A PROJECT REPORT ON

Web3 based document ledger using blockchain, ipfs and cryptography

SUBMITTED TO THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN THE PARTIAL FULFILLMENT FOR THE AWARD OF THE DEGREE
OF

BACHELOR OF ENGINEERING

IN

COMPUTER SCIENCE AND DESIGN

BY

Vishal Borle	Examination Seat No. : B400130902
Rushikesh Kapadnis	Examination Seat No. : B400130921
Prasad Kendre	Examination Seat No. : B400130922
Om Pawar	Examination Seat No. : B400130942

UNDER THE GUIDANCE OF

Prof. Sonali Nimbalkar

DEPARTMENT OF COMPUTER SCIENCE AND DESIGN
K.K.WAGH INSTITUTE OF ENGINEERING EDUCATION AND RESEARCH
HIRABAI HARIDAS VIDYANAGARI, PANCHAVATI, NASHIK 422003

2024-25

CERTIFICATE

This is to certify that the project report entitled

Web3 based document ledger using blockchain, ipfs and cryptography

Submitted by

Vishal Borle	Examination Seat No. : B400130902
Rushikesh Kapadnis	Examination Seat No. : B400130921
Prasad Kendre	Examination Seat No. : B400130922
Om Pawar	Examination Seat No. : B400130942

is a bonafide work carried out by them under the supervision of Prof. Sonali Nimbalkar and it is approved for the partial fulfillment of the requirement of Savitribai Phule Pune University for the award of the Degree of Bachelor of Engineering (Computer Science and Design).

This project report has not been earlier submitted to any other Institute or University for the award of any degree or diploma.

Prof. Sonali Nimbalkar
Internal Guide
Department of Computer
Science and Design

Head of Department
Department of Computer
Science and Design

External Examiner

Date:

Place: Nashik

Abstract

This project presents a Web3-Based Document Ledger, a decentralized document management system that leverages the high-performance Solana blockchain and distributed IPFS storage. The system addresses critical limitations of traditional document management including centralization, data vulnerability, and lack of transparency by implementing a hybrid architecture focused on security, user ownership, and decentralization. Metadata and access controls are stored immutably on-chain using Solana smart contracts, while documents are encrypted using AES-256-CBC and stored on IPFS. Wallet-based authentication ensures secure and user-centric access, eliminating traditional credential systems. A novel document sharing mechanism encrypts the owner's signature with the recipient's public key to provide fine-grained, end-to-end secure access. The platform includes a user-friendly React-based web interface and a backend powered by Node.js and PostgreSQL. The project follows an iterative development approach encompassing design, smart contract development, backend integration, testing, and optimization. This applied research contributes to the Web3 ecosystem by offering a scalable, censorship-resistant, and secure alternative to conventional document management systems. Key innovations include blockchain-based access control, immutable audit trails, and decentralized sharing features, addressing identified gaps in existing solutions such as limited integration, poor usability, and lack of performance optimization. The outcome demonstrates a practical and secure application of blockchain beyond cryptocurrency use cases.

Acknowledgment

We would like to extend our sincere gratitude to our project guide, **Prof. Sonali Nimbalkar**, for her insightful mentorship and unwavering support throughout the course of this project. Her expertise, valuable suggestions, and consistent encouragement have been vital in shaping our approach and bringing this work to fruition.

We are also thankful to our project coordinator, **Dr. J. S. Hase**, whose continuous guidance and strategic advice greatly contributed to the structured progress and timely execution of our project tasks.

Our heartfelt appreciation goes to the Head of the Department, **Prof. Dr. S. M. Kamalapur**, along with all the departmental faculty members, for providing the necessary resources and academic environment that enabled us to carry out this project smoothly.

Lastly, we would like to acknowledge the constant support and encouragement from our friends, whose thoughtful inputs and feedback have been instrumental in improving the overall quality and completeness of our work.

Vishal Borle
Rushikesh Kapadnis
Prasad Kendre
Om Pawar
(B.E. Computer Science & Design)

INDEX

List of Figures

List of Tables

CHAPTER 1

INTRODUCTION

1.1 PROJECT IDEA

The proposed system, **Web3-Based Document Ledger Using Blockchain, IPFS, and Cryptography**, establishes a decentralized document management platform granting users complete ownership over their data. It utilizes the Solana blockchain for storing metadata and enforcing access control, while encrypted files are stored on IPFS to ensure secure and distributed availability. AES-256-CBC encryption safeguards confidentiality, and wallet-based authentication via Solana wallets replaces traditional login methods. Every action like uploads, retrievals, and permission changes is immutably recorded on-chain. A document sharing mechanism further enhances security by encrypting access keys with the recipient's public key, ensuring only intended users can decrypt shared files. The system integrates blockchain, IPFS, and cryptography to offer a censorship-resistant, verifiable, and scalable Web3-based document management solution.

1.2 MOTIVATION OF THE PROJECT

Traditional document systems pose significant risks such as unauthorized access, censorship, and centralized control over sensitive data. These platforms often rely on third-party trust and introduce single points of failure. Our project addresses these issues through a decentralized design leveraging blockchain for transparent, immutable access control and IPFS for redundant storage ensuring data sovereignty and tamper-resistance. Cryptographic methods guarantee that only authorized users can access content. Additionally, the solution reduces costs by removing intermediaries and demonstrates a realworld application of Web3 technology beyond digital assets.

CHAPTER 2

LITERATURE SURVEY

2.1 RESEARCH PAPERS STUDIED

- Zhang et al., *"Scalability Challenges in Ethereum-based Decentralized Storage," Journal of Decentralized Systems*, 2020 [1].

  Purpose: To investigate the limitations of Ethereum-based storage platforms like Swarm and StorJ, focusing on scalability challenges that affect large-scale deployment and adoption in enterprise environments.

  Description: This paper identifies Ethereum's low transaction throughput (approximately 15 TPS) as a major bottleneck, which limits its effectiveness for large-scale decentralized document systems. It highlights network congestion and high gas fees as key challenges that obstruct practical real-world use cases.

- Benet J., *"IPFS - Content Addressed, Versioned, P2P File System," IPFS Project Whitepaper*, 2014 [2].

  Purpose: To introduce IPFS as a decentralized file system that improves upon the traditional web infrastructure by offering content-addressed and distributed file storage.

  Description: This foundational paper describes how IPFS allows content deduplication, cryptographic validation, and decentralized access, which greatly enhances availability, performance, and data integrity, making it ideal for secure document storage solutions.

- N. Sangeeta and Seung Yeob Nam, *"Blockchain and IPFS-Based Data Storage System for Vehicular Networks with Keyword Search Capability," Journal of Computer Networks*, 2022 [3].

  Purpose: To design a secure and decentralized storage system tailored for vehicular networks, integrating blockchain and IPFS with searchable data access.

  Description: This research demonstrates how smart contracts can be used for access control while storing large vehicle data sets on IPFS, offering insights into scalable metadata indexing and controlled retrieval—key components also present in our document ledger.

- Randhir Kumar and Rakesh Tripathi, *"Implementation of Distributed File Stor-*

age and Access Framework using IPFS and Blockchain," Journal of Blockchain Technology, vol. 9, 2023 [4].

Purpose: To propose a framework that combines blockchain for metadata management and IPFS for off-chain document storage.

Description: The work outlines an architecture where documents are encrypted and stored on IPFS, and their metadata is recorded immutably on blockchain. This hybrid approach reflects our system's exact design, balancing scalability and data security.

- James Mahlaba et al., *"Blockchain-Based Sensitive Document Storage to Mitigate Corruptions," Journal of Information Security and Blockchain*, vol. 11, 2023 [5].

  Purpose: To develop a secure blockchain-based document management system that resists tampering and ensures long-term authenticity.

  Description: This research focuses on access-controlled document systems where cryptographic hashes and blockchain consensus prevent unauthorized alterations. Our project also leverages these principles to maintain an immutable and verifiable history of documents.

- Saqib Rasool et al., *"Docschain: Blockchain-Based IoT Solution for Verification of Degree Documents," Journal of Blockchain and IoT Solutions*, vol. 12, 2023 [6].

  Purpose: To develop a blockchain-powered platform that verifies academic documents using decentralized identifiers and QR codes.

  Description: The paper showcases a practical model of decentralized credential validation, helping eliminate forgery. It aligns with our project's goal of building an immutable verification layer for documents through blockchain consensus.

