

Grade Appeal - HW5: Team *Promptopia*

Team: Guy Raz (302632740), Gil Chen (316019975) | **Repository:** [GitHub Link](#)

Option #3 (RAG)

Our Self Score: 98 / 100

Level 4: 90–100 Exceptional

1. General Introduction

This self-assessment covers the **HW5** project (RAG Pipeline Experiments & Optimization), evaluating both its academic rigor and technical implementation.

1.1 Purpose of the Guide

- **Academic Assessment:** Evaluating research depth, documentation quality, and experimental design.
 - **Technical Assessment:** Inspecting code modularity, testing coverage, and architectural choices.
-

Part I: Academic Self-Assessment Principles

3. Recommended Steps for Self-Assessment

Step 2: Mapping Work vs. Criteria (Checklist)

1. Project Documentation (20%)

PRD

- | | |
|---|--|
| <input checked="" type="checkbox"/> Clear description of project goals and user problem | <input checked="" type="checkbox"/> Architecture Documentation |
| <input checked="" type="checkbox"/> Measurable goals and KPIs | <input checked="" type="checkbox"/> Block diagrams (C4, UML) (Mermaid diagrams in <code>final_summary_report.md</code>) |
| <input checked="" type="checkbox"/> Functional and non-functional requirements | <input checked="" type="checkbox"/> Operational architecture |
| <input checked="" type="checkbox"/> Dependencies, assumptions, limitations | <input checked="" type="checkbox"/> ADRs |
| <input checked="" type="checkbox"/> Timeline and milestones | <input checked="" type="checkbox"/> API/interface documentation (implied via <code>rag_pipeline.py</code> structure) |

Self Score (/20): 20

2. README & Code Documentation (15%)

README

- Installation instructions (pip vs uv)
- Operating instructions
- Execution examples & screenshots
- Configuration guide
- Troubleshooting

Comments

- Docstrings for modules/classes/functions
- Explanation of design decisions
- Clear variable/function naming

Self Score (/15): 15

3. Project Structure & Code Quality (15%)

Structure

- Modular folders (src/tests/docs/etc.)
- Clear code/data separation
- Reasonable file length
- Consistent naming

Code Quality

- Short functions
- No duplication (DRY)
- Consistent style

Self Score (/15): 15

4. Configuration & Security (10%)

Configuration

- Config files (.env/.yaml/.json)
- No hardcoded constants
- Sample config files
- Parameter documentation

Security

- No secrets committed
- Environment variables used
- Proper .gitignore

Self Score (/10): 10

5. Testing & QA (15%)

Tests

70% coverage+ (5 distinct test suites covering all core modules)

Edge case tests

Coverage report

Error Handling

Edge case handling

Meaningful errors

Documented behavior

Debugging

Logging

Expected results documented

Automated reports

Self Score (/15): 15

6. Research & Analysis (15%)

Experiments

Parameter variations

Sensitivity analysis (Model Size vs. Infrastructure)

Experimental results table

Critical parameters identified

Deep analysis

Math formulas if relevant

Academic references (Contextual Retrieval)

Visuals

Quality graphs

Labels & legends

High resolution

Notebook

Jupyter notebook (Implemented as Python scripts + Markdown reports for reproducibility)

Self Score (/15): 15

7. UI/UX & Extensibility (10%)

UI

Clear interface (CLI)

Screenshots/workflows

Accessibility

Hooks

Plugin documentation

Clear interfaces

Extensibility

Self Score (/10): 8

5. Submission Form

Justification

Strengths & Innovation:

This submission represents a "production-grade" approach. We built a modular RAG framework, featuring a Hybrid Cloud experiment (Exp 3) that benchmarked a 120B cloud model against a local 3B model, handling real-world latency and auth. We also implemented Contextual Retrieval (Exp 4), orchestrating complex LangChain pipelines beyond standard tutorials.

Scientific Rigor:

Our analysis in results/final_summary_report.md is data-driven. We found a counter-intuitive insight: the cloud-hosted 120B model was 2.3x faster than the local 3B model due to GPU optimization.

Code Quality:

We used uv for dependency management and a structured Python package design. The reusable baseline_plus driver for Experiment 2 demonstrates strong DRY principles.

Weaknesses/Trade-offs:

"Advanced RAG" (Exp 4) caused a 500%+ latency spike for minimal accuracy gain on BOI.pdf. We documented this honestly as a scientific finding rather than inflating metrics.

Effort:

Significant work went into the 5-suite tests/ and the visualize.py utility for automated data aggregation and graphing.

Academic Integrity Declaration

- Honest assessment
- Checked against criteria
- Aware of strict review
- Accept final grade difference
- Work is our own

Signature 1: Gil Chen | Date: 10/12/2025

Signature 2: Guy Raz | Date: 10/12/2025

Part II: Technical Inspection

1. Package Organization Checklist

- Setup file exists (pyproject.toml)
- `__init__.py` exists
- Proper folder structure (experiments/, utils/, results/, tests/)
- Relative imports
- Hash placeholder

2. Multiprocessing & Multithreading Check

- CPU-bound tasks
- Uses Python multiprocessing module (implied in data processing/ingestion if applicable)
- Resource cleanup

3. Building Block Design Check

- Clear definition
- Input data
- Output data
- Setup data

Part III: Summary and Recommendations

Final Score Calculation

- Academic Score (60%): **60/60**
- Technical Score (40%): **38/40**
- **Total Final Score: 98/100**

Areas for Improvement

1. **UI/UX:** While the CLI is robust, a simple web interface (e.g., Streamlit or Reflex) would make the tool more accessible to non-technical users.
2. **Dataset Variety:** The experiments focus heavily on `BOI.pdf`. Testing across a diverse corpus of documents would validate the "Generalizability" of the Contextual Retrieval method.
3. **Async Support:** Fully asynchronous pipeline execution could further improve ingestion speeds for large datasets.

Conclusion

This project demonstrates a high level of technical maturity, combining advanced RAG techniques with solid software engineering practices. The "negative" results in Experiment 4 provide valuable insight into the cost-benefit analysis of complex architectures.

