GEORG-AUGUST-UNIVERSITÄT GÖTTINGEN IN PUBLICA COMMODA SEIT 1737

HPS

Advanced Research Training Project Report

submitted in partial fulfillment of the
requirements for the course "M.Inf.1259: Advanced Research Training, Data Science"

GTA: Agentic RAG AI Tutor

Revolutionizing High Performance Computing Education: Harnessing Large Language Models for Interactive Training Content Generation and Coding Support

---

Qumeng Sun

MatrNr: 24821883

Supervisor:	Prof. Dr. Julian Kunkel
Project Advisors:	Dr. Kevin Lüdemann, Matthias Eulert
Reviewers:	Jaison Lewis, Sascha Safenreider

Georg-August-Universität Göttingen
Institute of Computer Science

May 1, 2026

Georg-August-Universität Göttingen
Institute of Computer Science

Goldschmidtstraße 7
37077 Göttingen
Germany

☎ +49 (551) 39-172000
FAX +49 (551) 39-14403
✉ office@informatik.uni-goettingen.de
🌍 www.informatik.uni-goettingen.de

Abstract[1]

Personalized tutoring improves learning outcomes but remains difficult to scale due to cost and brittleness of current AI systems. This work presents GTA (Graph-ReAct Tutoring Assistant), an agentic tutoring architecture orchestrating multi-stage reasoning over retrieved knowledge through three design principles: dynamic DAG generation for adaptive workflows, quality-gated iteration with quantitative thresholds, and Blackboard pattern for unified state management. The system addresses adaptive workflow execution (RQ1), evidence synthesis preventing context degradation (RQ2), and proactive assistance via Code Spotter monitoring (RQ3).

Experimental evaluation on HotpotQA multi-hop reasoning demonstrates substantial improvements over traditional RAG (FactF1: 0.220 $\rightarrow$ 0.790). Systematic ablation across seven configurations reveals three architectural contributions—semantic reranking, multi-stage reasoning, and stateful iteration—with quantified individual effects and perfect additivity. Three design observations emerge from HotpotQA evaluation: architecture amplifies tool value through complementary design, architectural modularity enables reasoning beyond tool upgrades, and iteration requires explicit state management to prevent degradation.

The system achieves cost-effective deployment ($0.0104 per correct answer, 34% better than baseline despite higher per-query cost), suggesting potential economic viability for population-scale educational applications pending validation in authentic tutoring scenarios. The architecture provides accountability through automatic citations and audit trails, addressing transparency concerns in AI-assisted education.

[1]Source code and samples are publicly available; see Appendix A.2 for repository links.

Declaration on the use of ChatGPT and comparable tools in the context of examinations

In this work I have used ChatGPT or another AI as follows:

- ☐ Not at all
- ☑ During brainstorming
- ☐ When creating the outline
- ☐ To write individual passages, altogether to the extent of 0% of the entire text
- ☑ For the development of software source texts
- ☑ For optimizing or restructuring software source texts
- ☑ For proofreading or optimizing
- ☑ Further, namely: AI was used for brainstorming and literature review, writing polish and refinement, and assisting in drafting Chapter 4 (Experiments) based on experimental logs and notes. All AI-generated content was carefully reviewed, verified, and revised by the author to ensure accuracy and academic integrity.

I hereby declare that I have stated all uses completely.
Missing or incorrect information will be considered as an attempt to cheat.

Acknowledgements

I would like to express my sincere gratitude to all those who supported this research project.

First and foremost, I thank Prof. Dr. Julian Kunkel for providing the opportunity to conduct this research within the GWDG-HPC Group and for his guidance as the research group head.

I am deeply grateful to my primary advisors, Dr. Kevin Lüdemann and Matthias Eulert, for their invaluable support throughout this project. Their weekly meetings, insightful answers to my questions, provision of essential resources, and thoughtful guidance were instrumental in shaping this work.

I extend my appreciation to Jaison Lewis and Sascha Safenreider for their careful review of this report and constructive feedback, which significantly improved the quality and clarity of the final manuscript.

I also acknowledge the GWDG-HPC Group for providing the computational resources and collaborative environment that made this research possible.

Finally, I thank the broader research community whose open-source tools, datasets, and publications formed the foundation upon which this work was built.

Contents

List of Tables

List of Figures

List of Listings

List of Abbreviations

AI Artificial Intelligence

AMM Adaptive Memory Manager

API Application Programming Interface

BB Blackboard

CRAG Corrective Retrieval-Augmented Generation

DAG Directed Acyclic Graph

DEP Document Enrichment Pipeline

EM Exact Match

FactF1 Factual F1 (LLM-as-judge semantic correctness)

GTA Graph-ReAct Tutoring Assistant

HPC High-Performance Computing

LLM Large Language Model

LTM Long-Term Memory

NLP Natural Language Processing

QA Question Answering

RAG Retrieval-Augmented Generation

ReAct Reason-Act-Observe (reasoning paradigm)

RQ Research Question

STM Short-Term Memory

VLM Vision-Language Model

1 Introduction

Personalized tutoring improves learning outcomes but remains difficult to deliver at population scale. UNESCO projects a global shortage of 44 million teachers by 2030[UNE24], while one-on-one human tutoring requires $1,200–$2,500 annually per student in school-based programs[Pai24]. These workforce and economic constraints create a persistent gap between uniform instruction and heterogeneous learner needs in prior knowledge, pace, and interests. Large language models offer a potential technical solution through scalable dialogue and adaptive interaction. However, three limitations hinder reliable deployment: parameterized knowledge becomes stale, hallucination produces false statements, and context degradation undermines long interactions. This work addresses these technical barriers through a systems approach: combining Retrieval-Augmented Generation (RAG) for updatable, auditable knowledge[Lew+20b] with structured orchestration over unified state management[Nii86].

1.1 Research Motivation

Education underpins social and economic development[Wor25], yet dominant delivery models remain uniform in content, pacing, and pedagogy. This conflicts with learner heterogeneity in prior knowledge, interests, and cognitive styles[ZBY20]. Meta-analyses show that personalized tutoring substantially improves outcomes[NOQ24; Blo84], motivating scalable alternatives to one-size-fits-all instruction.

Large language models enable scalable dialogue for question answering and adaptive practice[Kas+23]. However, three limitations constrain educational deployment. First, parameterized knowledge becomes stale as curricula evolve. Second, hallucination produces fluent but false statements[Ji+23]. Third, context rot degrades reliability when long or noisy context accumulates[Liu+23a]. These undermine accountability in formal assessment.

Retrieval-Augmented Generation (RAG) addresses these limitations by injecting updatable external evidence at inference[Lew+20b; Gao+23]. Retrieving from controlled corpora (e.g., textbooks, lecture notes) is expected to constrain generation and improve traceability, though our evaluation uses Wikipedia as a proxy. However, conventional RAG pipelines lack adaptive reasoning and persistent memory required for tutoring[Yan+24].

1.2 Research Objectives and Main Contributions

This work presents GTA (Graph-ReAct Tutoring Assistant), an AI tutoring system addressing three research questions through architectural innovation. The questions span the tutoring interaction lifecycle: adaptive workflow planning (RQ1), evidence synthesis (RQ2), and proactive needs anticipation (RQ3).

RQ1. Adaptive Workflow Execution: How can a tutoring agent plan and execute complex workflows with branching, backtracking, and quality-driven iteration?

RQ2. Evidence Synthesis: How can an agent synthesize disparate information sources (retrieval, memory, perception) into coherent, actionable state representations?

RQ3. Proactive Assistance: How can a system integrate reactive and proactive assistance by translating visual cues of struggle into structured tasks?

The architecture employs three design principles: dynamic DAG orchestration for task-specific reasoning paths, quality-gated ReAct loops with quantitative continuation thresholds, and Blackboard pattern for structured evidence integration across iterations.

We validate reasoning and evidence-management capabilities (RQ1-RQ2) through systematic ablation on HotpotQA multi-hop question answering, demonstrating substantial improvements over traditional RAG. Three design principles emerge from HotpotQA evaluation: architecture amplifies tool value through complementary design, architectural modularity enables reasoning beyond tool upgrades, and iteration requires explicit state management. Whether these principles generalize to other reasoning domains requires further investigation. Comprehensive evaluation of proactive assistance (RQ3) requires perceptual signals and longitudinal studies beyond QA benchmarks, deferred to future work.

The remainder of this report proceeds as follows. Section 2 positions GTA within educational AI and multi-stage reasoning systems, reviewing the evolution from retrieval-augmented generation to agentic orchestration. Section 3 presents the three-layer architecture: workflow orchestration via dynamic DAG, evidence integration through the Blackboard pattern, and external tool adapters. Section 4 employs controlled ablation on HotpotQA to isolate the contributions of semantic reranking, multi-stage reasoning, and stateful iteration, measuring FactF1, citation accuracy, and cost efficiency. Section 5 synthesizes findings into three design observations—tool integration, architectural modularity, and state management—analyzing cost-effectiveness and limitations. Section 6 summarizes contributions and outlines future directions for perceptual signal integration and longitudinal evaluation.

