CHAPTER 1: INTRODUCTION

1.1 Problem and Solution Definition

The current methods for document sharing—uploading files to college portals, job application sites, or centralized storage—result in a complete loss of user control and transparency. This reliance on centralized custodians creates significant security and privacy vulnerabilities that cannot be resolved by conventional cloud solutions. Key problems driving the need for DSMS include:

- Loss of Control: Users don't know where the file is physically stored, who manages the server, and lose all ability to track or modify access permissions once the file leaves their device.
- Data Misuse and Retention: Third parties, having assumed complete custodianship, can keep, copy, or misuse the file indefinitely. The user cannot retroactively retrieve the document or demand its deletion, leading to permanent data exposure.
- Lack of Audit Trail: There is no verifiable proof of who accessed the file, when, or for what purpose. This absence of transparency undermines user trust and fails to meet modern compliance standards. Furthermore, logs, if they exist, can be easily altered by the platform administrator.
- Vendor Lock-in and Censorship Risk: Users are tied to a specific service provider, and their files are susceptible to removal or account suspension based on the host's arbitrary policies.

DSMS addresses this systemic failure by placing the user as the sole owner and decision-maker who dictates the terms of access (time limit, view-only permission, download restrictions) and can revoke access instantly at any time.

DSMS is building the future of safe, transparent document sharing by giving users full control of their data. The system allows users to upload documents once to their own secure space, encrypting them before transit. They can then securely share them via a unique link with highly customizable limits (e.g., specifying how long the link is active or how many times the file can be viewed).

The platform is inherently designed to be censorship-resistant and tamper-proof because files are stored in pieces across a global, decentralized network utilizing IPFS (InterPlanetary

File System). This architecture ensures that no single entity—not even the DSMS platform itself—has complete control over the stored document. The solution is presented through the Trusted Document Bridge, which enables a secure, temporary handshake between the user's data and external requesting websites. This is achieved using a robust technology stack that includes Node.js/Express.js for the core API, Firebase for rapid authentication and metadata, and IPFS for decentralized storage.

1.2 Literature Review

Traditional document management systems (e.g., Google Drive, Dropbox, and institutional portals) are built upon centralized architectures, where a single organization manages all data,permissions,andlogs.

While convenient, this model inherently transfers data ownership and control away from users, making files vulnerable to misuse, unauthorized access, and censorship.

The Decentralized Secure Management System (DSMS) overcomes these limitations by leveraging IPFS (InterPlanetary File System) and blockchain-backed verification mechanisms, fundamentally redefining data custody, access transparency, and digital trust.

Key Comparative Advantages:

1. Data Storage

- Traditional Systems:
    - Store files on centralized corporate servers, fully managed by service providers.
    - Users cannot verify where their data resides or how it is replicated.
    - Vulnerable to data loss, policy-based censorship, and server compromise.
- DSMS:
    - Uses IPFS, a decentralized, content-addressed storage network.
    - Files are fragmented and distributed globally, eliminating a single point of failure.
    - Ownership remains with the user, not the platform.
    - Guarantees tamper-proof and censorship-resistant storage.

2. Sharing Mechanism

- Traditional Systems:
    - Generate static URLs or share links that remain active indefinitely.
    - Offer limited control (only “view” or “edit”) without expiration or watermarking.
- DSMS:
    - Enables granular, rule-based sharing through:

        - Time-limited links (auto-expire after specified duration).
        - Role-limited permissions (e.g., view-only, restricted download).
        - Watermark-enabled sharing (planned feature to prevent redistribution).
    - Provides users a dynamic dashboard to monitor, modify, or revoke access in real time.

3. Access Control

- Traditional Systems:
    - Once shared, users lose the ability to revoke or modify access.
    - Files can continue to circulate without owner consent.
- DSMS:
    - Offers active and continuous control to the user.
    - Access can be instantly revoked, regardless of where the link is shared.
    - Ensures complete traceability and data sovereignty even after the document leaves the dashboard.

4. Tracking and Verification

- Traditional Systems:
    - Maintain internal, editable logs that can be modified by administrators.
    - Users have no independent way to verify access history.
- DSMS:
    - Logs every view, download, and revoke event in real-time.
    - Provides a transparent, verifiable audit trail accessible to users.
    - Future Phase (Phase 2)**:** Plans to secure all access logs via immutable blockchain smart contracts, ensuring data access history is permanently tamper-proof and publicly auditable.

5. Security and Privacy

- Traditional Systems:
    - Files are stored in readable formats on centralized servers.

    - Service providers and internal staff may gain unauthorized visibility.
- DSMS:
    - Encrypts documents before upload (client-side encryption).
    - The Content Identifier (CID) — the file's unique IPFS hash — is securely managed and hidden from unauthorized users.
    - No party, including DSMS itself, can access the document contents without user permission.
    - Enhances confidentiality, integrity, and non-repudiation.

6. Resilience and Independence

- Traditional Systems:
    - Prone to downtime, server outages, and account-based censorship.
    - Users depend entirely on provider infrastructure and terms of service.
- DSMS:
    - Decentralization ensures high redundancy and global availability.
    - Files cannot be censored, deleted, or altered by any single entity.
    - Eliminates vendor lock-in, empowering users with true data portability.

CHAPTER 2: REQUIREMENT SPECIFICATION

2.1 Functional Requirements

Requirement ID	Requirement Statement	Must/Want
FR01	The system shall support user registration and login via both traditional Email/Password (Web2) and a Web3 wallet (e.g., MetaMask).	Must
FR02	The user shall be able to upload documents to the platform, which the system will then upload to IPFS and retrieve the unique Content ID (CID).	Must
FR03	The system shall securely store the file metadata (File Name, CID, Owner, Timestamp) in the Firebase Firestore database.	Must
FR04	The user shall be able to generate a unique, secure access link for sharing any uploaded document.	Must
FR05	The user shall be able to set granular sharing permissions, including access expiry time and the number of allowed views, when generating the link.	Must
FR06	The user shall be able to revoke access to a shared document at any time, instantly invalidating the access link.	Must
FR07	The system shall log every access attempt (successful or denied) to a shared document, including the viewer's identity and timestamp.	Must
FR08	The user shall have access to a dashboard to view all uploaded documents and their complete, historical access logs.	Must
FR09	The system shall provide a mechanism (Trusted Document Bridge) for external websites to send a formal document request to the user, instead of requiring a blind upload.	Must
FR10	The document viewer page shall display a secure preview of the document (PDF/Image) only if the access link is valid and unexpired.	Must

2.2 Non-Functional Requirements

ID	Requirement Statement
NFR01	Security (Data Integrity)All files shall be encrypted before upload to IPFS, ensuring data is unreadable in transit and at rest without the user's explicit permission.
NFR02	Security (Authentication)The system must implement robust security practices (e.g., Firebase Authentication, JWTs) to protect all API endpoints from unauthorized access.
NFR03	Security (Access Logging)The system shall maintain an immutable, time-stamped log of every document access, ensuring non-repudiation for the owner.
NFR04	Decentralization Document storage must be implemented using IPFS CIDs (Content Identifiers), making files censorship-resistant and accessible via any compliant gateway.
NFR05	Performance The maximum document upload time (up to 10MB file) to IPFS and metadata recording in Firestore shall not exceed 10 seconds under typical network conditions.
NFR06	Reliability The system shall ensure data persistence by pinning uploaded IPFS files using a reliable service (e.g., Pinata or web3.storage).
NFR07	Scalability The backend architecture (Node.js/Express.js) and database (Firestore) shall be designed to handle up to 1,000 active users and 5,000 document access events per day without performance degradation**.**
NFR08	Usability The user interface (React.js/Tailwind CSS) shall be fully responsive and intuitive, providing an optimal experience on desktop, tablet, and mobile devices**.**
NFR09	Maintainability The code base must adhere to clean coding standards, with comprehensive inline comments and documentation (per Teammate 3's role) to facilitate future development and debugging.

CHAPTER 3: REQUIRED COMPONENTS

3.1 Hardware Requirements

Component	Description
Development Machine (Laptop/PC)	Main machine for coding, unit testing, and running local development servers (Node.js/React). Requires sufficient processing power and memory (e.g., 8GB RAM, modern processor) to handle IDEs, debugging tools, and multiple browser instances for component testing.
Internet Connectivity	Stable, high-speed connection is essential for interacting with external services: pushing code to GitHub, accessing the Firebase database, uploading and retrieving files via the IPFS gateway, and deploying code to Vercel/Railway.
User Access Device	Any device (Desktop, Tablet, Mobile) with a modern web browser capable of supporting the React frontend and Web3 integration (via MetaMask extension/app). This ensures broad accessibility for all users.
MetaMask-enabled Device (Optional)	A device capable of running the MetaMask wallet extension or mobile app, required for users choosing the optional Web3 wallet login feature.

3.2 Software Components

1.Front-End Components

Component	Purpose
Development Languages: HTML5 & CSS3	HTML5 provides the core structure and content organization. **CSS3** (including Tailwind CSS) is used for styling the components and building a clean, modern, and natively responsive interface across all device sizes.
Programming Language: Vanilla JavaScript	The core language used for all client-side logic, including user interaction, form validation, dynamic content updates, and direct communication with the backend APIs.
User Interface Libraries: Tailwind CSS	A utility-first CSS framework used for rapid, responsive styling of the interface (Login, Dashboard, Viewer), ensuring a modern and consistent look.
Client-Side Libraries: Fetch API / Axios	Used for making asynchronous HTTP requests from the client to the Node.js/Express APIs to handle user data, file uploads, and link validation.