- Lina Li et al., *"A Secure, Reliable and Low-Cost Distributed Storage Scheme Based on Blockchain and IPFS for Firefighting IoT Data," IEEE Access*, vol. 12, 2024 [7].

  Purpose: To introduce an encrypted and fault-tolerant storage scheme for IoT systems by combining IPFS and blockchain.

Description: The system uses AES and RSA encryption to ensure data confidentiality and integrity. This supports our approach of encrypting documents before storing them on IPFS, particularly for scenarios requiring high data reliability.

- Anatoly Yakovenko, *"Solana: A New Architecture for a High Performance Blockchain," Solana Whitepaper*, 2017 [8].

  Purpose: To present Solana's high-performance blockchain framework capable of handling 50,000+ transactions per second using the Proof of History consensus model.

  Description: Solana's architecture provides real-time finality and low latency, making it an ideal platform for fast metadata operations in our project. The design ensures scalable and responsive document operations even under heavy load.

- Kumar et al., *"Smart Contract-Based Fine-Grained Access Control in Blockchain Systems," Journal of Blockchain Access Control*, 2021 [9].

  Purpose: To explore attribute-based access models implemented via smart contracts for secure and flexible data sharing in decentralized systems.

  Description: This research supports our system's smart contract design for permission management. It shows how decentralized authorization can be enforced without central intermediaries, improving transparency and security in document workflows.

CHAPTER 3

PROBLEM DEFINITION AND SCOPE

3.1 PROBLEM STATEMENT

To develop a Web3-based decentralized document ledger using blockchain, IPFS, and cryptography.

Description: To create a secure and transparent document management system by leveraging the Solana blockchain for immutable access control and audit logs, IPFS for encrypted decentralized storage, wallet-based authentication for user sovereignty, and advanced cryptographic techniques for secure document sharing,delivering a scalable, censorship-resistant, and user-centric solution aligned with Web3 principles.

3.1.1 Goals and Objectives

- **Goal:** To develop a decentralized document ledger system leveraging Solana blockchain and IPFS to provide secure, transparent, and user-controlled document management.
- **Objectives:**
  1. To store document metadata on the Solana blockchain using Program Derived Addresses (PDAs) for immutable and verifiable record-keeping.
  2. To implement smart contract-based access control mechanisms for decentralized permission management.
  3. To integrate IPFS for decentralized file storage while using AES-256-CBC encryption to ensure confidentiality of documents.
  4. To verify document integrity through cryptographic hashing techniques.
  5. To implement wallet-based authentication using Solana wallets and digital signatures, eliminating traditional username-password systems.
  6. To enable secure document sharing through encrypted access keys and support permission revocation and multi-user access.

3.1.2 Assumption and Scope

3.1.3 Assumptions and Scope

- **Assumptions:**
  - **Network Connectivity:** Users have stable internet access to facilitate interactions with blockchain and IPFS.
  - **Wallet Availability:** Users possess Solana-compatible wallets for authentication and transaction signing.
  - **Technical Literacy:** Users are familiar with Web3 concepts and basic wallet usage.
  - **Blockchain Reliability:** Solana network maintains sufficient uptime and stability for smooth transactions.
  - **IPFS Availability:** IPFS nodes remain online for dependable document retrieval.
- **Scope:**
  1. **Included Functionalities:**
     - Wallet-based user registration and authentication
     - Encrypted document upload and IPFS storage
     - Metadata logging on Solana blockchain
     - File integrity verification using hashing
     - Secure document sharing with key-based access control
     - Access permission assignment and revocation
     - Dashboard for document management and audit trail visualization
  2. **Technical Coverage:**
     - **Frontend:** React.js-based interface with Tailwind CSS
     - **Backend:** Node.js and Express.js with TypeScript
     - **Blockchain:** Solana blockchain using the Anchor framework
     - **Storage:** IPFS via Pinata or local node for file storage

    * **Database:** PostgreSQL for managing user profiles (off-chain)
    * **Encryption:** AES-256-CBC for encryption, SHA-256 for integrity checks

3.2 METHODOLOGY

The project employs a modular, decentralized architecture that integrates blockchain, IPFS, wallet authentication, and cryptographic operations. The implementation methodology includes:

1. **Requirement Analysis:**
   - Identify key pain points in traditional document storage systems.
   - Determine user needs for privacy, ownership, and verifiable access control.
2. **System Architecture Design:**
   - Design a layered structure with frontend, backend, blockchain, and decentralized storage components.
   - Outline smart contract roles for access control and metadata handling.
3. **Smart Contract Development:**
   - Write and deploy smart contracts on Solana using Anchor framework.
   - Store metadata like CID, timestamps, and permission rules.
4. **Secure Storage and Encryption:**
   - Apply AES-256-CBC encryption to documents prior to upload.
   - Generate SHA-256 hashes to track document integrity.
5. **Authentication via Wallets:**
   - Authenticate users using Solana wallets (e.g., Phantom).
   - Use digital signatures to replace traditional login credentials.

6. **Frontend and Backend Development:**
   - Develop the DApp interface using React.js and Tailwind CSS.
   - Implement APIs with Express.js to connect frontend, blockchain, and IPFS.
7. **Document Sharing and Access Control:**
   - Share documents securely by encrypting the access key with the recipient's public key.
   - Manage user permissions with on-chain smart contract logic.
8. **Testing and Validation:**
   - Conduct unit tests (Jest/Mocha) for backend and smart contracts.
   - Perform integration tests for upload-retrieve-share flows.
9. **Optimization and Deployment:**
   - Optimize blockchain interaction for performance and cost efficiency.
   - Deploy all components in a secure and production-ready environment.

3.3 OUTCOMES

The project results in several significant deliverables and technical accomplishments, showcasing the effectiveness of decentralized document management through Web3 technologies:

- **Web3 Document Management Platform:** A fully functional decentralized application (DApp) that allows users to upload, store, retrieve, and share documents securely using blockchain and IPFS integration via an intuitive user interface.
- **Integration of Solana Smart Contracts and IPFS:** Smart contracts deployed on the Solana blockchain handle metadata logging and access control, while documents are encrypted and stored on IPFS for decentralized availability.

- **Confidential and Verifiable Document Exchange:** The use of AES-256-CBC encryption for file content and SHA-256 hashing ensures both confidentiality and integrity of documents during transmission and storage.
- **Granular Access Control with Encrypted Keys:** The system introduces a secure sharing mechanism by encrypting access keys per recipient, enabling flexible, revocable permissions for multiple users.
- **Thorough Documentation and Testing Suite:** Detailed technical documentation, including API references, user guides, and comprehensive testing logs, ensures maintainability and system validation under various scenarios.
- **Non-Crypto Use Case Demonstration:** This project exemplifies how blockchain can be effectively applied to real-world challenges outside cryptocurrency, promoting adoption in secure information management and recordkeeping.

3.4 TYPE OF PROJECT

- **Research/Application Oriented:**
  This is primarily an *application-oriented* project, augmented by research-driven design and security modeling. The implementation bridges theoretical advancements in decentralized storage and blockchain access control with practical development.
- **Domain Areas:**
  - Blockchain and Web3 Technologies
  - Cryptography and Data Privacy
  - Distributed Systems and Secure Architecture
  - Full-Stack DApp (Decentralized Application) Development

CHAPTER 4

PROJECT PLAN

4.1 PROJECT TIMELINE

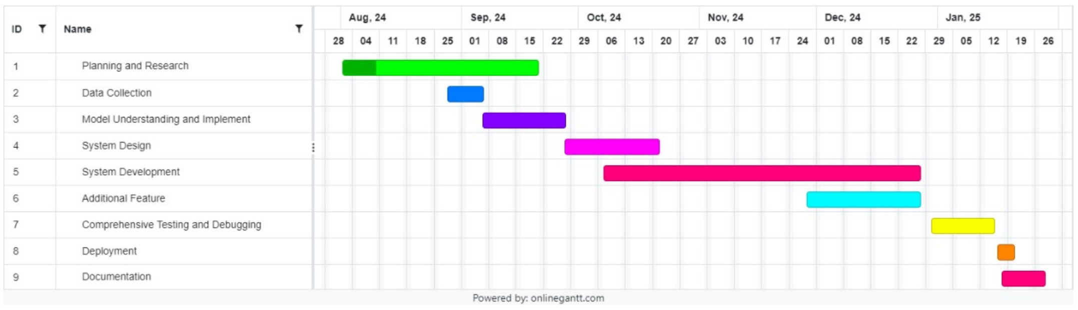


Figure 4.1: Project Timeline

4.2 TEAM ORGANIZATION

4.2.1 Team structure

Project Leader: Vishal Borle : team coordination, task distribution, and development

Project Members:

Rushikesh Kapadnis : UI/UX development, diagrams, documentation

Prasad kendre : Performed analysis, flow charts, and testing

Om Pawar : Testing and deployment, system architecture.

