AI / ML Engineer
Take-Home Assignment

Technical Assessment

Overview

Time allocation:	2-3 days from assignment receipt
Objective:	Build a production-ready AI agent demonstrating prompt engineering, optimization, and evaluation

The Challenge

Build an AI agent that demonstrates the engineering rigor needed for enterprise AI - not just "does it work" but "how do you know it works, and how do you make it better?"

Choose Your Industry Scenario

Pick ONE industry you understand or want to learn:

- **Finance:** Analyze financial documents + transaction data

  Documents: Annual reports, regulatory filings, financial statements
  Data: Transaction logs, portfolio data, market data

- **Healthcare:** Process medical information + patient data

  Documents: Medical guidelines, research papers, procedure manuals
  Data: Treatment outcomes, patient records (anonymized/synthetic)

- **Logistics:** Parse shipping documentation + operational data

  Documents: Shipping procedures, compliance docs, equipment manuals
  Data: Shipment tracking, delivery metrics, route data

- **Retail:** Product catalogs + inventory/sales data

  Documents: Product manuals, supplier contracts, policy documents
  Data: Inventory levels, sales transactions, customer data

- **Insurance:** Policy documents + claims data

Documents: Policy documents, coverage guidelines, legal terms
Data: Claims records, risk assessments, payout data

Or propose your own - just explain why it's relevant

Required Components (All Must Be Present)

1. RAG Implementation (35%)

Build a working document retrieval and generation system:

Must include:

- Document ingestion pipeline (PDF, text, or other formats)
- Chunking strategy (explain your approach and why)
- Vector database integration
- Semantic search and retrieval
- Query handling with source citations
- Basic error handling

What we're evaluating:

- Does it actually work for realistic queries?
- Is the chunking strategy justified?
- Are sources properly cited?
- Does it handle edge cases gracefully?

Data requirements:

- Minimum 10-15 documents (can be public datasets, documentation, papers, etc.)
- Must be relevant to your chosen industry
- Include a DATA_SOURCES.md explaining where you got data and why

2. Prompt Engineering & Optimization (30%)

This is critical - it's what AI engineers do daily.

Must demonstrate:

- At least **3 different prompt approaches** for the same core task
- Systematic comparison across:
- • **Accuracy/Relevance** (subjective or scored)
- • **Latency** (response time in ms)
- • **Cost** (token usage, estimated $/query)
- Clear documentation of which approach you selected and **why**

Example approaches to try:

- Zero-shot vs few-shot
- Different system prompts
- Chain-of-thought vs direct answer
- Structured output vs free-form
- Different temperature/parameters
- Query rewriting strategies
- Different retrieval strategies (top-k variations, reranking, etc.)

What we're evaluating:

- Can you systematically improve prompts?
- Do you understand accuracy/cost/latency trade-offs?
- Can you make data-driven decisions?
- Do you test rigorously?

3. Evaluation Framework (20%)

You can't improve what you don't measure.

Must implement:

- Test set: Minimum 10 representative queries with expected outcome types
- At least 3 evaluation metrics relevant to your use case
- Results across your different prompt approaches
- Analysis of what you learned

Example metrics (choose what fits):

- Answer correctness (binary or scored)
- Source citation accuracy
- Response completeness
- Latency (p50, p95, p99)
- Token usage / cost per query
- Retrieval precision (did it fetch relevant chunks?)
- Hallucination rate (answers not supported by sources)

What we're evaluating:

- Are your metrics appropriate for the task?
- Is your test set representative?
- Can you draw insights from results?
- Do you think like a scientist/engineer?

4. Production Considerations (15%)

Show you understand this isn't a toy project.

Must address:

Error Handling:

- What happens when retrieval fails?
- What if the LLM times out?
- What if the query is outside your system's scope?
- How do you handle malformed inputs?

Cost Analysis:

- Estimate cost per 1000 queries (your final approach)
- What are the main cost drivers?
- Where would you optimize if cost was 10x higher?

Monitoring Strategy:

- What metrics would you track in production?
- What alerts would you set up?
- How would you detect degradation?

Scaling Considerations:

- Current limitations (query throughput, data volume, etc.)
- What would break first at 10x scale?
- How would you address it?

