Appendix: LLM Usage and Integration Report

1. Purpose of LLM Assistance

During Iteration 2, Large Language Model (LLM) support was utilized in the following areas to enhance development efficiency and ensure structural integrity:

- Project Management & Collaboration: Creating task definitions for GitHub Issues and Trello, and planning a balanced workload distribution among team members.
- Documentation & UML: Clarifying relationships between classes (Aggregation vs. Composition) for the Design Class Diagram (DCD) in accordance with software engineering principles.
- Design Pattern Implementation: Obtaining structural suggestions on how to best integrate design patterns such as Strategy, Singleton, and Factory into the system architecture.
- Refactoring & Clean Code: Auditing written code blocks for compliance with SOLID principles and receiving optimization suggestions for sustainability.

2. LLM Output That Was Used (Reviewed & Modified)

The following outputs were generated by the LLM but were modified and integrated into the system according to specific project requirements:

- Class Templates: Base structures and skeleton code for Strategy interfaces, including QueryWriter, Reranker, and IntentDetector.
- Data Structures: Recommendations for efficient data structures based on FIFO (First-In-First-Out) logic for the QueryCache mechanism.
- Unit Test Scenarios: Essential test scenarios and basic unit test templates for the verification of critical components.
- Log & Metric Definitions: Structural templates for logging and metric definitions (e.g., latency, hit/miss ratio) to monitor system performance.

3. Limitations and Manual Adjustments

LLM outputs were not used directly; they underwent the following manual adjustments to ensure compatibility with the project's unique structure:

- Traceability: Every method and logical structure proposed by the LLM was manually verified and corrected to ensure alignment with the Use Case steps, SSD (System Sequence Diagram), and DSD (Design Sequence Diagram).
- Contextual Integrity: To prevent errors arising from the LLM’s inability to perceive the entire project scope simultaneously, class dependencies and data flow were restructured according to the project's global architecture.