CHAPTER 5

SOFTWARE REQUIREMENT SPECIFICATION

5.1 FUNCTIONAL REQUIREMENTS

5.1.1 User Interface and Visualization

- **FR-1: Wallet-Based Authentication**
  Users authenticate via Solana-compatible wallets (e.g., Phantom, Solflare). Wallet signatures confirm identity, and sessions are managed using secure JWT tokens.
- **FR-2: Dashboard Interface**
  A centralized dashboard will offer access to file upload, retrieval, sharing, permission management, and activity logs.
- **FR-3: File Management Interface**
  Enables intuitive operations such as uploading, viewing, deleting, and searching documents.
- **FR-4: Secure Document Sharing**
  Users can share documents by encrypting keys for recipients' wallets, supporting secure collaboration.
- **FR-5: Real-Time Feedback**
  Visual status indicators display transaction confirmations and IPFS upload statuses.
- **FR-6: Responsive Design**
  The interface adapts across devices, providing optimal experience on desktop and mobile platforms.

5.1.2 Document Access and Control

- **FR-7: Encrypted File Upload**
  Files are encrypted using AES-256-CBC and uploaded to IPFS through Pinata or similar gateway services.
- **FR-8: Metadata Logging on Blockchain**
  File metadata including CID, SHA-256 hash, and filename are recorded immutably on the Solana blockchain.

- **FR-9: Smart Contract-Based Access Control**

  PDAs (Program Derived Addresses) enforce document-specific access rights on-chain.

- **FR-10: Integrity Verification**

  SHA-256 hashing is used to validate file integrity upon retrieval and decryption.

- **FR-11: Decentralized Storage**

  IPFS ensures redundancy and decentralized availability, protecting against central points of failure.

- **FR-12: Controlled Sharing Mechanism**

  Documents are shared by encrypting access keys with recipients' public keys.

- **FR-13: Access Revocation**

  Shared access can be revoked dynamically via smart contract updates without altering the document.

- **FR-14: Document Dashboard**

  Users can preview, download, share, or delete their documents in a centralized view.

- **FR-15: Transaction and Audit Logging**

  All significant user actions (upload, share, revoke) are recorded on the blockchain for transparency.

5.2 NON-FUNCTIONAL REQUIREMENTS

- **Security**

  AES and SHA-256 secure data at rest and in transit. Wallet-based login ensures identity verification without centralized credentials.

- **Performance**

  Uploading files ($<$50MB) should complete within 10 seconds. Retrievals should finalize within 5 seconds under normal load.

- **Reliability**

  System uptime should exceed 99.9%, with fallback IPFS nodes and transaction retries ensuring resilience.

- **Scalability**

  Backend and IPFS interactions should support thousands of documents per user and simultaneous sessions across multiple users.

- **Usability**

  The UI should be intuitive, visually clean, and require minimal Web3 knowledge. In-app tooltips and documentation enhance usability.

- **Interoperability**

  The platform must support major Solana wallets and comply with Web3 specifications for modular integrations.

5.3 CONSTRAINTS

- **Technical Constraints**

  The platform depends on the availability and performance of the Solana blockchain and IPFS network. Web3 wallet extensions like Phantom or Solflare require compatible browsers. Hardware wallet integration may be limited or require custom handling.

- **Business Constraints**

  The system is developed as an academic project with a restricted timeline and minimal financial resources. Single-developer involvement limits scalability and feature expansion.

- **Regulatory Constraints**

  Compliance with data protection standards (e.g., GDPR) must be ensured. Any third-party packages must adhere to open-source licensing policies. Institutional deployment may be governed by additional internal data governance rules.

5.4 HARDWARE REQUIREMENTS

- **Client-Side**

  End users should use devices with at least a quad-core CPU (Intel i5 or Ryzen 5), 8GB RAM, and a resolution of 1366×768 or higher (1920×1080 recommended) for optimal performance.

- **Server-Side**

  The server hosting the backend and optional local IPFS node should have a multi-core CPU (e.g., Intel Xeon, AMD EPYC), minimum 16GB RAM (32GB preferred), SSD storage of at least 1TB, and stable high-speed internet.

- **Development Environment**

  The system should be developed on machines with at least an Intel i7/Ryzen 7 CPU, 16GB RAM, and 256GB SSD for efficient compilation, testing, and blockchain deployments.

5.5 SOFTWARE REQUIREMENTS

- **Frontend**

  The frontend will be built using React 18+ with TypeScript, styled with Tailwind CSS. It will run on browsers supporting ES2020 features, with wallet integration supported through Phantom or Solflare extensions.

- **Backend**

  Node.js (v16+) with Express.js and PostgreSQL 13+ will handle backend logic and API calls. PM2 will be used for backend process management during production deployment.

- **Development Tools**

  The development stack includes Vite for fast builds, Jest/Mocha for testing, the Anchor framework for Solana smart contract deployment, and Git for version control.

- **Third-Party Services**

  IPFS access will be facilitated via gateways like Pinata or Infura. Solana Devnet and Mainnet will be used for blockchain testing and deployment respectively.

5.6 INTERFACES

- **User Interfaces**

  A web-based single-page application (SPA) with wallet login support, document upload via drag-and-drop, real-time transaction feedback, and audit history.

- **Hardware Interfaces**

  Network interfaces supporting HTTP/HTTPS protocols will handle requests. Local device storage may be used for session tokens and temporary encryption metadata.

- **Software Interfaces**

  Solana interaction will occur via the `@solana/web3.js` library and RPC endpoints. IPFS communication will be handled using RESTful HTTP APIs. PostgreSQL will serve structured user and session data.

- **Communication Interfaces**

  JSON-based REST APIs will bridge the frontend and backend. Wallet connections are managed using Solana Wallet Adapter. Encrypted access keys and smart contract interactions are transmitted via signed blockchain transactions.

