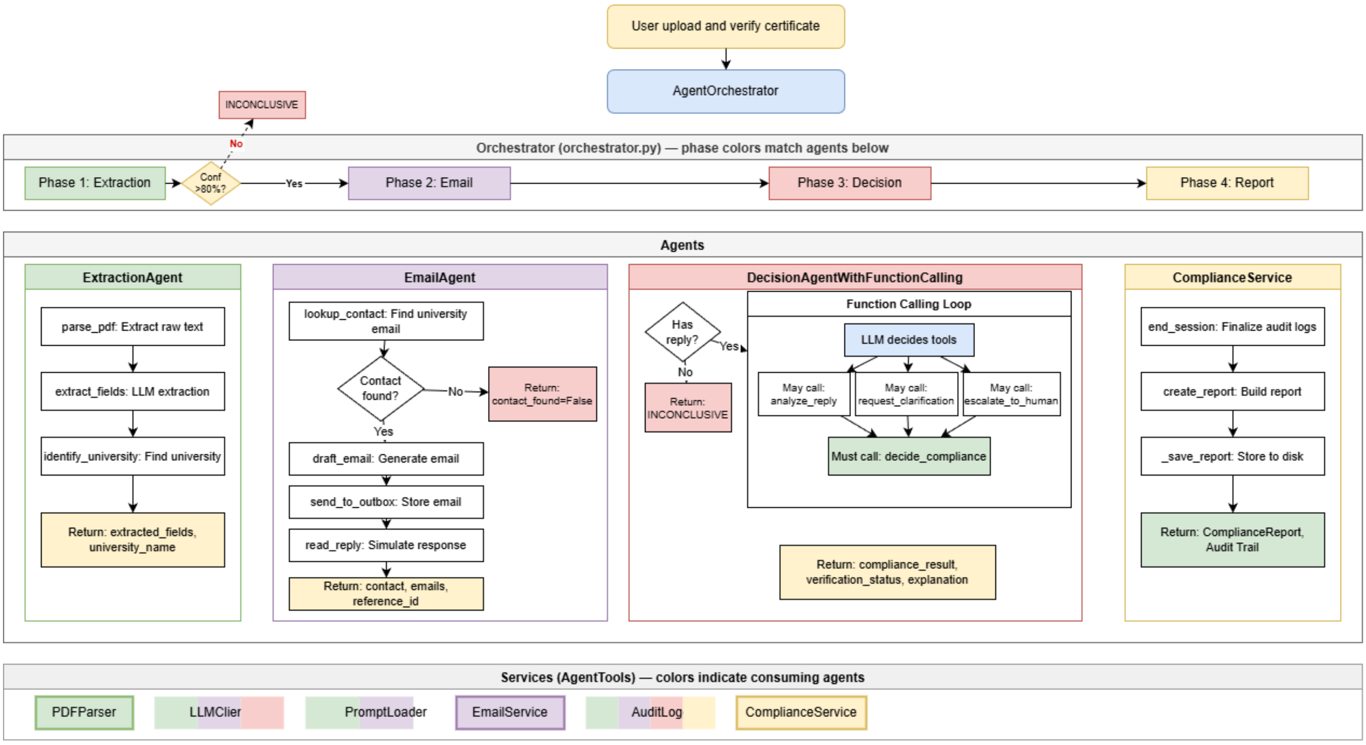
Research & Engineering Insight

1. Research Insight

Architecture Overview

The workflow follows a fixed sequential order: **Extraction** → **Email** → **Decision** → **Report**. Compliance processes require predictable, auditable execution paths—the Orchestrator enforces this strictly.


User upload and verify certificate
AgentOrchestrator
INCONCLUSIVE
No
Orchestrator (orchestrator.py) — phase colors match agents below
Phase 1: Extraction
Conf >80%?
Yes
Phase 2: Email
Phase 3: Decision
Phase 4: Report
Agents
ExtractionAgent
parse_pdf: Extract raw text
extract_fields: LLM extraction
identify_university: Find university
Return: extracted_fields, university_name
EmailAgent
lookup_contact: Find university email
Contact found?
No
Return: contact_found=False
Yes
draft_email: Generate email
send_to_outbox: Store email
read_reply: Simulate response
Return: contact, emails, reference_id
DecisionAgentWithFunctionCalling
Has reply?
Yes
No
Return: INCONCLUSIVE
Function Calling Loop
LLM decides tools
May call: analyze_reply
May call: request_clarification
May call: escalate_to_human
Must call: decide_compliance
Return: compliance_result, verification_status, explanation
ComplianceService
end_session: Finalize audit logs
create_report: Build report
_save_report: Store to disk
Return: ComplianceReport, Audit Trail
Services (AgentTools) — colors indicate consuming agents
PDFParser
LLMClier
PromptLoader
EmailService
AuditLog
ComplianceService

Three specialized agents handle each domain:

Agent	LLM Usage	Behavior
ExtractionAgent	Content generation (field extraction)	Deterministic tool sequence
EmailAgent	Content generation (email drafting)	Deterministic tool sequence
DecisionAgent	Tool selection via Function Calling	Adaptive—handles ambiguous replies

This separation enables clear audit trails—each agent's actions are logged independently with timestamps and inputs/outputs.

