DocuSenseLM — Product Requirements Document (PRD)

Purpose

DocuSenseLM is a **local-first document intelligence assistant** that lets users ingest a set of business documents (initially **PDF** and **DOCX**), extract/structure key fields, and chat with those documents using retrieval-augmented generation (**RAG**). The system is designed to work as a **desktop app** (Windows-first) with user data stored locally.

Target Users & Personas

- **Analyst / Ops**: needs to quickly find clauses, parties, dates, renewal terms, pricing, and obligations across many documents.
- **Legal / Compliance**: needs accurate extraction, traceability to source text, and controlled local storage.
- **Sales / Procurement**: needs quick Q&A across NDAs/pricing docs and standardized summaries.

Problem Statement

Users spend significant time manually reading documents to extract structured details and answer questions across a corpus. They want a fast desktop workflow to:

- upload many documents at once,
- see processing status,
- edit extracted fields,
- search and ask questions with cited sources,
- manage/backup their local dataset.

Goals (What success looks like)

- **Fast bulk ingestion**: uploading many files is quick; processing continues in the background.
- **Reliable processing**: no corruption of metadata/state under concurrency.
- **Useful Q&A**: chat answers grounded in indexed documents with source citations.
- **User trust**: local storage transparency, easy backup/restore, clear error states.

Non-Goals (for v1.0)

- Multi-user collaboration / shared server deployments
- Cloud storage sync
- Full offline LLM/embedding runtime (currently expects OpenAI API)
- Advanced PDF markup authoring (nice-to-have later)

Core User Journeys

1) First Run / Setup

1. User installs and launches the app.

2. User configures API key in Settings.
3. App verifies backend health and initializes RAG components.

2) Bulk Upload & Processing

1. User chooses a document type (e.g., NDA) and selects multiple files.
2. App uploads files quickly (upload phase).
3. App triggers background processing and shows per-file status until complete.
4. User can close the modal and continue using the app while processing proceeds.

3) Review & Edit Extracted Metadata

1. User opens a document entry.
2. User reviews extracted fields and edits values (with appropriate parsing/validation for dates).
3. User saves changes; they persist locally.

4) Ask Questions / Chat with the Corpus

1. User goes to Chat and asks a question.
2. App retrieves relevant chunks (hybrid retrieval) and sends to LLM.
3. App returns an answer with sources (document references).

5) Manage Data + Backup/Restore

1. User opens the local storage folder for transparency.
2. User downloads a backup zip via the app.
3. User restores from a backup (verification required before release).

Feature Requirements (High Level)

Document Ingestion

- Accept PDF and DOCX.
- Bulk upload should separate **upload** from **processing**.
- Show upload/processing status per file.
- Support reprocessing a document with immediate UI feedback.
- Security basics: file size limits, filename sanitization, type validation.

Document Library

- List documents with status, type, upload date, and workflow attributes.
- Search/sort/filter by type and key fields.
- Archive/unarchive and delete (as applicable).

Metadata Extraction & Editing

- Extract a set of type-specific fields using templates/prompts.
- Allow manual edits and persist edits.
- Maintain stable schema and prevent metadata corruption during concurrent updates.

Chat (RAG)

- Ask questions across processed documents.
- Include sources/citations (document references).
- Chat should remain responsive during document processing.
- Allow clearing chat history (and actually remove stored data).

Settings & Configuration

- Configure OpenAI API key.
- Configuration editor (with reset-to-default).
- Open local storage folder.
- Backup download / restore.

UX Requirements

- Clear startup experience (backend initializing → ready).
- Clear status indicator for documents: pending/uploaded/processing/processed/error.
- Bulk upload modal: allow closing after upload phase completes.
- Errors should be actionable and not block app interaction unnecessarily.

Quality & Compliance Requirements (Product-Level)

- **Local-first**: documents and derived metadata stored in user profile directories.
- **Transparency**: user can open local storage folder.
- **Safety**: do not silently lose/overwrite metadata during parallel processing.

Release Criteria (MVP)

- Windows install flow verified on a clean machine.
- Bulk upload + background processing stable under load (e.g., 10 docs).
- Chat usable while docs process.
- Backup/restore verified end-to-end.
- Known “white screen” failures mitigated or clearly recoverable.