Agent Development Internship

Technical Assessment Task

Task Overview

Build a multi-agent system that processes supplier invoices, extracts structured data, matches them against purchase orders, and intelligently flags discrepancies. This task tests your ability to build production-grade agentic AI systems that handle real-world messiness.

Objective

Create an invoice reconciliation agent system that can autonomously process invoices in various formats, extract line items, match them against a purchase order database, detect discrepancies, and provide intelligent recommendations for resolution.

Core Agent Architecture

You must implement at least these four agents:

- Document Intelligence Agent - Extract structured data from messy invoice PDFs (various formats, handwritten notes, stamps, poor scans, rotated images)
- Matching Agent - Compare invoice line items to PO database, handle partial matches, implement fuzzy matching logic
- Discrepancy Detection Agent - Flag price mismatches, quantity differences, missing PO references with confidence scoring
- Resolution Recommendation Agent - Suggest actions (auto-approve, request clarification, escalate to human) based on severity and confidence

Technical Constraints

- Must use an agentic framework (LangGraph preferred, but CrewAI, AutoGPT, or custom implementations are acceptable)
- Agents must communicate and handoff intelligently - no simple linear pipelines
- Must handle uncertainty with confidence scoring - agents should know when they're unsure and need human review
- Must work with multiple invoice formats (clean PDFs, scanned images, photos, rotated documents)
- Include reasoning transparency - explain why agents made specific decisions
- Performance target: Process all 5 test invoices in under 5 minutes total

Deliverables (72-hour deadline)

1. Working Code (GitHub Repository)

- Well-structured code with clear README

- Requirements.txt or package.json with all dependencies
- Setup instructions that actually work
- Output results as JSON following the provided schema

2. Demo Video (5 minutes maximum)

Record a video (Loom, YouTube unlisted, or similar) demonstrating:

- Processing all 5 test invoices
- Agent decision-making process and reasoning chain
- How your system handles the price discrepancy (Invoice 4)
- How your system handles missing PO reference (Invoice 5)
- What breaks your system and why (be honest about limitations)

3. Written Analysis (500 words maximum)

Answer these questions:

- Where does OCR/extraction fail? How do your agents compensate?
- How would you improve accuracy from 70% to 95%?
- How would you validate this system at 10,000 invoices/day scale?

Test Data Provided

You will receive:

- 5 invoice files (PDFs/images) with escalating difficulty
- Purchase order database (JSON file with 20 POs)
- Reconciliation rules document
- Expected output format (JSON schema)
- One complete example showing expected quality

Invoice Difficulty Progression

Invoice 1 - Easy Baseline:

Clean PDF with clear table structure. Perfect match to PO. Tests basic extraction and matching.

Invoice 2 - Real World Scan:

Scanned image, slightly rotated, minor quality issues. Tests OCR quality and preprocessing.

Invoice 3 - Format Variation:

Different template, line items in different order. Tests template-agnostic extraction and flexible matching.

Invoice 4 - Price Discrepancy (Critical Test):

Professional looking invoice with one line item showing 10% price increase vs PO. Everything else perfect. Tests discrepancy detection and confidence scoring.

Invoice 5 - Missing PO Reference (Critical Test):

Invoice with no PO number or invalid PO number, but products exist in database. Tests fuzzy matching and handling missing data.

Evaluation Criteria

Criteria	Weight
Agent orchestration - Do agents communicate intelligently? Good error handling? Clear handoffs?	35%
Extraction accuracy - Can it handle messy real-world documents? OCR quality?	25%
Matching logic - Does it catch discrepancies? What's the false positive rate? Confidence scoring quality?	20%
Code quality - Maintainable, extensible, well-documented, follows best practices?	20%

Red Flags (Automatic Rejection)

- Hardcoded rules instead of agent reasoning
- No error handling or recovery mechanisms
- Cannot explain agent decisions or reasoning chain
- Just wrapping API calls without real agentic behavior
- Missing both intentional test cases (price discrepancy and missing PO)
- Code doesn't run or has no documentation

Bonus Challenge (Optional)

Implement a feedback loop: add a 'Human Reviewer Agent' that simulates catching common conversion errors and sends feedback back through the system. Show that agents can learn from and adapt to review feedback. This demonstrates understanding of iterative agent systems.

Submission Instructions

Email the following to internships@niyamrai.com:

- GitHub repository link (make sure it's public or add our review account)

- Demo video link
- Written analysis (PDF or Markdown)

Deadline: 72 hours from when you receive this task package

Late submissions will not be considered. Start immediately.

What We're Looking For

This task is deliberately challenging. We expect most candidates to struggle with some aspects. What matters is:

- How you approach complex problems
- Your ability to build systems that handle uncertainty
- Whether you understand what 'agentic' actually means vs just prompt engineering
- Your honesty about limitations and failure modes
- Code quality and architectural thinking

This is hard. That's the point. Show us what you can do.

Questions?

For clarifications about the task (not technical help): internships@niyamrai.com

We will not provide technical support or debugging help during the assessment.

Good luck. We're looking for builders, not bullshitters.