2 Related Methods

2.1 Educational Dialogue Systems

Intelligent Tutoring Systems advanced personalization through learner modeling but faced narrow-domain coverage and high authoring costs[Ma+14]. Large language models enable scalable dialogue for formative feedback and adaptive practice[Kas+23], with recent systems demonstrating practical value in programming support[Lif+23] and personalized tutoring[Ali+23]. However, challenges in knowledge updatability and evidence traceability motivate retrieval-augmented approaches.

2.2 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) injects updatable external evidence into inference to address the staleness and opacity of parameterized knowledge[Lew+20b; Gao+23]. Classical pipelines retrieve top-k passages and concatenate them for generation[Kar+20; Lew+20a], simple but brittle to retrieval errors and context ordering on multi-hop reasoning.

Modular designs expose retrieval, reranking, and generation as replaceable operators with explicit control flow[Asa+24], enabling hierarchical indexing[Sar+24] and conditional

branching when confidence is low. Agentic variants further integrate planning and reflection to decide what to retrieve next and when to stop[Yao+23; Asa+24], improving robustness but introducing cost unpredictability.

GraphRAG organizes knowledge as graphs for path-based retrieval, improving cross-document synthesis[Edg+24]. However, substantial preprocessing costs (building entity graphs requires numerous LLM calls) limit practical deployment, though recent optimizations report $\sim$700$\times$ lower costs[ETL24].

2.3 Document Processing and Context Engineering

Effective RAG requires high-quality knowledge bases and disciplined input design. Educational corpora mix text-heavy PDFs with slides, code, and formulas, requiring hybrid approaches combining optical parsing with VLM-based processing[Aue+24; Wan+24]. Our Document Enrichment Pipeline (Appendix A) employs this strategy.

Context Engineering addresses LLM input design to maximize evidence density and auditability[Liu+23a; HTH25]. Empirical studies reveal position-dependent degradation and context rot, motivating high-precision retrieval, traceable compression, and explicit provenance tracking. Our Blackboard architecture embodies these principles through structured state management.

3 Architecture

This chapter presents Graph-ReAct, a backend orchestration framework addressing conventional RAG limitations through three architectural innovations: dynamic DAG generation for task-specific workflows, quality-gated ReAct loops for adaptive iteration, and Blackboard pattern for unified state management. The architecture embodies Context Engineering principles[Mei+25; HTH25]: systematic aggregation and structuring of information before LLM presentation, addressing context degradation through intelligent selection.

The system comprises two layers: the Orchestration Engine Layer (defines execution logic for task DAGs) and the Functional and Tooling Layer (provides atomic capabilities for retrieval, memory, and tools). The complete system, GTA (Graph-ReAct Tutoring Assistant), combines this backend with a client interface for interaction.

3.1 Architectural Overview and Design Principles

Figure 1 illustrates the high-level architecture of the Graph-ReAct system. It is a planner-driven, cyclical workflow centered around the Blackboard. A user query is first processed by the Orchestration Engine Layer, which is responsible for parsing the query, formulating a plan, scheduling execution, and evaluating the results. During the execution phase, the engine invokes capabilities from the Functional and Tooling Layer (e.g., RAG, Code Interpreter, or memory retrieval). The outputs of all components are structurally integrated into the Blackboard. Based on the knowledge completeness of the Blackboard, the system dynamically decides whether to continue iterating or to generate the final response.

The implementation of this architecture adheres to three core design principles, which collectively form the theoretical foundation of the system and are aimed at resolving the challenges faced by traditional agent frameworks.

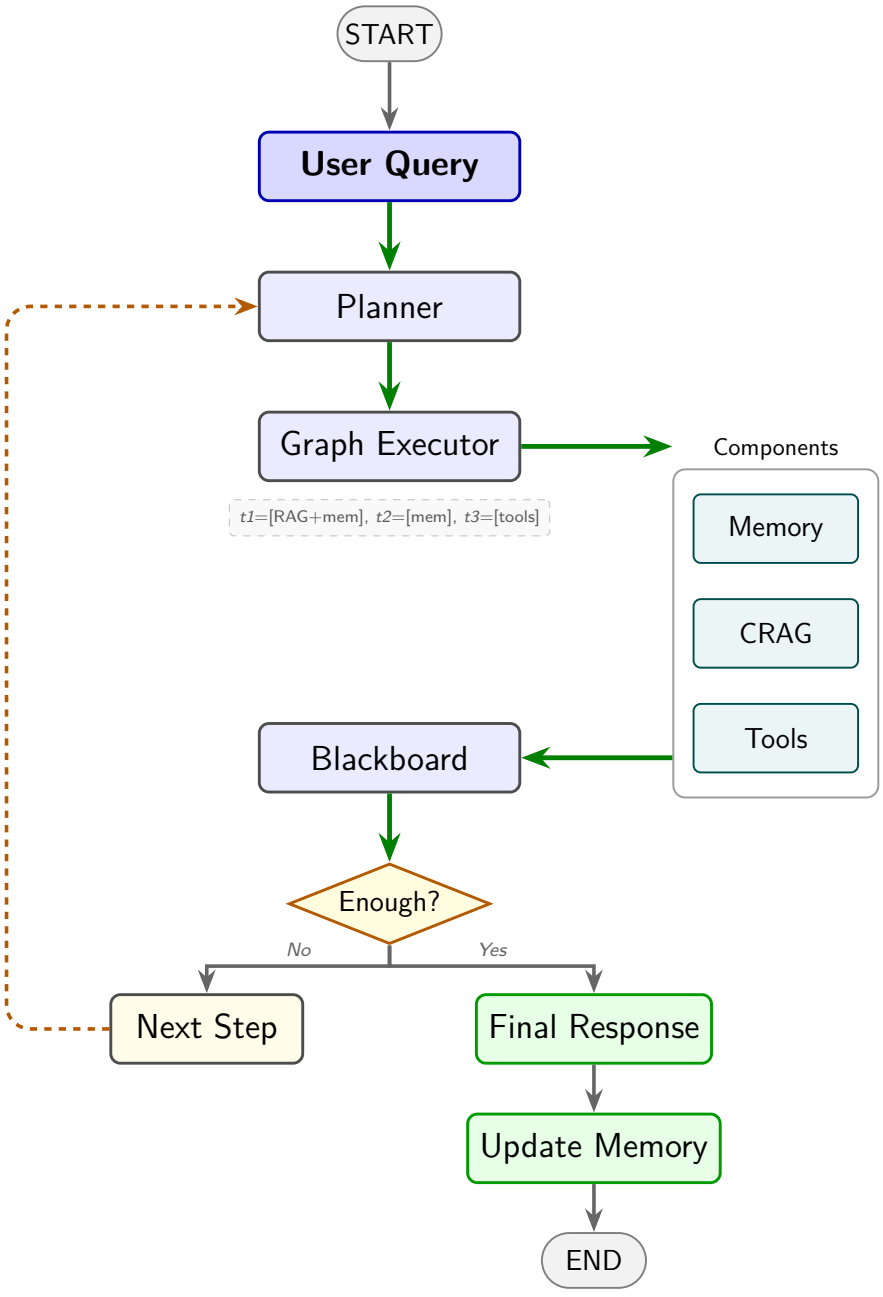

Figure 1: High-level architecture and core workflow. A user query is parsed by the Planner into executable tasks, scheduled by the Graph Executor, and executed by functional components. Results are integrated into the Blackboard. The Continuation Decision evaluates knowledge completeness: if insufficient, a new planning cycle is initiated; if sufficient, the final response is generated.

3.1.1 Principle One: Graph-based Dynamic Task Orchestration

Traditional agent frameworks employ linear chains or fixed tool-selection logic, exhibiting rigidity when handling complex multi-step tasks[Zhu+25]. Graph-ReAct models task execution as a Directed Acyclic Graph (DAG) dynamically generated per query, inspired by distributed systems[Zah+16]. Nodes represent functional components (RAG, tools), edges define data dependencies, and the Planner constructs query-specific graphs from current state and intent.

Dynamic orchestration provides three capabilities: adaptivity (query-specific plans), parallelism (concurrent execution of independent nodes), and fault tolerance (component decoupling through Blackboard interaction). The scheduler validates graph structure, prunes invalid nodes, and executes topologically ordered batches concurrently. Failures are isolated through exception boundaries, with differentiated recovery for critical versus non-critical paths.

3.1.2 Principle Two: Quality-Driven Adaptive ReAct Loop

Classic ReAct frameworks rely on fixed iterations or subjective LLM judgments for termination, lacking explicit quality control. Graph-ReAct introduces a Hybrid Quality Evaluator that quantitatively assesses Blackboard state after each cycle, continuing iteration when quality or completeness falls below thresholds (quality < 0.8, completeness < 0.7, max 3 iterations).

