P R O D U C T R E Q U I R E M E N T S
D O C U M E N T

AI-Powered Bid & Proposal Assistant

Engineering & Business Development Workflow Automation

A complete technical blueprint for building an AI workflow that processes a 120-page RFP in under 30 minutes — from document ingestion to submission-ready proposal pack.

Version	**1.0**
Status	**Draft for Review**
Classification	**Confidential**
Date	**March 2026**

Table of Contents

1. Executive Summary

This Product Requirements Document defines the complete technical and functional specification for an AI-Powered Bid & Proposal Assistant. The system automates the end-to-end proposal lifecycle for engineering and business development teams — from RFP ingestion to submission-ready document generation.

1.1 Problem Statement

Engineering firms spend 40–120 person-hours per proposal cycle on repetitive document review, requirement extraction, compliance cross-checking, and response drafting. This creates three critical business risks:

- Revenue leakage from missed bid deadlines or under-resourced submissions
- Compliance exposure from overlooked requirements or inconsistent commercial positions
- Knowledge loss when experienced staff leave without codifying institutional proposal knowledge

1.2 Solution Overview

An orchestrated multi-agent AI workflow that processes RFP documents through a nine-step pipeline: ingest, classify, extract, retrieve, draft, validate, flag, deliver, and learn. The system uses LLM-powered agents coordinated through a graph-based orchestrator, with human-in-the-loop approval gates for risk decisions.

1.3 Target Outcome

TARGET SCENARIO

A firm receives a 120-page RFP for a $4M infrastructure contract. Within 30 minutes the AI classifies all documents, extracts 47 requirements, retrieves matching CVs and past proposals, drafts a compliant technical response, and flags 3 commercial risks for human review — submission-ready.

2. Product Vision & Objectives

2.1 Vision Statement

Transform proposal management from a labor-intensive, knowledge-siloed process into an AI-augmented digital production line where humans focus on strategy and judgment while AI handles extraction, drafting, validation, and institutional memory.

2.2 Strategic Objectives

#	Objective	Key Result	Metric
O1	Reduce proposal cycle time	Process 120-page RFP in <30 minutes	Time-to-first-draft
O2	Improve bid quality	Zero missed compliance requirements	Requirement coverage %
O3	Increase bid throughput	Handle 3x more concurrent proposals	Proposals per quarter
O4	Capture institutional knowledge	Reuse rate >80% on similar bids	Template reuse rate
O5	Reduce commercial risk	Flag 100% of high-risk clauses	Risk detection recall

2.3 User Personas

Persona	Role	Primary Need	Key Workflow
Bid Manager	Leads proposal response	End-to-end visibility and control	Upload RFP, review AI output, approve final
Technical Lead	Engineers the solution	Quick extraction of technical scope	Review requirements, validate technical response
Commercial Manager	Owns pricing and risk	Risk-flagged commercial terms	Review pricing, approve assumptions
BD Director	Wins the work	Pipeline throughput and win rate	Dashboard analytics, go/no-go decisions
System Admin	Manages the platform	Reliability and compliance	Configure integrations, manage access

3. System Architecture

The system follows a four-layer architecture separating presentation, orchestration, execution, and data concerns. This design enables independent scaling, clear security boundaries, and technology substitution without full-stack refactoring.

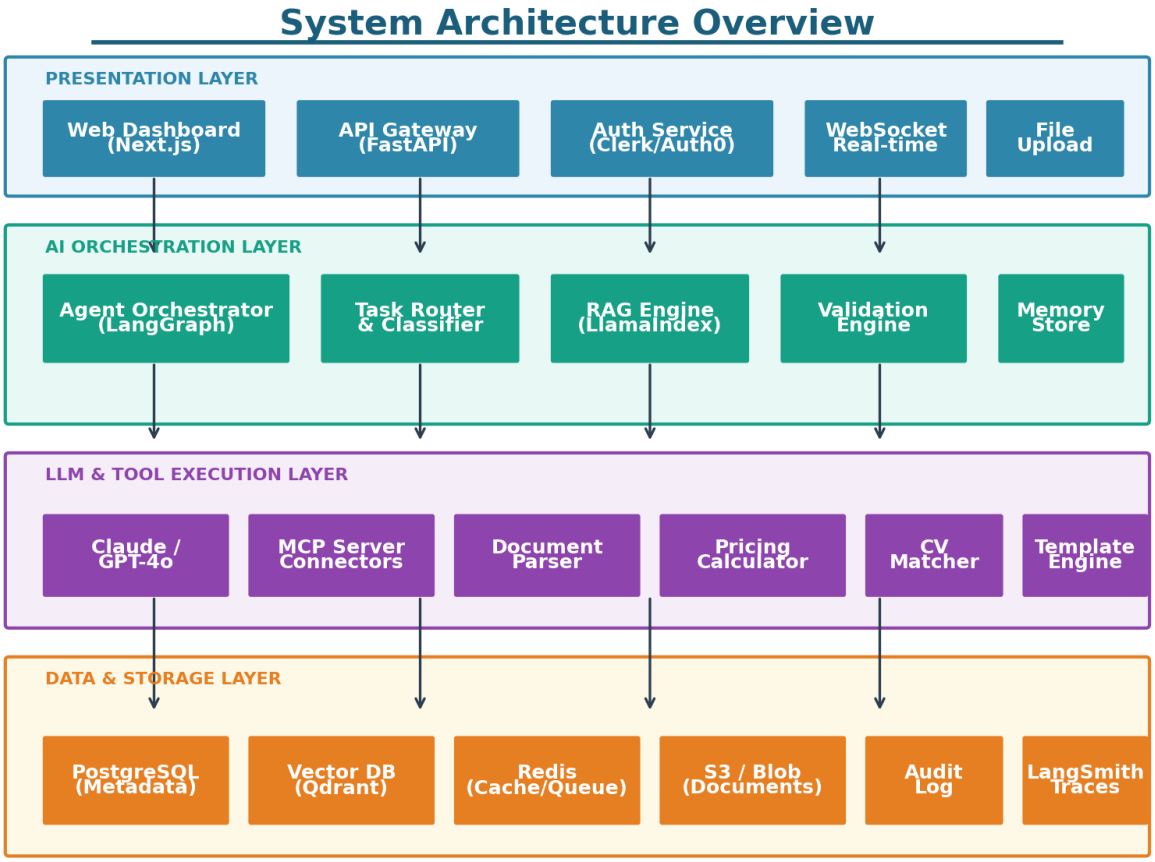

Figure 1: Four-Layer System Architecture

3.1 Layer Descriptions

Presentation Layer

The user-facing tier built on Next.js 15 with React Server Components for optimal performance. It includes a real-time WebSocket connection for live progress updates during AI processing, a RESTful API gateway via FastAPI, and authentication through Clerk or Auth0 with SSO support.