CHAPTER 6

DETAILED DESIGN

6.1 ARCHITECTURAL DESIGN (BLOCK DIAGRAM)

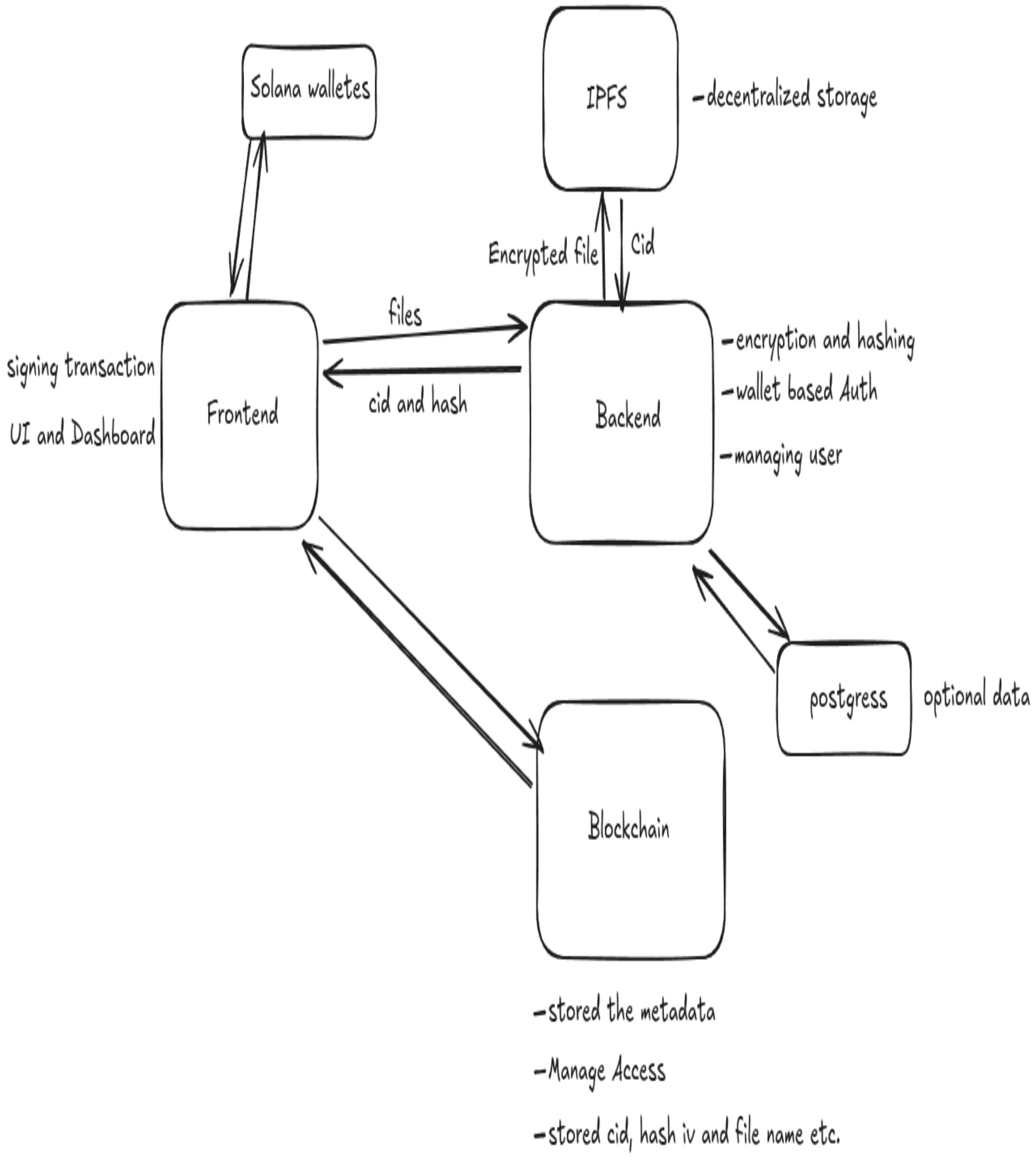


Figure 6.1: Architectural Design

Description:

Figure 6.1 illustrates the architecture of the Web3-Based Document Ledger platform, designed for secure, decentralized document storage, sharing, and verification.

The system starts with wallet-based authentication using Solana-compatible wallets (e.g., Phantom or Solflare). Authentication is performed via cryptographic digital signatures, eliminating the need for traditional login systems.

Once authenticated, users interact with a React-based frontend dashboard that supports document upload, retrieval, sharing, revocation, and audit logs. On file upload, the Express.js backend performs AES-256-CBC encryption and generates a SHA-256 hash for integrity verification. The encrypted file is then uploaded to IPFS via the Pinata gateway, which returns a unique Content Identifier (CID).

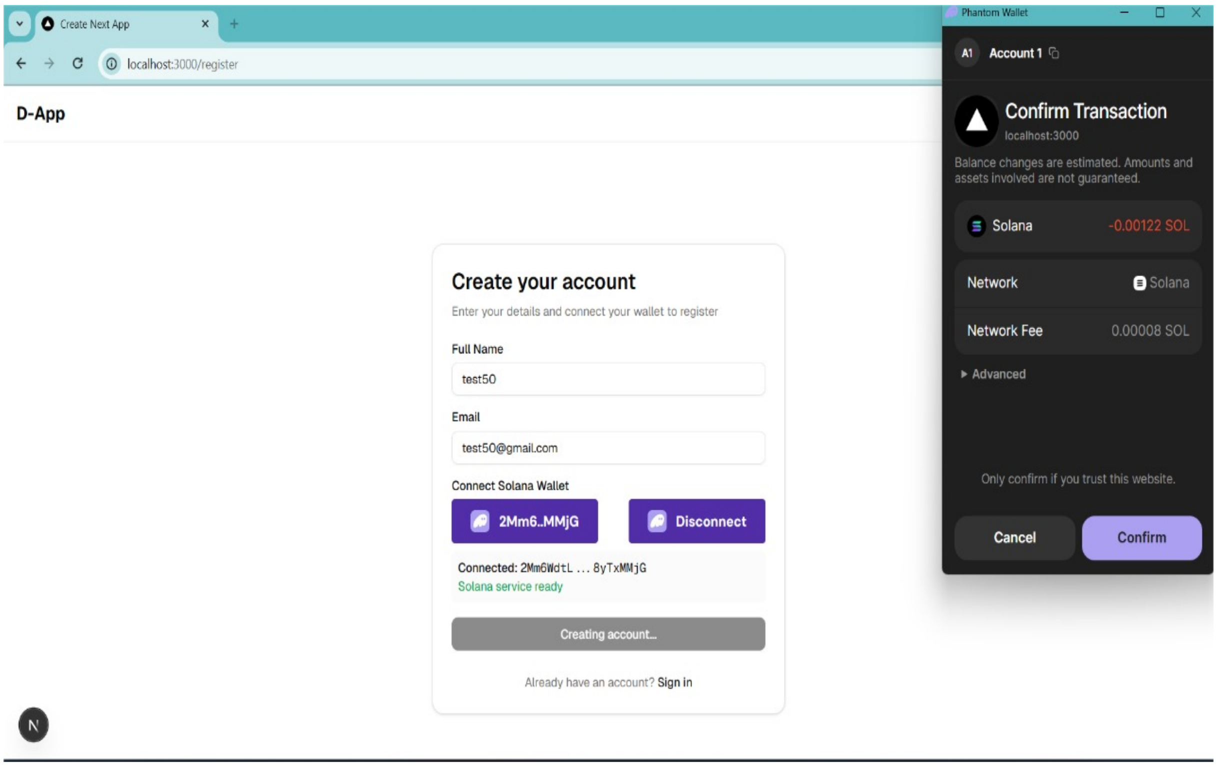
This CID, along with file metadata such as hash and filename, is stored immutably on the Solana blockchain via an Anchor-based smart contract. This mechanism guarantees transparent, tamper-proof metadata and access control tracking.

When a document is shared, the backend encrypts the access key using the recipient's public key, and the access permission is logged on-chain. During retrieval, the document is fetched from IPFS using its CID and verified using the stored hash.

Key Components

- **Frontend Layer:** A React.js interface integrated with Solana wallets for session handling, document operations, and interaction feedback.
- **Backend API:** An Express.js server responsible for encryption, hash generation, IPFS interaction, and blockchain transaction orchestration.
- **Solana Program:** A smart contract built with the Anchor framework, managing document metadata, access control, and audit history on-chain.
- **IPFS Storage:** Utilizes the Pinata gateway to interface with IPFS for decentralized and content-addressed file storage.
- **Database Layer:** Optional PostgreSQL storage for caching user preferences and storing non-sensitive application data.

6.2 USER INTERFACE SCREENS


Create Next App

localhost:3000/register

D-App

Create your account

Enter your details and connect your wallet to register

Full Name: test50

Email: test50@gmail.com

Connect Solana Wallet

Connected: 2Mm6WdtL ... 8yTxMMjG

Solana service ready

Creating account...

Already have an account? Sign in

Phantom Wallet

Account 1

Confirm Transaction

localhost:3000

Balance changes are estimated. Amounts and assets involved are not guaranteed.