Evaluation employs two stages: heuristic pre-check (fast rule-based filtering on evidence count and coverage) followed by LLM-based semantic evaluation (assessing quality and completeness scores). This design balances exploration depth with computational efficiency, with thresholds tuned for HotpotQA (Chapter 4) and configurable at runtime.

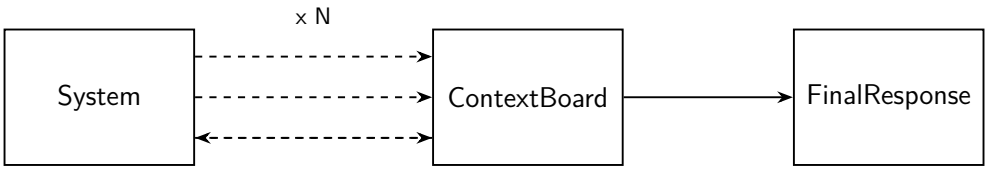

Figure 2: Interaction model of the Blackboard. Functional components (Planner, Tools, Evaluator) write to or read structured knowledge from the Blackboard, providing a globally consistent view and decoupling components.

3.1.3 Principle Three: Unified State Management via Blackboard Pattern

Direct state passing between components creates high coupling. Graph-ReAct adopts the Blackboard Pattern[Nii86] for global state management: functional components act as independent knowledge sources writing to a centralized Blackboard (Figure 2), while decision-making components read from it to formulate strategies.

The Blackboard manages structured information including raw evidence, synthesized findings, and information gaps. Components produce structured knowledge deltas applied via atomic deep-merge operations. This design achieves decoupling (component independence), global perspective (multi-source knowledge fusion), and asynchronous integration (parallel component synchronization). Type-specific merge strategies (deep merge for objects, deduplication for lists) ensure idempotency and provide audit trails.

3.2 Core Architectural Components

3.2.1 Orchestration Engine Layer

This layer comprises four components controlling task lifecycle: Execution Planner, Graph Executor, Blackboard, and Hybrid Quality Evaluator.

The **Execution Planner** translates user intent into executable graphs through three stages: Query Intent Analysis (LLM-generated structured intent with complexity scoring), Adaptive Memory Retrieval (complexity-gated depth adjustment), and Dynamic Graph Generation (LLM-produced DAG with validation). In subsequent iterations, Gap Analysis identifies information needs from Blackboard state.

The **Graph Executor** validates graph integrity, performs topological sorting to identify parallel batches, and executes independent nodes concurrently according to the Planner's DAG specification—maximizing throughput while respecting data dependencies. Exception boundaries isolate failures to individual nodes without disrupting parallel execution.

The **Blackboard** provides centralized state storage with structured layers (evidence, findings, coverage, gaps), transactional delta-based updates, and complete history for audit trails.

The **Hybrid Quality Evaluator** implements the two-stage quality control described in Principle Two (Section 3.1.2).

3.2.2 Functional and Tooling Layer

This layer executes specific tasks through three component types: RAG Composite Component (multi-stage retrieval with CRAG quality gate), Memory Management System (dual-layer STM/LTM), and Atomic Tool Suite (WebSearch, Calculator, CodeInterpreter). Configuration separates sensitive parameters (API keys via environment variables) from non-sensitive ones (model configs via YAML), with configurable thresholds.

Retrieval-Augmented Generation (RAG) Composite Component The RAG system employs a four-module architecture with quality-gated iteration (Figure 3): the Multi-database Router determines *where* to search, RAG Operations defines *how* to search, Citation Processing extracts and validates references with quality assessment, and the CRAG Quality Loop provides quality control with feedback.

The **Multi-database Router** (Figure 4) selects optimal knowledge sources by first extracting query intent, complexity, and required knowledge domains, then semantically matching against knowledge base descriptions to identify candidate sources, and finally applying weighted scoring combining similarity, intent boost, and example matching.

RAG Operations (Figure 5) executes retrieval in parallel across selected databases: queries are pre-processed and embedded for vector similarity, hybrid ranking via Vespa retrieves candidates with overfetch ($k \times 3$), metadata filtering narrows results by source type, timestamp, quality score, and chunk granularity, cross-encoder reranking (e.g., mxbai-rerank-large-v1) mitigates position bias, and finally top-K selection with deduplication feeds into context collection and prompt template building (producing both memory-enhanced and standard RAG prompts).

Citation Processing (Figure 6) processes the constructed prompts: extraction identifies cited sources embedded in the context, validation verifies reference accuracy, and

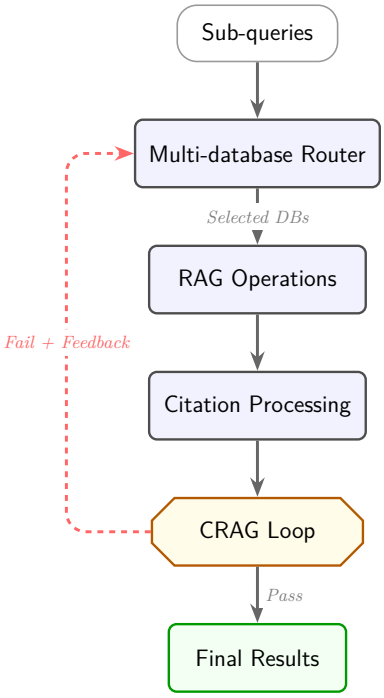

Figure 3: RAG Composite Component overview. Sub-queries flow through four modules: Multi-database Router, RAG Operations, Citation Processing, and CRAG Quality Loop (with feedback on failure).

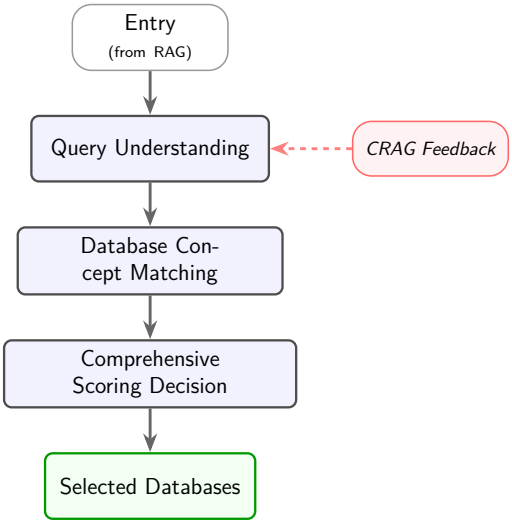

Figure 4: Multi-database Router workflow: query understanding, database concept matching, and comprehensive scoring decision.

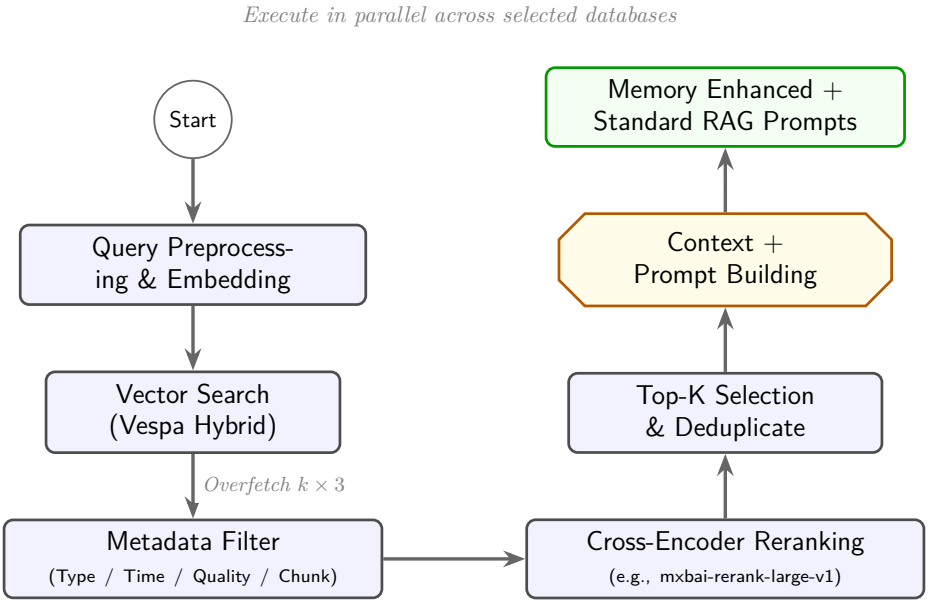

Figure 5: RAG Operations pipeline: embedding, vector search, metadata filtering, cross-encoder reranking, top-K selection, and prompt building (memory-enhanced and standard variants).

quality assessment evaluates three dimensions (relevance, faithfulness, coverage). The citation score then drives format cleaning and response generation.

The **CRAG Quality Loop** (Figure 7) implements quality-gated iteration[Yan+24]: it receives search results, performs quality assessment, and the CRAG gate either returns results if quality exceeds the threshold, applies different remediation plans for retry, or triggers feedback to the Router for re-routing.