- Web Dashboard — Upload RFPs, monitor processing, review drafts, approve submissions
- API Gateway — RESTful endpoints with OpenAPI 3.1 documentation, rate limiting, and request validation

- Real-time Updates — WebSocket channels for pipeline progress, agent status, and notification delivery

AI Orchestration Layer

The brain of the system. A LangGraph-based supervisor agent manages task planning, delegation, and state transitions across specialized sub-agents. The RAG engine (LlamaIndex) provides hybrid retrieval combining vector similarity search with keyword matching (BM25) for maximum recall.

- Agent Orchestrator — LangGraph state graph managing multi-step workflows with conditional branching
- Task Router — Classifies incoming documents and routes to appropriate processing agents
- RAG Engine — Hybrid retrieval (dense + sparse) across company knowledge bases
- Validation Engine — Multi-pass compliance checking against extracted requirement registry
- Memory Store — Persistent conversation and project memory for cross-bid learning

LLM & Tool Execution Layer

Houses the LLM providers (Claude Opus/Sonnet, GPT-4o as fallback), MCP server connectors for enterprise integrations, and specialized tools for document parsing, pricing calculations, CV matching, and template rendering.

Data & Storage Layer

PostgreSQL 16 for relational metadata, Qdrant for vector embeddings, Redis 7 for caching and job queues, and S3-compatible object storage for documents. All operations are logged to an immutable audit trail with LangSmith trace integration for observability.

4. AI Processing Pipeline

The core value of the system is a nine-step AI pipeline that transforms raw RFP documents into a submission-ready proposal pack. Each step is implemented as a discrete agent node in the LangGraph state graph, enabling retry, branching, and human-in-the-loop intervention at any point.

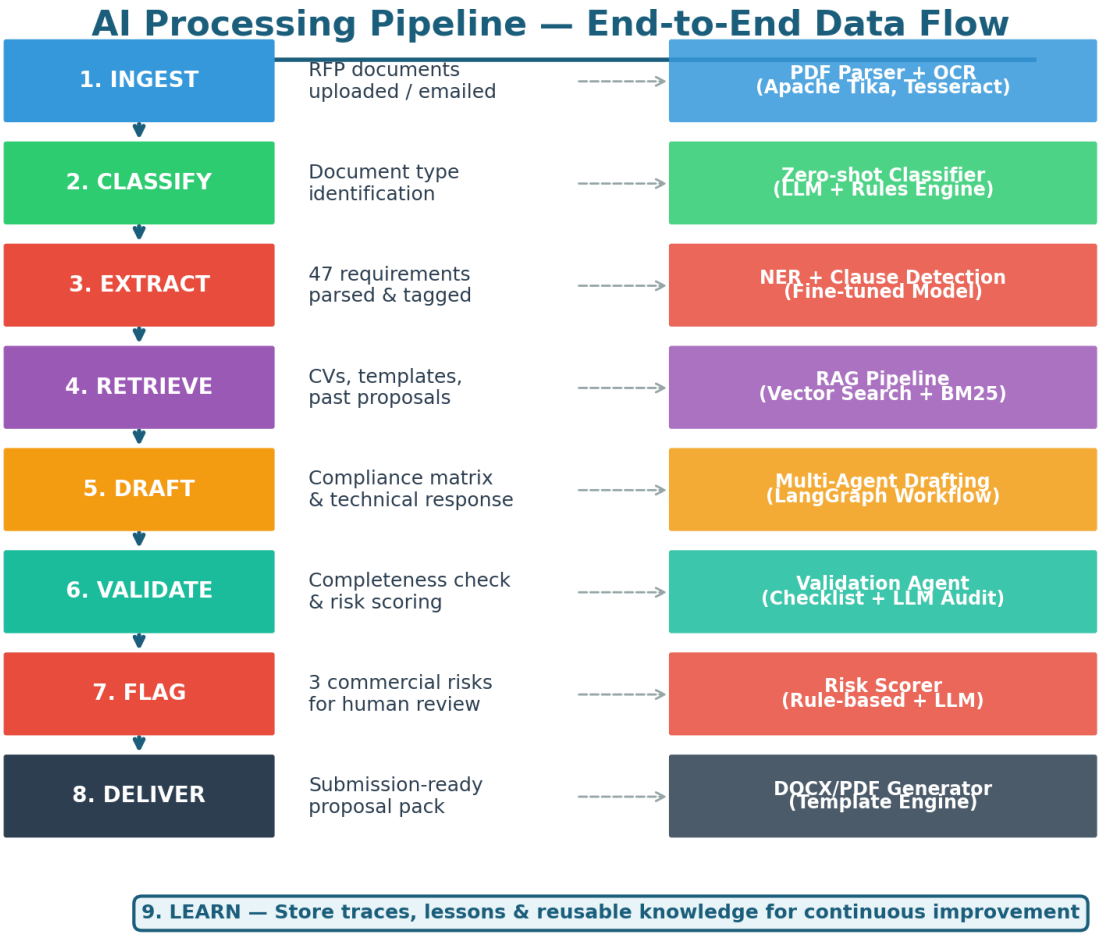

Figure 2: Nine-Step AI Processing Pipeline with Technology Mapping

4.1 Pipeline Step Specifications

Step	Name	Description	Technology	Output
1	Ingest	Accept RFP documents via upload, email, or API. Extract text from PDF/DOCX/scanned images.	Apache Tika, Tesseract OCR, PyPDF2	Raw text corpus + document metadata

2	Classify	Identify document types: technical specs, commercial terms, compliance requirements, appendices, drawings.	LLM zero-shot classification + rule engine	Classified document registry
3	Extract	Parse requirements, deadlines, evaluation criteria, technical clauses, and commercial conditions. Tag with priority and category.	NER model + LLM extraction + regex patterns	Structured requirement registry (47 items in sample)
4	Retrieve	Search company knowledge base for matching CVs, past proposals, rate cards, technical datasheets, and standard responses.	Qdrant vector search + BM25 hybrid retrieval	Ranked retrieval results per requirement
5	Draft	Generate compliance matrix, technical response sections, assumptions, exclusions, and executive summary using retrieved context.	LLM generation with structured prompts + template engine	Draft proposal document
6	Validate	Check draft against requirement registry for completeness. Verify pricing consistency, cross-reference assumptions.	Validation agent + rule-based checks + LLM audit	Validation report with coverage score
7	Flag	Identify commercial risks, ambiguous clauses, unusual terms, and items requiring human judgment.	Risk scoring model + LLM analysis	Risk register (3 flags in sample)
8	Deliver	Compile final proposal pack in required format. Generate DOCX/PDF with correct styling and structure.	python-docx, ReportLab, template engine	Submission-ready proposal pack
9	Learn	Store processing trace, human corrections, and reusable patterns for future bid improvement.	LangSmith traces + PostgreSQL + vector store update	Updated knowledge base + audit trail

4.2 Pipeline Performance Targets

Metric	Target	Measurement
End-to-end processing time	< 30 minutes for 120-page RFP	Clock time from upload to draft ready
Requirement extraction recall	> 95% of all stated requirements	Manual audit of sample RFPs

Draft quality score	> 80% usable without major edits	Human reviewer satisfaction rating
Risk detection recall	> 98% of high-risk clauses flagged	Comparison with legal review
System availability	99.5% uptime during business hours	Monitoring and alerting

5. Technology Stack

The technology stack is selected for production readiness, community support, and alignment with emerging AI engineering best practices. All components are either open-source or have transparent pricing models.

