SAVITRIBAI PHULE PUNE UNIVERSITY

A PRELIMINARY PROJECT REPORT ON

BlockShare - Blockchain Based Secure Data Sharing Platform

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

BACHELOR OF ENGINEERING
(Computer Engineering)

SUBMITTED BY

Group ID : B06

Mr. Sahane Abhijit Rajaram	Exam No: B400100415
Mr. Shinde Rohit Nivrutti	Exam No: B400100428
Mr. Sayyad Mohammadsaani Shahid	Exam No: B400100422
Mr. Kankate Sairaj Chandrakant	Exam No: B400100378

Under The Guidance of

Prof. K. U. Rahane

DEPARTMENT OF COMPUTER ENGINEERING
Amrutvahini College of Engineering, Sangamner
Amrutnagar, Ghulewadi - 422608

2024-25

AMRUTVAHINI COLLEGE OF ENGINEERING,SANGAMNER
DEPARTMENT OF COMPUTER ENGINEERING

CERTIFICATE

This is to certify that the Project Entitled

BlockShare - Blockchain Based Secure Data Sharing Platform

Submitted by

Group ID: B06

Mr. Sahane Abhijit Rajaram	Exam No: B400100415
Mr. Shinde Rohit Nivrutti	Exam No: B400100428
Mr. Sayyad Mohammadsaani Shahid	Exam No: B400100422
Mr. Kankate Sairaj Chandrakant	Exam No: B400100378

are bonafide students of this institute and the work has been carried out by them under the supervision of Prof. K. U. Rahane 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 Engineering).

Prof. K. U. Rahane Project Guide Dept. of Computer Engg.	Dr. D. R. Patil Project Coordinator Dept. of Computer Engg.
Dr. S. K. Sonkar H.O.D. Dept. of Computer Engg.	Dr. M.A. Venkatesh Principal AVCOE Sangamner

SAVITRIBAI PHULE PUNE UNIVERSITY

CERTIFICATE

This is to certify that,

Group ID: B06

Mr. Sahane Abhijit Rajaram	Exam No: B1901004294
Mr. Shinde Rohit Nivrutti	Exam No: B1901004307
Mr. Sayyad Mohammadsaani Shahid	Exam No: B1901004301
Mr. Kankate Sairaj Chandrakant	Exam No: B1901004257

of BE Computer Engineering was examined in the Project Examination Semester I entitled