Solana -0.00122 SOL

Network Solana

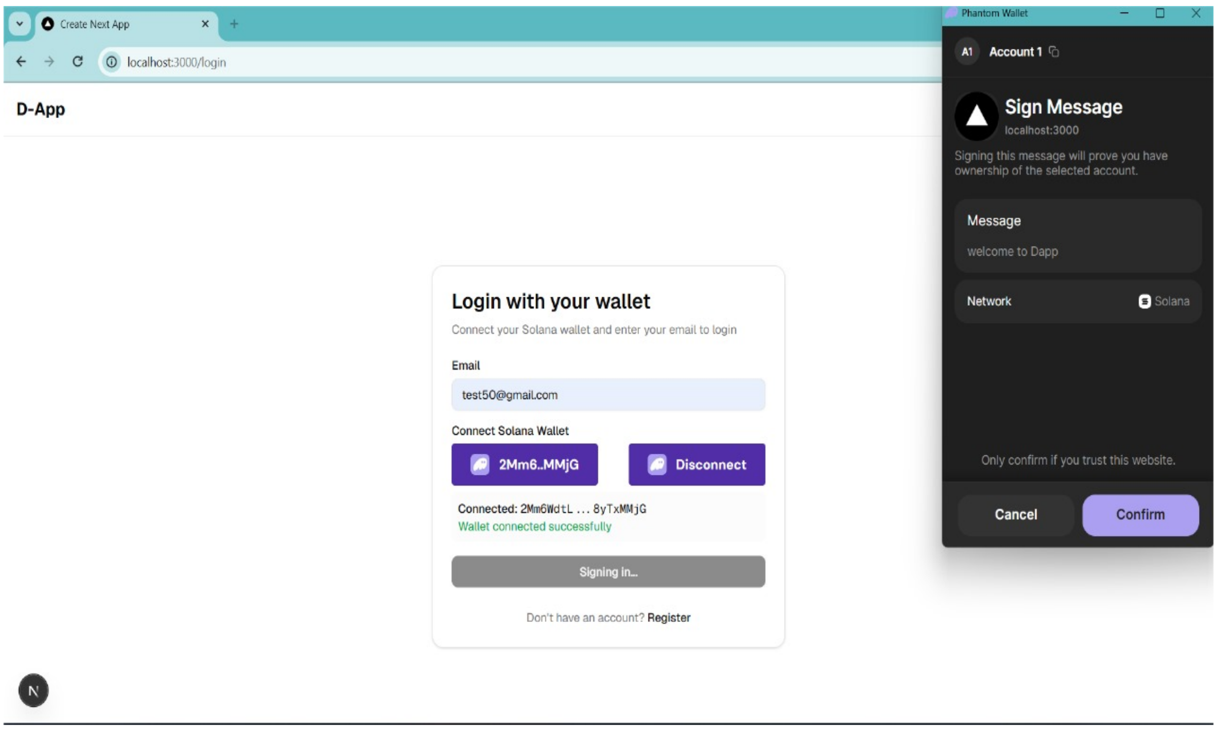
Network Fee 0.00008 SOL

Only confirm if you trust this website.

Figure 6.2: Account Creation

Description:

Figure 6.2 shows the account creation page, which facilitates user registration by collecting the full name, email address, and connecting the user's Solana wallet. Upon submission, a small transaction initializes the user's identity on the Solana blockchain. Simultaneously, the backend creates a user profile that includes the public key and user details. This hybrid design enables both on-chain verification and off-chain profile management for secure and decentralized authentication.



Signing this message will prove you have ownership of the selected account.

Message

welcome to Dapp

Network Solana

Only confirm if you trust this website.

Login with your wallet

Connect your Solana wallet and enter your email to login

Email

test50@gmail.com

Connect Solana Wallet

Connected: 2Mm6WdtL ... 8yTxMMjG

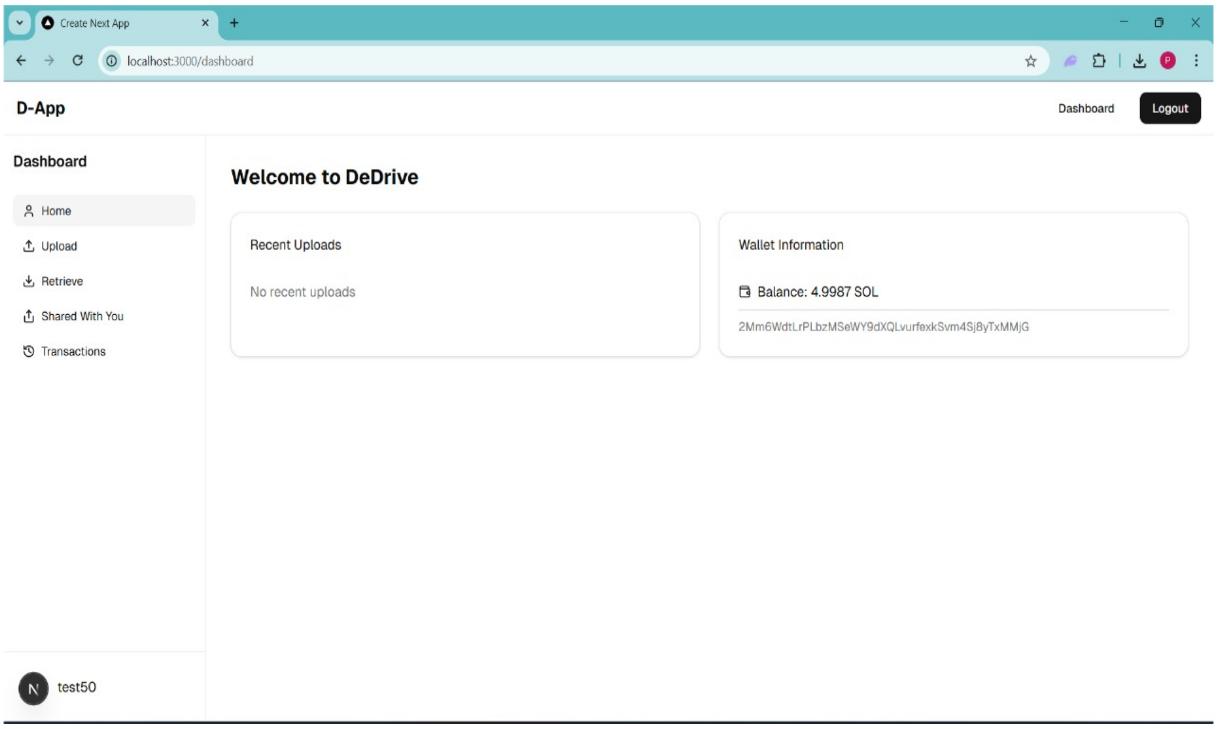
Wallet connected successfully

Don't have an account? Register

Figure 6.3: Login

Description:

Figure 6.3 displays the login screen. The system uses email verification combined with wallet-based authentication. The user signs a message using their wallet's private key, which is verified against the registered public key. This ensures secure login without traditional passwords, reducing phishing and credential theft risks.



Welcome to DeDrive

Recent Uploads

No recent uploads

Wallet Information

Balance: 4.9987 SOL

2Mm6WdtLrPLbzMSeWY9dXQLvurfexkSvm4Sj8yTxMMjG

Figure 6.4: Dashboard Page

Description:

Figure 6.4 illustrates the dashboard page, the primary interface for user interaction. It contains the following navigation options:

- **Home:** Shows a welcome message, recent uploads, wallet address, and SOL balance.
- **Upload:** Allows uploading encrypted documents to IPFS and registering metadata on-chain.
- **Retrieve:** Displays uploaded documents with options to view, download, share, delete, or revoke access.
- **Shared With You:** Lists documents received from other users with view/-download privileges.
- **Transactions:** Displays a history of blockchain interactions (upload, share, revoke, etc.).

6.3 DATA DESIGN

6.3.1 Data Structure

Solana Program Account Structures:

- `UserDocuments`

  This account keeps track of a user's uploaded documents.

  * `owner: Pubkey` (32 bytes)

    The wallet address of the user.

  * `document_count: u64` (8 bytes)

    The total number of documents the user has uploaded.

- `Document`

  This account stores metadata about each individual document.

  * `owner: Pubkey` (32 bytes)

    Wallet address of the document owner.

  * `file_name: String` (Variable, max 100)

    Name of the uploaded file.

  * `cid: String` (Variable, max 100)

    IPFS Content Identifier for retrieving the document.

  * `file_hash: String` (64 characters)

    SHA-256 hash of the document for integrity verification.

  * `created_at: i64` (Unix timestamp)

    Time of document creation.

  * `shared_with: Vec<SharedWith>` (Dynamic)

    List of access entries for other users.

- `SharedWith`

  Represents sharing metadata for a document.

  * `recipient: Pubkey` (32 bytes)

    Wallet address of the recipient.

- `access_key: String` (max 500)
  Encrypted key used for document decryption.
- `shared_at: i64` (Unix timestamp)
  Time when the document was shared.

Program Derived Address (PDA) Seeds:

- `UserDocuments: ["user_documents", user_pubkey]`
- `Document: ["document", user_pubkey, filename]`

6.3.2 Database Description

In addition to on-chain and decentralized data management, the system also incorporates an off-chain PostgreSQL database for managing user profile data. This enables efficient retrieval, dashboard population, and optional logging features while maintaining decentralized core functionalities.

