Architecture Document – DocuChain

1. System Overview

The system consists of a React frontend, Node.js backend, Solidity smart contracts deployed on Polygon Blockchain. IPFS can be used in future for file storage.

2. Technology Stack

Layer	Technology Used
Frontend	React + Material UI
Backend	Node.js + ethers.js (for blockchain interactions)
Smart Contract	Solidity Contract (UUPS upgradeable), deployed on Polygon Amoy Testnet
File Hashing	All uploaded offer letters are hashed client-side using Keccak-256.

3. High-Level Architecture

- Web Application (Frontend)
  - Built with React + Material UI
  - Interfaces for issuers, verifier and viewers
- Smart Contracts (Blockchain)
  - Deployed on Ethereum-compatible chain (Polygon)
  - Stores the hash and metadata of issued offer letters
- Optional Decentralized File Storage (IPFS) - Future upgrade if needed
  - Actual PDFs may be uploaded to IPFS and linked to their hash
  - Ensures off-chain document storage with on-chain verification

4. Component Descriptions

Component	Description
Frontend (React + MUI)	Interfaces for issuing, verifying, and managing offer letters
Backend (Node.js)	Handles API requests, wallet auth, and communication with smart contracts
Smart Contracts (Solidity)	Core logic for issuing, revoking, and validating offer letters on-chain

Blockchain Layer (Polygon)	Stores immutable hash and metadata of issued documents
Storage (IPFS - Future)	Optional storage of original offer letters in a decentralized system

5. Security Considerations

- Secure API endpoints between frontend and backend
- Smart contract functions restricted by role-based access modifiers
- Keccak-256 hashing to prevent tampering or duplicate uploads

6. Deployment

Deployed on Vercel (frontend), Railway (backend) and Smart Contract on Polygon Amoy Testnet

7. Hashing Logic

The platform uses Keccak-256 hashing in both frontend and backend to ensure compatibility with the Ethereum Virtual Machine. Uploaded offer letters are hashed client-side before being submitted and the hash is stored on-chain. This prevents tampering and ensures integrity during verification.

8. Upgradeability

The smart contract is upgradeable via the UUPS (Universal Upgradeable Proxy Standard) model. This allows new features or bug fixes without changing the proxy address or losing data.

9. Role-Based Access

- Issuer: Authorized institution or employer who can issue, revoke and extend offer letters.
- Verifier: Public user (e.g. student, immigration officer) who verifies document authenticity using hash.
- Admin: Platform owner who manages issuers and can upgrade contracts or burn documents if needed.

10. Limitations & Future Enhancements

- Files must be preserved exactly to match the hash during verification (sensitive to re-saving).
- IPFS and document link integration is roadmap-based.
- Cross-border document issuing and credentialing is under research.