DocuSenseLM — Software Requirements Specification (SRS)

1. Scope

DocuSenseLM is a desktop application that:

- runs a local UI (Electron + web frontend),
- runs a local HTTP API backend (Python/FastAPI) on a fixed port,
- stores documents, metadata, templates, and vector DB data in a user profile directory,
- provides document ingestion, processing, metadata editing, and RAG chat over the document corpus.

This SRS defines **functional requirements**, **interfaces**, **data/storage**, **non-functional requirements**, and **acceptance criteria** for release readiness.

2. Definitions

- **Document**: a user-supplied file (PDF or DOCX) stored locally.
- **Processing**: extraction + chunking + embedding/indexing + metadata inference for a document.
- **RAG**: retrieval of relevant chunks from the index + LLM answer generation.
- **User Data Directory**: OS-specific directory where the app persists all user data.

3. System Overview (Architecture)

3.1 Components

- **Desktop Shell**: Electron main process
  - launches the UI window
  - starts and monitors the Python backend
  - provides IPC events (startup status, backend ready/error)
  - provides backup download behavior via Electron download manager
- **Frontend UI**: web app bundled into Electron
  - calls backend HTTP endpoints (localhost)
  - displays documents, statuses, chat, settings
- **Backend API**: Python FastAPI
  - handles upload, processing queue, documents list, config read/write, chat
  - persists metadata.json and a local vector DB

3.2 Ports / Local Endpoints

- Backend port: **14242** (fixed)
- Health check: `GET /health`

4. Data & Storage Requirements

4.1 Storage Locations

The application MUST store all user data under a user profile directory (platform dependent).

At minimum, these directories/files exist:

- `documents/`
  - document files (PDF/DOCX)
  - `metadata.json` (document registry + extracted fields)
- `templates/` (prompt/templates or user-defined extraction templates)
- `chroma_db/` (vector database persistence)
- `config.yaml` (user configuration override)
- `prompts.yaml` (user prompt override)

4.2 Metadata Integrity

- Concurrent read/write of `metadata.json` MUST be thread-safe.
- If a document processing operation updates metadata, it MUST NOT corrupt the file.

4.3 Backup/Restore

- Backup MUST produce a zip containing all required user data (documents, metadata, templates, config, DB).
- Restore MUST replace/merge in a deterministic way and must preserve integrity (acceptance testing required).

5. Functional Requirements

5.1 Startup & Backend Lifecycle

- **FR-START-01**: On app launch, Electron MUST start the backend process.
- **FR-START-02**: Electron MUST poll `GET /health` until ready or until max attempts are exceeded.
- **FR-START-03**: Frontend MUST not assume the backend is ready until:
  - it receives an IPC “python-ready” event OR
  - health checks succeed (fallback).
- **FR-START-04**: If the backend fails to start, the user MUST see an actionable error state (not just console logs).

5.2 Document Upload (Single)

- **FR-UP-01**: User can upload a single PDF/DOCX.
- **FR-UP-02**: Backend MUST validate file type and size limits.
- **FR-UP-03**: Backend MUST sanitize filenames and prevent path traversal.
- **FR-UP-04**: Backend MUST update document status appropriately (uploaded/processing/processed/error).

5.3 Bulk Upload (Multi-file)

- **FR-BULK-01**: UI MUST support selecting multiple files.
- **FR-BULK-02**: Upload phase MUST upload all files first (can be parallelized).
- **FR-BULK-03**: After upload phase completes, UI MUST trigger processing via a dedicated endpoint.
- **FR-BULK-04**: UI MUST allow user to close the modal after upload completes; processing continues.

- **FR-BULK-05**: UI MUST show per-file status transitions (uploading → uploaded → processing → completed/error).

5.4 Processing Engine

- **FR-PROC-01**: Backend MUST process documents in the background without blocking UI.
- **FR-PROC-02**: Backend MUST support parallel processing with a bounded worker count.
- **FR-PROC-03**: Processing MUST be resumable via “reprocess” action per document.
- **FR-PROC-04**: If processing fails for a document, status MUST reflect error and logs MUST capture reason.

