Fidera
Privacy First AI Document Sharing & Intelligence Platform.

1. COMPLETE WORKFLOW
Privacy-First AI Document Sharing & Intelligence Platform
This is the **authoritative system flow**. If something isn't here, it doesn't exist.

A. User Onboarding & Trust Establishment
1. User signs in (JWT-based auth).
2. System establishes:
    - User identity
    - Role = Owner
3. No file access without authentication.

Trust begins **before AI**.

B. Upload → Transparency Gate (Your Core Moat)
B1. Ephemeral Intake
- User uploads file into **temporary staging**.
- File is never written to permanent storage.

B2. Metadata Exposure
- System extracts **all metadata**.
- UI shows:
    - **Metadata Present (Raw)**
    - **Metadata After Removal (Preview)**

User **must** acknowledge the difference.

B3. Explicit Consent
- User confirms upload.
- If cancelled → file destroyed instantly.

This step makes your privacy claim provable.

C. Sanitization & Secure Storage
1. Metadata is stripped.
2. File is revalidated.
3. Only the sanitized file is stored.
4. User sets:
    - Expiry duration (mandatory)

No expiry = no storage.

D. Automatic AI Intelligence Pipeline
Triggered immediately after storage.

D1. Document Understanding
- Text extraction

- OCR (only if needed)
- Language + structure detection

D2. Privacy Risk Intelligence (Owner-Only)

AI computes:

- Metadata exposure risk
- Content sensitivity risk
- Distribution suitability

Outputs:

- Risk level
- Human-readable explanation

D3. Sensitive Data Awareness

- Detects PII, financial markers, identifiers
- Highlights risk zones (no auto-redaction)

D4. AI Knowledge Indexing

- Chunking
- Embeddings
- Secure vector indexing
- All AI artifacts inherit file expiry

No orphan intelligence.

D5. Intelligence Snapshot

Auto-generated:

- Executive summary
- Key points
- Risk notes

Owner-only. Always.

E. Controlled Sharing (Zero Trust by Default)

E1. Share Link Creation

Owner defines:

- Viewer access
- Expiry (cannot exceed file expiry)

E2. Shared Viewer Capabilities

- View-only document
- AI chat (source-bound)
- No downloads
- No resharing
- No metadata visibility

E3. Viewer Safeguards

- Watermarked sessions
- Logged access
- Rate-limited AI responses

F. AI Chat (Permission-Aware)

Owner Chat
- Full document context
- Can request summaries, explanations, risks

Viewer Chat
- Limited scope
- No full-document summarization
- No bulk extraction

Chat is **assistance**, not exfiltration.

G. Expiry & Erasure (Hard Stop)
At expiry:
- File deleted
- AI embeddings deleted
- AI outputs deleted
- Share links invalidated
- Chat disabled

No recovery. No exceptions.
This is where most products lie. You won't.

2. FINAL TECH STACK (CORRECT, DEFENSIBLE, SCALABLE)
This stack is **appropriate** for Fidera. Anyone telling you otherwise is optimizing for comfort, not correctness.

Backend (Source of Truth)
Language
- Python 3.11+

Framework
- FastAPI
  - Async uploads
  - Background tasks
  - OpenAPI schema (frontend contracts)

Core Backend Modules
- Auth & JWT
- File ingestion & staging
- Metadata extraction & stripping
- AI pipeline orchestration
- Permission-aware RAG
- Expiry & deletion workers

AI & Intelligence Layer
Metadata Processing
- ExifTool
- PyPDF / pikepdf

- python-docx

Document Understanding

- pdfplumber / PyMuPDF
- Tesseract (OCR, conditional)

Embeddings

- SentenceTransformers (local)

Vector Store

- FAISS (local, expiry-bound)

LLM

- Local via Ollama OR
- API-based (fallback)

Important:
LLM is **replaceable**, architecture is not.

Storage
Object Storage

- S3 / MinIO
- Sanitized files only

Database

- PostgreSQL
- Strong relational constraints (files, permissions, audit logs)

No Mongo. No shortcuts.

3. Frontend: You're Thinking Correctly (With One Warning)
Yes, you **should** build:

- A good-looking website
- Animated UI
- Smooth transitions
- Trust-building UX

But here's the warning:
Do not mix "marketing website" and "application UI" concerns.
They are different products.

4. Frontend Architecture (Correct Way)
Split the Frontend into Two Layers
1️⃣ **Marketing Website (Public)**
Tech

- Next.js
- Tailwind CSS
- Framer Motion
- GSAP (optional)

Purpose

- Explain Fidera
- Build trust
- Show animations
- No sensitive logic

This is where:

- Animated CSS
- Scroll effects
- Hero transitions
- Brand storytelling lives

No backend auth here except signup/login redirects.

2️⃣ Application Dashboard (Private)

Tech

- Next.js (same codebase, different routes)
- React Query / TanStack Query
- Zustand / Context
- Tailwind + motion (subtle, not flashy)

Purpose

- File upload
- Metadata preview
- AI insights
- Chat
- Sharing controls

This UI must be:

- Fast
- Predictable
- Boring in the good way

Security beats aesthetics here.

5. Backend ↔ Frontend Integration (This Is Key)

This is where many teams screw up. Don't.

Integration Model (Clean)

FastAPI exposes **pure APIs**
Frontend consumes via **typed contracts**

Step 1: Define API Contracts First

- Upload
- Metadata preview
- Confirm upload
- AI status
- Chat
- Share

Frontend does not guess behavior.

Step 2: Auth Flow

- Frontend handles login UI
- Backend issues JWT
- Frontend stores token securely
- Every request includes token

No frontend-side permission logic. Ever.

Step 3: File Upload Flow
Frontend:
- Upload file
- Receive metadata preview
- Render before/after
- User confirms

Backend:
- Controls staging
- Controls deletion
- Controls AI trigger

Frontend only **reflects state**, not decides it.

Step 4: Real-Time Feel (Without WebSockets Yet)
Use:
- Polling
- Background status endpoints
- Optimistic UI updates

You don't need sockets for MVP.

6. Is Your Plan Correct?
What You're Doing Right
✅ Python + FastAPI
✅ Backend-first thinking
✅ AI as controlled intelligence
✅ Strong frontend focus
✅ Clear separation of roles
Where You Must Be Careful
⚠️ Don't let frontend dev invent backend behavior
⚠️ Don't prioritize animation over trust cues
⚠️ Don't blur marketing UI with app UI
⚠️ Don't copy GitHub repos blindly

Final Verdict
You are on a **good path**, but only if you treat:
- Palmr as a **mechanical reference**
- FastAPI as **authority**
- Frontend as **a consumer, not a decision-maker**
- AI as **constrained intelligence**

If you want next, I can:
- Design the **exact API schema** frontend should consume
- Review a frontend wireframe for **security UX mistakes**
- Define a **frontend-backend integration checklist** so nothing leaks

Say what you want next.