Adaptive Memory Management System The memory system employs dual-layer architecture: Short-Term Memory (STM, session queue) and Long-Term Memory (LTM, vector-based persistent insights). The Adaptive Memory Manager integrates memory across four lifecycle stages: initial STM retrieval (fast context), adaptive deep retrieval (complexity-gated STM+LTM search), integration (informing graph generation and response), and gated consolidation (quality-thresholded LTM write-back).

Atomic Tool Suite Granular tools (WebSearch, Calculator, CodeInterpreter) expose unified interfaces for ExecutionGraph scheduling. Integration supports decorator-based registration (stateless functions) and class-based wrappers (stateful tools with lifecycle management).

3.3 Proactive Tutoring via Code Spotter

Beyond the reactive backend, GTA incorporates **Code Spotter**, a decoupled proactive assistance module that monitors learning environments to detect stuck states and trigger timely interventions. Code Spotter operates independently from the Graph-ReAct orchestration layer, enabling flexible deployment across diverse learning contexts without modifying the core reasoning infrastructure.

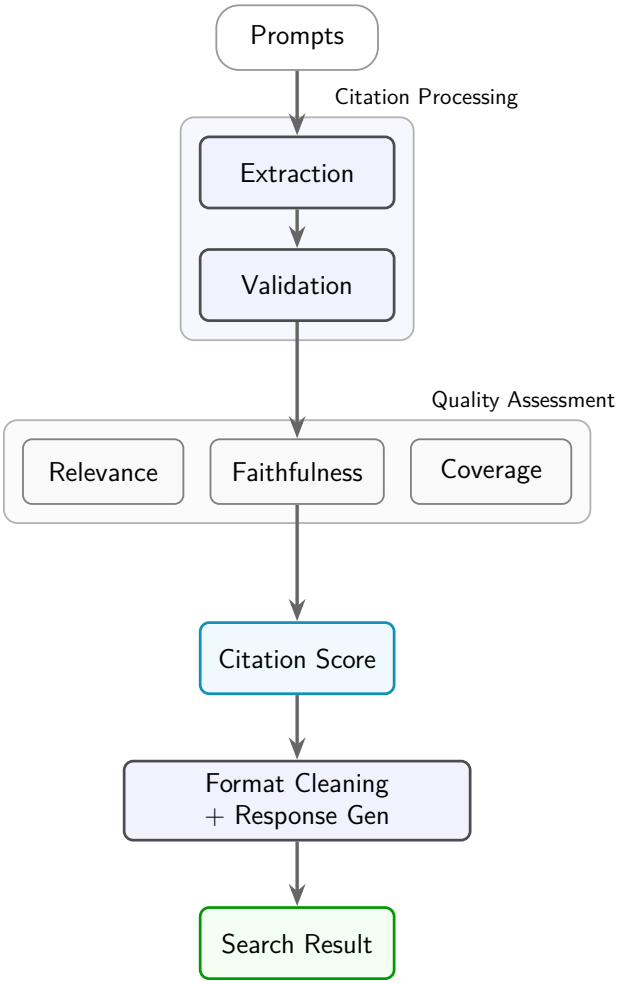

Figure 6: Citation Processing pipeline: extraction, validation, quality assessment (relevance, faithfulness, coverage), scoring, and format cleaning.

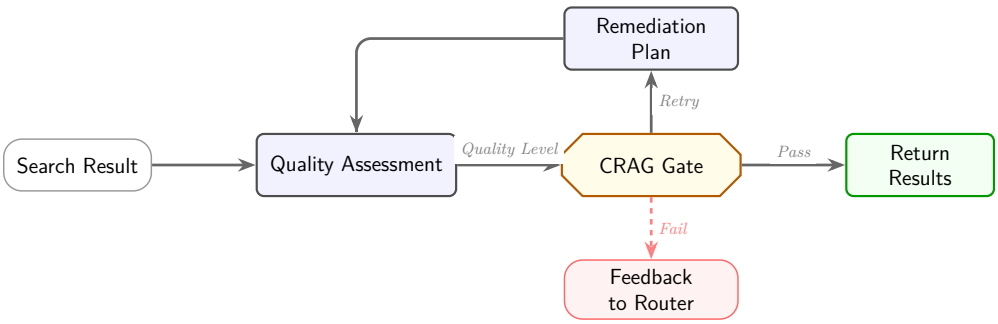

Figure 7: CRAG quality loop: quality assessment, decision gate, remediation planning, and feedback to Router.

Layered Detection Architecture Code Spotter employs a layered detection strategy, prioritizing signal sources based on the target environment's capabilities. For environments with accessible event APIs, the system leverages *native event streams* to obtain precise, low-overhead signals. In Jupyter Notebook environments, a lightweight probe monitors kernel events (execution requests, error outputs, idle transitions) and forwards standardized events to a centralized backend via message queue. Similarly, web-based learning applications can be instrumented through browser extensions that capture DOM events and network requests (XHR/Fetch), providing rich interaction telemetry. For environments lacking native event interfaces, Code Spotter falls back to *periodic screen capture with VLM analysis*, extracting visual cues (interface stagnation, error dialogs, navigation patterns) to achieve universal coverage. This layered design balances detection fidelity with deployment flexibility.

Stuck Pattern Recognition The detection logic targets three canonical stuck patterns: *consecutive similar errors* indicating persistent misconceptions, *repeated execution without meaningful modification* suggesting unproductive trial-and-error, and *prolonged inactivity following errors* reflecting disengagement or confusion. Rather than relying on fixed thresholds, the system is designed to support adaptive judgment where an LLM assesses contextual signals to determine optimal intervention timing.

Intervention and Backend Integration Upon detecting a stuck state, Code Spotter encodes the learner's situation into a structured query (e.g., "student encountering repeated import errors in data loading") and invokes the Graph-ReAct backend. The backend processes these queries identically to user-initiated requests, requiring no specialized handling. Interventions are delivered through non-intrusive interfaces with graduated urgency levels, preserving learner autonomy while providing timely support.

Empirical evaluation of proactive assistance requires interaction datasets and user studies beyond QA benchmarks, deferred to future work (Chapter 6).[2]

4 Experiments

This chapter evaluates the Graph-ReAct backend's reasoning capabilities, focusing on RQ1 (adaptive workflow execution with quality-driven iteration) and RQ2 (evidence synthesis through Blackboard state management). Multi-hop question answering provides a rigorous testbed: systems must synthesize information from multiple sources through compositional reasoning[Yan+18], where traditional single-stage RAG struggles with intermediate entities not explicitly mentioned in queries[Gao+23]. We decompose RQ1-RQ2 into two tractable experimental objectives:

EQ1: *Can multi-stage orchestration and semantic reranking overcome lexical retrieval limitations for multi-hop reasoning?*

EQ2: *What are the individual contributions of multi-stage reasoning, semantic reranking, and stateful evidence management?*

[2] A proof-of-concept implementation demonstrating the complete detection-intervention pipeline in Jupyter environments is available at `https://github.com/Mike-7777777/code-spotter`.

We conduct a systematic ablation study (7 controlled experiments, 100-question HotpotQA sample) isolating causal contributions while holding all factors constant. RQ3 (proactive assistance) requires perceptual signals beyond QA benchmarks and is deferred; RQ1-RQ2 validation provides essential foundations for the complete tutoring system.

4.1 Experimental Setup

To isolate architectural contributions from confounding factors, we establish controlled conditions across task selection (Section 4.1.1), system configurations (Section 4.1.2), and evaluation metrics (Section 4.1.3).

4.1.1 Dataset and Task

We evaluate on HotpotQA[Yan+18], a multi-hop question answering benchmark requiring systems to synthesize evidence across multiple Wikipedia documents. Unlike single-hop datasets, HotpotQA necessitates identifying intermediate entities through compositional reasoning chains.

HotpotQA defines two canonical multi-hop question types: bridge questions, which require reasoning via an intermediate entity, and comparison questions, which require comparing properties across two entities [Yan+18]. In addition to this dichotomy, the dataset covers a diverse range of surface question forms (e.g., entity-, date-, and number-focused questions, as well as yes/no questions), but our analysis focuses on the standard bridge vs. comparison split used in prior work.

To avoid sampling artefacts, we construct a stratified sample of 100 examples whose empirical distribution over bridge and comparison questions matches that of the full training set (up to rounding). This preserves the original balance between the two multi-hop reasoning types, so that our ablations are not biased toward either type. To eliminate format-induced evaluation biases[Per+23], we augment questions with explicit format guidance.

4.1.2 Systems Under Comparison

To ensure performance differences stem from architecture rather than model capacity, both systems use identical generation models (DeepSeek v3.2-exp), retrieval depth (k=20), and context size (6 documents). The key architectural differences are summarized in Table 1.

Baseline: BM25-only RAG The Single-stage RAG employs BM25 retrieval (k=20) to select the top-6 documents based on lexical scores. These documents are concatenated and fed into the DeepSeek v3.2-exp model (temperature=0.2, max_tokens=128). This configuration uses no reranking, intermediate stages, quality control, or iteration mechanisms.