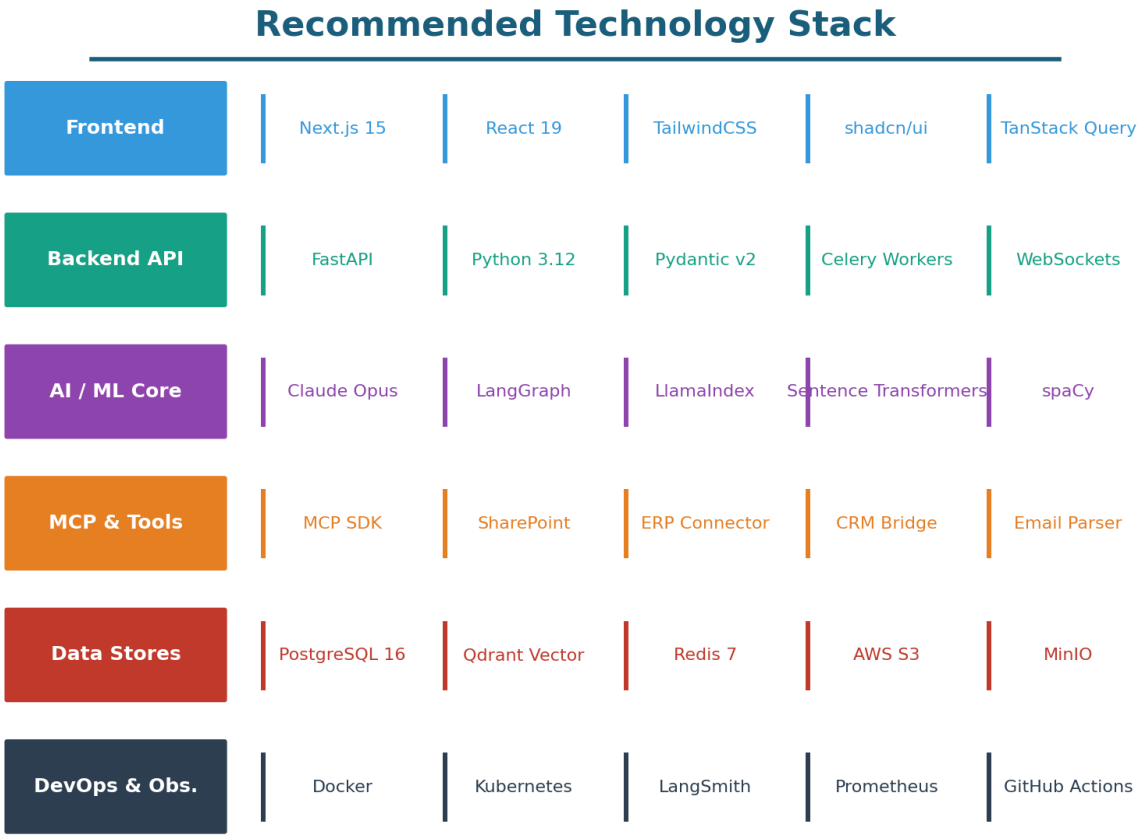

Figure 3: Full Technology Stack by Layer

5.1 Detailed Technology Specifications

Layer	Technology	Purpose	Why This Choice
Frontend	Next.js 15	Web application framework with RSC	Server components reduce bundle size; built-in API routes
Frontend	React 19	UI component library	Concurrent features, Suspense, Server Actions
Frontend	TailwindCSS + shadcn/ui	Styling and component system	Rapid UI development with accessible, customizable components
Backend	FastAPI (Python	REST API server	Async native, Pydantic

	3.12)		validation, automatic OpenAPI docs
Backend	Celery + Redis	Task queue for async processing	Reliable distributed task execution with retry and monitoring
AI Core	Claude Opus / Sonnet	Primary LLM for reasoning and generation	Best-in-class document understanding, structured output, tool use
AI Core	LangGraph	Agent orchestration framework	Graph-based state machine; native support for cycles, branching, HITL
AI Core	LlamaIndex	RAG framework	Advanced retrieval strategies, document connectors, hybrid search
AI Core	Sentence Transformers	Embedding generation	Local embeddings for privacy; configurable models
Tools	MCP SDK	Model Context Protocol connectors	Standardized tool interface; broad enterprise integration
Tools	Apache Tika	Document parsing and text extraction	Handles 1,000+ file formats including legacy Office, PDF, CAD
Tools	Tesseract OCR	Optical character recognition	Open-source, supports 100+ languages, works with scanned PDFs
Data	PostgreSQL 16	Relational database	JSON support, full-text search, row-level security
Data	Qdrant	Vector database	Native hybrid search, filtering, horizontal scaling
Data	Redis 7	Cache and message broker	Sub-millisecond latency, Streams for event sourcing
Data	S3 / MinIO	Object storage for documents	Industry standard; MinIO for on-prem deployments
DevOps	Docker + Kubernetes	Containerization and orchestration	Reproducible environments, horizontal auto-scaling
DevOps	LangSmith	LLM observability and tracing	End-to-end trace visualization, evaluation, dataset management
DevOps	GitHub Actions	CI/CD pipeline	Native GitHub integration, reusable workflows

5.2 Emerging Technology Considerations

The architecture is designed to incorporate these emerging capabilities as they mature:

- Multi-modal Models — Process engineering drawings, P&IDs, and site photos directly within the pipeline using vision-capable LLMs
- Structured Output / Constrained Decoding — Enforce JSON schema compliance on LLM outputs for zero-parsing-error extraction
- Model Context Protocol (MCP) 2.0 — Leverage evolving MCP specifications for richer tool discovery, streaming results, and bidirectional communication
- Edge Inference — Run lightweight classification and extraction models locally for latency-sensitive or air-gapped environments
- Retrieval-Augmented Fine-Tuning (RAFT) — Fine-tune smaller models on company-specific proposal patterns to reduce LLM API costs
- Agentic Coding Assistants — Use Claude Code or similar tools for automated test generation and code review within the development workflow

6. MCP Integration Architecture

The Model Context Protocol (MCP) provides a standardized interface between the AI orchestration layer and enterprise systems. Rather than building custom integrations for each system, MCP enables a single protocol for tool discovery, invocation, and result handling across all connected services.