What we're evaluating:

- Do you think beyond happy paths?
- Do you understand real-world constraints?
- Can you anticipate problems?
- Is your thinking practical and concrete?

Deliverables Checklist

Code & Configuration

■ All source code with clear structure

■ requirements.txt or pyproject.toml

■ .env.template (with API keys removed)

■ Clear directory organization

Documentation (All Required)

■ **README.md -** Setup, architecture, how to run

■ **DATA_SOURCES.md -** Where you got data and why

■ **PROMPTS.md -** All approaches, comparison, final choice

■ **EVALUATION.md -** Test set, metrics, results, analysis

■ **PRODUCTION.md -** Error handling, cost, monitoring, scaling

Demo (Choose One)

■ Video walkthrough (5-10 min showing it work + explaining decisions)

■ Screenshots with explanations

■ Well-documented script output

■ Live demo (if invited to interview)

Evaluation Rubric

Component	Criteria	Points
RAG Implementation (35 points)	• Core retrieval works correctly • Chunking strategy justified • Source citations accurate • Error handling present	15 pts 8 pts 7 pts 5 pts
Prompt Engineering (30 points)	• Tested 3+ approaches • Systematic comparison with metrics • Clear data-driven justification	10 pts 12 pts 8 pts
Evaluation Framework (20 points)	• Appropriate metrics selected • Representative test set with results • Meaningful insights drawn	7 pts 8 pts 5 pts
Production Considerations (15 points)	• Error handling strategy • Cost analysis with optimization ideas • Monitoring/scaling plan	5 pts 5 pts 5 pts

Minimum passing score: 65/100

What Success Looks Like

Minimum Viable Submission (Pass)

- RAG agent works for your test queries
- Tried 3 prompt approaches with basic comparison
- Has evaluation results showing what you measured
- Documented production considerations clearly
- Code runs without errors

Strong Submission

- RAG handles edge cases well
- Sophisticated prompt comparison (e.g., tested combinations)
- Comprehensive evaluation with insights that drove decisions
- Production plan shows understanding of real constraints
- Clear, well-organized code and documentation

Exceptional Submission

Everything above, plus:

- Novel approach to a problem (chunking, retrieval, prompting)
- Deep domain insights applied to implementation
- Cost optimization analysis with actual trade-offs tested
- Monitoring strategy tied to real failure modes you discovered

Data Sourcing Guidelines

Don't overthink this - we want to see your engineering, not your data collection skills.

Good data sources:

- Public datasets (Kaggle, HuggingFace, government open data)
- Documentation/manuals (company websites, GitHub repos)
- Research papers (arXiv, PubMed)
- Synthetic/generated data (if you explain the generation)
- Sample data from previous projects (if you have it)

What we're NOT looking for:

- Perfect, production-ready datasets
- Large volumes (10-15 documents is fine)
- Real proprietary data (use public or synthetic)
- Extensive time spent on data cleaning (focus on the agent)

Industry Context Bonus (+10 points)

If you have domain expertise in your chosen industry:

Add a section to your README:

- **Domain Challenges:** What makes this hard in your industry
- **Regulatory/Compliance:** Relevant regulations you considered
- **Enterprise Deployment:** How you'd adapt this for real production use
- **Domain-Specific Evaluation:** Additional metrics that matter in this industry

This helps us see your domain expertise and how you think about industry requirements (a key part of the role).

Submission

Submission details including deadline, repository access, and contact information will be provided when you receive this assignment.

What We Value

In priority order:

1. **Engineering rigor** - Systematic approach to optimization and evaluation
2. **Working software** - It runs, handles test cases, has error handling
3. **Clear thinking** - Decisions explained with data/reasoning
4. **Production mindset** - Understanding of real-world constraints
5. **Communication** - Clear documentation and explanations

What we don't expect:

- Perfect code or zero bugs
- Production-scale infrastructure
- Exhaustive test coverage
- Polished UI (focus on backend)
- Novel research contributions

Red Flags (Automatic Rejection)

- Code doesn't run or has no setup instructions
- No prompt comparison (core requirement)
- No evaluation results (can't assess anything)
- Copy-pasted code without understanding
- No documentation of any decisions
- Submitted after deadline without prior communication

We look forward to seeing your work!