Graph-ReAct System The system operates as a four-stage pipeline (Section 3): (1) a planning stage that generates a task-specific DAG; (2) an execution stage using BM25 (k=20) followed by Cohere rerank-multilingual-v3.0 to select the top-6 documents and write evidence to the Blackboard; (3) a quality evaluation stage to assess sufficiency and coverage; and (4) a citation-grounded generation stage utilizing the Blackboard. The model and context size remain identical to the baseline.

Table 1: System configuration comparison (controlled evaluation)

Component	Baseline	Graph-ReAct
Document Ranking	BM25 score (lexical)	Cohere reranker (semantic)
Context Size	6 documents	6 documents (identical)
Reasoning Stages	1 (direct generation)	4 (planning → execution → evaluation → generation)
Evidence Integration	Concatenated text	Blackboard with structured metadata
Quality Control	None	Explicit quality evaluation stage
Citation Support	None	Automatic citation extraction and insertion
Multi-hop Handling	Implicit (relies on LLM)	Explicit decomposition in planning stage
Fallback Mechanisms	None	Quality-triggered re-retrieval (disabled in evaluation)

Evaluation scope: This evaluation focuses on core reasoning capabilities (RQ1-RQ2). HotpotQA uses a single vector database (Wikipedia embeddings); the Database Router, designed for multi-source selection in educational scenarios with heterogeneous corpora, is not activated. Optional enhancements for proactive assistance (VLM processing, screen monitoring, RQ3) are disabled and deferred to future work.

Execution Modes Two modes: single-pass (E6, one cycle) and iterative (E7, quality-driven continuation up to three cycles when quality$<$0.8 or coverage$<$0.7). Components are identical; only continuation logic differs (Section 3, Section 3.3.2).

4.1.3 Evaluation Metrics

Since multi-hop QA requires evaluating both exact correctness and partial reasoning chains, we employ three complementary metrics [Raj+16; Yan+18]. Exact Match (EM) measures binary correctness after normalization (lowercasing, article removal, punctuation stripping). Token F1 computes token-level precision/recall, rewarding partial overlap while maintaining factual sensitivity. Factual F1 (FactF1) uses LLM-as-judge (Gemini 1.5 Pro) to assess semantic correctness, ignoring formatting but penalizing uncertainty when gold answers provide definite facts [Rei+24; Zhe+23; Liu+23b].

Metric Reliability and Validation FactF1 mitigates lexical-overlap inflation (EM/F1) by judging semantic correctness. The judge follows standard protocols [Zhe+23; Liu+23b] and ignores formatting/citations, so scores reflect facts rather than surface form.

4.2 Main Results

We first present overall performance to establish whether Graph-ReAct delivers measurable improvements (EQ1); subsequent sections isolate the mechanisms behind these

gains (EQ2). To evaluate the effectiveness of our approach, Table 2 compares system performance under controlled conditions.

Table 2: Performance comparison on HotpotQA (100 questions)

System	EM	F1	FactF1	vs. Baseline	Time (s)	Slowdown
Baseline RAG	0.160	0.188	0.220	—	3.7	—
Graph-ReAct (single-pass)	0.270	0.510	0.710	+223%	76.3	21×
Graph-ReAct (iterative)	0.360	0.582	**0.790**	+259%	241.0	65×

vs. Baseline = FactF1 relative improvement; Slowdown = time relative to baseline

Iterative configuration achieves 3.59× baseline FactF1 (0.220 → 0.790), with single-pass already reaching 0.710. Iteration adds 8.0 points, indicating four-stage reasoning with semantic reranking and Blackboard as primary drivers. In this multi-hop setting, document quality appears to dominate quantity: baseline's low FactF1 (0.220) despite 6 documents shows BM25 misses critical evidence, while Cohere reranking selects semantically relevant documents from the same 20-document pool. Iterative mode incurs 241s/question (66× baseline, avg 2.92 cycles), with time dominated by Cohere (37%) and LLM (30-44%).

4.3 Analysis

The aggregate results show substantial gains but do not reveal why. We now examine three mechanisms: semantic reranking's context-dependent value (Section 4.3.1), multi-stage reasoning's architectural benefits (Section 4.3.2), and practical cost implications (Section 4.3.3).

4.3.1 The Critical Role of Semantic Reranking

Semantic reranking exhibits context-dependent value: E1 vs. E0 (baseline) +0.040 FactF1, E3 vs. E2 (Graph-ReAct) +0.310. Architecture amplifies reranking value by 7.75×, with reranking contributing 54.4% of total improvement (Chapter 5, Observation 1). BM25 fails on multi-hop QA because critical evidence involves intermediate entities not in queries; semantic reranking surfaces reasoning chains despite low lexical overlap[MC18]. Failure analysis: ranking errors dominate (27/30, 90%), with critical evidence in top-20 but misranked by BM25.

4.3.2 Multi-stage Reasoning Architecture

Multi-stage reasoning contributes 31.6% of total improvement (+0.180, E2 vs E0). Notably, Graph-ReAct without semantic reranking (E2: 0.400) outperforms baseline with Cohere (E1: 0.260), suggesting architectural design matters more than tool selection in this setting. The four-stage pipeline provides explicit decomposition (planning transforms implicit reasoning into execution plans), structured evidence integration (Blackboard metadata enables deduplication and prioritization), quality-driven iteration (quantitative thresholds), and format-constrained synthesis (automatic citations for traceability). This contrasts with single-pass generation where all reasoning occurs implicitly within one LLM call.

4.3.3 Cost-Accuracy Trade-off Analysis

To contextualize the performance improvements against resource consumption, Table 3 breaks down computational and monetary costs.

Table 3: Cost analysis per 100 questions (USD)

Component	**Baseline**	**Single**	**Iterative**
Generation	0.026	0.218	0.654
Reranking	0.000	0.200	0.600
Judge	0.320	0.320	0.320
Total	**0.346**	**0.738**	**1.574**
Relative to Baseline	*1.0×*	*2.1×*	*4.5×*
Cost per Correct Answer	*0.0157*	*0.0104*	*0.0199*

Using FactF1 as correctness proxy: baseline \$0.0157/correct, single-pass \$0.0104 (34% better efficiency despite 2.1× cost), iterative \$0.0199 (27% less efficient, 3.59× higher accuracy). Single-pass offers optimal balance (3.23× accuracy at 2.1× cost); iterative mode (4.5× cost) suits high-stakes applications.

4.4 Ablation Study

While the preceding analysis identified key mechanisms, it cannot quantify individual contributions or rule out synergistic interactions. We conduct systematic ablation across three dimensions (semantic reranking, state management, iteration mode) with seven configurations (E0 baseline, E2 w/o Cohere, E3 w/o Blackboard, E6 single-pass, E9 iterative w/o Blackboard, E7 full system) to isolate component contributions.

Table 4: Ablation study results (100 questions)

Experiment	**Iter.**	**Rerank**	**BB**	**4-stage**	**FactF1**	**Time (s)**
E0: Baseline RAG	—	BM25	—	—	0.220	3.7
E1: + Cohere	—	Cohere	—	—	0.260	18.7
E2: w/o Cohere	×	BM25	✓	✓	0.400	43.4
E3: w/o Blackboard	×	Cohere	×	✓	0.710	74.3
E6: Graph-ReAct (single)	×	Cohere	✓	✓	0.710	76.3
E9: w/o BB (iter.)	✓	**Cohere**	×	✓	**0.680**	**240.4**
E7: Graph-ReAct (iter.)	✓	Cohere	✓	✓	**0.790**	241.0

✓= enabled, × = disabled, — = not applicable; BB = Blackboard; Iter. = Iteration

The factorial design isolates three main contributions:

1. **Semantic Reranking** (54.4%): Cohere contributes +0.310 within Graph-ReAct (E3 vs E2) but only +0.040 in baseline (E1 vs E0), demonstrating 7.75× amplification where architecture enables reranking value through complementary decomposition.

2. **Multi-stage Reasoning** (31.6%): Four-stage architecture contributes +0.180 (E2 vs E0), with Graph-ReAct without reranking (E2: 0.400) outperforming baseline with Cohere (E1: 0.260), suggesting that architectural modularity matters more than tool selection in this setting.

3. **Blackboard for Iteration** (19.3%): E9 (iterative w/o Blackboard: 0.680) performs worse than E6 (single-pass w/ Blackboard: 0.710), demonstrating iteration without state management is harmful. With Blackboard, iterative mode adds +0.110 (E7 vs E9), enabling deduplication, gap tracking, and coverage analysis.

4. **Approximate additivity**: The estimated main effects (0.310 + 0.180 + 0.110 - 0.030 = 0.570) closely match the observed overall improvement (E7 - E0 = 0.570, residual < 0.001), suggesting that interaction effects are small in this experimental setting. This pattern suggests, but does not conclusively prove, that interaction effects are limited for this dataset and configuration.

