Specification Document – DocuChain

1. Smart Contract Interface

Below are the public functions exposed by DocuChain's smart contract (Solidity, UUPS Upgradeable):

function initialize(address contractOwner, address initialIssuer) public initializer
function issueOfferLetter(bytes32 issuerId, bytes32 subjectIdHash, bytes32 fileHash, uint256 expiryDate) external onlyIssuer
function revokeOfferLetter(bytes32 fileHash) external
function updateExpiry(bytes32 fileHash, uint256 newExpiry) external onlyIssuer
function burnLetter(bytes32 fileHash) external onlyOwner
function isLetterValid(bytes32 fileHash) external view returns (bool)
function getLetterDetails(bytes32 fileHash) external view returns (...)
function addIssuer(address issuer) external onlyOwner
function removeIssuer(address issuer) external onlyOwner

2. API Endpoints

- POST /api/issue – Upload and issue offer letter (calls smart contract)
- GET /api/verify?hash=.. – Verifies offer letter by file hash
- POST /api/login – Authenticates issuer

3. Data Models

- Offer Letter:

```
{
 fileHash: bytes32,
 issuerId: bytes32,
 subjectIdHash: bytes32,
 expiryDate: uint256,
 isRevoked: bool
}
```

- User:

```
{
 walletAddress: string,
 email: string,
 role: 'issuer' | 'admin'
}
```

4. Frontend Routes

- /login – Users will sign in using traditional sign-up methods . In our MVP, we are using a json file to sign up.
- /verify – Upload and verify an offer letter
- /issue – Upload form to issue a new offer letter
- /dashboard – View issued letters and revoke/extend them (Future upgrades)

5. Validation Rules

- Only valid PDF files can be uploaded (extension and MIME check)
- File is hashed using Keccak-256 before sending to smart contract
- Duplicate file hashes are rejected by the contract
- Expiry date must be in the future
- File must not be revoked or expired to pass verification

6. Integrations

- Polygon (Amoy Testnet via Alchemy) – Blockchain platform for smart contract deployment
- Railway – Hosts Node.js backend and APIs
- Vercel – Hosts frontend React app
- IPFS (Planned) – Optional decentralized file storage for PDFs

7. Assumptions

- Users are responsible for preserving the exact original PDF to match hash during verification
- Frontend is configured to use Keccak-256 hashing, matching Solidity's keccak256()

5. Validation Rules

- Uploaded letter must be PDF
- Hash must match on-chain

6. Integrations

- IPFS
- Metamask Wallet (used by Platform)
- Alchemy RPC