2 Back-End, Database, and Decentralized Services

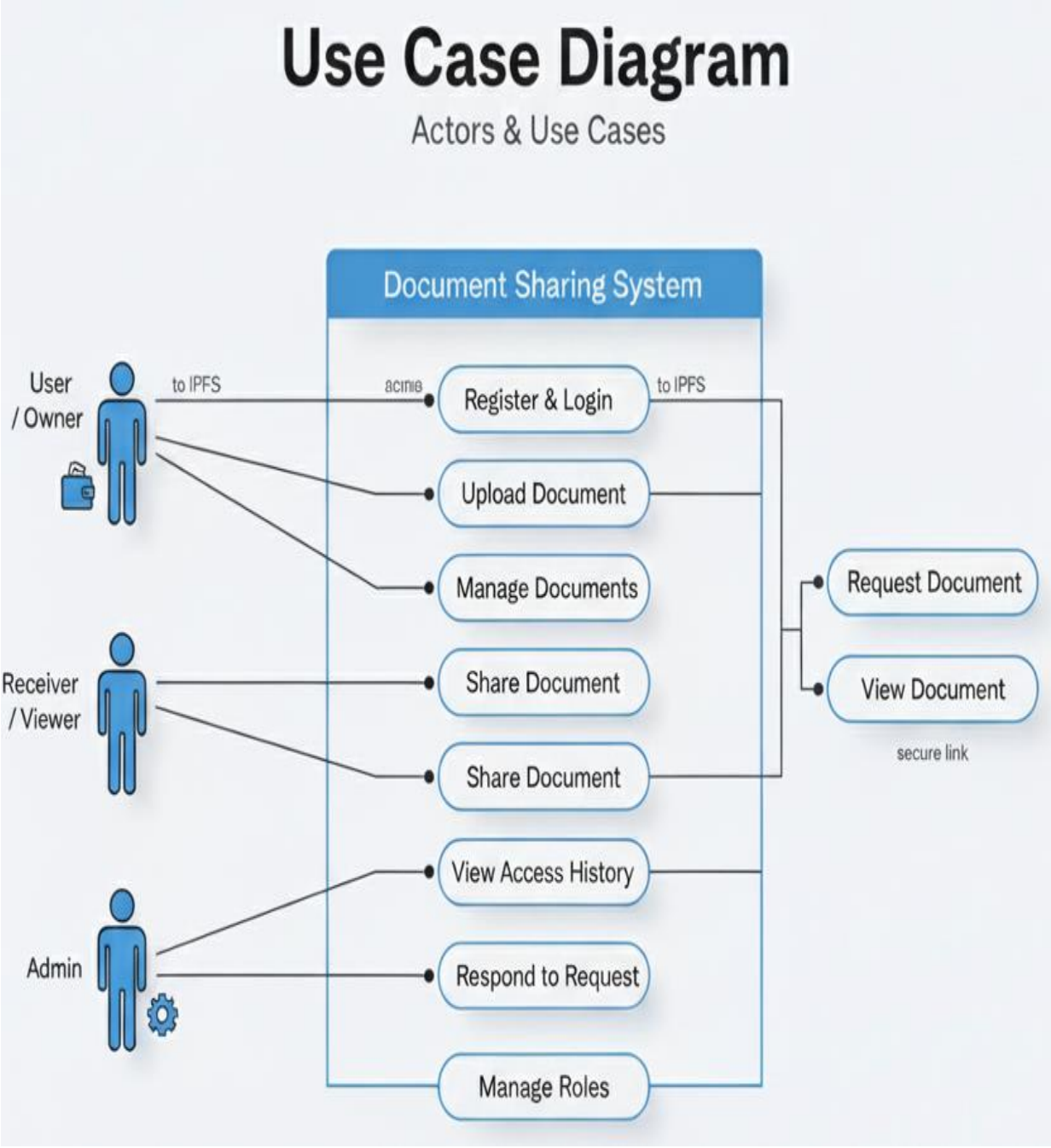
Component	Purpose
API Framework: Node.js & Express.js	Node.js is the runtime environment, and Express.js is the framework used to build the fast and scalable REST APIs that manage user roles, sharing logic, access logging, and communication between the frontend and Firestore/IPFS.
Database: Firebase Firestore (NoSQL)	A scalable, flexible NoSQL database used to store all file metadata**,** user profiles, role assignments, unique share links, and real-time access logs**.**
Decentralized Storage: IPFS Integration	Used via SDKs (web3.storage or Pinata**)** to handle file uploads. Files are addressed by their Content Identifier (CID)**,** ensuring data immutability and decentralization, moving storage ownership away from centralized servers.
Authentication: Firebase Authentication	Provides secure, industry-standard authentication for email/password login and manages user sessions (e.g., JWT if required), crucial for securing API endpoints.
eb3 Integration: MetaMask & Ethers.js	MetaMask acts as the decentralized identity provider, and Ethers.js is the JavaScript library used to connect the application to the user's wallet for the optional Web3 login and to interact with Smart Contracts (Phase 2).
Smart Contracts (Phase 2): Solidity	The programming language used to write the smart contract logic for storing immutable access logs on the blockchain (e.g., Polygon Mumbai Testnet).

3 Deployment and Management Tools

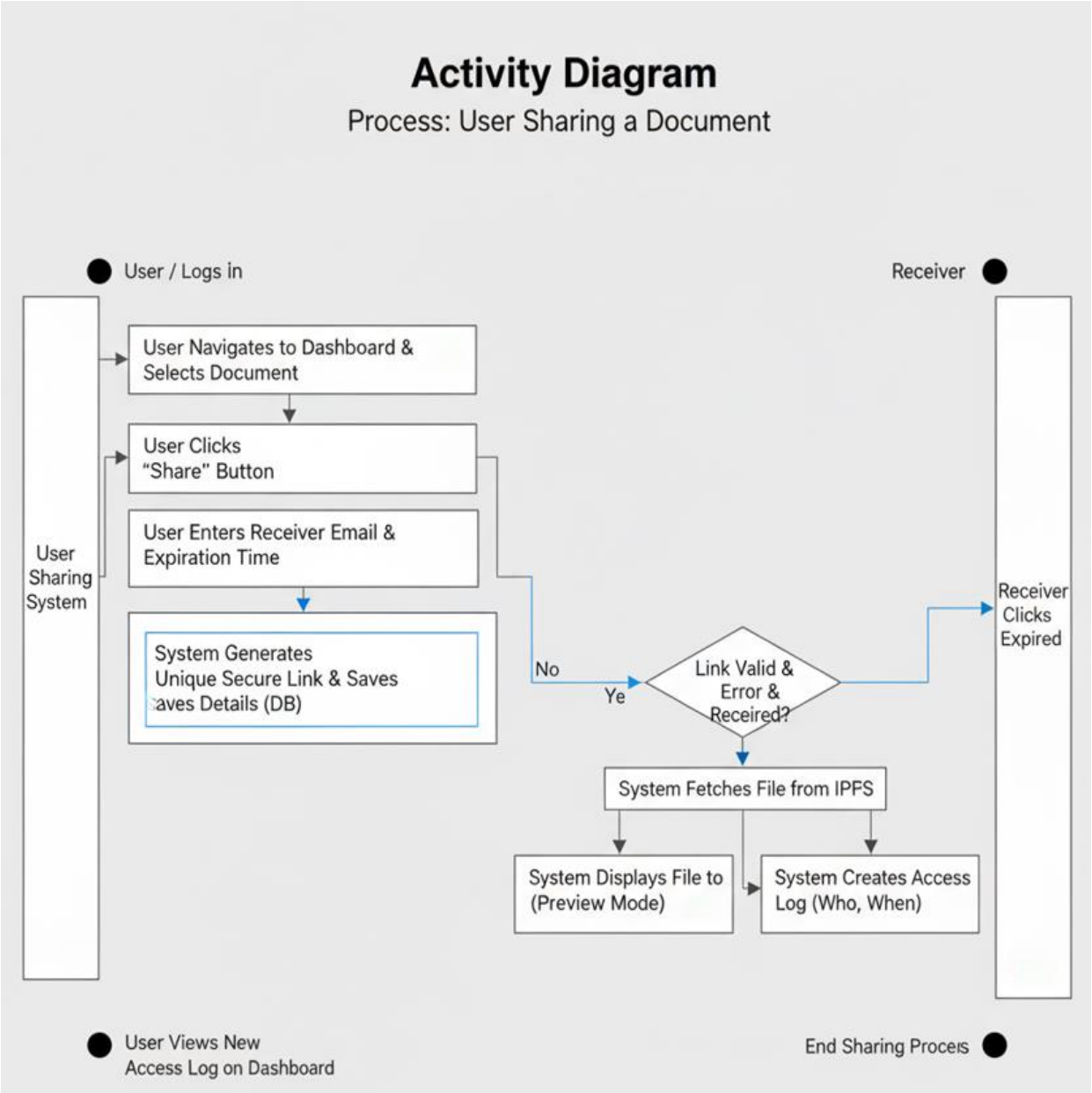
Component	Purpose
Frontend Deployment: Vercel	Used for rapidly deploying the static HTML/CSS/JS application globally via its CDN (Content Delivery Network), ensuring minimal latency and fast load times.
Backend Deployment: Railway / Render	Used for deploying the Node.js/Express REST API layer, providing continuous deployment and auto-scaling capabilities.
Version Control: Git & GitHub	Essential for collaborative development, managing code versions, feature branching, and running automated deployment actions.