Why Handcrafted Orchestration + Function Calling?

The alternative was full LangChain/LangGraph—letting the framework handle both orchestration and tool execution. Given the prototype scope and timeline, the **handcrafted approach** was chosen: zero additional dependencies, and the LLM decides only where needed instead of every step.

LangChain excels at dynamic agent behavior—and with more time, it could be configured to enforce sequential phases. However, for a fixed compliance workflow, handcrafted orchestration achieves the same result with less complexity. **Function Calling in DecisionAgent** provides the balance: the workflow stays deterministic, but the LLM retains intelligent tool selection where ambiguity exists (interpreting email replies). This avoids both extremes—fully autonomous agents and fully hardcoded logic. If requirements evolve (e.g., dynamic phase ordering), LangGraph would be reconsidered.

Trade-offs Accepted

What I Gave Up	Why It's Acceptable
Dynamic workflow changes—cannot reorder phases at runtime	Compliance workflows are inherently fixed; dynamic ordering would break audit requirements
Framework ecosystem—no LangChain integrations (memory, callbacks, tracing)	Prototype scope doesn't require these; can add later if needed
Less "agent-like"—most steps are deterministic, not LLM-driven	Deliberate choice: predictability over autonomy for regulated processes

Function Calling in DecisionAgent compensates for the third point—the system retains intelligent decision-making where it matters most.

2. Engineering Solution Thinking

Tools, Prompts, and Models Structure

Tools use a mixin pattern to group domain-specific logic, simplifying code structure. **Prompts** are externalized to **Jinja2 templates** (`config/prompts/*.j2`) for independent versioning. **University Mappings** are config-driven via `config/universities.json`—no code deployment needed for updates.

Failure Cases

Failure	Handling
PDF unreadable	LLM Vision API extracts text from rendered images; handles scanned documents
Low-quality/damaged PDF	Vision API detects visual damage; extraction confidence ≤80% triggers `INCONCLUSIVE`
University not found	Return `INCONCLUSIVE`; flag for manual review
Ambiguous/fraud reply	Lower confidence score triggers human review queue
LLM hallucination	Schema validation on all outputs; reject malformed responses

The decision to use **LLM Vision for all PDFs** (instead of traditional Python PDF readers like PyMuPDF) was deliberate: it avoids arbitrary fallback thresholds and provides consistent handling regardless of PDF type—digital or scanned.

Auditability

Every tool call creates an `AuditLogEntry` with: timestamp, agent name, tool name, sanitized inputs/outputs, and success status. Credentials are redacted and long values truncated in logs. PII sanitization is extensible via config. The audit trail enables **timeline reconstruction** (sequential step numbering), **accountability** (which agent made which decision), and **reproducibility** (Session ID links all related logs).

3. Real-World Applicability

Production Extensions

Current (Prototype)	Production Path
Simulated outbox/inbox	SendGrid/AWS SES for sending; IMAP/Gmail API for receiving
LLM Vision for PDFs	Already production-ready; handles scanned certificates
JSON university config	Database + external APIs (My eQuals, National Clearinghouse, etc.)

Scaling to 1,000+ Checks/Day

Architecture: `FastAPI → Redis Queue → Celery Workers`. Features include horizontal scaling via worker pool, per-university rate limiting (avoid spam triggers), and dead letter queue for failed verifications.

Monitoring & Reliability: Target 30s processing time (SLA); alert if queue exceeds threshold. Retries use exponential backoff (10s → 30s → 60s) for failures. Observability via Prometheus metrics on processing time, result distribution, and error rates.

4. Security & Compliance Thoughts

Data Privacy

Audit logs automatically redact sensitive credentials. Production plans include standard encryption (AES-256, TLS 1.3) and GDPR/CCPA-compliant retention policies.

Model Hallucination Risks

Mitigation	Implementation
Schema Validation	Pydantic models validate all LLM outputs to ensure structural integrity
Structured Outputs	Function Calling restricts responses to valid enum values, preventing open-ended fabrications
Extraction Confidence	Vision API assesses document quality; confidence ≤80% escalates to human review
Decision Confidence	Reply analysis scores below 0.7 trigger fallback to human review

Traceability

The detailed audit trail allows for complete timeline reconstruction. A unique **Session ID** links every log entry for a specific verification request, ensuring that every automated decision can be traced back to its inputs and timestamp.

Human-in-the-Loop

This architecture prioritizes safety by treating the AI as an **accelerator, not a replacement**. Full automation is strictly reserved for clear-cut success paths.

Automatic Breaks are triggered at two checkpoints:

1. **Extraction Phase**: Document quality confidence ≤80% immediately returns `INCONCLUSIVE`—damaged or altered documents are flagged before verification proceeds.
2. **Decision Phase**: Any result marked `INCONCLUSIVE` / `NOT_COMPLIANT`, or `COMPLIANT` with low confidence, is routed to a human review queue.

Only high-confidence extractions from high-quality documents with clear `COMPLIANT` responses bypass this safety layer.