



AI Legal Document Explainer – Product Requirements Document

1. Purpose & Goals

- **Problem:** Legal documents (leases, contracts, terms) are often written in complex, technical language that non-lawyers find confusing. This creates a risk of misunderstanding obligations or missing critical terms ¹ ².
- **Solution:** Use AI to automatically summarize contracts in plain English and highlight important clauses and risks. By translating legal jargon into clear summaries, users can quickly grasp the document's content ² ¹.
- **Outcome:** Empower non-lawyer users to understand agreements on their own. Quickly uncover key obligations, penalties, auto-renewals, and unusual terms. Reduce the need for external legal advice for straightforward review tasks.




2. Target Audience

- **Primary Users:** Renters and tenants reviewing lease agreements; small business owners and freelancers reviewing service contracts or partnership agreements. These are individuals with no legal training who need clarity on their rights and duties ³.
- **Secondary Users:** HR or operations staff at small companies, landlords, and gig workers who regularly encounter standard contracts (NDAs, EULAs) but lack easy access to legal counsel.
- **Use Case Rationale:** Most people seeking online legal help don't have a lawyer on hand ⁴, so an AI explainer that provides clear, actionable guidance (avoiding jargon) fills a critical gap ⁴ ¹.




3. User Stories & Use Cases

- **Lease Review:** *As a tenant*, I upload my rental lease. The tool highlights the lease term, rent escalation, and termination clauses. It then summarizes my obligations (e.g. rent due, notice period) in plain language ² ⁵.
- **Freelancer Contract Check:** *As a freelancer*, I upload a client service contract. The system flags any auto-renewals or penalty clauses (●■) and provides a simple summary of payment terms. I can ask questions like "What happens if I end the contract early?" and receive a contextual answer ⁶ ⁵.
- **Employee Agreement:** *As an HR manager*, I drop in an employment agreement to verify there are no illegal clauses. The tool highlights unusual non-compete or indemnity clauses and compares them to typical standards. It assigns risk levels and explains any red flags.
- **Legal Q&A:** *As a small business owner*, I upload a partnership agreement and ask, "Is there a penalty for late delivery of services?" The chatbot finds the relevant clause, highlights it, and answers based on the document's context ⁶ ⁷.



