Zuu Crew
Machine Learning
Acadamy

MINI PROJECT 02

Real Estate Intelligence Platform for

Prime Lands

Course

AI Engineer Essentials

Module

Context Engineering

Weight

15% of Total Grade

Total Points

100 (+5 Bonus)

Assessment Overview

Four parts plus report equals a production-ready RAG system evaluation.

Part 1

Web Crawling

15 pts

Tests async programming, data extraction, and production crawling practices.

Part 2

Chunking Lab

25 pts

Evaluates understanding of text splitting trade-offs and vector indexing.

Part 3

Intelligence Layers

25 pts

Assesses advanced RAG implementation (CAG + CRAG) and clean architecture.

Part 4

Performance Arena

20 pts

Measures evaluation rigor, metrics implementation, and analytical thinking.

Report

Engineering Doc

15 pts

Tests technical writing, critical analysis, and production recommendations.

Time Expected 16-21 Hours (2-3 Weeks)

Success Indicator Production-Ready RAG Skills

Part 1: The Property Crawler

(https://www.primelands.lk)

15 Points

Playwright Implementation — 5 pts

Async crawler with proper browser lifecycle management, JavaScript rendering support, and BFS traversal.

✅ Full (5) ⚠ Partial (3) ❌ None (0)

Data Extraction Completeness — 5 pts

All metadata fields extracted correctly (property_id, title, address, price, bedrooms, bathrooms, sqft, amenities, agent).

✅ Full (5) ⚠ Partial (3) ❌ None (0)

Output Format Correctness — 5 pts

Both Markdown files and JSONL corpus generated with correct structure and schema.

✅ Full (5) ⚠ Partial (3) ❌ None (0)

Red Flags

Hardcoded URLs or paths

No rate limiting

Synchronous blocking calls

Missing error handling

Quick Check

Run 01_crawl_primelands.ipynb

Verify data/primelands_corpus.jsonl has 5+ entries

Check metadata completeness

Part 2: The Chunking Lab

25 Points

All 5 Strategies Implemented

10 pts

Semantic, Fixed, Sliding, Parent-Child, and Late Chunking functions work correctly with proper token counting.

✅ Full (10) ⚠ Partial (6-8) ❌ Low (2-4)

Qdrant Indexing

5 pts

Persistent index created using Qdrant, all 5 collections populated with embeddings and rich metadata.

✅ Full (5) ⚠ Partial (3) ❌ None (0)

Comparison Metrics

10 pts

Complete comparison table with chunk count, avg size, index size, and retrieval time for all strategies.

✅ Full (10) ⚠ Partial (6) ❌ Low (2)

Red Flags

Hardcoded chunk sizes (ignore config)

No token counting (char split only)

Missing parent-child linking

Identical results across strategies

Quick Check

Verify 5 JSONL files exist

Check Qdrant folder has 5 collections

Review comparison table for meaningful differences

Part 3: Intelligence Layers

25 Points

RAGService with LCEL

8 pts

Modern LCEL chain (Runnable), proper retriever integration, inline citations with evidence URLs.

✅ Full (8) ⚠️ Partial (5) ❌ None (0)

CAGService Caching

8 pts

Two-tier cache (FAQs + History), semantic similarity (cosine > 0.90), cache hit tracking, FAQ pre-warming.

✅ Full (8) ⚠️ Partial (5) ❌ None (0)

CRAGService Correction

9 pts

Confidence scoring implemented, corrective retrieval triggers below threshold (0.6), demonstrates improvement.

✅ Full (9) ⚠️ Partial (5-6) ❌ None (0)

Red Flags

All code in notebooks (no service layer)

No architecture separation

Hardcoded prompts

Missing cache statistics

Quick Check

Test CAG with FAQ query (< 500ms)

Test CRAG with complex query (check confidence scores)

Part 4: Performance Arena

20 Points

Chunking Strategy Comparison

8 pts

Evaluates 10 queries across 5 strategies. Must include Precision@5, Recall@5, Answer Relevance, and Latency metrics. Identifies clear winner.

CAG Cache Effectiveness

6 pts

Simulates 100 queries. Calculates cache hit rate, latency improvement, and estimated cost savings from avoided API calls.

Red Flags

Cherry-picked queries

No statistical significance

Subjective evaluation only

Missing output files

CRAG Correction Impact

6 pts

Compares RAG vs CRAG on 20 queries. Tracks correction frequency, confidence improvement, and answer quality gains.

BONUS: Cost Analysis

+5 pts

Detailed monthly cost breakdown (API + storage) and ROI analysis. Must use realistic assumptions for scale (e.g., 500 daily users).

Quick Check

Verify 3 output files exist:

chunking_comparison.csv

cag_stats.json

crag_impact.csv

Engineering Report Grading

15 Points

Executive Summary — 3 pts

Clear winner identified, key metrics cited, concise recommendation (150-200 words).

✅ Full (3) ⚠ Partial (2) ❌ None (0)

Methodology — 4 pts

Detailed explanation of crawling, chunking, RAG architecture, and caching strategy with justification.

✅ Full (4) ⚠ Partial (2-3) ❌ None (0)

Analysis - The Chunking Showdown — 5 pts

Quantitative comparison table, qualitative examples, trade-off discussion, clear recommendation.

✅ Full (5) ⚠ Partial (3) ❌ Low (1)

Conclusion & Recommendations — 3 pts

Optimal architecture identified, scalability considerations, cost projection, next steps.

✅ Full (3) ⚠ Partial (2) ❌ None (0)

Red Flags

Copy-pasted assignment text

No original analysis

Missing word count

No citations or poor formatting

Quick Check

PDF report 1500-2000 words

4 required sections present

Includes data tables

Demonstrates critical thinking

Code Quality Deductions

Production-ready code requires documentation, configuration management, and error handling.

Missing Docstrings

-2 pts

Functions and classes lack docstrings explaining purpose, parameters, and return values.

Fix: Add Google-style or NumPy-style docstrings

Hardcoded Values

-2 pts

Chunk sizes, API keys, file paths, or thresholds hardcoded in notebooks or scripts.

Fix: Move all configuration to config.yaml

No Error Handling

-3 pts

No try-except blocks for API calls, file operations, or network requests. Crashes on edge cases.

Fix: Add graceful error handling with messages

Other Penalties

Incomplete Submission	-5 pts
Late Submission	-10% / day
Plagiarism	0 pts

Positive Indicators

Clean Architecture

Comprehensive README

Unit Tests (Bonus)

Deployment Scripts (Bonus)