Implementation Plan

Template Intelligence Engine

AdvisoryAI Internal Platform

Document Purpose

This document defines a phased, execution-ready implementation plan for building the Template Intelligence Engine. It reflects how a senior engineer would structure delivery: backend-first, incremental, resilient, and demo-focused.

1 Execution Strategy

The system will be implemented in clearly defined phases. The backend will be completed end-to-end before any frontend work begins. Each phase produces concrete, verifiable deliverables.

2 Phase 0: Project Initialization and Guardrails

Goal

Establish a stable foundation so all future work compounds cleanly.

Sub-phases

- Repository and Python environment setup
- FastAPI application bootstrap
- Logging directory and configuration
- Database, Redis, and object storage connectivity

Deliverables

- Application starts cleanly
- Health endpoint operational
- Logs written to file
- Infrastructure connectivity verified

3 Phase 1: Backend Domain Skeleton

Goal

Lay down the complete backend structure before implementing behaviour.

Sub-phases

- Domain scaffolding for template, section, document, job, and audit
- API router scaffolding with request and response schemas
- Worker process scaffolding

Deliverables

- Backend compiles
- Routes are registered
- Workers start successfully

4 Phase 2: Persistence Layer and Schema

Goal

Implement all persistence before adding intelligence.

Sub-phases

- PostgreSQL schema and migrations
- Repository implementations per domain
- Object storage integration for templates and outputs

Deliverables

- Versioned data persistence
- Audit logs recorded
- Files stored and retrievable from object storage

5 Phase 3: Job System and Pipeline Orchestration

Goal

Make the system asynchronous, resilient, and observable.

Sub-phases

- Job lifecycle management
- Event-driven pipeline orchestration
- Job status and error APIs

Deliverables

- Jobs survive restarts
- Failures are visible and recoverable
- Pipelines resume correctly

6 Phase 4: Template Parsing and Structure Inference

Goal

Convert Word documents into structured, machine-readable representations.

Sub-phases

- Word document ingestion and validation
- Parsing via Python libraries
- LLM-assisted structure inference

Deliverables

- Parsed template representations stored
- Structural consistency validated

7 Phase 5: Section Classification

Goal

Automatically identify static and dynamic sections.

Sub-phases

- Rule-based classification
- LLM semantic classification
- Section metadata persistence

Deliverables

- Sections classified without human input
- Prompt configurations stored

8 Phase 6: Content Generation and Document Assembly

Goal

Generate complete, formatted Word documents.

Sub-phases

- Section-level content generation
- Document assembly with formatting preservation
- Versioned output storage

Deliverables

- Documents open cleanly in Word
- Formatting preserved
- Version history maintained

9 Phase 7: Regeneration and Change Handling

Goal

Enable safe iteration without reprocessing everything.

Sub-phases

- Section-level regeneration
- Full document regeneration
- Template version updates

Deliverables

- Regeneration produces new versions
- Audit chain preserved

10 Phase 8: Observability and Demo Hardening

Goal

Make the system stable, debuggable, and demo-ready.

Sub-phases

- Structured logging finalisation
- Error scenario validation
- Demo data seeding

Deliverables

- Clear error visibility
- Stable demo flows

11 Phase 9: Frontend Implementation

Goal

Expose backend functionality via an internal dashboard.

Sub-phases

- React + TypeScript frontend setup
- Core pages for templates, documents, and jobs
- Backend API integration

Deliverables

- End-to-end flow visible in UI
- No manual backend interaction

12 Phase 10: Final Polish and Submission

Goal

Make the system intentionally complete and senior-grade.

Sub-phases

- Code cleanup and consistency
- Full happy-path validation
- Submission packaging

Deliverables

- Submission-ready system
- Clear demo narrative

13 Conclusion

This implementation plan is sequential, execution-focused, and designed to minimise rework. It reflects senior engineering judgement under time constraints while preserving long-term extensibility.