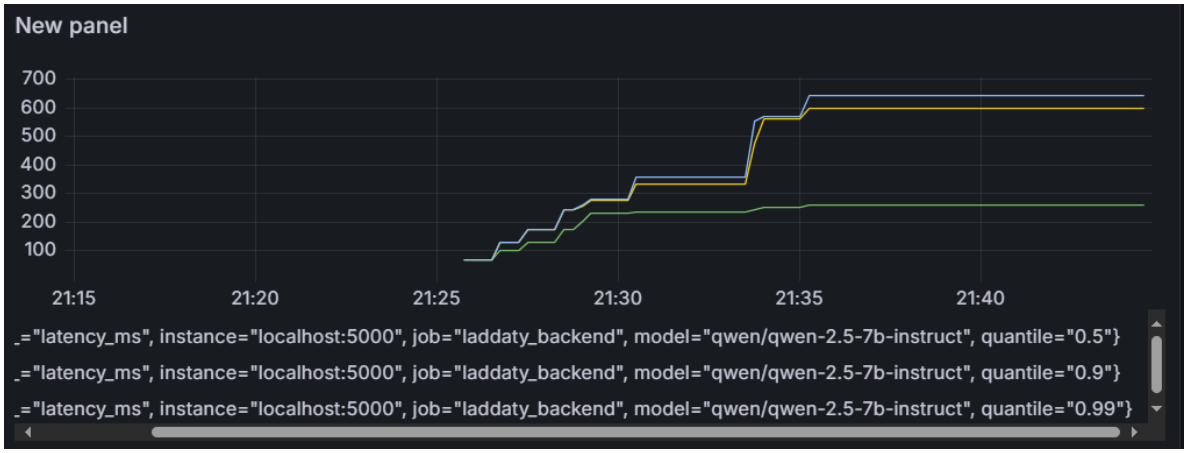
To ensure reliable performance of our LLM-based ingredient substitution system, **Prometheus** and **Grafana** were integrated for real-time monitoring.

Key Metrics:

- *latency_ms (Summary):* Tracks latency percentiles (P50, P90, P99) per model.
- *single_request_latency_ms (Gauge):* Records each request's duration in milliseconds.
- *error_count (Counter):* Counts model errors (e.g. timeouts, parsing failures).

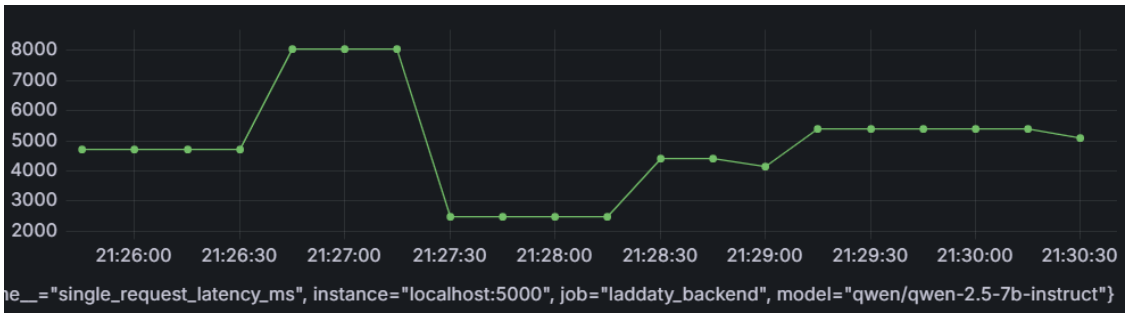
Dashboards:

1) Model Latency Percentiles Over Time (P50, P90, P99).



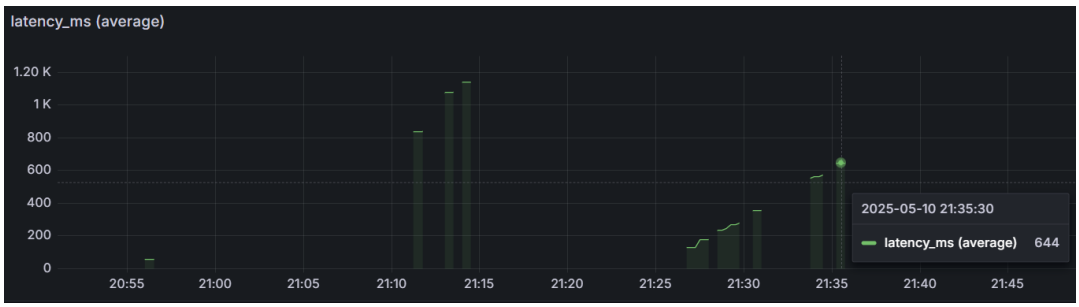
- *The 50th, 90th, and 99th percentile latencies show increasing trends over time, with **P99 peaking around 600 ms**, indicating that a small number of requests experience significantly higher latency. The gap between P50 and P99 highlights tail latency issues that may affect worst-case user experience.*

2) Live Latency per Substitution Request (ms).