CHAPTER 4: PROJECT DESIGNING

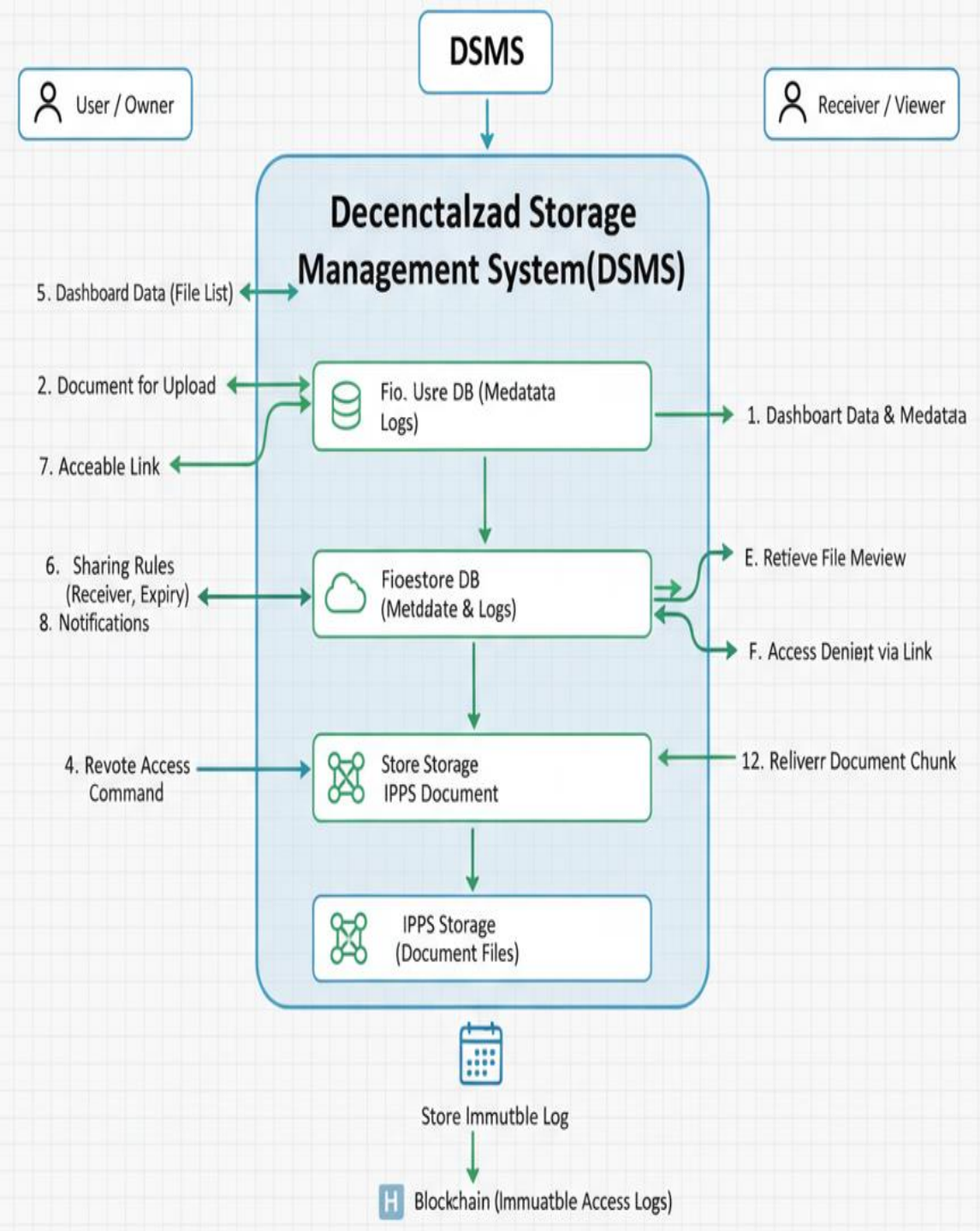
1.1 Use Case Diagram



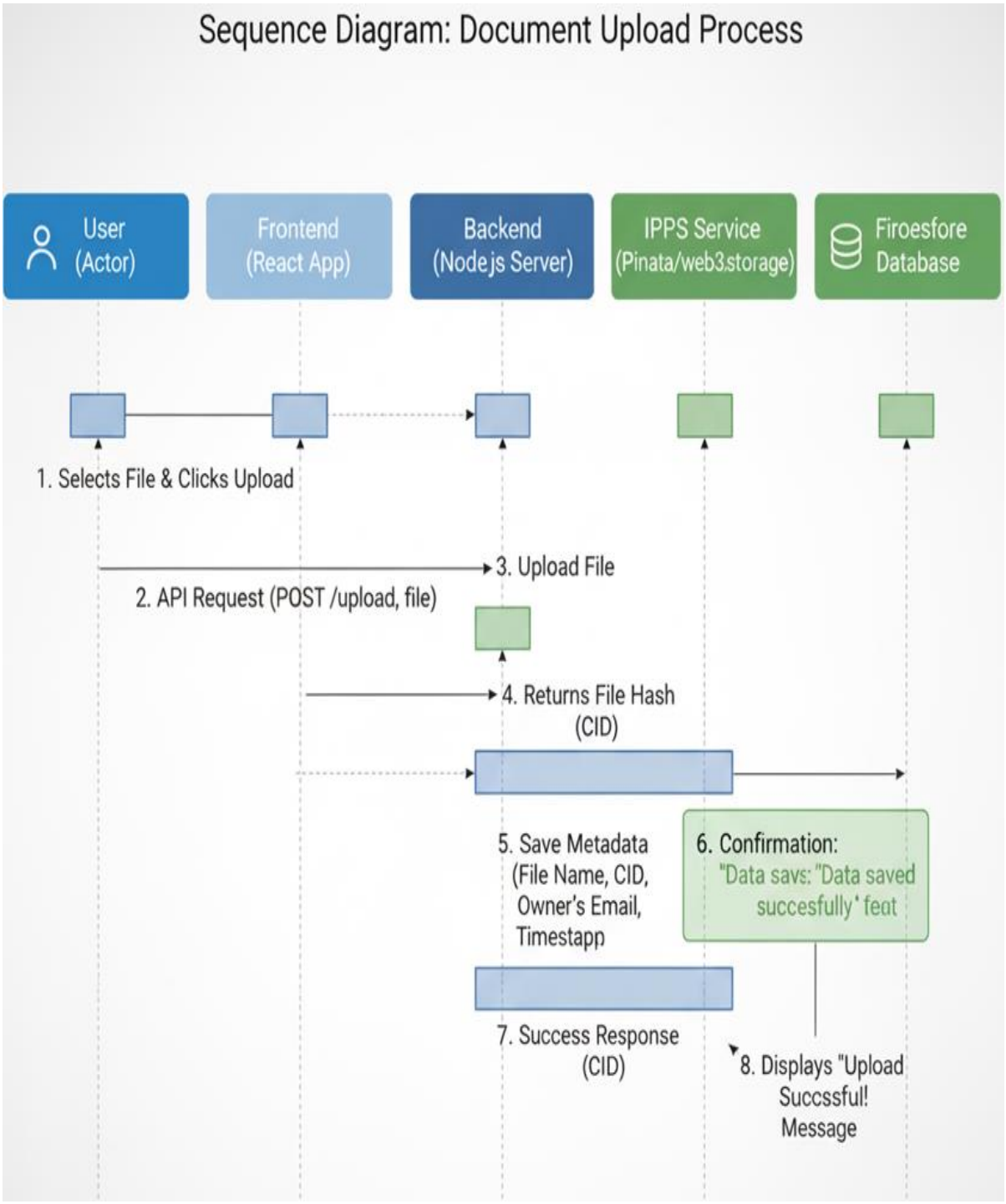
1.2 Activity Diagram



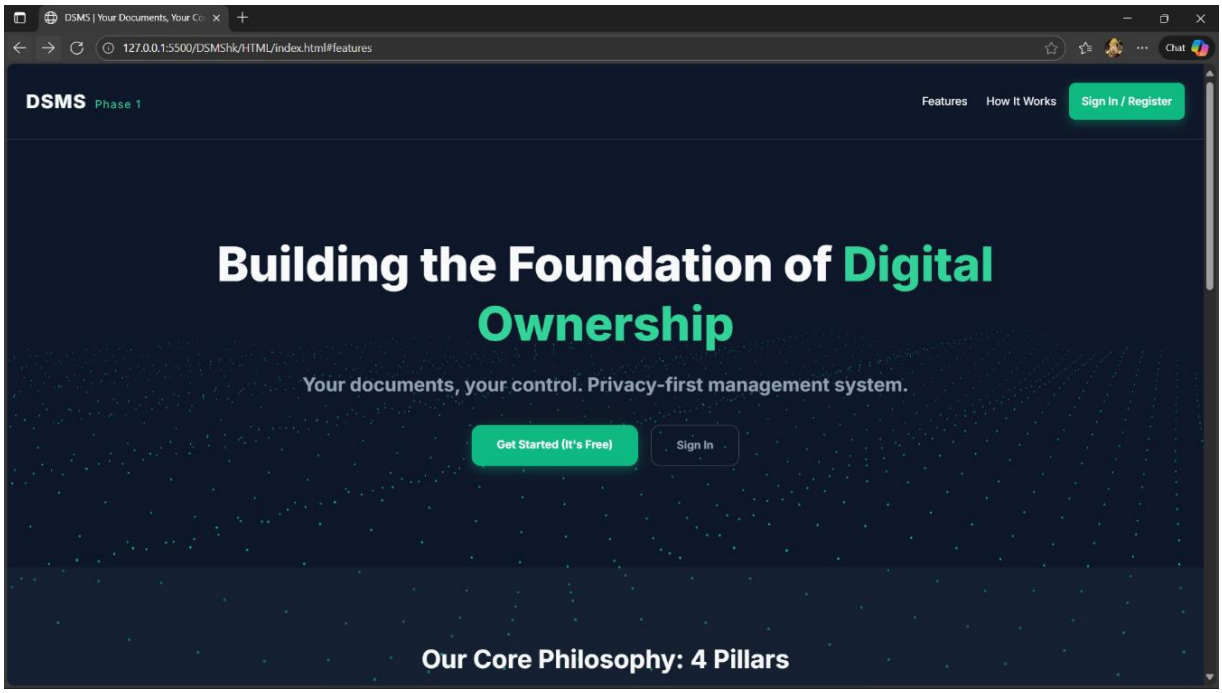
1.3 Fow Diagram

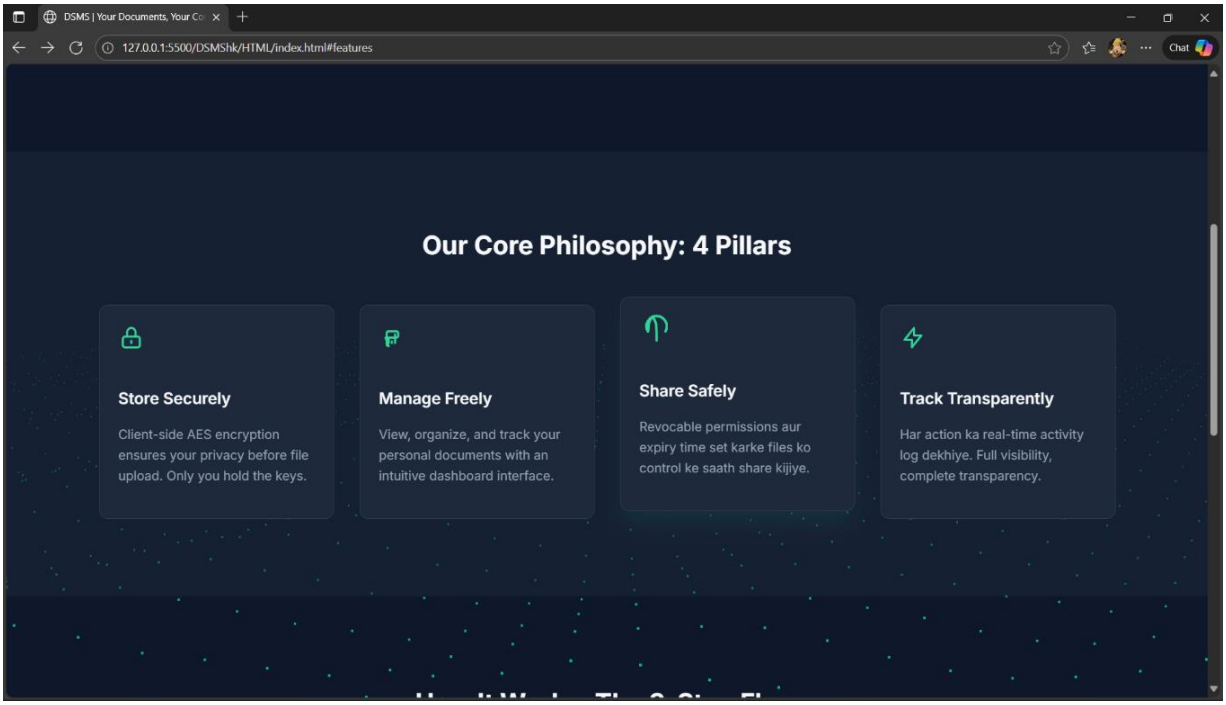


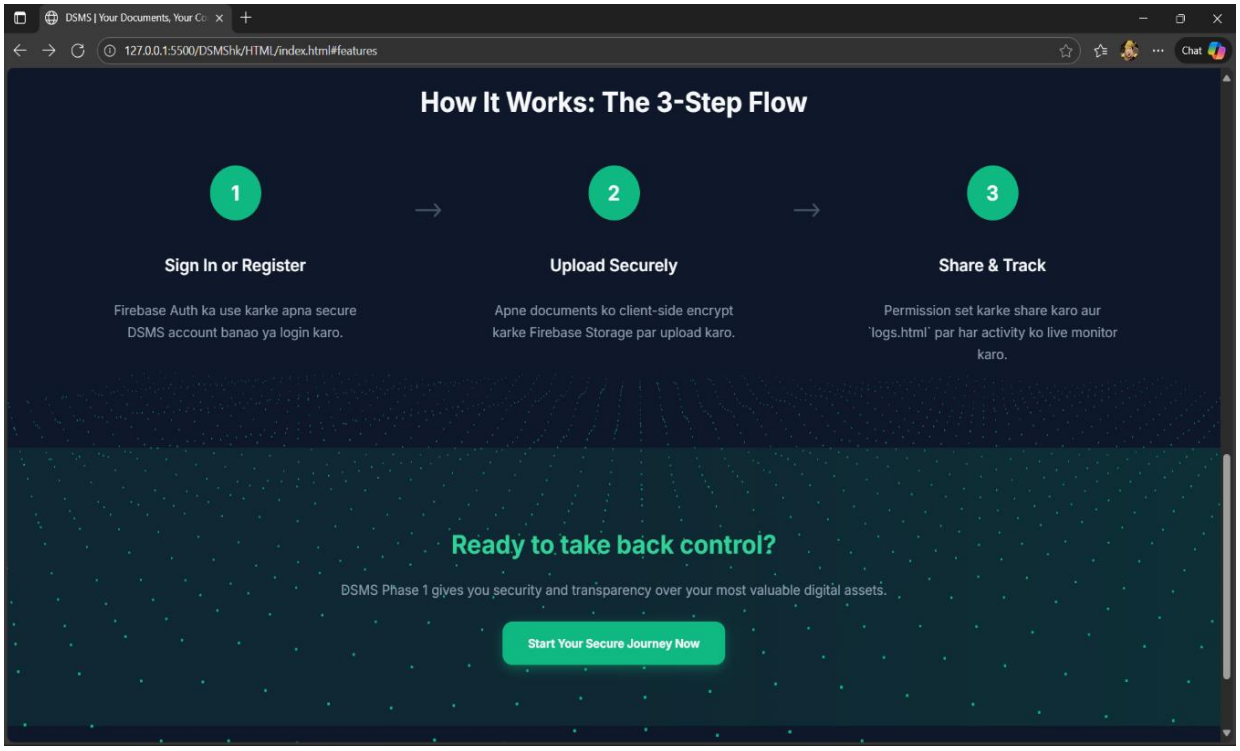
1.4Sequence Diagram

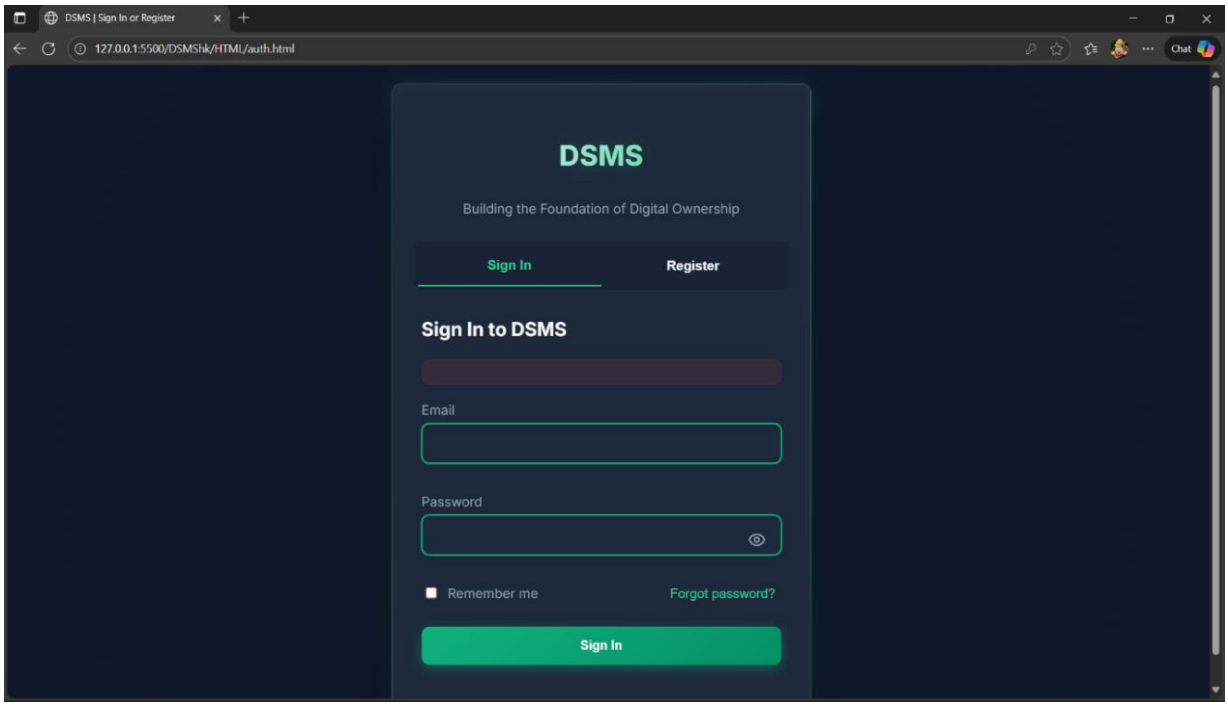


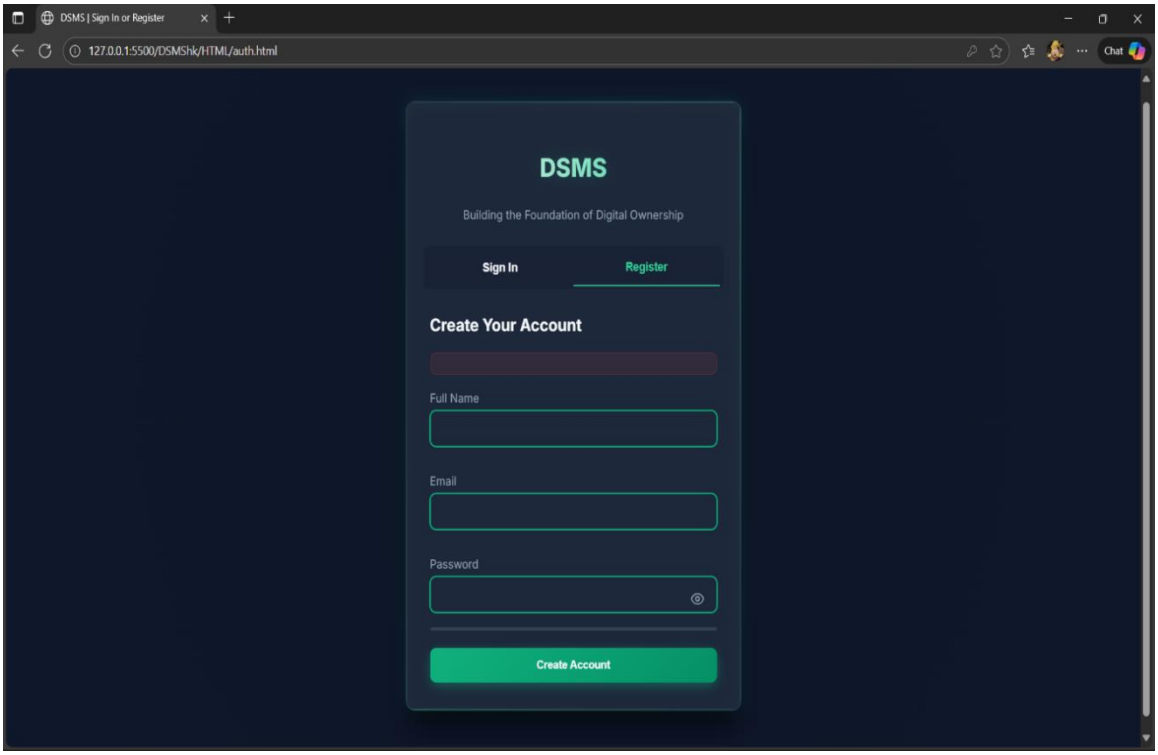
CHAPTER 5: IMPLEMENTATION DIAGRAMS

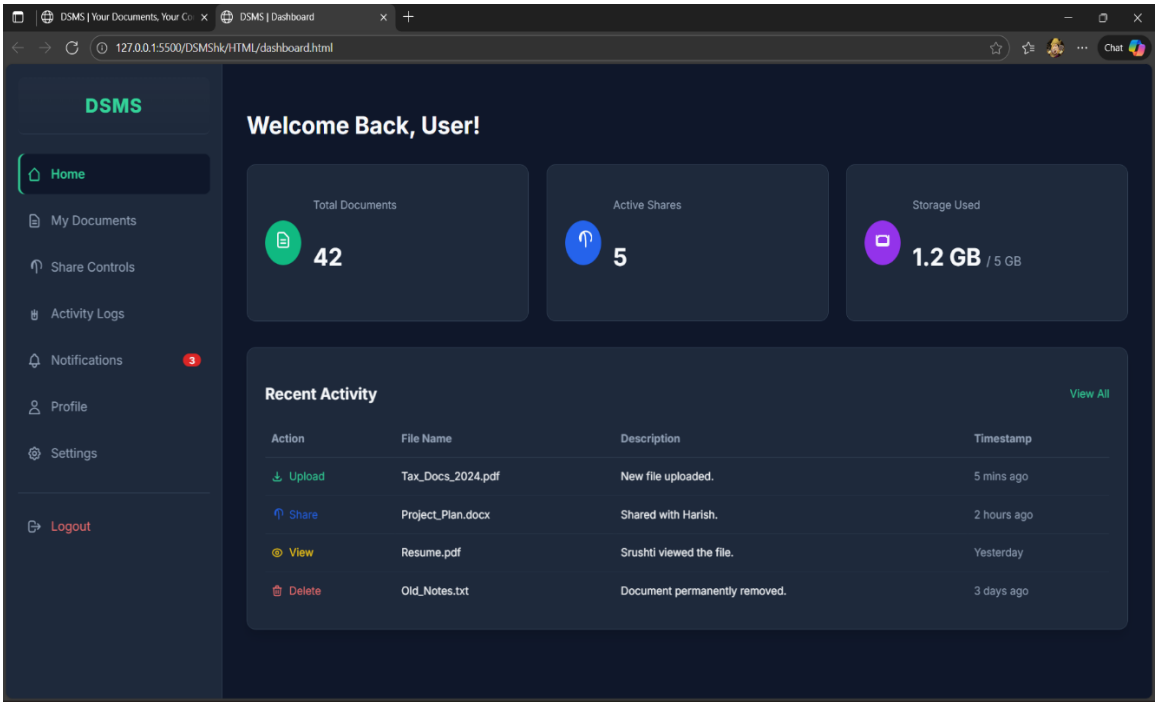


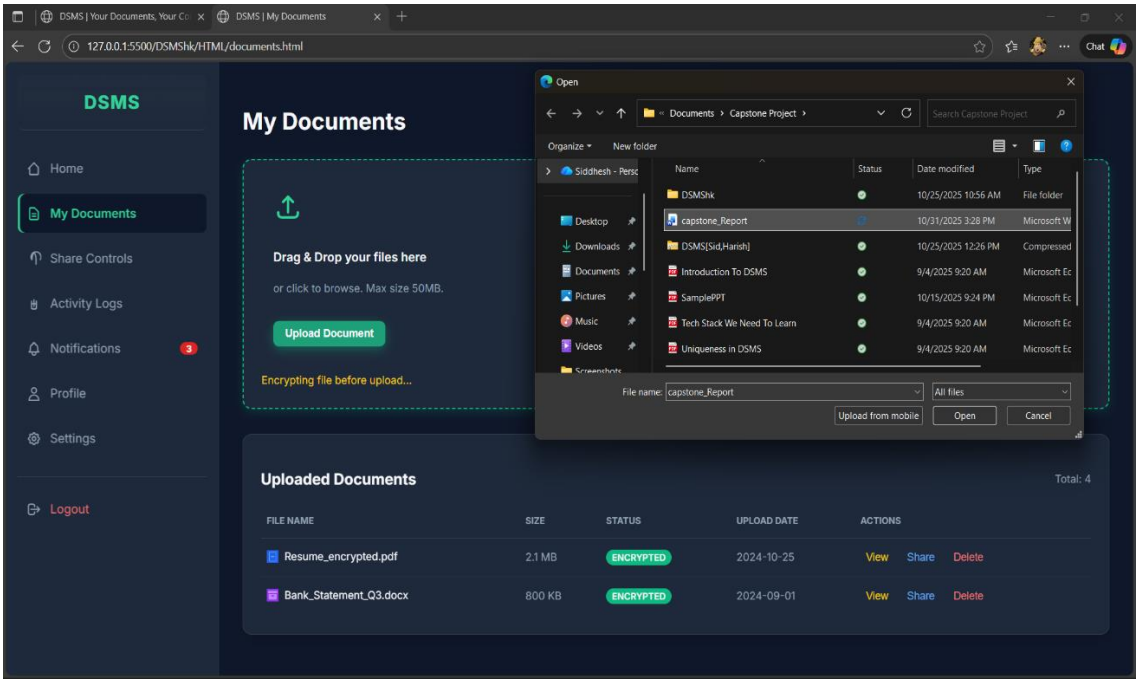