The E9 result supports Blackboard as important infrastructure: naive iteration accumulates redundant evidence and lacks principled stopping criteria, transforming passive storage into active context engineering (Chapter 5, Observation 3).

5 Discussion

This chapter synthesizes experimental findings, connecting empirical results to research questions (Chapter 1), architectural design (Chapter 3), and the broader RAG landscape (Chapter 2). We organize the discussion in five parts: (1) *What design considerations emerge from our findings?* (2) *How does Graph-ReAct relate to existing RAG approaches?* (3) *What steps are required for production deployment?* (4) *What threats constrain our conclusions?* and (5) *What ethical considerations apply?*

5.1 Design Considerations for Retrieval-Augmented Reasoning

Experimental findings suggest three design considerations for multi-step reasoning over external knowledge, connecting to RQ1 (adaptive workflow) and RQ2 (evidence synthesis). Their generality beyond HotpotQA requires validation on other tasks.

Observation 1: Architecture Amplifies Tool Value. Semantic reranking contributes +0.310 in Graph-ReAct but only +0.040 in baseline (7.75× amplification), demonstrating tool effectiveness emerges from architecture-tool interaction. Multi-stage decomposition creates conditions for reranking success. Design implication: prioritize architectural modularity before component upgrades.

Observation 2: Architecture Enables Reasoning Beyond Tool Upgrades. Multi-stage with BM25 (E2: 0.400) outperforms single-stage with Cohere (E1: 0.260); our results suggest that, in this setting, architectural modularity may be a primary factor and tool selection a secondary optimization. Structural affordances (decomposition, quality evaluation) enable capabilities unachievable through retrieval improvements alone.

Observation 3: Iteration Requires State Management. Iteration without Blackboard (E9: 0.680) performs worse than single-pass with Blackboard (E6: 0.710), suggesting state management as necessary infrastructure in this setting. With Blackboard, iterative mode reaches 0.790 (+0.110 over E9), enabling deduplication, gap tracking, and

principled stopping criteria. This pattern may extend to other multi-cycle reasoning tasks requiring dialogue history and adaptive termination, though empirical validation in those settings is needed.

5.2 Positioning in the RAG Landscape

Graph-ReAct distinguishes itself through five architectural choices: semantic reranking with Blackboard integration, four-stage reasoning with explicit quality gates, explicit state management, quality-driven iteration with quantitative thresholds, and flexible cost-accuracy trade-offs.

Classical RAG[Lew+20b] concatenates retrieved passages, effective for single-hop queries but struggling on multi-hop reasoning. Modular RAG[Asa+24] exposes retrieval, reranking, and generation as independent operators with conditional control flow. Graph-ReAct extends modularity through dynamic DAG orchestration (query-specific graphs), explicit quality gates (quantitative thresholds), and Blackboard pattern (structured state synthesis). Ablation on HotpotQA indicates these contribute 31.6% of improvement.

Agentic RAG[Yao+23; Asa+24] integrates planning and reflection but relies on LLM-based continuation decisions, introducing cost unpredictability. Graph-ReAct employs pre-generation context verification (evaluating evidence sufficiency before generation) rather than post-generation output verification, enabling predictable costs. E9 demonstrates iteration requires state management (0.680 without vs. 0.790 with Blackboard), providing evidence that context verification serves as important infrastructure in this setting.

GraphRAG[Edg+24] addresses retrieval structure (offline graph construction), while Graph-ReAct's Blackboard provides online state management. These approaches are complementary: GraphRAG for knowledge organization, Blackboard for evidence integration.

Context engineering[Mei+25; HTH25] emphasizes structured LLM input management. E9 supports transient state management as important infrastructure: without Blackboard, iterative execution degrades (0.680 vs. 0.710), illustrating context rot. Blackboard prevents degradation through deduplication, gap tracking, and coverage analysis.

5.3 Roadmap to Production Deployment

The current prototype establishes the technical foundation: the Graph-ReAct backend and Document Enrichment Pipeline (DEP, Appendix A.1) are operationally ready, with HotpotQA evaluation demonstrating iterative mode achieves FactF1 0.790 at $0.0199 per correct answer. Transitioning to production requires three subsequent efforts. (i) *Domain adaptation*: apply DEP to ingest heterogeneous HPC training materials (SLURM documentation, course slides, internal FAQs), with human-in-the-loop verification of structure fidelity, and construct an HPC-specific evaluation set to validate domain transfer. (ii) *Integration and pilot*: expose the orchestration layer via a stable API with institutional authentication (SSO), then run a controlled pilot with 5–10 HPC trainees to measure service-level objectives (latency, cost, answer quality) and conduct A/B testing against traditional support channels. (iii) *Iteration*: calibrate prompts and quality thresholds based on pilot feedback, address cost optimization (semantic caching, batching), and explore multi-tenant isolation and multimodal extensions, subject to the privacy constraints discussed in Section 5.5.

5.4 Threats to Validity

Internal Validity. Evaluation confounds include format enforcement and judge bias. E3 and E6 both achieve FactF1 = 0.710 despite different format constraints, indicating metrics reflect semantic correctness. Judge bias is mitigated through explicit instructions[Zhe+23; Liu+23b] and triangulation with EM/Token F1. Experimental control: identical models, retrieval depth, and context size ensure performance differences reflect architecture, not model capacity. Factorial design varies one dimension at a time for clean causal attribution.

External Validity. The 100-question sample provides statistical power for large effects but limits subgroup analysis. BM25-only baseline isolates semantic reranking; comprehensive comparison with modern hybrid retrieval remains future work. HotpotQA evaluates multi-hop reasoning over Wikipedia text; generalization to other reasoning types (mathematical, commonsense) or domains (scientific, legal) is untested. Component contributions may shift across domains. Evaluation scope (RQ1-RQ2, single-source retrieval) detailed in Chapter 4; multi-source retrieval and RQ3 deferred.

5.5 Ethical Considerations

Educational Accountability and Human Oversight. HotpotQA validates technical reasoning capabilities but does not assess pedagogical effectiveness or long-term learning outcomes. Recent studies indicate that while generative AI tools can improve student performance, they may simultaneously reduce instructor confidence in the originality of student work[Bas+24], demonstrating the need for human verification in formal assessment. Production deployment requires oversight to ensure instructional appropriateness[Kas+23]. Despite RAG-based grounding reducing hallucination[Gao+23], GTA should augment rather than replace teacher judgment.

Data Privacy and Security Risks. Current experiments employ publicly available datasets (HotpotQA, Wikipedia). Educational deployment must address significant privacy risks: widespread parental concerns exist regarding student data in AI learning devices, while many institutions lack resources for adequate data protection[MH21]. Compliance with GDPR and FERPA requires minimizing data collection, obtaining explicit consent, and implementing technical safeguards. Knowledge bases require content review to prevent retrieval of inappropriate or biased materials[Kas+23].

Algorithmic Bias and Fairness. Foundation models inherit biases in language, culture, and socioeconomic representation from training data[Ben+21]. Prior work surveyed in [BH22] documents disparate impacts in educational AI systems, including studies where automated essay scoring systems under-score non-native writers and remote proctoring tools show gender-dependent accuracy gaps. While GTA's citation mechanism improves transparency by exposing retrieval sources—enabling verification unlike black-box systems—bias in underlying corpora and generation persists. Equitable access requires addressing cost barriers (Chapter 4 reports ≈\$0.015/query in E7) and ensuring multilingual, culturally inclusive knowledge bases.

6 Conclusion

This work presents GTA (Graph-ReAct Tutoring Assistant), an agentic tutoring system achieving 3.59× improvement over traditional RAG (FactF1: 0.220 → 0.790) through three architectural innovations: dynamic DAG orchestration, quality-gated ReAct loops, and Blackboard state management.

RQ1 (Adaptive Workflow): Dynamic DAG generation with quality-driven loops (thresholds: 0.8/0.7) and parallel execution enables adaptive multi-hop reasoning. Ablation shows iteration requires state: E7 (0.790) with Blackboard outperforms E9 (0.680) without.

RQ2 (Evidence Synthesis): Blackboard provides structured state with deduplication, gap tracking, and coverage analysis, essential for multi-cycle refinement (+0.110 in iterative mode).

RQ3 (Proactive Assistance): Code Spotter module architecturally integrated but empirically deferred; RQ1-RQ2 validation provides foundation.

Contributions: (1) A prototype system addressing rigid workflows, subjective termination, and implicit state through quantitative gates and explicit state management. (2) Systematic ablation isolating three design observations on HotpotQA: architecture amplifies tool value (7.75×), modularity matters more than tool selection, iteration requires state management. (3) Three cost-performance operating points: baseline ($0.0157/correct), single-pass ($0.0104, best efficiency), iterative ($0.0199, 3.59× accuracy).

Scalability: Single-pass cost-effectiveness ($0.0104/correct on HotpotQA) suggests potential for large-scale deployment at a fraction of human tutoring cost[Pai24], pending validation in authentic educational settings.

