LLMOps Engineer Take-Home Assessment

Overview

Time Allocation: 3-4 hours
Focus Areas: System design, monitoring, versioning, and practical implementation

---

Scenario

Thrive is launching a new LLM-powered feature: **"Career Path Navigator"** that generates personalized career recommendations based on user profiles and job market data.

Current Requirements:

- Use both GPT-4 and Claude-3.5-Sonnet for A/B testing
- Average 500 tokens per request (input + output)
- Expected volume: 1,000 requests/day initially, scaling to 10,000/day
- 95th percentile latency target: < 3 seconds
- Monthly budget constraint: $5,000 for LLM costs

Technical Context:

- Main application: Ruby on Rails
- Experimentation/ML work: Python + Databricks
- Configuration approach: JSON files that Rails consumes
- No existing dedicated LLM infrastructure

Your Task

You're the first LLMOps Engineer at Thrive. Your mandate is to establish foundational infrastructure for this feature that will scale to support future LLM-powered products.

Part 1: System Design (30-40 minutes)

Create an **architecture diagram** showing:

- How LLM requests flow from Rails to LLM providers

- Where monitoring/observability happens
- Where configuration lives and how it's consumed
- How you'd handle provider failover

Deliverable: Architecture diagram (use any tool: draw.io, Excalidraw, hand-drawn + photo, etc.)

Part 2: Configuration & Versioning Strategy (45-60 minutes)

Design a **prompt versioning and configuration system** that allows the AI team to:

- Version prompts independently of code deployments
- A/B test between providers and prompt variations
- Roll back to previous versions quickly
- Track which version generated which outputs

Deliverables:

1. **JSON Schema Design** - Create example configuration files that demonstrate:
    - Prompt versioning
    - Provider configuration (with failover)
    - Feature flags for A/B testing
    - Any other metadata you think is important
2. **Brief Written Explanation** (300-500 words):
    - How Rails would consume these configs
    - Your versioning strategy (how versions are named, stored, deployed)
    - Rollback process

Example structure to consider (adapt as you see fit):

```json
{
  "feature": "career_path_navigator",
  "version": "???",
  "config": {
    // Your design here
  }
}
```

Part 3: Monitoring Implementation (60-90 minutes)

Build a **Python proof-of-concept** that demonstrates your monitoring approach.

Requirements:

- Create a simple wrapper around mock LLM API calls
- Log structured telemetry data for each request
- Implement basic provider failover logic
- Track key metrics (latency, tokens, costs, errors)

You don't need:

- Real API keys (use mock responses)
- A database (in-memory or file-based logging is fine)
- A complete production system

Part 4: Technical Recommendations (30-45 minutes)

Write a **technical memo** (1-2 pages) addressing:

1. **Cost Optimization Strategy**
    - How would you keep costs under $5,000/month initially?
    - What monitoring would help identify cost optimization opportunities?
    - Caching strategy considerations?
2. **A/B Testing Approach**
    - How would you structure A/B tests between providers?
    - What metrics would determine the winner?
    - How long should tests run?
3. **Failure Scenarios & Mitigations**
    - Identify 3 realistic failure modes
    - Propose mitigation strategies for each
    - How would you detect these failures quickly?
4. **Quality Evaluation**
    - How would you measure output quality for this use case?
    - What automated checks could you implement?

Deliverable: Written document (PDF or Markdown)

Submission Guidelines

Please submit:

1. Architecture diagram (PNG, PDF, or JPG)
2. JSON configuration examples (as files or in a document)
3. Python code (as `.py` files with requirements.txt)
4. Technical memo (PDF or Markdown)
5. A brief README with:
    - How to run your code
    - Any assumptions you made
    - What you'd do differently with more time

Submission Method:

- GitHub repository

Timeline: Please complete within 3 days of receiving this assessment. If you need more time, just let us know.

Questions?

If anything is unclear, please email ali@thrivemycareer.com. We want you to succeed and are happy to clarify scope or requirements.

We're excited to see your approach to this challenge. Good luck!

Internal Evaluation Criteria

Looking for:

System Thinking (30%)

- Practical, scalable architecture decisions
- Understanding of production constraints
- Balance between simplicity and completeness

Technical Implementation (30%)

- Code quality, structure, and readability
- Error handling and edge cases
- Logging and observability approach

LLM-Specific Knowledge (25%)

- Understanding of LLM cost drivers
- Provider-specific considerations
- Prompt versioning best practices

Communication (15%)

- Clarity of documentation
- Justification of design decisions
- Ability to articulate tradeoffs

Notes for Candidates

- **Don't over-engineer**: We value pragmatic solutions over perfect ones
- **Document your thinking**: Explain *why* you made certain choices
- **It's okay to make assumptions**: Just document them clearly
- **Focus on what matters**: You won't have time to do everything perfectly—prioritize what showcases your strengths
- **This is representative of real work**: The scenario mirrors actual challenges you'd face in this role