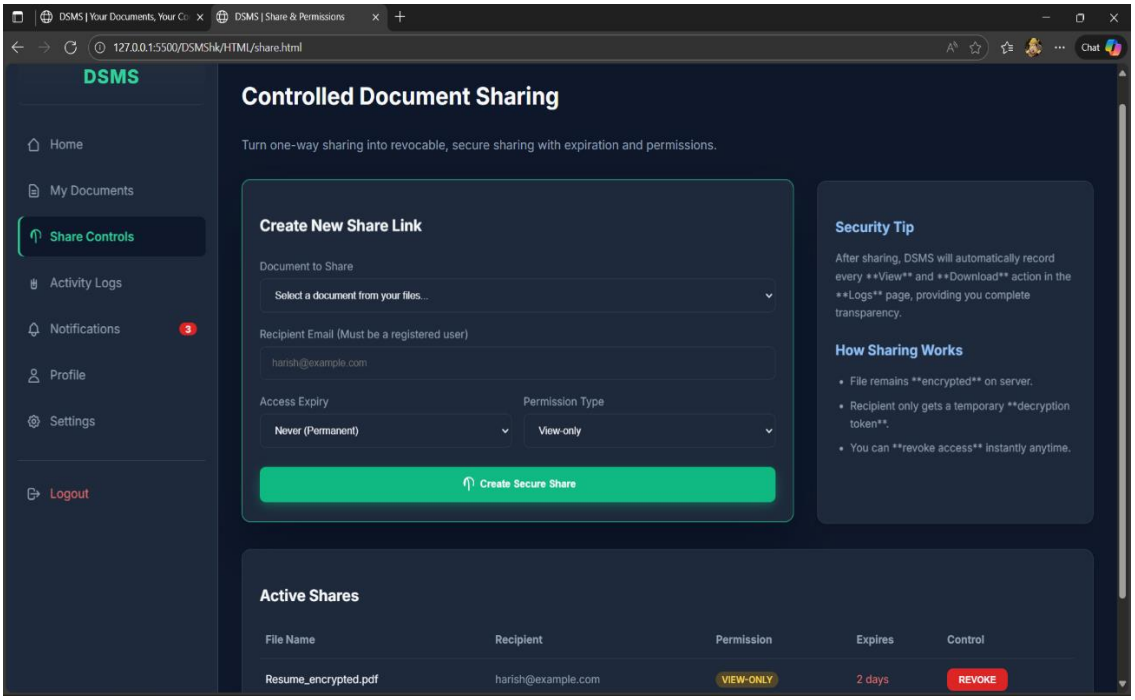

DSMS | Your Documents, Your Co
127.0.0.1:5500/DSMShk/HTML/index.html#features
Chat
How It Works: The 3-Step Flow
1
Sign In or Register
Firebase Auth ka use karke apna secure DSMS account banao ya login karo.
2
Upload Securely
Apne documents ko client-side encrypt karke Firebase Storage par upload karo.
3
Share & Track
Permission set karke share karo aur `logs.html` par har activity ko live monitor karo.
Ready to take back control?
DSMS Phase 1 gives you security and transparency over your most valuable digital assets.