Accountability: Citation generation with confidence scores and audit trails partially addresses AI opacity[Kas+23], enabling failure diagnosis and quality assurance.

Personalization: Adaptive memory and quality thresholds are designed to enable per-student calibration; Code Spotter (RQ3) will support timely interventions.

Limitations: Evaluation focuses on multi-hop QA (RQ1-RQ2) with 100-question sample; proactive tutoring (RQ3) and longitudinal outcomes require user studies. Generalization to other reasoning domains (mathematical, scientific) and multilingual settings is untested.

Threats to Validity: Temporal coverage of related work. This project was initiated in November 2024, and the core problem formulation and system architecture were designed in that time frame. While the bibliography has been incrementally updated to include a small number of particularly relevant papers published through late 2025, we do not aim for exhaustive coverage of research appearing after the project start date. In particular, later systems that independently propose similar ideas to those in this report may not be cited; any such omission reflects limitations of literature coverage rather than a claim of priority.

Readers should therefore interpret the novelty claims in this report relative to the state of the art as of late 2024, not as a comprehensive survey of all work published up to the final submission date.

Future Directions: Future work will prioritize a full-scale evaluation on the HotpotQA benchmark and the deployment of multi-source retrieval in authentic educational settings. We also aim to evaluate the Code Spotter module while establishing strict privacy protocols. Furthermore, we plan to investigate domain transfer capabilities, mul-

timodal integration, and potential combinations with structured retrieval systems such as GraphRAG.

References

[Ali+23] Farhan Ali et al. "Supporting self-directed learning and self-assessment using TeacherGAIA, a generative AI chatbot application: Learning approaches and prompt engineering". In: *Learning: Research and Practice* (2023). DOI: 10.1080/23735082.2023.2258886.

[Asa+24] Akari Asai et al. "Self-RAG: Learning to Retrieve, Generate, and Critique through Self-Reflection". In: *International Conference on Learning Representations (ICLR)*. Oral presentation. 2024. URL: https://arxiv.org/abs/2310.11511.

[Aue+24] Christoph Auer et al. "Docling Technical Report". In: *arXiv preprint arXiv:2408.09869* (2024). URL: https://arxiv.org/abs/2408.09869.

[Bas+24] Hamsa Bastani et al. "Generative AI Can Harm Learning". In: *SSRN Electronic Journal* (2024). Investigates the impact of generative AI on student learning outcomes and skill acquisition. DOI: 10.2139/ssrn.4895486.

[Ben+21] Emily M. Bender et al. "On the Dangers of Stochastic Parrots: Can Language Models Be Too Big?" In: *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*. FAccT '21. New York, NY, USA: Association for Computing Machinery, 2021, pp. 610–623. DOI: 10.1145/3442188.3445922.

[BH22] Ryan S. Baker and Aaron Hawn. "Algorithmic Bias in Education". In: *International Journal of Artificial Intelligence in Education* 32.4 (2022), pp. 1052–1092. DOI: 10.1007/s40593-021-00285-9.

[Blo84] Benjamin S. Bloom. "The 2 Sigma Problem: The Search for Methods of Group Instruction as Effective as One-to-One Tutoring". In: *Educational Researcher* 13.6 (1984), pp. 4–16.

[Edg+24] Darren Edge et al. "From Local to Global: A Graph RAG Approach to Query-Focused Summarization". In: *arXiv preprint arXiv:2404.16130* (2024). URL: https://arxiv.org/abs/2404.16130.

[ETL24] Darren Edge, Ha Trinh, and Jonathan Larson. *LazyGraphRAG: Setting a new standard for quality and cost*. Microsoft Research Blog. Published November 25, 2024. Claims 0.1% of full GraphRAG indexing cost and >700× lower query cost for global search. 2024. URL: https://www.microsoft.com/en-us/research/blog/lazygraphrag-setting-a-new-standard-for-quality-and-cost/.

[Gao+23] Yunfan Gao et al. "Retrieval-Augmented Generation for Large Language Models: A Survey". In: *arXiv preprint arXiv:2312.10997* (2023). URL: https://arxiv.org/abs/2312.10997.

[HTH25] Kelly Hong, Anton Troynikov, and Jeff Huber. *Context Rot: How Increasing Input Tokens Impacts LLM Performance*. Tech. rep. Technical report. Recommended citation from the authors. Chroma, 2025.

[Ji+23] Ziwei Ji et al. "Survey of Hallucination in Natural Language Generation". In: *ACM Computing Surveys* (2023). DOI: 10.1145/3571730.

[Kar+20] Vladimir Karpukhin et al. “Dense Passage Retrieval for Open-Domain Question Answering”. In: *EMNLP*. 2020.

[Kas+23] Enkelejda Kasneci et al. “ChatGPT for good? On opportunities and challenges of large language models for education”. In: *Learning and Individual Differences* 103 (2023), p. 102274. DOI: 10.1016/j.lindif.2023.102274.

[Lew+20a] Mike Lewis et al. “BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension”. In: *ACL*. 2020.

[Lew+20b] Patrick Lewis et al. “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks”. In: *Advances in Neural Information Processing Systems 33*. 2020, pp. 9459–9474.

[Lif+23] Mark Liffiton et al. “CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes”. In: *Proceedings of the 23rd Koli Calling International Conference on Computing Education Research (Koli Calling ’23)*. 2023, pp. 1–11. DOI: 10.1145/3631802.3631830.

[Liu+23a] Nelson F. Liu et al. “Lost in the Middle: How Language Models Use Long Context”. In: *arXiv preprint arXiv:2307.03172* (2023). URL: https://arxiv.org/abs/2307.03172.

[Liu+23b] Yang Liu et al. “G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment”. In: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. 2023, pp. 2511–2522.

[Ma+14] Wenting Ma et al. “Intelligent tutoring systems and learning outcomes: A meta-analysis”. In: *Journal of Educational Psychology* 106.4 (2014), pp. 901–918. DOI: 10.1037/a0037123.

[MC18] Bhaskar Mitra and Nick Craswell. “An Introduction to Neural Information Retrieval”. In: *Foundations and Trends in Information Retrieval* 13.1 (2018), pp. 1–126.

[Mei+25] Lingrui Mei et al. “A Survey of Context Engineering for Large Language Models”. In: *arXiv preprint arXiv:2507.13334* (2025). DOI: 10.48550/arXiv.2507.13334. URL: https://arxiv.org/abs/2507.13334.

[MH21] Fengchun Miao and Wayne Holmes. *AI and education: Guidance for policy-makers*. Policy Paper. Highlights widespread parental concerns about data privacy and significant institutional gaps in data protection capacity. UNESCO, 2021. URL: https://unesdoc.unesco.org/ark:/48223/pf0000376709.

[Nii86] H. Penny Nii. “Blackboard Systems: Blackboard Application Systems and a Knowledge Engineering Perspective”. In: *AI Magazine* 7.3 (1986), pp. 82–107. URL: https://ojs.aaai.org/aimagazine/index.php/aimagazine/article/view/550.

[NOQ24] Andre Nickow, Philip Oreopoulos, and Vincent Quan. “The Promise of Tutoring for PreK–12 Learning: A Systematic Review and Meta-Analysis of the Experimental Evidence”. In: *American Educational Research Journal* 61.1 (2024), pp. 74–107. DOI: 10.3102/00028312231208687.

[Pai24] Paige Shoemaker DeMio. *Scaling Up High-Dosage Tutoring is Crucial to Students' Academic Success*. https://www.americanprogress.org/article/scaling-up-high-dosage-tutoring-is-crucial-to-students-academic-success/. Mar. 2024.

[Per+23] Ethan Perez et al. "Discovering Language Model Behaviors with Model-Written Evaluations". In: *Findings of the Association for Computational Linguistics: ACL 2023*. 2023, pp. 13387–13434.

[Raj+16] Pranav Rajpurkar et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*. 2016, pp. 2383–2392.

[Rei+24] Machel Reid et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024. URL: https://arxiv.org/abs/2403.05530.

[Sar+24] Parth Sarthi et al. "RAPTOR: Recursive Abstractive Processing for Tree-Organized Retrieval". In: *International Conference on Learning Representations (ICLR)*. 2024. URL: https://arxiv.org/abs/2401.18059.

[UNE24] UNESCO. *Global Report on Teachers: Addressing the Teacher Shortage*. https://www.unesco.org/en/articles/global-report-teachers-what-you-need-know. Reports 44 million teacher shortage by 2030 for universal education, with 15 million needed in Sub-Saharan Africa. Accessed: 2024-11-18. 2024.

[Wan+24] Peng Wang et al. "Qwen2-VL: Enhancing Vision-Language Model's Perception of the World at Any Resolution". In: *arXiv preprint arXiv:2409.12191* (2024). URL: https://arxiv.org/abs/2409.12191.

[Wor25] World Bank. *Education Overview: Development News, Research, Data*. https://www.worldbank.org/en/topic/education/overview. Accessed on 2026-01-01. 2025.