4. Core Features

1. **Summarization (Plain English):** Generates a concise summary of the entire document, rewriting complex terms in everyday language. This accelerates understanding: studies show AI can “translate...claims into plain English,” improving client comprehension ² ⁸ .
2. **Key Clause Highlighting:** Automatically detect and highlight important clauses (e.g. termination, payment, confidentiality). For instance, the model identifies “key clauses and explanations” of legal implications ⁵ . Highlighted clauses are annotated (e.g. “This clause imposes a heavy penalty”).
3. **Color-Coded Risk Indicators:** Assign each clause a risk rating ( High,  Medium,  Low) based on content analysis. AI compares clauses against organizational standards and flags “high-risk terms” ⁹ . This visual risk cue helps users prioritize review.
4. **Risk Explanation Flags:** For each flagged clause, provide an explanation of the risk. The AI notes why it’s risky or unusual (e.g. “Penalty is unusually high” or “Missing standard indemnity term”). This aligns with contract analysis best practices of surfacing risks for human review ⁵ ⁹ .
5. **Interactive Q&A (Contextual Chat):** Users can ask natural-language questions about the document (e.g. “What is the notice period?”). A retrieval-based system (using LangChain or similar) fetches relevant text from the document and answers with context. This mirrors legal chatbots that give “instant, contextual answers” on contract clauses ⁶ ⁸ .
6. **Multi-Language Support:** Accept and analyze documents in English, Tamil, and Sinhala. Modern LLMs (like GPT-4 or LLaMA) can process multiple languages ¹⁰ , enabling summarization and Q&A in all three. (If needed, embed a translation step for Tamil/Sinhala terms.)
7. **Confidence Scores & Advice:** Each AI output shows a confidence level. If confidence is low (e.g. unusual phrasing or incomplete data), display a warning and recommend consulting a lawyer. This user-centric caution aligns with findings that AI should guide users to help without “paralyzing” them ⁴ .
8. **Risk Visualization Charts:** Present clause-level data graphically (e.g. a heatmap of risk categories per section, or a radar chart of obligation types). Research shows “clear visualizations simplify exploration” of documents ¹¹ , enabling users to quickly grasp overall risk distribution.
9. **Glossary Pop-Ups:** Provide definitions for legal terms. Clicking on a highlighted term (like “force majeure”) shows an explanation in simple language. The AI can use its knowledge to extract “legal definitions” and interpret obligations ¹² ¹³ . This educates users and reduces confusion over jargon.
10. **What-If Simulation:** Allow users to edit key clause parameters (e.g. adjust late fees, toggle auto-renewal) and immediately re-run analysis. This shows how the summary, risk scores, and compliance change under hypothetical scenarios, aiding negotiation and understanding.
11. **Community Clause Comparison:** Compare the document’s clauses against a crowdsourced or built-in clause library of standard vs. risky clauses. If a clause deviates from common practice, flag it. AI contract analysis tools often “spot unusual patterns” and highlight deviations from standards ⁹ ⁵ . This feature crowdsources insight.
12. **Offline Mode:** Provide a lightweight analysis mode when offline. For core tasks (summaries, highlights, basic Q&A), run a compact LLaMA model locally on the device. Recent tech enables running LLaMA 3B/1B on mobile, offering “instant processing” and “enhanced privacy” without cloud reliance ¹⁰ .
13. **Transparency Slider:** Offer a toggle between “Very Simple” and “Legal Detailed” modes for explanations. This empowers users to choose depth of explanation, addressing diverse needs (novices vs. power users). This aligns with guidelines that users prefer clarity and explanation context ¹³ ¹⁴ .

5. UI/UX Requirements

- **Mobile-First Responsive Design:** The interface must adapt seamlessly from mobile screens to desktop. Touch-friendly layouts (large buttons, swipe support) ensure usability on smartphones and tablets.
- **Clear Navigation Flow:** A linear workflow with a progress indicator: **Upload Document → View Summary → Clause List → Ask Question**. Each step is clearly labeled.
- **PDF Viewer with Highlights:** Integrate a PDF.js reader to display the uploaded document. Overlay highlights on clauses in their actual text positions, colored by risk rating. Hovering/tapping a highlight shows the clause summary and risk note.
- **Chat Q&A Panel:** A persistent sidebar or modal for the Q&A chatbot. It should scroll through the conversation and allow uploading follow-up docs.
- **Interactive Glossary:** Legal terms in text appear underline or colored; tapping them opens a tooltip with a definition. Ensure glossary terms are easily noticed (e.g. question-mark icon or popover).
- **Visualization Dashboard:** Embed Recharts or similar charts on a summary screen. Include a legend explaining   . Charts should be interactive (hover to see exact values).
- **Simple Language UI:** All buttons, labels, and summaries use plain language (no unexplained legal jargon) in line with user literacy best practices ¹³.
- **Accessibility:** Use high-contrast themes and consider colorblind-friendly palettes (e.g. patterns in addition to color). Font sizes should be adjustable. Include alternative text for all non-text elements.

