Internship Description: One-Month ML/MLOps Engineering Internship (DDQ Project)

Project Overview

DDQ is an internal project that has already been implemented. The goal of this internship is to **refactor and optimize the DDQ system** with a focus on improving code quality, performance, observability, and maintainability.

The intern will gain hands-on experience with modern ML tools, vector databases, and infrastructure automation through real-world contributions.

Duration

1 Month (Full-time)

Internship Goals

- Understand and improve the DDQ codebase
- Optimize vector handling and pipeline performance
- Enhance the development workflow with CI/CD and observability
- Deliver a functional and well-documented system improvement

---

Internship Plan and Steps

Week 1: Onboarding and Setup

1. **Set up Local Development and Test Environment**
    - Clone the DDQ repository
    - Install necessary dependencies
    - Verify project runs locally and passes basic tests
2. **Basic Tutorials and Learning**
    - Short learning modules on:
        - Large Language Models (LLMs)
        - Embeddings and vector stores
        - ML pipelines and orchestration

Week 2: Codebase Familiarization and Initial Refactoring

3. **Explore DDQ Codebase**
    - Read and understand the current code structure
    - Trace the data and model pipelines
    - Note key components, responsibilities, and data flows
4. **Document the Current System Flow**
    - Create a visual/system flow diagram
    - Write notes explaining the flow of data and API interactions
5. **Identify Bottlenecks and Poor Design Patterns**
    - Use profiling tools to identify slow or resource-heavy operations
    - Highlight hardcoded logic, redundancy, or non-modular code

---

Week 3: Optimization and Infrastructure Enhancements

6. **Propose Optimization Plan**
    - Present a short document or slide deck with:
        - Refactoring proposals
        - Expected performance gains or code improvements
        - Timeline and steps for implementation
7. **Implement Optimization**
    - Refactor inefficient or messy parts of the code
    - Improve modularity and maintainability
    - Optimize vector collection handling and API interactions
8. **Begin Work on Centralized Model APIs**
    - Design a shared interface for multiple models
    - Standardize input/output structure
9. **CI/CD Configuration**
    - Set up or improve CI/CD pipelines
    - Add automated testing, linting, and deployment steps
10. **Add Observability Tools**

- Integrate tools such as:
    - **Prometheus** for metrics
    - **Grafana** for dashboards
    - Basic logging and alerting

Week 4: Finalization and Deployment

11. **Write Monitoring Scripts**

- Health check endpoints
- Uptime monitoring
- Basic performance logging

12. **Complete Optimization Tasks**

- Finalize all code updates
- Ensure compatibility with existing components

13. **Deploy Updated Services**

- Deploy to local and/or staging environments
- Ensure system stability and monitor initial behavior

14. **Monitor and Log System Performance**

- Use observability stack to track improvements
- Validate system behavior under test load

15. **Documentation**

- Summarize changes and improvements
- Update project README and technical documentation

16. **Final Presentation**

- Prepare a brief presentation
- Share work completed, lessons learned, and next steps
- Demo performance and architecture improvements to the team

Skills and Tools You'll Use

- Python (for backend and ML)
- Git & GitHub
- Docker
- Prometheus, Grafana
- CI/CD tools (e.g., GitHub Actions, Jenkins)
- Vector DBs, LLM libraries (e.g., LangChain, Transformers, milvus)