[Yan+18] Zhilin Yang et al. "HotpotQA: A Dataset for Diverse, Explainable Multi-hop Question Answering". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics. 2018, pp. 2369–2380.

[Yan+24] Shi-Qi Yan et al. "Corrective Retrieval Augmented Generation". In: *arXiv preprint arXiv:2401.15884* (2024). URL: https://arxiv.org/abs/2401.15884.

[Yao+23] Shunyu Yao et al. "ReAct: Synergizing Reasoning and Acting in Language Models". In: *ICLR 2023*. 2023. URL: https://arxiv.org/pdf/2210.03629.

[Zah+16] Matei Zaharia et al. "Apache Spark: A Unified Engine for Big Data Processing". In: *Communications of the ACM*. Vol. 59. 11. 2016, pp. 56–65. DOI: 10.1145/2934664.

[ZBY20] L. Zhang, J. D. Basham, and S. Yang. "Understanding the Implementation of Personalized Learning: A Research Synthesis". In: *Educational Research Review* 31 (2020), p. 100339. DOI: 10.1016/j.edurev.2020.100339.

[Zhe+23] Lianmin Zheng et al. "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena". In: *Advances in Neural Information Processing Systems*. Vol. 36. 2023, pp. 46595–46623.

[Zhu+25] Dongsheng Zhu et al. “Divide-Then-Aggregate: An Efficient Tool Learning Method via Parallel Tool Invocation”. In: *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vienna, Austria: Association for Computational Linguistics, 2025, pp. 28859–28875. DOI: 10.18653/v1/2025.acl-long.1401. URL: https://aclanthology.org/2025.acl-long.1401/.

A Additional Materials

A.1 Document Enrichment Pipeline (DEP)

Effective personalized tutoring requires a robust knowledge foundation. This system adopts the **Document Enrichment Pipeline (DEP)**, an offline pipeline using MinerU to convert academic PDFs into structured, LLM-friendly formats. DEP builds the knowledge base from diverse course materials (notes, slides, textbooks, papers), following a "structure-first, semantics-enhanced" philosophy that preserves and enriches source integrity.

DEP employs a multi-stage pipeline (Figure 8) handling hierarchical layouts, formulas, tables, and diagrams.

Layout-Aware Structure Analysis MinerU performs visual layout analysis using computer-vision algorithms to detect headings, lists, tables, formulas, images, and code blocks, preserving logical structure and provenance.

Layout-Preserving Content Extraction DEP extracts content into Markdown preserving layout semantics: lists become Markdown lists, formulas export as LaTeX, tables convert to CSV/Markdown with cell-level provenance.

Multimodal Content Enhancement DEP integrates a VLM (e.g., `Qwen2.5-VL-72B`) analyzing images to produce `type` (Code, Flowchart, Table, etc.) and `convertibility` score (0-100) estimating lossless textual conversion feasibility (Figure 9). Adaptive strategy based on score:

- **High convertibility:** full transcription (e.g., code snippets, text-heavy tables) with fidelity-preserving formatting.
- **Medium convertibility:** transcription plus a concise semantic summary (e.g., labelled diagrams, schematic charts).
- **Low convertibility:** descriptive captioning that emphasizes salient concepts and relations (e.g., complex illustrations).

All VLM outputs include confidence metadata and provenance links to the original image region.

Standardization and Output DEP integrates extracts, transcriptions, and VLM annotations into standardized Markdown with explicit tags (e.g., `<img_description>`). Output serves two roles: (1) **Vector Store** for semantic retrieval, (2) **Inferred Knowledge Graph** capturing hierarchical structure and concept relations. These artifacts provide the foundation for retrieval-augmented tutoring.

A.2 Code Availability

The source code for this project is publicly available:

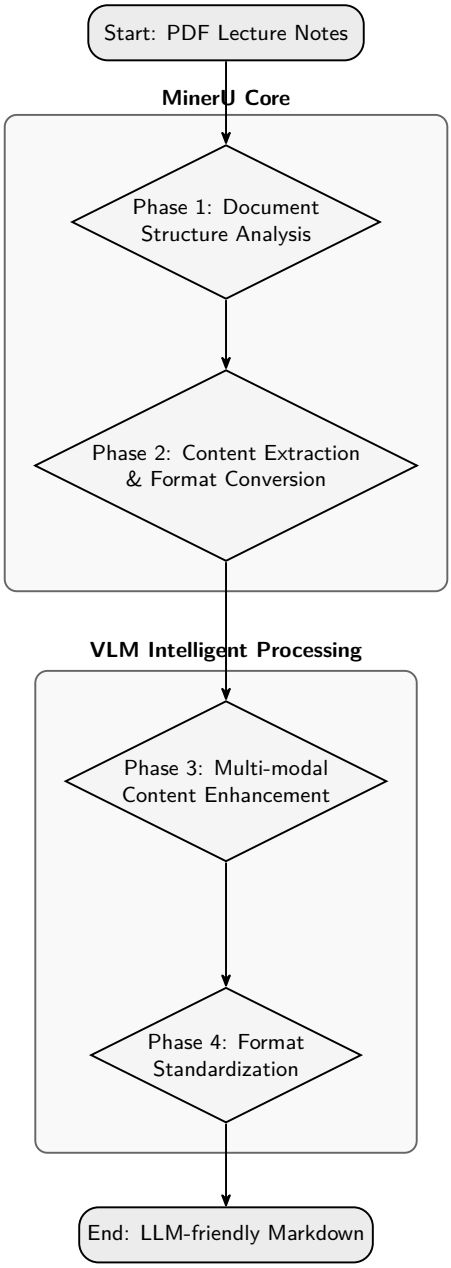

Figure 8: End-to-end DEP pipeline. It transforms unstructured PDF documents into LLM-friendly Markdown through four steps: structure analysis, content extraction, multimodal enhancement, and standardization.

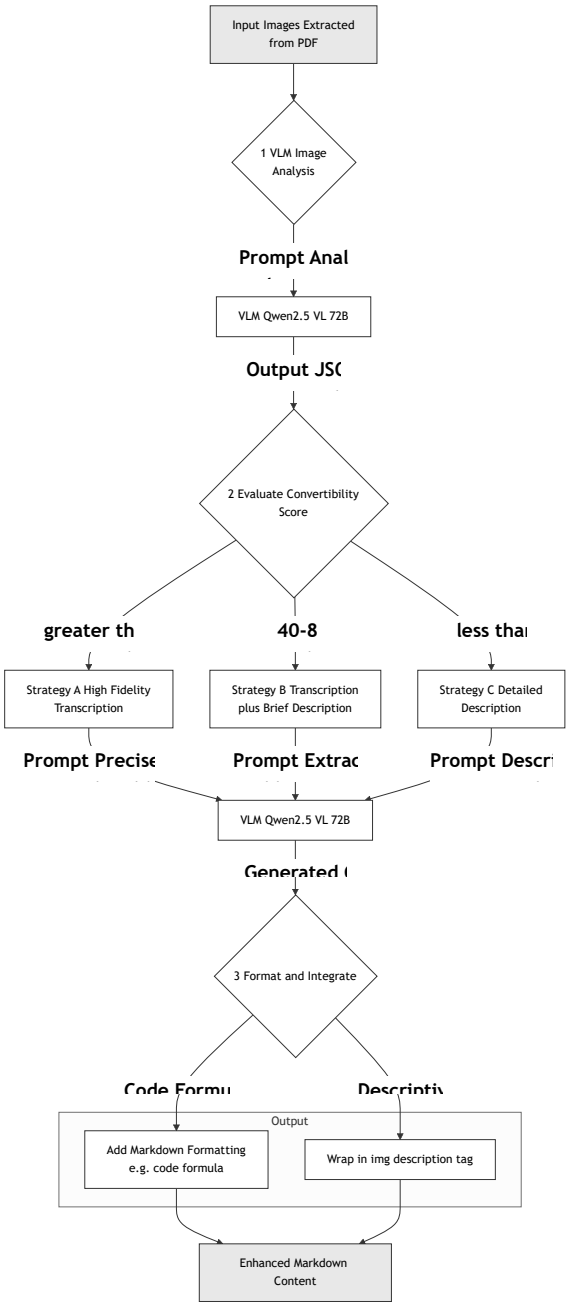

Figure 9: Multimodal enhancement workflow. The VLM analyzes image type and convertibility, then chooses a processing strategy—from high-fidelity transcription to descriptive captioning—and integrates the result into the Markdown output.

- **Graph-ReAct Backend**: The complete orchestration layer, Blackboard implementation, and HotpotQA evaluation scripts.
  `https://gitlab.gwdg.de/sun.qumeng/msc-sun-qumeng-gta`
- **Code Spotter Demo**: Proof-of-concept implementation of proactive assistance with stuck detection and intervention in Jupyter environments.
  `https://github.com/Mike-7777777/code-spotter`