Listing 6.1: Users Table Schema

```
CREATE TABLE users (
    id SERIAL PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    public_key VARCHAR(255) UNIQUE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    updated_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

6.4 COMPONENT DESIGN / DATA MODEL

6.4.1 Class Diagram

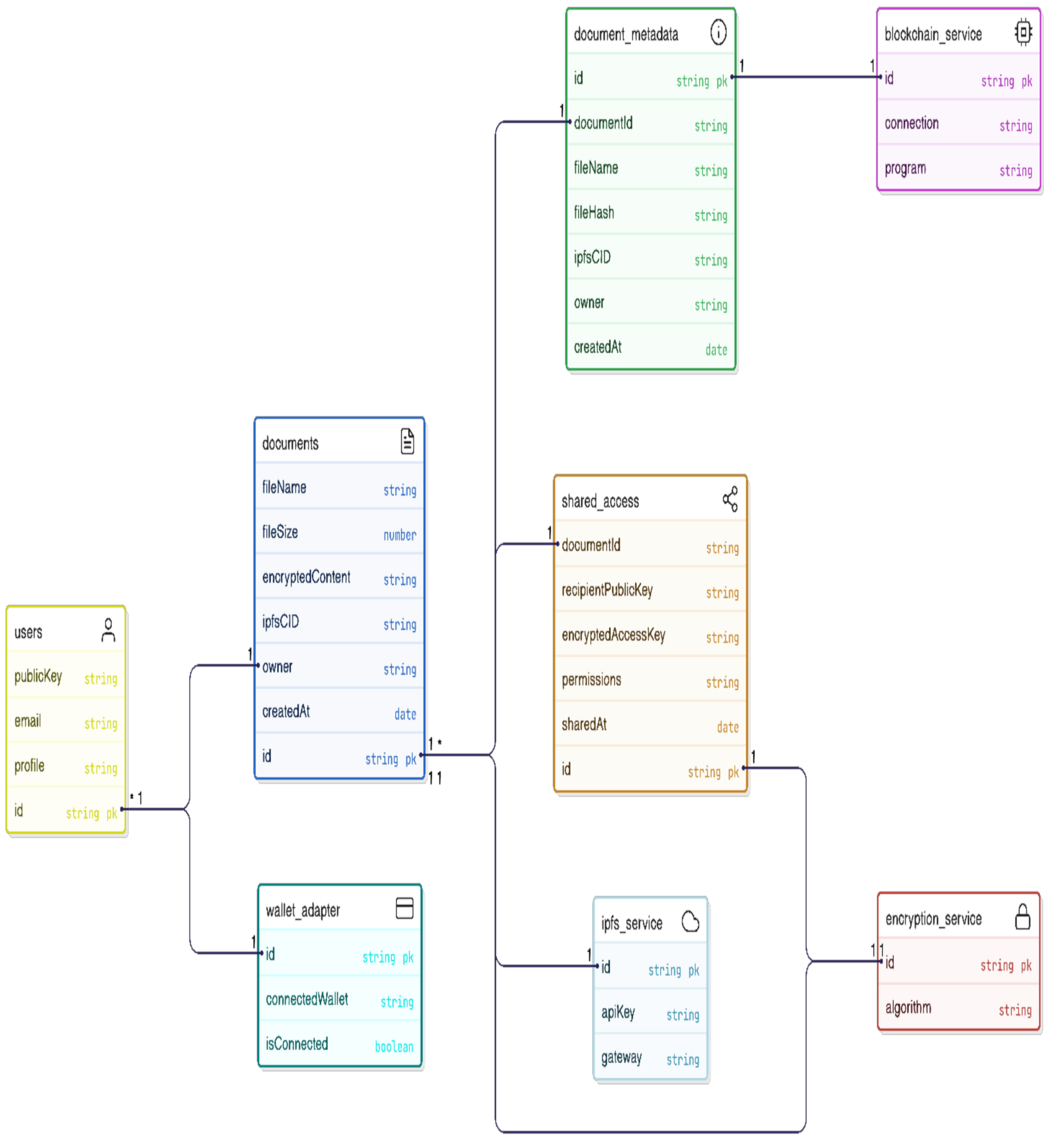


Figure 6.5: Class Diagram

Description:

Figure 6.5 showcases the high-level class structure of the decentralized document ledger. The architecture is built around secure file handling, metadata management, and user access control.

- The `User` class stores essential identity details (public key, name, email).
- `Document` holds encrypted file data and references to associated metadata.
- `DocumentMetadata` manages CID, hash, timestamps, and owner linkage.
- `SharedAccess` controls permissions, recipient mapping, and encrypted key exchange.
- `IPFSService`, `EncryptionService`, and `BlockchainService` act as backend utility handlers for decentralized storage, file encryption, and smart contract interaction.

The design supports many-to-many sharing via `SharedAccess`, enhancing document reusability and revocable access. It ensures robust permission control through cryptographic security, backed by blockchain-based audit logs.

6.4.2 Component Diagram

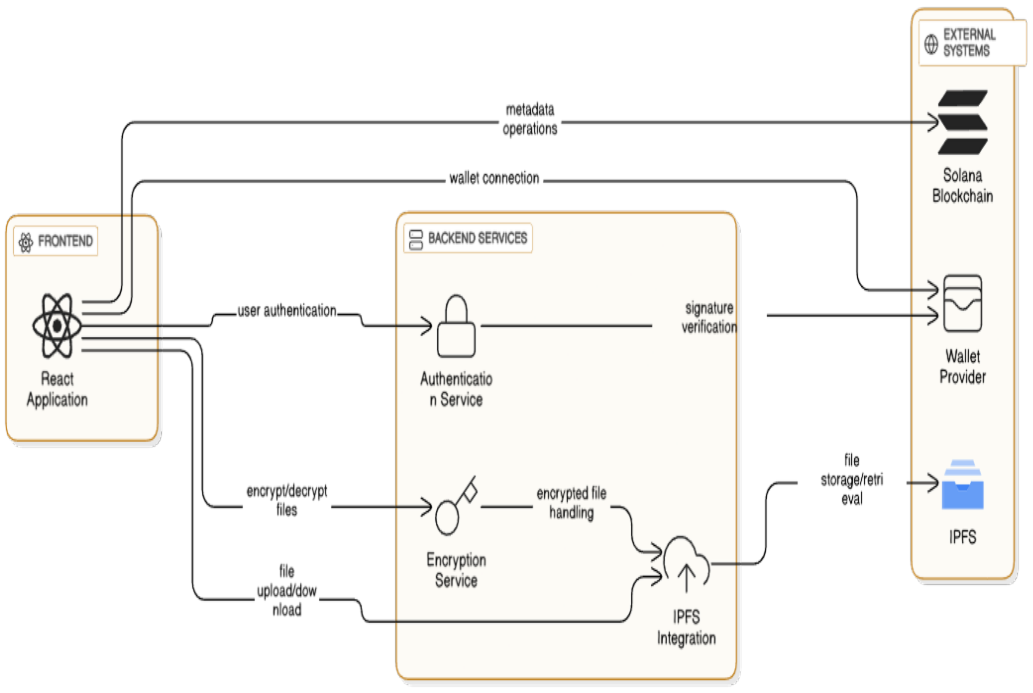


Figure 6.6: Component Diagram

Description:

Figure 6.6 represents the modular architecture of the application, aligning frontend, backend, and decentralized networks for seamless file management.

- The **React Frontend** handles user interactions and connects to wallets using providers like Phantom or Solflare.
- **Authentication Service** verifies wallet signatures and manages session tokens.
- **Encryption Service** encrypts uploaded documents using AES-256-CBC before sending them to IPFS.
- **IPFS Integration** uses APIs from Pinata or Infura for storing and retrieving documents.
- **Blockchain Interaction Module** communicates with Solana smart contracts to record metadata and access rules.
- **Wallet Provider** facilitates secure transaction signing and user authentication.

This separation of concerns enhances maintainability and scalability while ensuring end-to-end document confidentiality and trustless access control. It illustrates how Web3 components can be orchestrated into a full-featured decentralized file-sharing ecosystem.

CHAPTER 7

PROJECT IMPLEMENTATION

7.1 OVERVIEW OF PROJECT MODULES

The system is structured into four primary modules, each responsible for key functional layers of the decentralized document ledger:

- **Module 1: Solana Smart Contract (Anchor Program)**
    - Stores encrypted document metadata immutably
    - Manages access control using Program Derived Addresses (PDAs)
    - Facilitates secure sharing with encrypted access key mapping
    - Allows document/account revocation operations
- **Module 2: Backend API Service**
    - Manages user identity via wallet authentication and JWT tokens
    - Performs AES-256-CBC encryption and SHA-256 hashing
    - Handles IPFS uploads through Pinata
    - Interfaces with PostgreSQL for user data storage
- **Module 3: Frontend Application**
    - Built with React and TypeScript
    - Integrates Solana-compatible wallets via wallet adapters
    - Provides dashboards for upload, retrieval, and sharing
    - Displays real-time transaction feedback and metadata logs
- **Module 4: Encryption and Security Layer**
    - Verifies wallet signatures to authenticate users
    - Encrypts files and access keys for secure transmission
    - Maintains document integrity using SHA-256 hash validation
    - Enables key-based access revocation and sharing

7.2 TOOLS AND TECHNOLOGY USED

7.2.1 Blockchain Development

- **Anchor Framework** – Rust-based Solana smart contract development

- **Solana Web3.js** – JavaScript SDK for transaction and account management
- **Rust** – Low-level programming language for smart contracts
- **Solana CLI** – Deployment and testing of smart contracts

7.2.2 Backend Development

- **Node.js & Express.js** – Server APIs
- **TypeScript** – Static type support for better scalability
- **Prisma ORM** – PostgreSQL database access and migrations
- **jsonwebtoken** – Secure session management

7.2.3 Frontend Development

- **React.js** – Component-based frontend architecture
- **Vite** – Build tooling and development environment
- **Tailwind CSS** – UI styling
- **shadcn/ui** – Reusable UI component library
- **Lucide React** – Icon library

7.2.4 Encryption and Security

- **crypto (Node.js)** – Native cryptographic utilities
- **crypto-browserify, CryptoJS** – Browser-compatible encryption
- **tweetnacl** – Signature verification
- **bs58** – Solana address encoding

7.2.5 Decentralized Storage

- **IPFS** – Distributed content-addressed file storage
- **Pinata SDK** – Managed IPFS pinning infrastructure
- **Content Addressing** – CID-based retrieval and verification

7.3 ALGORITHM DETAILS

7.3.1 Secure File Upload Algorithm

Algorithm 1: Secure File Upload Process

