AI Vendor Contract Management System — System Overview

> **Scope note**: This overview is generated strictly from the project structure you shared (VS Code screenshots). No assumptions beyond visible files/folders are made.

---

1. Project Size (What I Know From Structure)

Based on the visible tree:

Backend (Python / FastAPI-style)

- **~35–40 Python source files**
- **Key folders**:
- `app/models`
- `app/routes`
- `app/schemas`
- `services/advanced_nlp`
- `services/ml_models`
- `services/time_series`
- **Supporting scripts**: training, seeding, verification, testing
- **Heavy assets**:
- PDFs (contracts)
- Embeddings
- Trained ML models

Frontend (React + Vite + TypeScript)

- **~15–18 TypeScript/TSX files**
- Components-heavy dashboard UI
- API abstraction layer

📌**Total visible files (excluding node_modules, envs, PDFs): ~55–60 files**

This already places the project in **large academic / early-enterprise prototype scale**.

---

2. High-Level Architecture

```
PDF Contracts
   ↓
PDF Parsing Service
```

```
    ↓
NLP Pipeline (NER, Classification, Embeddings)
    ↓
Clause Library + Similarity Engine
    ↓
Risk Scoring + Explainability
    ↓
Time-Series Forecasting (SLA / Risk)
    ↓
Alerts + Vendor Scoring
    ↓
REST API (FastAPI)
    ↓
React Dashboard (Analytics, Comparison, Forecasts)
```

---

3. Backend Architecture

3.1 Core Entry Points

- `main.py` — Application bootstrap
- `config.py` — Environment & configuration
- `database.py` — DB connection & session management

3.2 API Layer (`app/routes`)

Responsible for **HTTP interaction only**.

File	Responsibility
`contract.py`	Contract CRUD, upload, metadata
`enhanced_contract.py`	AI-enhanced contract analysis
`vendor.py`	Vendor management
`forecasting_routes.py`	SLA / risk forecasting APIs
`similarity_routes.py`	Contract similarity search
`ai_explanations.py`	Explainable AI outputs
`ml_routes.py`	ML inference endpoints

📌**Rule**: No business logic here — only request/response orchestration.

3.3 Data Models (`app/models`)

Represents **database entities & domain objects**.

File	Purpose
`contract.py`	Contract entity
`vendor.py`	Vendor entity
`sla.py`	SLA representation
`embedding.py`	Vector storage / reference

3.4 Schemas (`app/schemas`)

- Input/output validation
- API contracts

File	Purpose
`contract_schema.py`	Contract DTOs
`vendor_schema.py`	Vendor DTOs

---

4. Service Layer (System Intelligence)

This is the **heart of the project**.

4.1 Advanced NLP Services (`services/advanced_nlp`)

File	Function
`legalbert_classifier.py`	Clause / risk classification
`ner_service.py`	Entity extraction (dates, penalties, parties)
`embedding_service.py`	Text → vector embeddings
`embedding_similarity.py`	Vector similarity
`similarity_engine.py`	Clause comparison logic
`explainable_risk.py`	NLP-level explainability

Algorithms used: - Transformer-based NLP (LegalBERT) - Cosine similarity - Rule-enhanced ML outputs

4.2 Machine Learning Models (`services/ml_models`)

File	Function
`feature_engineering.py`	Feature construction
`risk_model.py`	Risk prediction model
`predictor.py`	Unified inference
`trainer.py`	Training pipeline
`train_model.py`	Model training execution
`model_registry.py`	Model versioning

Algorithms: - Logistic Regression / Tree-based models - Explainability (SHAP-like logic)

4.3 Time Series Services (`services/time_series`)

File	Function
`forecasting.py`	SLA breach forecasting

Algorithms: - Classical time-series / ML forecasting

4.4 Business Services (Cross-cutting)

File	Responsibility
`risk_service.py`	Risk aggregation & scoring
`vendor_scoring.py`	Vendor-level risk scoring
`alert_service.py`	Risk & SLA alerts
`summary_service.py`	Contract summarization
`pdf_service.py`	PDF parsing
`similarity_service.py`	High-level similarity orchestration

---

5. Data Flow (End-to-End)

1. User uploads contract PDF
2. PDF parsed → raw text

3. NLP pipeline extracts clauses + entities
4. Embeddings generated
5. Similarity engine compares against clause library
6. Risk model computes risk score
7. Explainability layer generates reasons
8. Time-series forecasts SLA violations
9. Vendor score updated
10. Alerts generated
11. Frontend dashboard updated via API

---

6. Frontend Architecture

6.1 Core Stack

- React + TypeScript
- Vite
- REST-based API integration

6.2 Key UI Components

Component	Purpose
`DashboardCards.tsx`	KPI overview
`ContractsTable.tsx`	Contract listing
`ContractComparison.tsx`	Side-by-side analysis
`EnhancedContractCard.tsx`	AI insights
`SimilaritySearch.tsx`	Clause similarity
`MLPrediction.tsx`	Risk predictions
`ForecastingDashboard.tsx`	SLA forecasting
`AlertsList.tsx`	Alerts & notifications
`VendorsTable.tsx`	Vendor analytics

---

7. Non-Functional Goals

- Explainable AI (not black-box)
- Modular service isolation
- Scalable ML lifecycle
- Industry-realistic workflows

- Academic clarity + enterprise relevance

---

8. Known Constraints

- Large files (PDFs, embeddings, models) excluded from AI indexing
- AI tools must be used **per-module**, not globally
- System understanding maintained via this document

---

9. How This Project Should Be Evaluated

This is **NOT** a single-problem AI project.

It demonstrates: - Multi-model AI orchestration - Real business workflows - Explainable decision systems - Full-stack engineering

📌Suitable classification:

> **AI-Powered Enterprise Contract Intelligence Platform**

---

10. Ownership Statement

Architecture, integration, and system-level decisions are **human-designed**. AI tools are used as **assistive components**, not authors.

End of System Overview