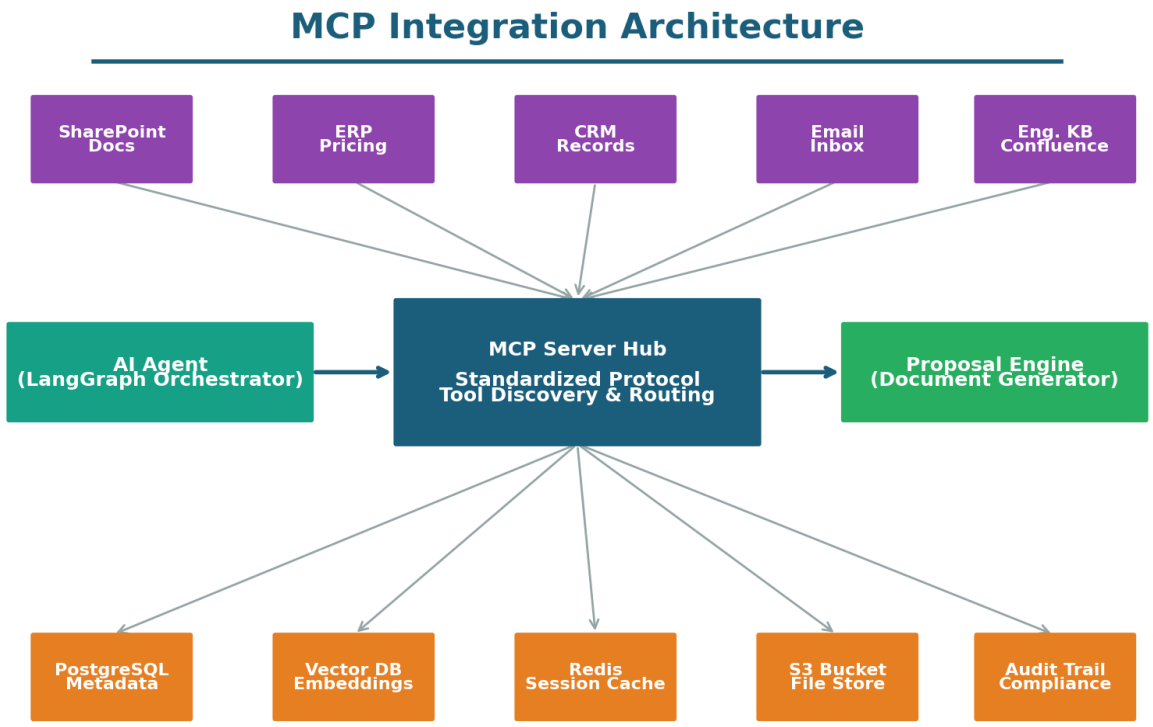

Figure 4: MCP Hub Connecting AI Agents to Enterprise Systems

6.1 MCP Server Specifications

MCP Server	Connected System	Exposed Tools	Data Flow
Document Server	SharePoint / Google Drive	search_documents, get_file, list_folders	Read-only: fetch tender docs, templates, past proposals
ERP Server	SAP / Oracle / Custom	get_rates, calculate_pricing, get_material_costs	Read-only: pricing data, resource rates, material costs
CRM Server	Salesforce / HubSpot	get_opportunity, get_contact, get_account_history	Read-only: client history, opportunity details, contact info

HR / CV Server	Internal CV Database	search_cvs, get_qualifications, match_requirements	Read-only: staff profiles, qualifications, availability
Email Server	Outlook / Gmail	search_inbox, get_attachments, parse_correspondence	Read-only: tender correspondence, clarifications, addenda
Knowledge Base	Confluence / Notion	search_articles, get_standard_responses, get_lessons	Read-only: engineering standards, lessons learned, SOPs

6.2 MCP Security Model

- All MCP connections use OAuth 2.0 with scoped permissions (read-only by default)
- Each MCP server runs in an isolated container with network policies restricting lateral access
- All tool invocations are logged with request/response payloads for audit compliance
- Rate limiting per server prevents runaway agent loops from overwhelming enterprise systems
- PII detection middleware scrubs sensitive data before it enters the LLM context

7. Multi-Agent Orchestration Design

The system uses a supervisor-worker pattern implemented in LangGraph. A central Supervisor Agent manages task planning and delegation to five specialized worker agents, all sharing a common state graph for coordination.

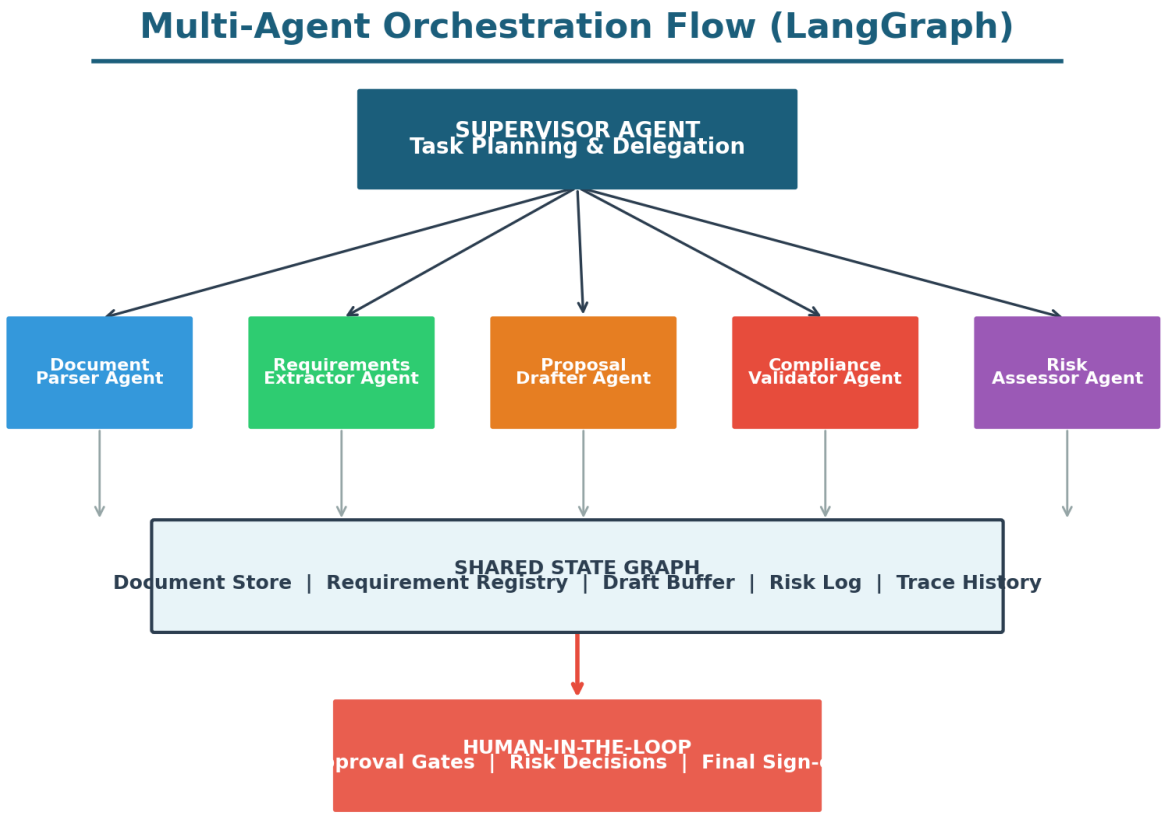

Figure 5: Supervisor-Worker Agent Architecture with Shared State

7.1 Agent Specifications

Agent	Responsibility	Tools Available	Output
Supervisor	Task planning, delegation, state management, human escalation	All sub-agents, state graph, notification system	Execution plan, delegation orders, escalation triggers
Document Parser	Ingest, OCR, text extraction, document classification	Tika, Tesseract, PDF parser, classification model	Structured document corpus with type labels
Requirements	NER, clause	NER model, regex patterns,	Requirement registry with