Start Your Secure Journey Now

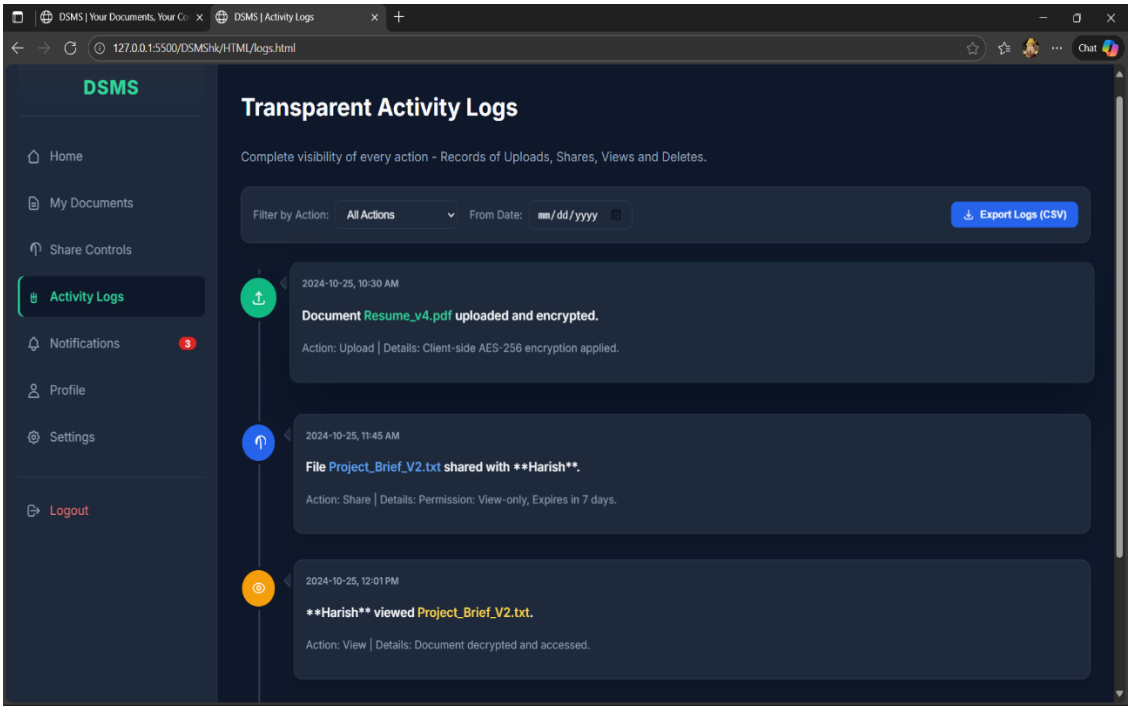

DSMS | Sign In or Register
127.0.0.1:5500/DSMShk/HTML/auth.html
Chat
DSMS
Building the Foundation of Digital Ownership
Sign In
Register
Sign In to DSMS
Email
Password
Remember me
Forgot password?
Sign In

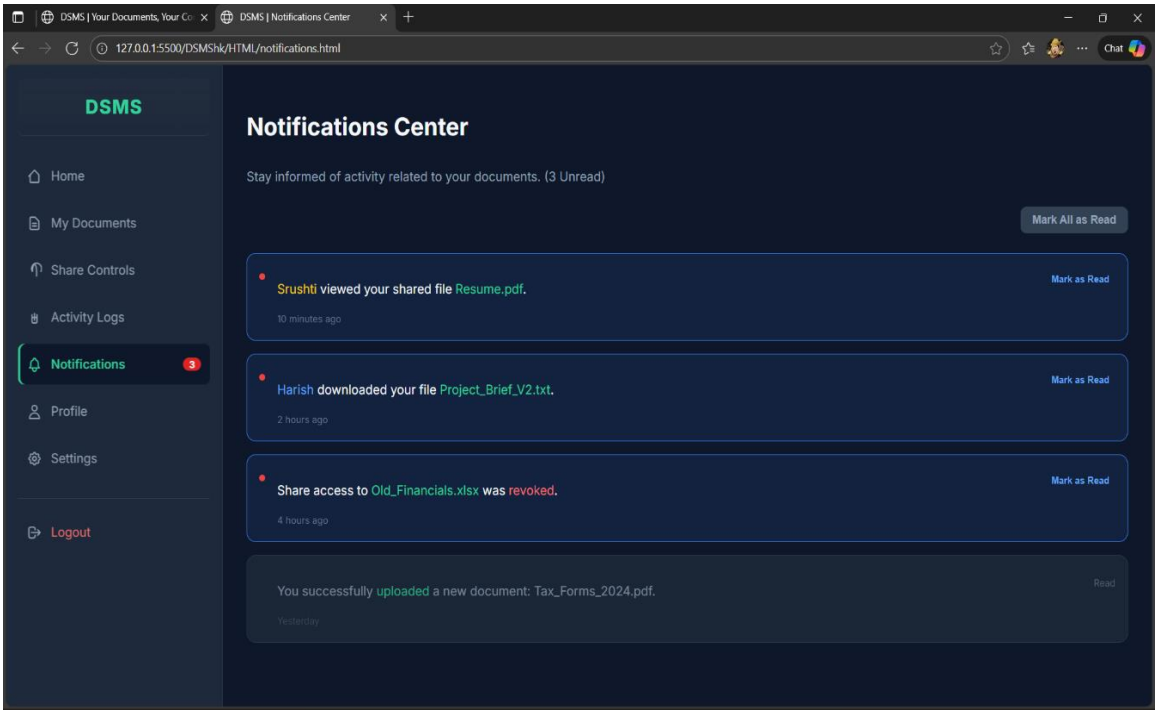

DSMS | Sign In or Register
127.0.0.1:5500/DSMShk/HTML/auth.html
Chat
DSMS
Building the Foundation of Digital Ownership
Sign In
Register
Create Your Account
Full Name
Email
Password
Create Account