5.5 Documents List / Library

- **FR-DOCS-01**: UI MUST list all known documents with key attributes:
  - filename, type, upload date, status, archived flag, extracted fields
- **FR-DOCS-02**: UI MUST provide search and sorting.
- **FR-DOCS-03**: UI MUST support archive/unarchive (if implemented).
- **FR-DOCS-04**: UI MUST support delete with confirmation (if implemented).

5.6 Metadata Editing

- **FR-META-01**: UI MUST allow editing extracted metadata fields.
- **FR-META-02**: Backend MUST persist edits to metadata storage.
- **FR-META-03**: Date fields MUST be parsed/validated and stored in a consistent format.

5.7 Chat (RAG)

- **FR-CHAT-01**: User can ask a question and receive an answer grounded in document content.
- **FR-CHAT-02**: Retrieval MUST use indexed content and return relevant sources.
- **FR-CHAT-03**: UI MUST display sources/citations alongside answers.
- **FR-CHAT-04**: Chat MUST remain usable while documents are processing.
- **FR-CHAT-05**: “Clear chat history” MUST remove stored conversation state.

5.8 Settings & Configuration

- **FR-SET-01**: User can set OpenAI API key (config or env).
- **FR-SET-02**: UI MUST allow editing config and prompts, including reset-to-default behavior.
- **FR-SET-03**: UI MUST allow opening the local user-data folder.
- **FR-SET-04**: UI MUST support backup download and restore (restore verification required).

6. External Interfaces

6.1 IPC Interfaces (Electron → Frontend)

- `startup-status`: string status messages during startup
- `python-ready`: signal that backend is ready
- `python-error`: error message if backend fails/crashes

6.2 HTTP API (Frontend → Backend) (Illustrative)

The backend MUST provide endpoints supporting:

- health/config
- document listing
- upload (including optional skip-processing behavior)
- start-processing (batch trigger)
- chat (RAG)
- backup/restore

Exact payloads/URLs may evolve; acceptance tests MUST verify current implementation.

7. Non-Functional Requirements

7.1 Reliability

- **NFR-REL-01**: No metadata corruption under concurrent processing (thread-safe writes).
- **NFR-REL-02**: Background processing MUST not freeze the UI.

7.2 Performance

- **NFR-PERF-01**: Bulk upload should complete upload phase quickly (bounded by disk/network).
- **NFR-PERF-02**: Parallel processing worker count MUST be bounded to avoid resource exhaustion.

7.3 Security

- **NFR-SEC-01**: File upload MUST validate size/type and sanitize names.
- **NFR-SEC-02**: Sensitive credentials (API keys) SHOULD NOT be stored in plaintext long-term (post-MVP improvement noted).

7.4 Maintainability

- **NFR-MAINT-01**: Provide stable, testable UI selectors/attributes for automation (post-MVP/e2e workstream).

8. State Model

8.1 Document Status Values

Document status MUST include at least:

- `pending` (known but not processed)
- `processing` / `reprocessing`
- `processed`
- `error` / `failed`

UI MUST map these states to clear labels and icons.

9. Acceptance Criteria (Release Gate)

9.1 Smoke Acceptance

- App installs and launches on Windows without admin.
- Backend becomes ready and UI transitions out of loading.
- Upload 1 PDF and see it appear in documents list.
- Upload 5–10 docs in bulk: upload phase completes quickly; processing begins; statuses update; no corruption of metadata.
- During processing, user can open Chat and ask a question; app responds (may not include in-progress docs).

9.2 Data Integrity

- After stress run (parallel processing), `metadata.json` remains valid JSON and contains all documents with correct status.

9.3 Backup/Restore (Required before “done”)

- Backup download produces a zip file saved to user-chosen location.
- Restore from that zip recreates all documents and metadata and the app functions afterward.

10. Open Issues / Known Gaps

- User-facing surfacing of backend crash errors (IPC `python-error`) needs a dedicated UI error component/state.
- Auto-update verification is pending.
- PDF viewer periodic white-screen issue remains pending.