- *Each point represents an individual LLM substitution request. Latency fluctuates widely, ranging from ~2500ms to **over 8000ms**, indicating variable response times.*

3) Average LLM Latency (ms) Over Time



- *The average latency shows a rising pattern with recent spikes reaching **>600ms**. This metric is valuable for tracking general system responsiveness and the impact of prompt complexity.*

2) Model Performance Monitoring

To monitor changes in the distribution of key features over time, we performed a drift analysis on the ingredient substitution system using the **Evidently** framework. Three columns were monitored: ingredient, substitute, and picked.

- ✓ *Ingredient drift:* A significant distributional shift was observed in the ingredient column (Chi-square p-value < 0.05), suggesting that the types of ingredients being substituted have changed over time. This may reflect evolving recipe trends or input diversity in recent data.

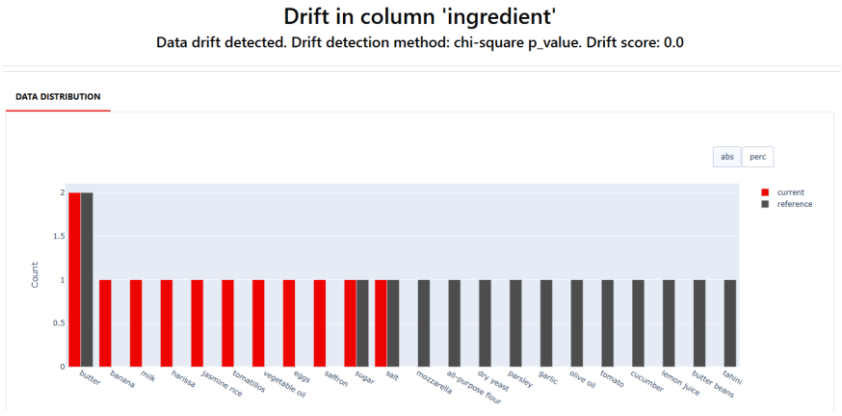


Figure 5: Drift of column 'ingredient'.

- ✓ *Substitute drift*: Similar drift was detected in the substitute column, indicating that the model or users are selecting a different set of alternatives compared to the earlier dataset. This could be due to updated model outputs, seasonal preferences, or shifts in dietary behavior.

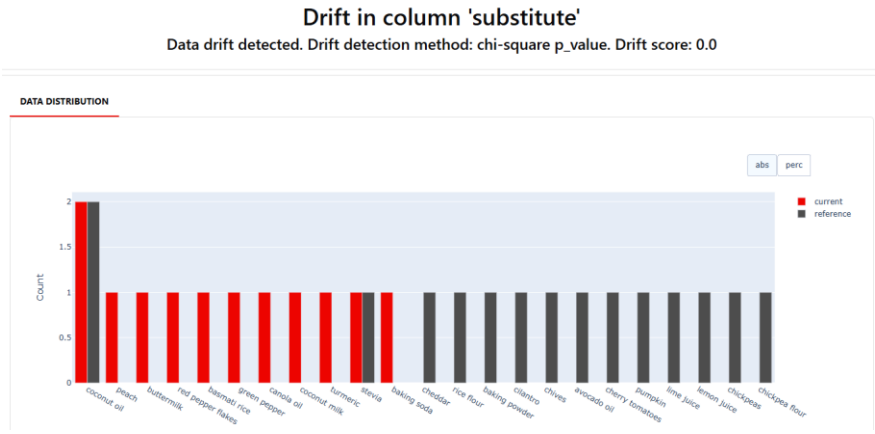


Figure 6: Drift of column 'substitute'.

- ✓ *Picked drift*: No drift was found in the picked column, meaning the final choices made by users or selected substitutes remained stable. This consistency suggests robustness in the decision mechanism or user preferences.