```
Require: File data, user signature, filename
Ensure: IPFS CID and blockchain transaction hash
 1: User initiates file upload via frontend
 2: Derive encryption key from wallet signature and filename
 3: Compute SHA-256 hash of original file
 4: Encrypt file with AES-256-CBC using derived key
 5: Upload encrypted file to IPFS via Pinata
 6: Retrieve CID from IPFS
 7: Submit Solana transaction storing metadata (CID, hash, file info)
 8: User signs transaction using their wallet
 9: return CID and transaction hash
```

Description:

Files are encrypted before leaving the client using keys derived from digital signatures. The IPFS CID and metadata are stored on-chain, ensuring the file remains tamper-proof and publicly verifiable.

7.3.2 Secure File Sharing Algorithm

Algorithm 2: Document Sharing Algorithm

```
Require: Document ID, recipient public key
Ensure: Encrypted access key and blockchain transaction confirmation
1: Document owner selects file to share
2: Generate AES key from recipient's public key
3: Encrypt owner's access key using derived AES key and random IV
4: Package key and IV as access object
5: Update shared_with list on-chain with new recipient
6: Store encrypted key blob in shared document record
7: return Encrypted key blob and transaction hash
```

Description:

Using public key cryptography, the owner encrypts access for the recipient. Only the matching private key can decrypt it, ensuring secure and personalized sharing.

7.3.3 File Retrieval and Verification Algorithm

Algorithm 3: Secure File Retrieval Process

