Legal Document Change Analyzer - Technical Interview Task

Background

Law firms often need to review changes between contract versions to identify which modifications carry legal significance versus mere formatting updates. You've been asked to build a proof-of-concept system that can intelligently analyze and categorize these changes.

Your Task

Build a system that compares two versions of a legal document and intelligently categorizes the changes found.

Provided Materials

- Two versions of a Data Processing Agreement (provided as .pdf files)

Requirements

Change Detection

- Identify all changes between document versions
- Handle complex changes (multi-line modifications, moved sections, etc.)

Change Categorization

Classify each change into one of these categories:

- **Critical (Legally Significant)**: Changes that materially affect rights, obligations, or legal interpretation
- **Minor (Non-substantive)**: Clarifications, wording improvements, or administrative updates
- **Formatting**: Punctuation, capitalization, spacing, or other cosmetic changes

Impact Analysis

For critical changes, provide:

- Brief explanation of potential legal implications
- Which party (Data Controller/Processor) is most affected
- Severity rating (high/medium/low)

Output Format

Generate a structured JSON report containing required data.

Bonus Features (if time allows)

- Simple web interface for uploading and viewing results

- Side-by-side diff visualization
- Export functionality for legal teams

Technical Constraints

- Use any LLM API (OpenAI, Anthropic, Cohere, etc.)
- Include clear setup/deployment instructions
- Document your approach to handling edge cases
- Time limit: 3-4 hours max

Deliverables

1. Source code with dependencies listed
2. README with setup and usage instructions
3. Sample output from analyzing the provided documents
4. Brief explanation of your approach and any limitations

Interview Discussion Topics

Example topics to cover during discussion:

- How you'd extend this to handle other document types
- Strategies for improving accuracy
- How to handle documents in different languages
- Integration with existing legal workflows
- Testing strategies for legal correctness