Extractor	detection, requirement tagging, deadline extraction	LLM extraction, vector search	priority and category tags
Proposal Drafter	Content generation, template population, section assembly	RAG retrieval, template engine, LLM generation, formatting tools	Draft proposal sections with citations
Compliance Validator	Completeness checking, cross-referencing, consistency audit	Requirement registry, draft buffer, rule engine, LLM audit	Validation report with coverage score and gap list
Risk Assessor	Commercial risk scoring, clause analysis, anomaly detection	Risk scoring model, commercial rules database, LLM analysis	Risk register with severity ratings and recommendations

7.2 State Graph Design

All agents operate on a shared LangGraph state object that serves as the single source of truth for the entire pipeline. The state graph uses typed channels for each data category:

State Channel	Type	Description
document_store	Dict[str, DocumentChunk]	Ingested and classified document fragments with metadata
requirement_registry	List[Requirement]	Extracted requirements with ID, text, category, priority, status
retrieval_results	Dict[str, List[RetrievalHit]]	RAG results mapped to each requirement ID
draft_buffer	Dict[str, DraftSection]	Generated proposal sections keyed by section ID
validation_report	ValidationReport	Coverage score, gap list, consistency checks
risk_register	List[RiskItem]	Identified risks with severity, clause reference, recommendation
trace_log	List[TraceEntry]	Immutable log of all agent actions and decisions
human_decisions	Dict[str, Decision]	Human approvals, overrides, and annotations

7.3 Human-in-the-Loop Gates

The system enforces mandatory human approval at three critical decision points:

1. Risk Approval Gate — All flagged commercial risks require human review before the proposal can be finalized. The Risk Assessor agent presents risks with severity ratings, clause references, and recommended positions.
2. Commercial Approval Gate — Pricing assumptions, exclusions, and commercial terms require sign-off from the Commercial Manager before inclusion in the final document.
3. Submission Approval Gate — The final proposal pack requires Bid Manager sign-off before export. This gate includes a completeness checklist and a diff view showing all AI-generated content.

8. Detailed Feature Specifications

8.1 Feature Priority Matrix

ID	Feature	Priority	Phase	Effort (Days)	Dependencies
F01	RFP Document Upload & Ingestion	P0 Critical	Phase 1	5	S3, Tika, OCR
F02	Automatic Document Classification	P0 Critical	Phase 1	4	LLM, Rules Engine
F03	Requirement Extraction Engine	P0 Critical	Phase 1	8	NER, LLM, Vector DB
F04	RAG-Powered Knowledge Retrieval	P0 Critical	Phase 2	10	Qdrant, LlamaIndex
F05	AI Proposal Drafting Engine	P0 Critical	Phase 2	12	LangGraph, LLM, Templates
F06	Compliance Validation Engine	P0 Critical	Phase 2	8	Requirement Registry
F07	Commercial Risk Flagging	P0 Critical	Phase 2	6	Risk Model, Rules DB
F08	Proposal Document Generator	P1 High	Phase 3	6	python-docx, ReportLab
F09	MCP Enterprise Connectors	P1 High	Phase 3	10	MCP SDK, OAuth
F10	Real-time Progress Dashboard	P1 High	Phase 3	5	WebSocket, Next.js
F11	Human Approval Workflow	P0 Critical	Phase 3	6	State Graph, Notifications
F12	Audit Trail & Trace Logging	P1 High	Phase 3	4	LangSmith, PostgreSQL
F13	Institutional Memory System	P2 Medium	Phase 4	8	Vector DB, Feedback Loop
F14	Analytics & Reporting Dashboard	P2 Medium	Phase 4	6	Recharts, PostgreSQL
F15	Multi-tenant Administration	P2 Medium	Phase 4	5	RBAC, Row-level Security

8.2 Core Feature Details

F01: RFP Document Upload & Ingestion

The system must accept documents in multiple formats and extract machine-readable text for downstream processing.

Attribute	Specification
Supported Formats	PDF, DOCX, XLSX, PPTX, scanned images (JPG/PNG/TIFF), ZIP archives
Max File Size	500 MB per file, 2 GB per submission
Upload Methods	Web UI drag-and-drop, REST API, email forwarding (monitored inbox), MCP connector
OCR Capability	Tesseract 5 with language auto-detection; falls back to cloud Vision API for low-confidence pages
Processing	Parallel page-level extraction with progress reporting via WebSocket
Output	Structured document corpus: page text, detected tables, images, metadata (page count, language, author)
Error Handling	Quarantine corrupt files, notify user, continue processing remaining documents

F05: AI Proposal Drafting Engine

The crown jewel of the system — generates a structured, compliant proposal draft using retrieved context and company templates.

Attribute	Specification
Input	Requirement registry, retrieval results, company templates, style guide, commercial rules
Generation Strategy	Section-by-section generation using requirement-specific prompts with retrieved context injection
Template Support	Configurable proposal structure: executive summary, technical approach, team CVs, pricing, assumptions, exclusions
Citation System	Every generated paragraph includes source citations: which requirement it addresses, which retrieved documents it used
Quality Controls	Max hallucination threshold (configurable), mandatory factual grounding in retrieved documents
Output Format	Structured draft in intermediate JSON format, rendered to DOCX/PDF via template engine
Fallback	If LLM confidence is below threshold, section is flagged for manual drafting with suggested outline

F07: Commercial Risk Flagging

Automatically identifies and scores commercial risks embedded in RFP terms and conditions.

Attribute	Specification
Risk Categories	Unlimited liability, IP assignment, payment terms <30 days, liquidated damages, performance bonds, retention clauses
Scoring Model	3-tier severity: Critical (must-negotiate), High (should-negotiate), Medium (acceptable with note)
Detection Method	LLM clause analysis + rule-based pattern matching against configurable commercial policy
Output	Risk register with: clause text, severity, category, recommended position, link to source document page
Human Gate	All Critical and High risks require Commercial Manager approval before proposal finalization
Learning	Human overrides and corrections are stored to improve future risk detection accuracy

9. API Specifications

The system exposes a RESTful API documented with OpenAPI 3.1. All endpoints require Bearer token authentication and return JSON responses. Below are the core API endpoints.