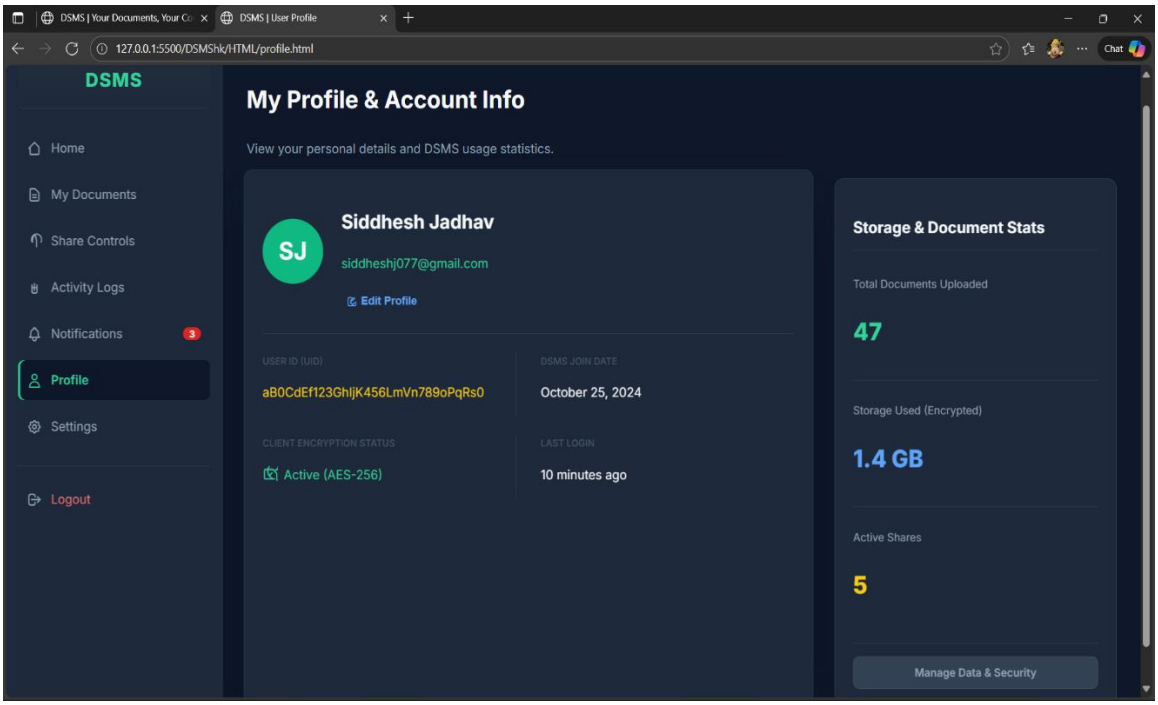

DSMS | Your Documents, Your Co
DSMS | Dashboard
127.0.0.1:5500/DSMShk/HTML/dashboard.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
Welcome Back, User!
Total Documents
42
Active Shares
5
Storage Used
1.2 GB / 5 GB
Recent Activity
View All
Action
File Name
Description
Timestamp
Upload
Tax_Docs_2024.pdf
New file uploaded.
5 mins ago
Share
Project_Plan.docx
Shared with Harish.
2 hours ago
View
Resume.pdf
Srushti viewed the file.
Yesterday
Delete
Old_Notes.txt
Document permanently removed.
3 days ago

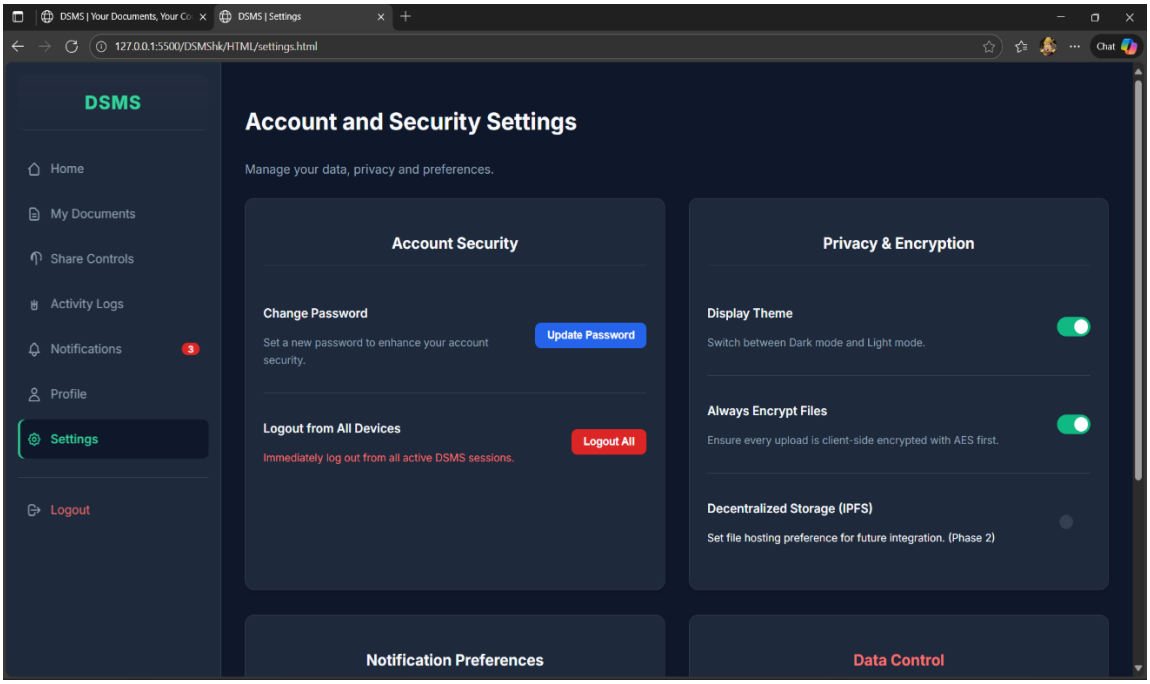

DSMS | Your Documents, Your Co
DSMS | My Documents
127.0.0.1:5500/DSMShk/HTML/documents.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
My Documents
Drag & Drop your files here
or click to browse. Max size 50MB.
Upload Document
Encrypting file before upload...
Open
Documents › Capstone Project
Search Capstone Project
Organize
New folder
Siddhesh - Perso
Desktop
Downloads
Documents
Pictures
Music
Videos
Screenshots
Name
Status
Date modified
Type
DSMShk
10/25/2025 10:56 AM
File folder
capstone_Report
10/31/2025 3:28 PM
DSMS[Sid,Harish]
10/25/2025 12:26 PM
Compressed
Introduction To DSMS
9/4/2025 9:20 AM
SamplePPT
10/15/2025 9:24 PM
Tech Stack We Need To Learn
9/4/2025 9:20 AM
Uniqueness in DSMS
9/4/2025 9:20 AM
File name: capstone_Report
All files
Upload from mobile
Open
Cancel
Uploaded Documents
Total: 4
FILE NAME
SIZE
STATUS
UPLOAD DATE
ACTIONS
Resume_encrypted.pdf
2.1 MB
ENCRYPTED
2024-10-25
View
Share
Delete
Bank_Statement_Q3.docx
800 KB
ENCRYPTED
2024-09-01
View
Share
Delete

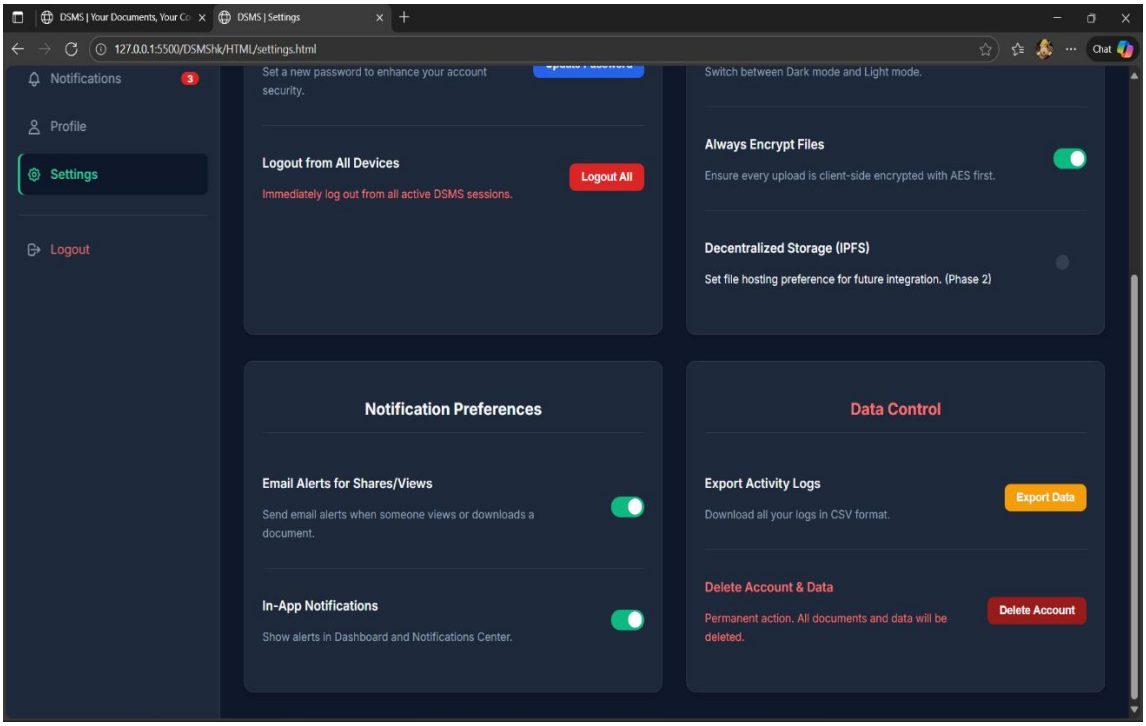

DSMS | Your Documents, Your Co
DSMS | Share & Permissions
127.0.0.1:5500/DSMShk/HTML/share.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
Controlled Document Sharing
Turn one-way sharing into revocable, secure sharing with expiration and permissions.
Create New Share Link
Document to Share
Select a document from your files...
Recipient Email (Must be a registered user)
harish@example.com
Access Expiry
Never (Permanent)
Permission Type
View-only
Create Secure Share
Security Tip
After sharing, DSMS will automatically record every **View** and **Download** action in the **Logs** page, providing you complete transparency.
How Sharing Works
File remains **encrypted** on server.
Recipient only gets a temporary **decryption token**.
You can **revoke access** instantly anytime.
Active Shares
File Name
Recipient
Permission
Expires
Control
Resume_encrypted.pdf
harish@example.com
VIEW-ONLY
2 days
REVOKE