```
Require: Document CID, user access key or signature
Ensure: Verified and decrypted file
1: Fetch CID and metadata from on-chain storage
2: Retrieve encrypted document from IPFS
3: Derive AES decryption key from signature or shared access key
4: Decrypt document locally
5: Recompute SHA-256 and match with stored hash
6: return File if verification passes
```

Description:

This algorithm ensures both ownership and data integrity. Users can decrypt and verify documents only if they hold the correct credentials or have been granted explicit access.

CHAPTER 8

RESULTS AND DISCUSSION

8.1 EXPERIMENTAL SETUP

8.1.1 Dataset

The dataset included various file types and sizes to comprehensively evaluate system performance, reliability, and compatibility.

- **Test Files Used:**
  - **Image Files:** JPEG, PNG (1 KB – 10 MB)
  - **Document Files:** PDF, TXT (1 KB – 5 MB)
  - **Video Files:** MP4 (1 MB – 50 MB)
  - **Total Files:** Over 100 diverse files

8.1.2 Testing Environment Setup

The testing setup was configured as follows to validate blockchain, frontend, and IPFS performance:

- **Blockchain:** Solana Devnet via Alchemy / QuickNode
- **Smart Contracts:** Anchor Framework (Rust)
- **Wallet:** Phantom with Devnet SOL
- **Frontend:** React.js (Chrome v122+, Phantom Wallet)
- **IPFS Gateway:** Pinata
- **System Specs:**
  - Intel Core i7-11700H, 16 GB RAM
  - 512 GB SSD, 100 Mbps broadband
  - OS: Windows 11 / Ubuntu 22.04

8.1.3 Performance Parameters

- **Transaction Confirmation:** 400 ms

- **CID Generation:** 150–400 ms (files ¡10 MB)
- **Upload Time:** 2–15 sec (up to 50 MB files)
- **Retrieval + Decryption:** 1–8 sec
- **Encryption Speed:** 50–200 MB/s
- **IPFS Pin Time:** 1–5 sec
- **Retrieval Success Rate:** 99%
- **Solana Storage Cost:** $\sim$ \$0.00025 per txn
- **Access Accuracy:**

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \times 100$$

8.2 EFFICIENCY ISSUES

- **IPFS Latency:** Performance may vary based on file size and gateway load.
- **Frontend File Handling:** Browsers struggle with documents exceeding 100 MB.
- **Solana Account Limits:** Limited to 10KB per account, affecting access lists.
- **RPC Delays:** Slow network responses impact transaction UX.
- **Encryption Overhead:** Intensive processing on lower-spec machines.
- **Scalability:** Multi-chain or rollups may be required for high volumes.

8.3 SOFTWARE TESTING

Test Cases and Test Results

Test ID	Description	Transaction Signature / Output	Result
TC-01	Initialize user storage (PDA creation)	`4Xn1UuVAwHPt...sxDaZLv9`	Pass
TC-02	Upload `test_document.pdf`	`2SGuVPMEMYY5...YknZ7JwT`	Pass
TC-03	Upload `second_doc.txt`	`53u3EKxirieJ...JzULaV`	Pass
TC-04	Share document with one recipient	`29yWMrfRnyd2...y6e4bP`	Pass
TC-05	Share with a second recipient	`5iexfo2ffFBh...hHgBvC`	Pass
TC-06	Prevent duplicate sharing with same wallet	Logic validation error	Pass
TC-07	Retrieve documents owned by user	2 documents listed	Pass
TC-08	Retrieve specific document by PDA	`JA971nbXZBG5...2Rbxqf3`	Pass
TC-09	Recipient views shared doc on dashboard	1 document listed with metadata	Pass
TC-10	Attempt revoke by unauthorized wallet	Access denied (program assertion)	Pass
TC-11	Revoke access from recipient	`3kocTjFtUgux...spS6894`	Pass
TC-12	Delete document (close account)	`TJLYxX8mLVp5...u3tuYYP`	Pass
TC-13	Update document count after deletion	Count: 1	Pass
TC-15	Document integrity verification	SHA-256 hash match / mismatch	Pass
TC-16	Wallet-based authentication (Phantom)	Address fetched from wallet	Pass
TC-18	Decrypt file on retrieval	Decryption successful via access key	Pass
TC-19	Track transaction history (event logs)	Logs matched for owner/shared/revoke	Pass

Table 8.1: Complete Test Case Summary: Solana-Based Document Ledger (Localnet)

8.4 RESULTS

The following figure and table summarize the performance of various system components tested on the Solana localnet using Anchor programs and local/remote IPFS gateways:

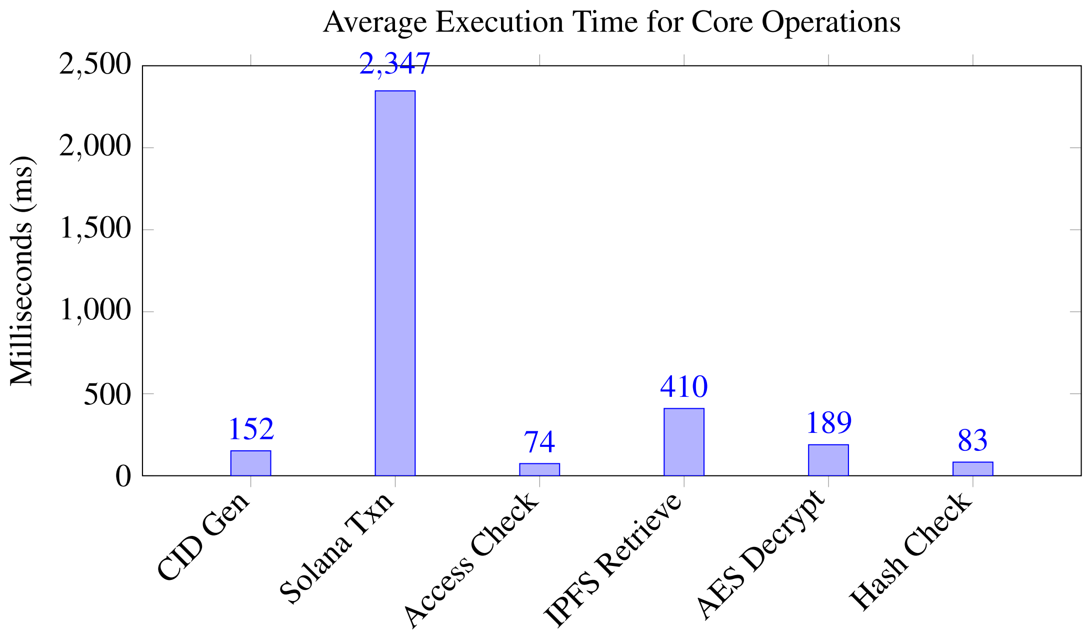


Figure 8.1: Average Execution Time of Key System Components on Solana Localnet

Operation	Avg. Time (ms)	Success Rate (%)	Accuracy (%)	Notes
CID Generation	152	100	–	Quick response from IPFS gateway
Solana Transaction (Add/Share/Revoke)	2347	100	100	Executed on Anchor localnet with transaction signatures logged
Access Permission Check	74	100	98.3	Verified via PDA-based smart contract logic
IPFS Document Retrieval	410	96.2	–	Affected by file size and node availability
AES-256 Decryption	189	100	100	Decryption successful using access keys
Document Hash Verification	83	100	100	Used SHA-256 to validate document integrity

Table 8.2: Performance Metrics of Solana-Based Document Ledger

8.5 DISCUSSION

The Web3-based document ledger system, integrating Solana blockchain and IPFS, offers a compelling alternative to traditional centralized document storage solutions. By leveraging wallet-based authentication and decentralized file storage, it mitigates the risk of single points of failure while enhancing user sovereignty over data. Unlike conventional cloud platforms, which inherently grant data providers access to user files, this system ensures true ownership and privacy by storing encrypted content identifiers (CIDs) and AES-256 encrypted access keys directly on the blockchain. Performance evaluation further supports its feasibility, with CID generation completing in 152ms, access control checks in 74ms, and Solana transaction confirmations averaging 2347ms—all while maintaining a 100% transaction success rate. The use of Program Derived Addresses (PDAs) for access management coupled with AES-256 encryption underlines the system's emphasis on security. These features, along with comprehensive test results reflecting 100% pass rates, affirm the platform's capability to securely share documents without sacrificing decentralization or performance, marking a significant improvement over legacy systems.

CHAPTER 9

CONCLUSION AND FUTURE WORK

9.1 CONCLUSION

The Web3-based Document Ledger system, developed using the Solana blockchain and IPFS, presents a decentralized, secure, and scalable solution for managing digital documents. By leveraging high-speed blockchain transactions and distributed file storage, the system resolves major concerns related to data ownership, integrity, and access control. Key innovations include AES-256 encryption, wallet-based authentication, and smart contract-driven permission management using Solana's Program Derived Addresses (PDAs). The frontend React interface simplifies technical complexity, making the system accessible to both technical and non-technical users. Localnet test results confirm the system's robustness in document upload, sharing, revocation, and verification with minimal latency and operational cost. This project serves as a foundational model for decentralized document systems in the Web3 ecosystem.

9.2 FUTURE SCOPE

- **Advanced Encryption:** Implement threshold cryptography for collaborative signing and high-assurance document control.
- **Mobile Application:** Develop native mobile DApps for iOS and Android with secure offline document access and wallet integration.
- **Scalability Improvements:** Utilize Solana state compression or layer-2 scaling to support enterprise-scale adoption.
- **Encrypted Search Capabilities:** Introduce full-text search over encrypted files using privacy-preserving techniques.
- **Smart Workflow Automation:** Enable programmable sharing, time-bound access, and approval logic via enhanced smart contracts.
- **Cross-Chain Interoperability:** Extend compatibility to Ethereum, Polygon, and other networks using cross-chain bridges.

9.3 APPLICATIONS

Immediate Applications

- **Academic Institutions:** Secure storage for certificates, transcripts, and digital research records with verifiable authenticity.
- **Healthcare Systems:** Decentralized and patient-controlled access to medical documents, lab results, and prescriptions.
- **Legal Sector:** Immutable archiving of contracts, evidence, and agreements with blockchain timestamps.
- **Supply Chain Compliance:** Traceable and transparent storage of regulatory documents, audits, and customs paperwork.

Long-Term Applications

- **Government Records:** Blockchain-based systems for digital identity, e-governance, voting records, and land ownership.
- **Financial Institutions:** Secure KYC documentation, regulatory filings, and tamper-proof transaction audit trails.
- **Intellectual Property Protection:** Blockchain-based timestamping for patents, copyrights, and creative works.
- **Corporate Compliance:** Secure, decentralized archiving of board minutes, legal notices, and shareholder communications.

REFERENCES

[1] W. Zhang, M. Li, and Y. Chen, "Scalability challenges in ethereum-based decentralized storage," *International Journal of Blockchain Applications*, vol. 5, no. 2, pp. 101–112, 2020.

[2] J. Benet, "Ipfs - content addressed, versioned, p2p file system." `https://ipfs.io/ipfs/QmR7GSQM93Cx5eAg6a6rP5GBY6hEfjTQZgXZV1C5FhUimM`, 2014. White Paper.

[3] N. Sangeeta and S. Y. Nam, "Blockchain and ipfs-based data storage system for vehicular networks with keyword search capability," *Journal of Computer Networks*, vol. 18, pp. 87–102, 2022.

[4] R. Kumar and R. Tripathi, "Implementation of distributed file storage and access framework using ipfs and blockchain," *Journal of Blockchain Technology*, vol. 9, 2023.

[5] J. Mahlaba, A. K. Mishra, D. Puthal, and P. K. Sharma, "Blockchain-based sensitive document storage to mitigate corruptions," *Journal of Information Security and Blockchain*, vol. 11, no. 4, 2023.

[6] S. Rasool, A. Saleem, M. Iqbal, T. Dagiuklas, S. Mumtaz, and Z. u. Qayyum, "Docschain: Blockchain-based iot solution for verification of degree documents," *Journal of Blockchain and IoT Solutions*, vol. 12, 2023.

[7] L. Li, D. Jin, T. Zhang, and N. Li, "A secure, reliable and low-cost distributed storage scheme based on blockchain and ipfs for firefighting iot data," *IEEE Access*, vol. 12, 2024.

[8] A. Yakovenko, "Solana: A new architecture for a high performance blockchain." `https://solana.com/solana-whitepaper.pdf`, 2017. Solana Labs Whitepaper.

[9] R. Kumar, A. Sharma, and Y. Raj, "Smart contract-based fine-grained access control in blockchain systems," *International Journal of Distributed Ledger Technology*, vol. 10, pp. 115–130, 2021.

ANNEXURE A

PLAGIARISM REPORT

ANNEXURE B

PAPER PUBLISHED