9.1 Core API Endpoints

Method	Endpoint	Description	Auth Scope
POST	/api/v1/projects	Create new proposal project	projects:write
POST	/api/v1/projects/{id}/upload	Upload RFP documents	documents:write
GET	/api/v1/projects/{id}/status	Get pipeline processing status	projects:read
GET	/api/v1/projects/{id}/requirements	List extracted requirements	requirements:read
PUT	/api/v1/requirements/{id}	Edit or approve a requirement	requirements:write
GET	/api/v1/projects/{id}/draft	Get current proposal draft	drafts:read
POST	/api/v1/projects/{id}/draft/regenerate	Regenerate specific sections	drafts:write
GET	/api/v1/projects/{id}/risks	Get flagged commercial risks	risks:read
PUT	/api/v1/risks/{id}/decision	Approve or override a risk flag	risks:write
POST	/api/v1/projects/{id}/export	Export final proposal pack	export:write
GET	/api/v1/projects/{id}/audit	Get full audit trail	audit:read
WS	/ws/v1/projects/{id}/stream	Real-time pipeline status stream	projects:read

9.2 Webhook Events

The system emits webhooks for integration with external notification systems:

Event	Trigger	Payload
pipeline.started	RFP processing begins	project_id, document_count, estimated_time
pipeline.step_complete	A pipeline step finishes	project_id, step_name, step_output_summary

risk.flagged	A commercial risk is detected	project_id, risk_id, severity, clause_text
approval.required	Human approval gate reached	project_id, gate_type, pending_items
proposal.ready	Final proposal pack generated	project_id, download_url, validation_score

10. User Experience & Journey

The user experience is designed for speed and confidence. The Bid Manager should be able to go from RFP upload to submission-ready proposal in under 30 minutes, with clear visibility into what the AI is doing at each step.

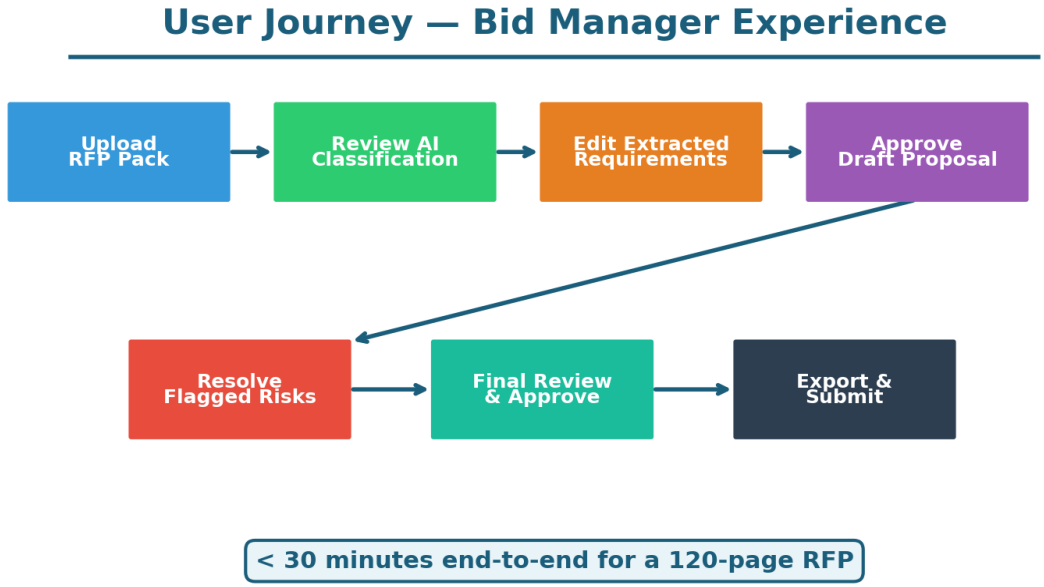

Figure 6: Bid Manager User Journey — 7-Step Workflow

10.1 Screen Specifications

Screen	Key Components	User Actions
Project Dashboard	Active project cards, status badges, timeline, alerts	Create project, view status, access any project
Upload & Ingest	Drag-and-drop zone, file list, processing progress bar, OCR confidence indicators	Upload files, monitor ingestion, retry failed files
Requirement Review	Requirement table with filters, AI confidence scores, source document links	Approve, edit, reject, merge, or add requirements
Proposal Editor	Section-by-section editor, AI draft with citations, side-by-side source view	Edit text, regenerate sections, approve sections
Risk Review	Risk cards with severity badges, clause excerpts, AI recommendations	Accept risk position, override, add notes

Export & Submit	Completeness checklist, format options, final diff view	Select format, review final, export and submit
Analytics	Win rate charts, cycle time trends, requirement coverage heatmaps	Filter by date, team, client, bid type

11. Security & Compliance Architecture

Security is a first-class architectural concern, not an afterthought. The system follows a zero-trust model with defense-in-depth across all layers. Compliance with NIST AI Risk Management Framework ensures responsible AI governance.