BlockShare – Blockchain Based Data Sharing Platform

on / / 2024

At

DEPARTMENT OF COMPUTER ENGINEERING

AMRUTVAHINI COLLEGE OF ENGINEERING, SANGAMNER

Internal Examiner

External Examiner

Acknowledgment

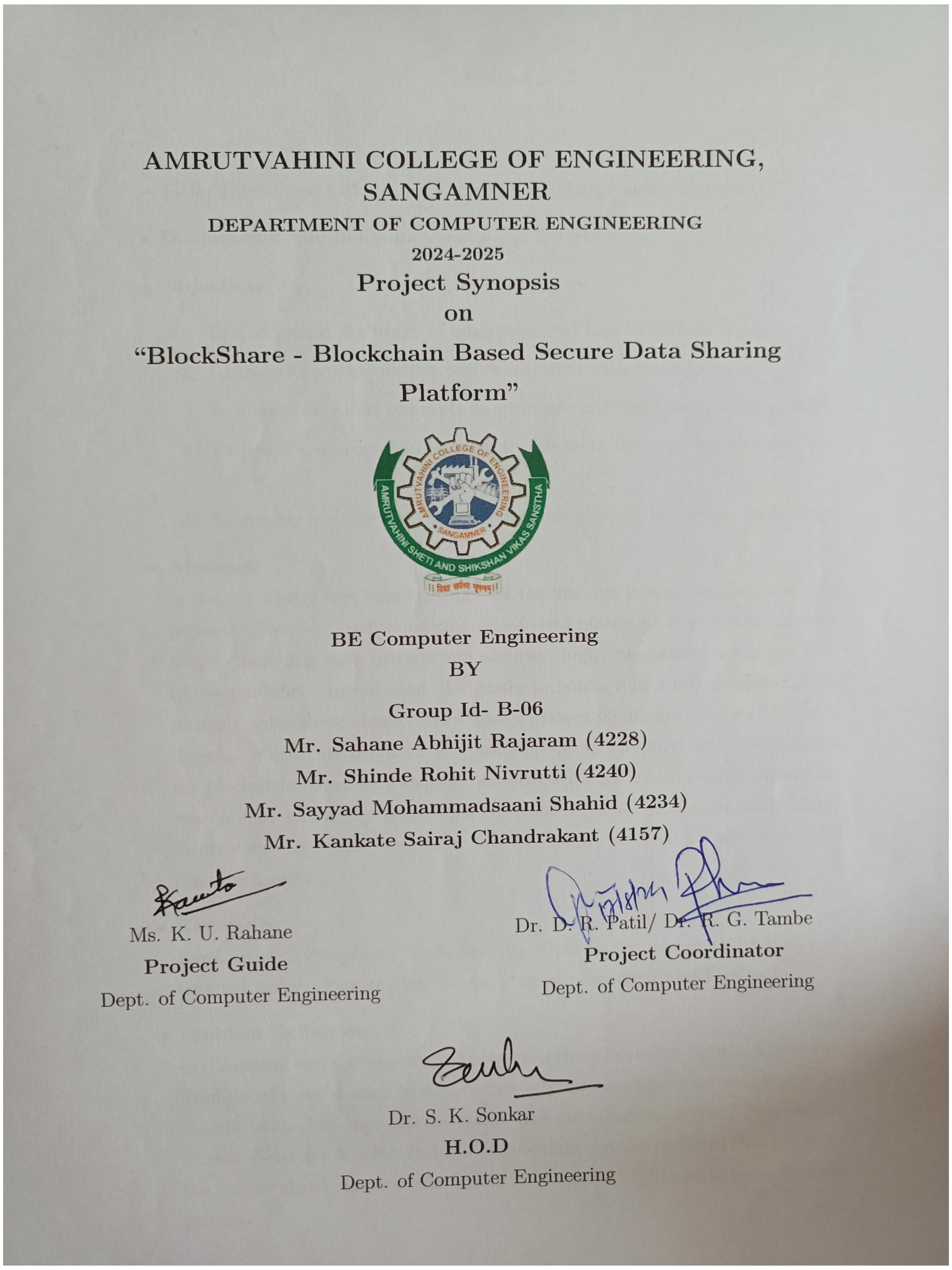
We are honored to present this project, “Blockshare - Blockchain based secure data sharing platform”. We extend our heartfelt gratitude to our guide, **Prof .K .U .Rahane**, for her valuable guidance and support throughout this project. We are also immensely grateful to our project coordinators **Dr. D. R. Patil**, and Head of Department **Dr. S. K. Sonkar**, for their continuous encouragement, valuable insights, and kind assistance, which were instrumental in shaping this project. Our sincere thanks go to our Principal, **Dr. M. A. Venkatesh**, for his unwavering support, motivation, and commitment to fostering a conducive learning environment. We would also like to express our appreciation to all the staff members who provided their cooperation and support. Lastly, we are profoundly grateful to our friends for their significant contributions, encouragement, and camaraderie, which made this journey truly rewarding. We extend our gratitude to everyone who directly or indirectly assisted us in completing this project.

Mr. Sahane Abhijit Rajaram
Mr. Shinde Rohit Nivrutti
Mr. Sayyad Mohammadsaani Shahid
Mr. Kankate Sairaj Chandrakant

Abstract

In today's digital world, data is incredibly valuable, and sharing it securely is more important than ever. However, centralized platforms that are commonly used for data sharing come with issues like privacy concerns, security risks, data breaches, data loss, and a lack of control over who accesses the data. These platforms also often rely on third parties like centralized servers and databases, making data vulnerable to misuse or breaches. To address these issues, our platform uses the core concepts of blockchain, such as decentralized storage and peer-to-peer communication. In a distributed system, there is no single point of control or failure, meaning that data is spread across multiple nodes, which improves both security and reliability. Peer-to-peer (P2P) sharing allows users to exchange data directly, without relying on any centralized authority, ensuring faster and more secure data transfer. By eliminating the need for central authorities or middlemen, we ensure that data remains private and is shared securely through peer-to-peer channels. Users will have more control over their data and can share it directly with others in a safe and efficient way. This new approach addresses the limitations of traditional data-sharing systems, offering a better solution for secure and trustworthy data exchange.

Synopsis



AMRUTVAHINI COLLEGE OF ENGINEERING, SANGAMNER

DEPARTMENT OF COMPUTER ENGINEERING

2024-2025

Project Synopsis

on

"BlockShare - Blockchain Based Secure Data Sharing Platform"

BE Computer Engineering

BY

Group Id- B-06

Mr. Sahane Abhijit Rajaram (4228)

Mr. Shinde Rohit Nivrutti (4240)

Mr. Sayyad Mohammadsaani Shahid (4234)

Mr. Kankate Sairaj Chandrakant (4157)

[illegible]

Ms. K. U. Rahane

Project Guide

Dept. of Computer Engineering

[illegible]

Dr. D. R. Patil/ Dr. R. G. Tambe

Project Coordinator

Dept. of Computer Engineering

[illegible]

Dr. S. K. Sonkar

H.O.D

Dept. of Computer Engineering

Title: BlockShare - Blockchain Based Secure Data Sharing Platform.

Domain and Sub-domain: Security and Blockchain.

Objectives:

1. To understand the basics of blockchain and how blockchain works.
2. To understand the security concepts related with blockchain.
3. To implement smart contracts to automate and secure data access control.
4. To create a decentralized system to eliminate the need for a central authority.
5. To develop a secure data sharing platform utilizing blockchain technology.