6. Functional Requirements

- **PDF Text Extraction:** On file upload, extract text via a library (pdf.js, PyMuPDF) or OCR if needed. Preserve structure (section headers, clause numbering).
- **Clause Mapping:** Implement logic (or use NLP) to segment text into clauses or sections. Tag each clause with metadata (page number, heading).
- **Text Chunking:** Break the document into smaller chunks (e.g. 1–2 page increments) for LLM input limits. Ensure each chunk retains clause context.
- **Summarization Engine:** Invoke the LLM on each chunk (or whole doc) to generate a plain-language summary. Use chain-of-thought prompts or prompt templates tuned for legal content. Merge chunk summaries into an overall summary.
- **Clause Classification & Risk Logic:** Use a trained model (e.g. fine-tuned legal-BERT or LLaMA) to classify clause risk. Alternatively, implement keyword/rule heuristics (e.g. “breach”, “penalty”). Risk model outputs High/Med/Low per clause.
- **Named Entity Recognition:** Extract defined terms and obligations (persons, dates, amounts) from text. Link each defined term to its definition for the glossary.
- **Multilingual Support:** Detect document language. For Tamil/Sinhala, either directly pass text to a multilingual LLM or translate to English, process, then translate outputs back.
- **Retrieval-Based QA:** Build an embeddings index of the document text. For each user query, retrieve relevant sentences or clauses and feed them with the question to the LLM for an answer. (LangChain RAG style) ⁶ ⁸.
- **Confidence Estimation:** Use model log-probabilities or an auxiliary classifier to estimate answer confidence. Display this score and trigger a “recommend lawyer” advisory if below threshold.
- **Visualization Generation:** Aggregate clause risk data into structured form and feed into Recharts. E.g., count of   per section for heatmap, or clause categories for radar chart.

- **Offline Mode Implementation:** Detect network status; if offline, switch to a local model (run via ONNX or a mobile-optimized runtime). Fall back to simpler outputs if model size limits detail.

7. Non-Functional Requirements

- **Performance:** The system should process an average 5-page document end-to-end in under 5 seconds. Use efficient I/O and parallelism (e.g. parallel chunk processing) to meet this.
- **Security & Privacy:** All documents and data in transit must be encrypted. Documents are not stored permanently; data is purged after the session. The system must comply with standards like GDPR and SOC 2 for handling sensitive legal data ¹⁵.
- **Accuracy:** Minimize AI hallucinations by grounding answers in the text. Use citation/attribution features if possible. Human review is recommended for final trust – aligning with best practices of human-in-loop in legal AI ¹⁵ ⁴.
- **Reliability:** The service should handle concurrent users (scale with container instances). Provide an offline fallback as noted. System should recover gracefully from failures.
- **Usability:** Prioritize clarity and readability. Goal: summaries should score at or below grade-8 reading level. Test with users to ensure explanations are genuinely “easy to understand” ¹³.
- **Extensibility:** Design APIs so the tool can later integrate additional languages or legal domains. Use modular components (e.g. separate summarizer, classifier) to allow upgrades.

8. Technical Stack Recommendations

- **Frontend:** React.js (component-driven UI) with Tailwind CSS for rapid styling. Use [PDF.js](#) to render PDF and overlay highlights. Recharts or Chart.js for risk visuals.
- **Backend:** FastAPI (Python) or Node.js (Express) for API endpoints. Use LangChain to orchestrate LLM calls and retrieval.
- **AI Models:** Leverage GPT-4o (OpenAI) or similar for high-quality summarization and Q&A. For offline mode, use Meta’s LLaMA 3 (1B/3B) via Ollama or GPT4All. Fine-tune or prompt-engineer for contract-specific accuracy.
- **Storage:** SQLite or lightweight MongoDB to cache embeddings, user sessions, and clause libraries. No permanent storage of document text.
- **Hosting:** Frontend on Vercel or Netlify. Backend on Render or Heroku (to allow hosting Python/Node easily). If needed, use a GPU-enabled cloud VM (AWS/GCP) for LLM if not using API.
- **Libraries:** Use PyMuPDF or pdfminer for reliable PDF parsing. Tesseract OCR for scanned docs. Pinecone or ChromaDB for vector store. Tailwind and React-PDF for UI. Tailwind ensures responsive design without heavy CSS.