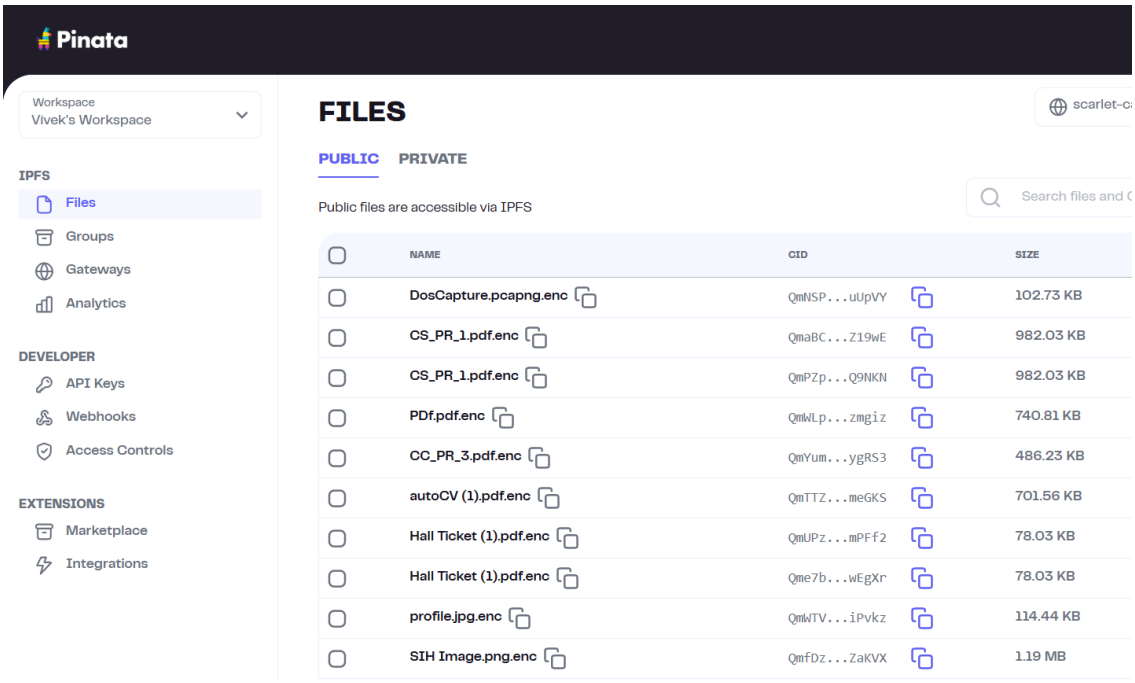

DSMS | Your Documents, Your Co
DSMS | Activity Logs
127.0.0.1:5500/DSMShk/HTML/logs.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
Transparent Activity Logs
Complete visibility of every action - Records of Uploads, Shares, Views and Deletes.
Filter by Action:
All Actions
From Date:
mm/dd/yyyy
Export Logs (CSV)
2024-10-25, 10:30 AM
Document Resume_v4.pdf uploaded and encrypted.
Action: Upload | Details: Client-side AES-256 encryption applied.
2024-10-25, 11:45 AM
File Project_Brief_V2.txt shared with **Harish**.
Action: Share | Details: Permission: View-only, Expires in 7 days.
2024-10-25, 12:01 PM
Harish viewed Project_Brief_V2.txt.
Action: View | Details: Document decrypted and accessed.


DSMS | Your Documents, Your Co
DSMS | Notifications Center
127.0.0.1:5500/DSMShk/HTML/notifications.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
Notifications Center
Stay informed of activity related to your documents. (3 Unread)
Mark All as Read
Srushti viewed your shared file Resume.pdf.
10 minutes ago
Mark as Read
Harish downloaded your file Project_Brief_V2.txt.
2 hours ago
Mark as Read
Share access to Old_Financials.xlsx was revoked.
4 hours ago
Mark as Read
You successfully uploaded a new document: Tax_Forms_2024.pdf.
Yesterday
Read


DSMS | Your Documents, Your Co
DSMS | User Profile
127.0.0.1:5500/DSMShk/HTML/profile.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
My Profile & Account Info
View your personal details and DSMS usage statistics.
SJ
Siddhesh Jadhav
siddheshj077@gmail.com
Edit Profile
USER ID (UID)
aB0CdEf123GhIjK456LmVn789oPqRs0
DSMS JOIN DATE
October 25, 2024
CLIENT ENCRYPTION STATUS
Active (AES-256)
LAST LOGIN
10 minutes ago
Storage & Document Stats
Total Documents Uploaded
47
Storage Used (Encrypted)
1.4 GB
Active Shares
5
Manage Data & Security


DSMS | Your Documents, Your Co
DSMS | Settings
127.0.0.1:5500/DSMShk/HTML/settings.html
Chat
DSMS
Home
My Documents
Share Controls
Activity Logs
Notifications
3
Profile
Settings
Logout
Account and Security Settings
Manage your data, privacy and preferences.
Account Security
Change Password
Set a new password to enhance your account security.
Update Password
Logout from All Devices
Immediately log out from all active DSMS sessions.
Logout All
Privacy & Encryption
Display Theme
Switch between Dark mode and Light mode.
Always Encrypt Files
Ensure every upload is client-side encrypted with AES first.
Decentralized Storage (IPFS)
Set file hosting preference for future integration. (Phase 2)
Notification Preferences
Data Control


DSMS | Your Documents, Your Co
DSMS | Settings
127.0.0.1:5500/DSMShk/HTML/settings.html
Chat
Notifications
3
Profile
Settings
Logout
Set a new password to enhance your account security.
Switch between Dark mode and Light mode.
Logout from All Devices
Immediately log out from all active DSMS sessions.
Logout All
Always Encrypt Files
Ensure every upload is client-side encrypted with AES first.
Decentralized Storage (IPFS)
Set file hosting preference for future integration. (Phase 2)
Notification Preferences
Email Alerts for Shares/Views
Send email alerts when someone views or downloads a document.
In-App Notifications
Show alerts in Dashboard and Notifications Center.
Data Control
Export Activity Logs
Download all your logs in CSV format.
Export Data
Delete Account & Data
Permanent action. All documents and data will be deleted.
Delete Account


Pinata
Workspace
Vivek's Workspace
IPFS
Files
Groups
Gateways
Analytics
DEVELOPER
API Keys
Webhooks
Access Controls
EXTENSIONS
Marketplace
Integrations
FILES
PUBLIC
PRIVATE
Public files are accessible via IPFS
Search files and
NAME
CID
SIZE
DosCapture.pcapng.enc QmNSP...uUpVY 102.73 KB
CS_PR_1.pdf.enc QmaBC...Z19wE 982.03 KB
CS_PR_1.pdf.enc QmPZp...Q9NKN 982.03 KB
PDf.pdf.enc QmWLp...zmgiz 740.81 KB
CC_PR_3.pdf.enc QmYum...ygRS3 486.23 KB
autoCV (1).pdf.enc QmTTZ...meGKS 701.56 KB
Hall Ticket (1).pdf.enc QmUPz...mPFf2 78.03 KB
Hall Ticket (1).pdf.enc Qme7b...wEgXr 78.03 KB
profile.jpg.enc QmWTV...iPvkz 114.44 KB
SIH Image.png.enc QmfDz...ZaKVX 1.19 MB

CHAPTER 6: COST ESTIMATION

1. Development Tools and Software

Item	Cost (INR)	Remarks
1. Development Tools & Environments	Free	No upfront cost required for standard development.
Visual Studio Code / IDE	Free	Lightweight open-source code editor for development.
Node.js & npm / Yarn	Free	Essential runtime and package manager for the Express.js backend and JavaScript libraries.
Git & GitHub	Free	Used for source control, collaboration, and private repositories (especially with a student account).
2. Core Technology Stack (Software)	Free / Pay-per-use	Primary costs only scale with production usage.
HTML5 / CSS3 / Vanilla JavaScript	Free	Core web languages and open-source foundation for the lightweight frontend.
Tailwind CSS	Free	Open-source, utility-first CSS framework.
Ethers.js / Web3.js	Free	Open-source JavaScript libraries required for Web3 wallet integration (MetaMask).
3. Backend & Cloud Services	Free / Pay-per-use	Usage-based costs for database and storage.
Firebase (Auth, Firestore)	Free / Pay-per-use	Free Spark plan is sufficient for development and demonstration. Costs scale based on database reads/writes and authentication volume.
IPFS (via web3.storage/Pinata)	Free / Pay-per-use	Services like Pinata offer a generous free tier for pinning decentralized files, sufficient for project demonstration. Costs scale based on storage volume and data retrieval.
4. Deployment Services	Free / Pay-per-use	Free tiers cover basic hosting needs.
Vercel / Railway / Render	Free / Pay-per-use	Hosting platforms offer a free tier sufficient for deploying the static HTML/JS frontend (Vercel) and the Node.js backend (Railway/Render).
5.Optional/Advanced Components	Free (Gas Fees Apply)	Costs tied to blockchain interaction.
Solidity / Smart Contracts	Free (Gas Fees Apply)	Solidity is free. Deployment (Phase 2) on a testnet like Polygon Mumbai is free, but actual mainnet deployment incurs gas fees (transaction costs).

Chart.js / D3.js	Free	Open-source visualization libraries used for the owner dashboard statistics/graphs.
6. Miscellaneous Costs	Low / Varies	Contingency and supporting costs.
Domain Name Registration (e.g., .com)	Approx. 500 – 1000/Yr	Required for a professional, easily memorable URL for the public application.
Testing Tools / API Clients (e.g., Postman)	Free / Low-cost	Free tier is sufficient for testing APIs; Pro versions have costs but are optional.