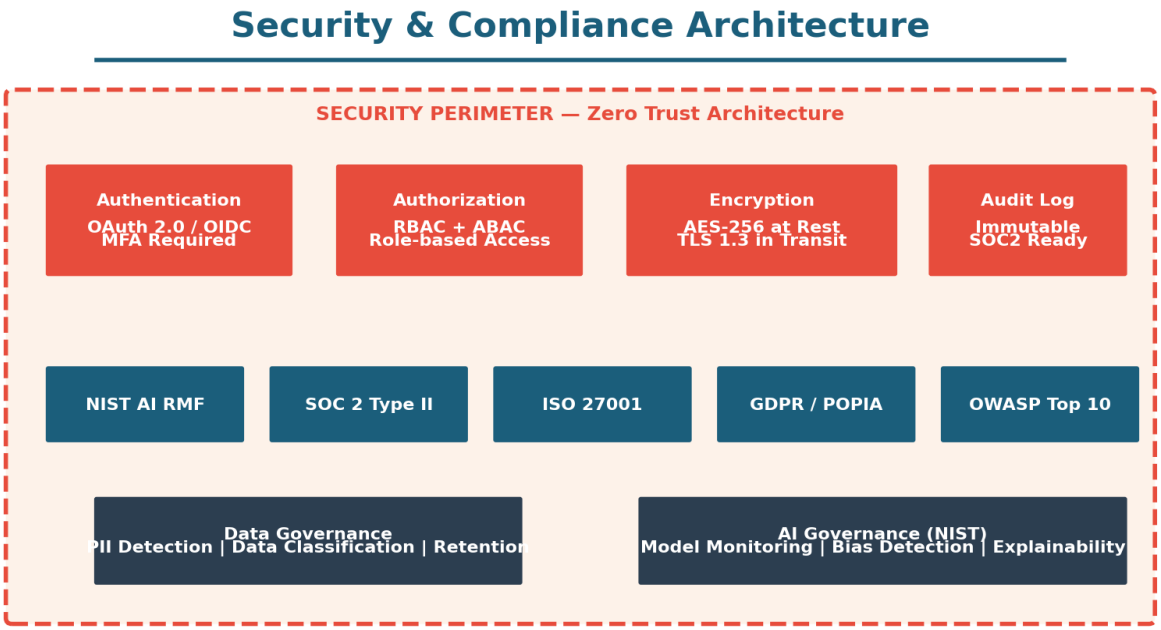

Figure 7: Security & Compliance Architecture

11.1 Security Requirements

Category	Requirement	Implementation
Authentication	Multi-factor authentication required for all users	Clerk/Auth0 with TOTP or WebAuthn
Authorization	Role-based + attribute-based access control	PostgreSQL row-level security + API middleware
Encryption at Rest	AES-256 encryption for all stored data	S3 server-side encryption, PostgreSQL TDE
Encryption in Transit	TLS 1.3 for all network communications	Nginx termination, internal mTLS between services
Audit Logging	Immutable log of all user and system actions	Append-only PostgreSQL table + S3 archive
PII Protection	Detect and redact PII before LLM processing	Microsoft Presidio + custom rules engine

LLM Data Privacy	No training on customer data; data residency controls	Anthropic API with zero-retention; self-hosted option
Vulnerability Management	Automated scanning and patching	Dependabot, Trivy container scanning, OWASP ZAP

11.2 Compliance Framework Alignment

Framework	Applicable Controls	Status
NIST AI RMF	Govern, Map, Measure, Manage functions for AI risk	Designed-in from architecture phase
SOC 2 Type II	Security, Availability, Confidentiality trust principles	Targeted for Phase 4 audit
ISO 27001	Information security management system controls	Control mapping complete
GDPR / POPIA	Data subject rights, lawful processing, data minimization	Privacy impact assessment required
OWASP Top 10	Web application security best practices	Automated testing in CI/CD pipeline

12. Non-Functional Requirements

Category	Requirement	Target
Performance	End-to-end pipeline for 120-page RFP	< 30 minutes
Performance	API response time (non-LLM endpoints)	< 200ms p95
Performance	Document upload and ingestion	< 60 seconds for 100 MB
Scalability	Concurrent proposal processing	10 simultaneous projects
Scalability	Knowledge base size	100,000+ document chunks
Scalability	User capacity	500 concurrent users
Availability	System uptime during business hours	99.5%
Availability	Recovery Time Objective (RTO)	< 1 hour
Availability	Recovery Point Objective (RPO)	< 15 minutes
Reliability	Pipeline completion rate	> 99% without manual intervention
Reliability	Data durability	99.999999999% (S3 standard)
Maintainability	Code test coverage	> 80% line coverage
Maintainability	Deployment frequency	Daily to staging, weekly to production
Observability	LLM trace capture rate	100% of all LLM calls traced
Observability	Alert response time	< 5 minutes for critical alerts

13. Development Roadmap

The project follows a 16-week delivery plan across four phases, each producing a deployable increment. The roadmap prioritizes core AI pipeline functionality first, then integrations and production hardening.

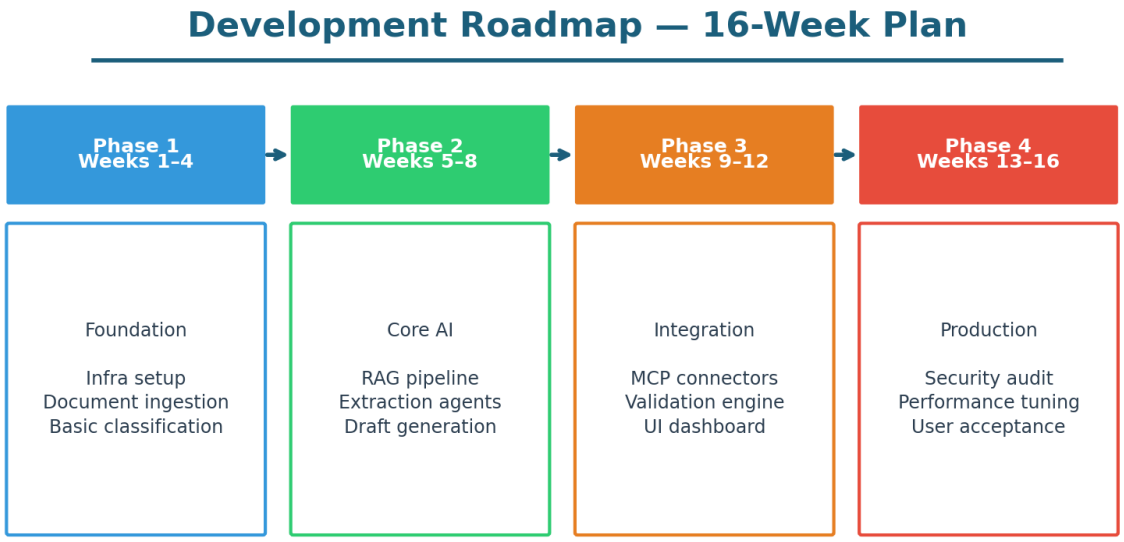

Figure 8: 16-Week Development Roadmap

13.1 Phase Details

Phase 1: Foundation (Weeks 1–4)

Establish infrastructure, document ingestion, and basic classification capabilities.

Deliverable	Description	Success Criteria
Infrastructure Setup	Docker Compose dev environment, PostgreSQL, Qdrant, Redis, S3	All services running locally with health checks
Document Ingestion	Upload API, PDF/DOCX text extraction, OCR pipeline	Process 120-page PDF in < 60 seconds
Document Classification	LLM-based document type classification with confidence scores	> 90% accuracy on test set of 50 documents
Basic UI Shell	Next.js app with auth, project creation, file upload	End-to-end upload flow working

Phase 2: Core AI (Weeks 5–8)

Build the core AI pipeline: extraction, retrieval, drafting, and validation agents.

Deliverable	Description	Success Criteria
Requirement Extraction	NER + LLM extraction pipeline with requirement registry	> 95% recall on test RFPs
RAG Pipeline	Hybrid retrieval with Qdrant + BM25, knowledge base indexing	Relevant results in top-5 for 90% of queries
Proposal Drafter	LangGraph workflow for section-by-section generation	Draft quality score > 70% on human review
Validation Engine	Completeness checking against requirement registry	100% requirement coverage verification
Risk Flagger	Commercial risk detection and scoring	Flag all Critical risks in test set

Phase 3: Integration (Weeks 9–12)

Connect enterprise systems via MCP, build the full UI, and implement human approval workflows.

Deliverable	Description	Success Criteria
MCP Connectors	SharePoint, CRM, ERP, and CV database integrations	Successful data retrieval from all connected systems
Approval Workflow	Human-in-the-loop gates for risks, commercial terms, and final submission	All three gates enforced in pipeline
Full Dashboard UI	All screens implemented: upload, review, edit, risks, export	Complete user journey testable end-to-end
Document Generator	DOCX/PDF output with company branding and correct structure	Output matches company template standards

Phase 4: Production (Weeks 13–16)

Security hardening, performance optimization, user acceptance testing, and launch.