9. Milestones & Timeline

1. **Day 1:** PDF ingestion + summarization. (Set up upload interface, text extraction, call LLM for initial summary).
2. **Day 2:** Clause detection + risk classification. (Implement clause parsing, run classifier to tag and color-code clauses).
3. **Day 3:** UI with highlights + Q&A. (Integrate PDF viewer showing highlighted clauses; implement chat interface for queries).
4. **Day 4:** Charts, glossary, confidence scoring. (Add Recharts for heatmap/radar; implement glossary pop-ups; display answer confidence bars).

5. **Day 5:** Simulations, community comparison, offline mode, polish. (Build “what-if” simulation UI; connect to clause library for comparison; integrate a simple offline model for fallback; finalize styling and fix bugs).

10. Success Metrics

- **Clause Detection Accuracy:** Percentage of correctly identified key clauses (target $\geq 90\%$).
- **Risk Scoring Alignment:** Agreement rate between AI-assigned risk levels and legal expert review.
- **Summary Clarity:** Readability scores (e.g. Flesch Reading Ease) and human ratings of summary usefulness. Aim for positive feedback that summaries are “easy to understand” ¹³.
- **QA Correctness:** Accuracy of the contextual Q&A (e.g. correctness of answers on benchmark questions extracted from documents).
- **User Satisfaction:** Collect user feedback (surveys or usability tests) measuring perceived clarity and trust. Aim for a high Net Promoter Score.
- **Performance:** Average processing time for a 5-page document remains under 5 seconds.

11. Differentiation Strategy

- **Multilingual Edge:** Offering Sinhala and Tamil support (in addition to English) addresses underserved user groups, a unique feature in legal tech.
- **Offline Capability:** The tool’s ability to function without internet (via on-device AI) is a key differentiator, especially in low-connectivity environments.
- **Interactive “What-If” Analysis:** This simulation of contract edits (rare in competitors) actively engages users in understanding consequences of changes, a powerful negotiation aid.
- **Community Clause Library:** By comparing clauses to a shared repository, we offer social proof on clause norms. This crowdsourced insight is novel for a hackathon project.
- **Transparency Controls:** The “Simple vs. Detailed” slider embodies the Stanford finding that AI should be transparent about its reasoning ¹³ ¹⁴. It lets users control jargon vs detail.
- **Visual Risk System:** The intuitive color-coded risk highlights (/●/●) and charts make risk immediately visible. As Spellbook notes, “clear visualizations simplify exploration” ¹¹, giving us a usability advantage.

Each of these innovative features adds concrete value: **plain-language summaries** save time ², **risk flags** direct user attention where needed, and **interactive insights** (Q&A, charts) turn passive review into active understanding. Together, they position our hackathon project as a practical, user-friendly leap in contract review tools.

Sources: AI for legal review (CaseStatus, Spellbook, Sirion) and legal-AI research informed these requirements ² ¹⁶ ⁹ ¹, ensuring our design follows best practices in AI-driven contract analysis.

¹ ³ ⁴ ⁷ ¹³ ¹⁴ Measuring What Matters: A Quality Rubric for Legal AI Answers – Justice Innovation
<https://justiceinnovation.law.stanford.edu/measuring-what-matters-a-quality-rubric-for-legal-ai-answers/>

² ¹⁵ AI for Legal Documents | Evaluating Popular AI Platforms | Case Status
<https://www.casestatus.com/blog/ai-for-legal-documents>

5 11 16 AI Legal Document Review: How AI Enhances Contract Analysis - Spellbook

<https://www.spellbook.legal/learn/ai-legal-document-review>

6 Unpacking legal AI: tools, trends, and use cases

<https://juro.com/learn/legal-ai>

8 9 12 AI for Legal Documents Analysis and Review: 2025 Guide

<https://www.sirion.ai/library/contract-ai/ai-legal-documents/>

10 Running Llama 3.2 on Android: A Step-by-Step Guide Using Ollama - DEV Community

<https://dev.to/koolkamalkishor/running-llama-32-on-android-a-step-by-step-guide-using-ollama-54ig>