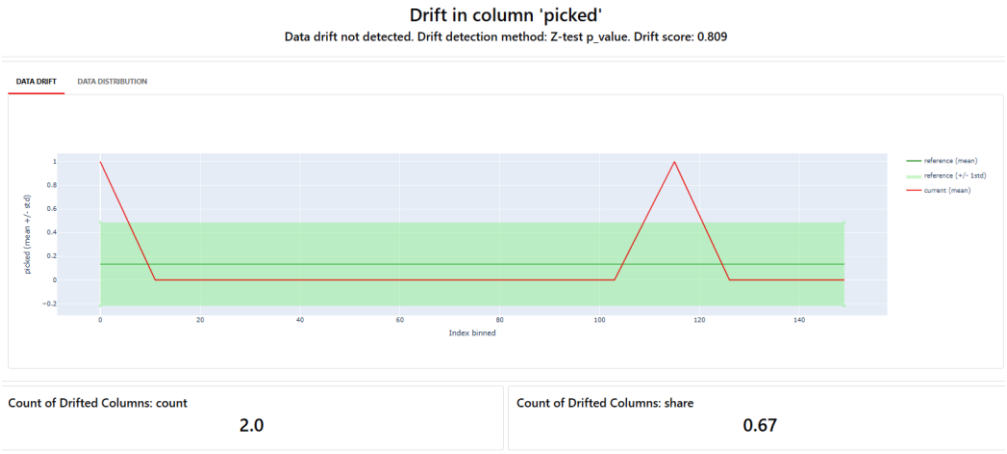


Figure 7: Drift of column 'picked'.

Overall, 2 out of 3 monitored features showed data drift, confirming the need for ongoing monitoring. These insights will guide potential fine-tuning to maintain relevance and performance.

3) Continual Learning: CT/CD pipeline

I implemented a lightweight continual learning pipeline using a CT/CD strategy based on few-shot learning and user feedback. Instead of retraining the model, I dynamically construct few-shot

prompts by sampling recent logs from the *SubstitutionLog* MongoDB collection. These logs capture user-picked substitutions alongside their original ingredient and recipe context.

During inference, the API injects up to 3 picked examples into the LLM prompt, allowing the model to adapt in real time to evolving user preferences and ingredient roles without retraining. This prompt-based adaptation acts as an implicit continual learning mechanism, leveraging live data without modifying model weights (which I do not have access to).

```
async function generateSubstitute(ingredient, recipe) {
    const variant = Math.random() < 0.5 ? 'A' : 'B';
    const model = variant === 'A' ? "qwen/qwen-2.5-7b-instruct" : "openai/chatgpt-4o-latest";
    const logs = await SubstitutionLog.find({ picked: 1 }).lean();

    const fewShot = buildFewShotPromptExamples(logs);

    const prompt =
        `${fewShot}
        Suggest the best 10 substitutes for "${ingredient}" in this recipe "${recipe}", depending on the
        For each substitute, provide:
        1. Ensure that the ratio provided can logically be applied **in both directions**.
        Provide the response in strict JSON format with:
        {
            "substitutes": [
                {"name": "Substitute Name", "ratio": "Substitution ratio as a decimal number"}
            ]
        }`;
    const maxRetries = 3;
    let retryCount = 0;
```

Figure 8: Code snippet of the updated generateSubstitute function.

An internal A/B testing setup randomly routes requests between multiple models (e.g., Qwen vs. GPT-4o), allowing us to evaluate real-world performance across variants. Prometheus monitors per-model latency (latency_ms), error rates (error_count), and single-request durations (single_request_latency_ms), enabling real-time observability.

This setup ensures continuous adaptation, low-latency deployment, and feedback-driven improvement; all without breaking production.

4) Pipeline orchestration

The entire substitution system is orchestrated using a modular, event-driven architecture without relying on heavyweight orchestration tools. Key components include the Express.js API server (serving as the controller), MongoDB (for state and log persistence), and Prometheus + Grafana (for monitoring and alerting). Each incoming substitution request triggers a pipeline consisting of input normalization, few-shot prompt assembly (based on SubstitutionLog), LLM inference (via OpenRouter), fallback JSON parsing, and result caching. All steps are encapsulated in asynchronous functions with built-in retries and latency/error tracking via Prometheus metrics. This lightweight

orchestration ensures maintainability and high availability while supporting real-time continual learning. Integration with CI/CD (via GitHub Actions) ensures consistent deployment and monitoring updates.

Conclusion

This project has been an incredibly valuable learning journey, both technically and personally. It pushed me to explore advanced concepts in MLOps, model evaluation, and real-world monitoring; skills that have directly elevated the capabilities of *Laddaty*. More importantly, this milestone served as a turning point, transforming isolated components into a robust, production-ready pipeline that can grow with the product and its users.

I would like to extend my deepest gratitude to **Dr. Mourhir** for her continuous, constructive, and insightful feedback throughout my academic journey. Her guidance challenged me to think critically, execute precisely, and always push one step further.

This is not the end -it's just the beginning. STAY TUNED 😊