Deliverable	Description	Success Criteria
Security Audit	Penetration testing, OWASP scan, access control review	No critical or high vulnerabilities
Performance Tuning	LLM caching, parallel	< 30 min for 120-page RFP

	processing optimization, load testing	under load
Institutional Memory	Feedback loops, pattern learning, knowledge base improvement	Demonstrated improvement on repeat bid types
User Acceptance Testing	Structured UAT with real proposal teams using real RFPs	> 85% user satisfaction score
Production Deployment	Kubernetes deployment with monitoring, alerting, and runbooks	Successful production launch

14. Success Metrics & KPIs

14.1 Business KPIs

KPI	Baseline (Current)	Target (6 Months)	Stretch (12 Months)
Avg. proposal cycle time	5–10 business days	< 2 business days	< 1 business day
Proposals submitted per quarter	8–12	20–30	40+
Requirement coverage per bid	~85% (manual audit)	> 98%	100%
Commercial risk detection rate	~60% (human review)	> 95%	> 99%
First-draft usability rate	N/A (manual drafting)	> 70%	> 85%
Bid win rate improvement	Baseline	+10%	+20%

14.2 Technical KPIs

KPI	Target	Alert Threshold	Measurement
Pipeline completion rate	> 99%	< 95%	Automated monitoring
LLM latency (p95)	< 10 seconds per call	> 15 seconds	LangSmith traces
Extraction accuracy (F1)	> 0.92	< 0.85	Weekly evaluation set
System uptime	> 99.5%	< 99%	Health check monitoring
API error rate	< 0.5%	> 2%	Prometheus metrics
Knowledge base freshness	< 24 hours stale	> 7 days	Indexing pipeline logs

15. Risks & Mitigations

#	Risk	Severity	Mitigation	Owner
R1	LLM hallucination in proposal drafts	Critical	Mandatory RAG grounding, citation requirements, validation agent, human review gates	AI Lead
R2	Enterprise system integration delays	High	MCP abstraction layer enables parallel development; mock servers for testing	Tech Lead
R3	User adoption resistance	High	Early UAT involvement, side-by-side comparison with manual process, training program	Product Owner
R4	LLM API cost overruns	Medium	Caching layer, smaller models for classification, budget alerts, cost-per-proposal tracking	Engineering Manager
R5	Data privacy concerns	High	PII scrubbing, data residency controls, zero-retention LLM APIs, privacy impact assessment	Security Lead
R6	Scope creep beyond proposals	Medium	Strict PRD scope, change request process, phased roadmap with clear boundaries	Product Owner
R7	Key person dependency	Medium	Documentation-first culture, pair programming, knowledge sharing sessions	Engineering Manager

16. Getting Started Guide for Developers

This section provides a step-by-step onboarding path for new developers joining the project. Follow this sequence to go from zero to contributing code.

16.1 Prerequisites

Tool	Version	Installation
Python	3.12+	pyenv install 3.12.0 && pyenv global 3.12.0
Node.js	20 LTS+	nvm install 20 && nvm use 20
Docker Desktop	Latest	docker.com/products/docker-desktop
Git	2.40+	Pre-installed on most systems
VS Code	Latest	code.visualstudio.com (recommended IDE)
Claude API Key	Active account	console.anthropic.com

16.2 Local Development Setup

1. Clone the repository: git clone <repo-url> && cd bid-proposal-ai
2. Copy environment template: cp .env.example .env and fill in API keys
3. Start infrastructure services: docker compose up -d (PostgreSQL, Qdrant, Redis, MinIO)
4. Install Python dependencies: pip install -r requirements.txt
5. Install Node dependencies: cd frontend && npm install
6. Run database migrations: alembic upgrade head
7. Seed test data: python scripts/seed_test_data.py
8. Start the backend: uvicorn app.main:app --reload
9. Start the frontend: cd frontend && npm run dev
10. Open http://localhost:3000 and upload a sample RFP to test the pipeline

16.3 Project Structure

Directory	Purpose
/ (root)	Docker Compose, README, .env, configuration files
/app	FastAPI backend application

/app/agents	LangGraph agent definitions (supervisor, parser, extractor, drafter, validator, risk)
/app/api	REST API route definitions organized by resource
/app/core	Configuration, security, database connection, shared utilities
/app/models	SQLAlchemy ORM models and Pydantic schemas
/app/services	Business logic layer (pipeline orchestration, RAG, document processing)
/app/mcp	MCP server connectors for enterprise integrations
/app/templates	Proposal document templates (DOCX, sections, styles)
/frontend	Next.js 15 web application
/frontend/app	App Router pages and layouts
/frontend/components	Reusable React components
/tests	Pytest unit and integration tests
/scripts	Utility scripts for seeding, migration, evaluation
/docs	Architecture decision records, API documentation

16.4 Key Development Patterns

Follow these patterns consistently throughout the codebase:

- **Agent Pattern:** Each agent is a Python class with a clear invoke() method that accepts shared state and returns updated state. Agents are registered with the LangGraph supervisor.
- **RAG Pattern:** All retrieval uses the hybrid search service (vector + keyword). Never call the vector database directly — always go through the retrieval service layer.
- **MCP Pattern:** Enterprise integrations use the MCP client abstraction. Mock MCP servers are provided for local development.
- **Error Pattern:** All agents use structured error handling with retry logic. Failed steps are logged to the trace and surfaced in the UI.
- **Testing Pattern:** Every agent has unit tests with mock LLM responses. Integration tests use a dedicated test database. Evaluation sets measure extraction and draft quality.

17. Glossary

Term	Definition
RFP	Request for Proposal — a formal document issued by a client inviting firms to submit proposals for a project
RFQ	Request for Quotation — a request focused on pricing for specified goods or services
LLM	Large Language Model — an AI model trained on large text datasets capable of understanding and generating text
RAG	Retrieval-Augmented Generation — a technique that enhances LLM outputs by retrieving relevant documents before generation
MCP	Model Context Protocol — an open standard for connecting AI agents to external tools and data sources
LangGraph	A framework for building stateful multi-agent AI workflows using graph-based state machines
Vector Database	A database optimized for storing and searching high-dimensional embeddings for semantic similarity
NER	Named Entity Recognition — an NLP technique for identifying and classifying named entities in text
HITL	Human-in-the-Loop — a design pattern where human judgment is required at specific workflow decision points
OCR	Optical Character Recognition — technology for extracting text from images and scanned documents
RBAC	Role-Based Access Control — an approach to restricting system access based on user roles
ABAC	Attribute-Based Access Control — a fine-grained access control model using user, resource, and environment attributes
NIST AI RMF	The National Institute of Standards and Technology AI Risk Management Framework for trustworthy AI systems

End of Product Requirements Document