🔑 Core Features & Security Guarantees

- ☐ End-to-end document confidentiality
  with IPFS + encryption
- ☐ Zero-trust access control
  (wallet-based + Passkey via WebAuthn)
- ☐ Version-controlled, immutable storage
- ☐ Blockchain anchoring on Polygon for verifiability
- ☐ DLP (Data Loss Prevention) module
  to prevent leaks, screenshotting, or unauthorized sharing
- ☐ Real-time document signing & audit trail
- ☐ Optional token/NFT-gated or role-gated access

🏗️ System Components

Frontend

User interface and document interaction

DLP

Data Loss Prevention system

Signature

Document verification workflow

Backend

API and business logic layer

Storage

IPFS and metadata management

Blockchain

Polygon smart contracts

Authentication

Passkey and access control

🔹 1. Frontend (Client)

Framework

Next.js (React)

Key Modules

- Secure document editor (Tiptap or ProseMirror)
- PDF viewer + diff viewer for versioning
- Wallet integration (via Wagmi, RainbowKit)
- **Passkey support** via WebAuthn (e.g., via **SimpleWebAuthn**)
- DLP UX hooks (disable copy/paste, watermarking, anti-screenshot overlays)

2. Backend (API Layer)

Stack

Node.js + tRPC or Express

Authentication

User/session authentication (SIWE + Passkey binding)

DLP Enforcement

Policy engine (e.g., restrict IPs, domains, times)

Document Management

Document metadata management

Version Control

Signature & version history index

Notifications

Notification engine (webhook or push)

Monitoring

Rate-limiting, logging, and monitoring

🔹 3. Storage Layer

Primary Storage

IPFS (via **Web3.Storage** or Infura IPFS)

Enhancements

- Versioning via CID chains
- Optional redundancy on Arweave

Encryption

- AES encryption before upload
- Decryption via client-side WebCrypto (key shared via Lit Protocol or Session-based access)

Metadata Indexing

PostgreSQL / Prisma

🔹 4. Blockchain Layer (Polygon)

Network

Polygon PoS or Polygon zkEVM

Smart Contracts

ClassifiedDocumentRegistry.sol:

- storeHash(bytes32 ipfsHash, uint docId, uint version)
- recordSignature(address signer, uint docId)
- toggleVisibility(uint docId, bool isPublic)

Optional: AccessControlNFT.sol – for NFT/role-based document access

Gas Efficiency

- Batched document hash submissions
- Merkle root option for bulk updates

🔷 5. Authentication & Access Control

Authentication

- Passkey-based login (WebAuthn + fallback to wallet auth)
- Optional multi-factor (email/SMS)

Access Control Logic

- Role-based (viewer, editor, approver)
- Token/NFT-gated viewing (e.g., hold ERC-20 or ERC-721)
- Invite-only encrypted link generation
- Expiring access tokens

🔹 6. Signature & Verification Workflow

1. **Document Upload**

   User uploads document to be signed

2. **Offchain Signing**

   EIP-712 typed signature of docId, CID, timestamp

   Stored in DB + hash optionally anchored onchain

3. **Onchain Option**

   Push hash and signer to Polygon for tamper-proof audit trail

   Public verifiability via explorer or frontend verifier

4. **Verification**

   Anyone can verify document authenticity and signatures

🔷 7. DLP (Data Loss Prevention) System

Integrated Hooks

- Realtime JS-based DOM controls (disable copying, blur on tab change, watermark session ID)
- Screenshot detection (e.g., canvas watermarking + session monitoring)
- Access policies (IP restriction, geofencing, time-limited viewing)

Admin Console

- Manage DLP rules per document or group
- Logs of policy violations
- Export of audit reports

🔁 Typical Workflow

1. User Authentication

User logs in via Passkey or Wallet

2. Document Upload

Uploads document, which is encrypted and stored on IPFS

3. Storage Confirmation

CID returned → metadata + version info saved to DB

4. Blockchain Anchoring

CID + hash submitted to ClassifiedDocumentRegistry smart contract on Polygon

5. Document Signing

Document is signed by one or more users (offchain or onchain)

6. DLP Enforcement

DLP module enforces read/write/view restrictions

7. Verification

Other users verify authenticity via onchain data or signature log

🔧 Dev Stack Summary

Layer	**Technology Stack**
Frontend	Next.js, Wagmi, WebAuthn, RainbowKit, Tiptap
Backend/API	Node.js, tRPC/Express, Prisma, PostgreSQL
Decentralized Storage	IPFS (**Web3.Storage** / Pinata / Infura)
Blockchain	Solidity, Polygon (PoS or zkEVM)
DLP	Custom policies, JS-based watermarking & guards
Auth	Passkey (WebAuthn), EIP-712, SIWE