CHAPTER 7: SOFTWARE TESTING

Test ID	Test Case Description	Objectives	Expected Result	Actual Result
TC01	Login with MetaMask Web3 wallet credentials	Verify decentralized user authentication	User logs in successfully and is redirected to the DSMS Dashboard	As Expected
TC02	Upload a document after client-side encryption	Test secure document upload to IPFS	The file is uploaded, encrypted status is shown, and a CID is generated	As Expected
TC03	Generate a "View-only" share link with a 24-hour expiry	Verify granular sharing controls	Unique link is created, and the access can be viewed for 24 hours but not downloaded	As Expected
TC04	Revoke access on an active share link	Test the core revocation feature	The recipient's link becomes immediately invalid, showing 'Access Denied'	As Expected
TC05	View a shared document using a valid, non-expired link	Confirm document retrieval from IPFS	The document is successfully retrieved, decrypted, and displayed in Preview Mode	As Expected
TC06	Check the Activity Logs after a view or download event	Ensure transparent and meticulous access tracking	A new log entry appears showing the recipient, action (View/Download), and timestamp	As Expected
TC08	Login with valid Firebase (Email/Password) credentials	Verify traditional login flow	User is successfully authenticated via Firebase Auth and lands on the dashboard	As Expected
TC09	Navigate to the "My Documents" page	Validate file metadata display	List of uploaded files, their size, and ENCRYPTED status are displayed correctly	As Expected
TC10	Export Activity Logs data	Check functionality for auditing.	A report downloads in a standard format (e.g., CSV)	As Expected

CHAPTER 8: SOFTWARE IMPLEMENTATION

1.Implementation Environment

Environment	Details
Development Machine (Local)	Standard Laptop/PC (Windows, macOS, or Linux) with the capacity to run all development tools, including Node.js and the project IDE.
Node.js Runtime	The foundational environment (version 16.x or later) required for executing the Express.js backend, running package scripts (npm/yarn), and processing server-side logic.
IDE (Visual Studio Code)	The primary code editor, configured with extensions for JavaScript, HTML, and CSS, to support efficient development and debugging of both frontend and backend code.
Version Control (Git)	Installed and configured for local tracking of all code changes, branching, merging, and synchronization with the remote GitHub repository.
API Client (Postman/Insomnia)	Essential tool for testing the RESTful APIs created in Express.js, ensuring that endpoints for upload, sharing, and revocation function correctly before frontend integration.
Firebase CLI	Command Line Interface tool used for local management of the Firebase project, including deploying database security rules and managing authentication settings.

2.Core Modules Implemented

1. Authentication and User Management Module
    - Role: Handles user identity, secure login (Email/Password and MetaMask Web3 Wallet), and maintains the user's document index in the database.
    - Implementation: Utilizes Firebase Authentication for traditional users and Ethers.js/MetaMask for Web3 integration. Secures the user profile data (but not the files themselves) in Firestore.
2. Decentralized Storage (Upload) Module
    - Role**:** Manages the secure transfer and storage of the user's document onto the decentralized network.

- Implementation: Takes the file from the user via the Node.js backend, encrypts it, and uploads the data to an IPFS Pinning Service (e.g., Pinata). It saves the unique, immutable Content ID (CID) to the user's Firestore profile.

3. Document Management Module
    - Role**:** Provides the user interface (dashboard) for the owner to view, organize, and select their stored documents.
    - Implementation**:** Frontend (HTML/CSS/JS) fetches metadata (file name, size, upload date) from Firestore using the owner's userId. Allows the owner to initiate the sharing process.
4. Secure Sharing and Access Control Module
    - Role**:** Generates time-limited, single-use, and revocable access links and enforces permissions when a receiver attempts to view the file.
    - Implementation: The Express.js backend generates a unique, cryptographically secure access token. This token, along with its expiry time and access count, is stored in Firestore.
5. Trusted Document Bridge (Viewing) Module
    - Role**:** The most critical module; it handles the secure, temporary delivery of the file to a receiver (external website or user) and ensures strict access logging.
    - Implementation**:** When an access token is validated, the backend uses the stored CID to retrieve the file from the IPFS Gateway, decrypts it on the server, and serves it as a single-view stream. It logs the viewer's ID, timestamp, and location (if available) to Firestore.
6. Access Revocation and Audit Trail Module
    - Role**:** Allows the owner to instantly block access to any generated share link and view a non-tamperable history of all access attempts.
    - Implementation**:** Revocation is achieved by instantly changing the status flag of the share token in Firestore. The audit trail displays the logs generated by the Document Bridge module.

3. Database Schema Overview

Field Name	Data Type	Description
userId	String	Unique Firebase User ID (auth.uid) or MetaMask address.
email	String	User's primary email address (for traditional login).
dateJoined	Timestamp	Date and time the user created the account.
walletAddress	String	The public key if the user signs in via MetaMask.
ownerId	String	Foreign Key: Links to the userId in the Users collection.
fileName	String	Original name of the uploaded file (e.g., 'Marksheet.pdf').
ipfsCid	String	The immutable Content ID (CID) of the encrypted file stored on IPFS.
fileHash	String	SHA-256 hash of the encrypted file for integrity check.
uploadDate	Timestamp	Date and time the document was uploaded.
fileSize	Number	Size of the file in bytes.
tokenId	String	The unique, cryptographically secure token included in the share link.
documentRef	DocumentRef	Foreign Key: Links directly to the document in the Documents collection.
receiverEmail	documentRef	The email address of the intended recipient (or a unique ID).
expiryDate	Timestamp	Date and time when the share link becomes invalid.
maxViews	Number	Maximum number of times this link can be used.
viewsUsed	Number	Current count of times the link has been accessed.
isRevoked	Boolean	Flag to instantly disable access (true/false) by the owner.
permissions	Array (Strings)	e.g., ['view-only', 'watermark'].
tokenRef	DocumentRef	Foreign Key: Links to the used ShareTokens document.
documentRef	DocumentRef	Foreign Key: Links to the document that was accessed.
accessTime	Timestamp	Date and time of the access attempt.

CHAPTER 9: FUTURE SCOPE

The successful completion of the basic DSMS version establishes a robust foundation for the project's planned advanced features (Phase 2). These future enhancements are designed to elevate the system's security, transparency, and user experience, cementing its position as a leader in data sovereignty.

- Immutable Audit Trail: The primary future goal is the implementation of the Solidity Smart Contract, which will log all document sharing and access events immutably on a public blockchain like Polygon Mumbai. This ensures a tamper-proof history of access that no party, including the DSMS administrators, can alter.
- Advanced Encryption: We plan to deploy end-to-end encryption before documents are uploaded to IPFS. This mechanism guarantees that files remain unreadable during transfer and storage, ensuring true privacy by making the content inaccessible without the owner's private key.
- Ecosystem Expansion (Mobile): Development of the Flutter-based mobile application will be undertaken to provide native access. This will include integrating advanced security features like Biometric Login (face/fingerprint) for secure device-based authentication.
- Enhanced Sharing Features: We will introduce mandatory watermarking on shared document previews. This security layer automatically overlays the viewer's ID and timestamp on the file to deter unauthorized copying or screen-capturing.
- Wider Integration (Open API): An Open API and SDK will be released to streamline the integration of the Trusted Document Bridge into third-party websites, making it the standard for secure document requests in areas like education and hiring.

CHAPTER 10: CONCLUSION

The DSMS project successfully delivered a robust, secure, and user-centric platform that fundamentally redefines digital document sharing. Our primary objective—to restore full control, transparency, and ownership to the user—has been successfully achieved through the strategic application of decentralized technology.

The seamless integration of IPFS ensures that documents are stored immutably across a global, distributed network, effectively eliminating the risks of single-point censorship or server failure inherent in traditional cloud services. Furthermore, all crucial document metadata and user access conditions are securely managed within the Firebase ecosystem.

A major success of this phase is the fully functional Access Control Module. This capability empowers users to generate highly granular, revocable, and time-limited share links, providing a level of control that is unprecedented in conventional centralized platforms. The foundation for the Trusted Document Bridge mechanism is also validated, proving the concept of secure document handshakes between external portals and the user's private vault without yielding possession.

The hybrid Web2 (Firebase Auth/Firestore) and Web3 (IPFS/MetaMask) approach has been validated as highly effective, offering the necessary speed and scalability for user management while leveraging decentralization for core storage. In summary, DSMS is not merely a storage application; it is a critical infrastructure designed to pioneer the future of digital identity and data sovereignty, empowering individuals to take back control from large centralized entities.

REFERENCES

1.Firebase Documentation
https://firebase.google.com/docs
Used for implementing Firestore database, user authentication, and secure cloud-based metadata storage.

2.Pinata IPFS Documentation
https://docs.pinata.cloud/
Reference for decentralized file storage and content pinning on IPFS network during document uploads.

3.Node.js Documentation
https://nodejs.org/en/docs
Consulted for setting up backend APIs, environment configuration, and asynchronous file handling.

4.Express.js Documentation
https://expressjs.com/
Used for building RESTful backend routes, middleware, and secure API endpoints for DSMS.

5.Google Cloud Documentation
https://cloud.google.com/docs
Referenced for environment variables, cloud function deployment, and scaling the backend.

6.Chart.js Official Documentation
https://www.chartjs.org/docs/latest/
Used to visualize analytics like uploads, shares, and storage trends on the DSMS dashboard.

7.Vanta.js & Three.js Documentation
https://www.vantajs.com/
https://threejs.org/docs/
Integrated for background animation and interactive visuals on the homepage.

8.Pinata API (REST Reference)
https://docs.pinata.cloud/api-pinning/pin-file
Used to upload, pin, and unpin files from IPFS network directly via JWT token authentication.

9.MDN Web Docs (Mozilla Developer Network)
https://developer.mozilla.org/
Used extensively for JavaScript DOM manipulation, fetch API handling, and event listeners.

10.Stack Overflow
https://stackoverflow.com/
Community-driven platform referenced for resolving integration issues, Node-Firebase setup errors, and CORS handling.

11.YouTube – Firebase + Node.js Full Stack Tutorials
https://www.youtube.com/
Assisted in learning backend integration, token-based authentication, and dashboard linking.

12.GitHub – Open Source Node.js Projects
https://github.com/
Used as reference for folder structuring, middleware patterns, and Express + Firebase integration examples.

13.Pinata Blog – Web3 and Decentralized Storage Concepts
https://blog.pinata.cloud/
Provided understanding of IPFS, pinning services, and decentralized content delivery used in DSMS.

14.Google Fonts Documentation
https://fonts.google.com/
Used for UI typography selection and integration into the DSMS frontend.

15.W3Schools Web Development Tutorials
https://www.w3schools.com/
General reference for HTML, CSS, and JavaScript concepts used in designing the static frontend templates.