Abstract:

In today's digital age, data sharing over the internet is very common and very popular. However, traditional centralized data platforms face significant challenges, including data privacy and security, high transaction costs, and lack of compatibility. Introducing blockchain technology into this domain can effectively solve these issues. Blockchain's system eliminates the need for middlemen, which cuts down on fees and speeds up transactions. Additionally, the blockchain based data sharing platform will offer robust decentralized data storage and exchange mechanisms, comprehensive access control, and reliable identity authentication, making it a revolutionary solution for secure and efficient data sharing.

Keywords:

Blockchain; Data Sharing; Data Security; Cryptography; Decentralization; Smart Contracts; Secure Communication; Peer-to-Peer; Distributed Ledger.

Problem Definition:

In the digital age, the security and privacy of shared data has become a concerns. Traditional data sharing systems often rely on centralized servers, which are susceptible to hacking, data breaches, and unauthorized access. This project aims to develop a blockchain-based secure data sharing platform that leverages the decentralized nature of blockchain technology to enhance data security and privacy.

List of Modules:

1. User Interface Development
2. User Registration and Authentication
3. Data Encryption and Storage
4. Smart Contract Integration
5. Data Sharing and Retrieval

Current Market Survey:

The current market for data sharing platforms is dominated by centralized systems, which pose significant security risks. Numerous data breaches and unauthorized access incidents have highlighted the vulnerabilities of these systems. Blockchain technology, with its decentralized and immutable nature, offers a promising solution to these issues. However, there is a need for platforms specifically addressing data sharing security and privacy concerns ,Blockchain will be a great alternative for this problem.

Scope of the Project:

Developing a secure data sharing platform based on blockchain technology. The platform will support secure data storage, sharing, and access control through decentralized mechanisms. Key functionalities will include user authentication, data encryption, smart contract-based access control, and audit trails. Focusing mostly on the core security and privacy features of the platform.

Literature Survey:

1. Title - Blockchain-Empowered Trustworthy Data Sharing: Fundamentals, Applications, and Challenges (2023).

Authors - Linh T. Nguyen, Lam Duc Nguyen, Thong Hoang, Dilum Bandara, Qin Wang, Qinghua Lu, Xiwei Xu, Liming Zhu, Petar Popovski, Fellow, IEEE, and Shiping Chen.

DOI - 10.48550/arXiv.2303.06546

2. Title - A Secure Data Sharing Platform Using Blockchain and Interplanetary File System (2019).

   Authors - Muqaddas Naz, Fahad A. Al-zahrani, Rabiya Khalid, Nadeem Javaid 1, Ali Mustafa Qamar, Muhammad Khalil Afzal and Muhammad Shafiq

   DOI - https://doi.org/10.3390/su11247054

3. Title - A Consent Model for Blockchain-Based Health Data Sharing Platforms (2020).

   Authors - Vikas Jaiman and Visara Urovi

   DOI - 10.1109/ACCESS.2020.3014565

4. Title - A Survey of Blockchain-Based Schemes for Data Sharing and Exchange (2023).

   Authors - Rui Song, Bin Xiao, Yubo Song, Songtao Guo and Yuanyuan Yang.

   DOI - 10.1109/TBDATA.2023.3293279

5. Title - Subscription-Based Data-Sharing Model Using Blockchain and Data as a Service (2020).

   Author - Fahad Ahmad Al-Zahrani.

   DOI - 10.1109/ACCESS.2020.3002823

Software and Hardware Requirement of the Project:

Software:

1. Operating System - Windows 7/8/10 / Linux / Mac
2. Front-end Frameworks - React.js ,Tailwind CSS
3. Decentralized platform - eg.(Ethereum)

4. API -eg(web3.js ,ether.js)
5. Programming Languages - Solidity ,JavaScript

Hardware:

1. Ram :- 8GB
2. Rom :- 256 GB ssd
3. processor :- 3.0 GHZ

Contribution to Society:

Probable Date of Project Completion: December 2024

Outcome of the Project:

1. A fully functional blockchain-based secure data sharing platform.
2. Enhanced data security and privacy through decentralized mechanisms.
3. Implementation of smart contracts for automated access control.

Abbreviation

BLT	Blockchain Technology
P2P	Peer-to-Peer
IPFS	Inter Planetary File System
LibP2P	Library Peer-to-Peer
MQ	Message Queue
CID	Content Identifier
SRS	Software Requirement Specification
API	Application Programming Interface
CSS	Cascading Style Sheets
IDE	Integrated Development Environment
UI	User Interface
DFD	Data Flow Diagram
UML	Unified Modeling Language

List of Figures

List of Tables

INDEX

CHAPTER 1

INTRODUCTION

In today's digital world, data is extremely valuable. Also data is being generated rapidly. This rapidly generated data plays a key role in making decisions, empowering technology, and driving innovation across industries. In today's world data is a new fuel which drives the industries and there operations.
From personal information to important business data, sharing data safely and easily is crucial for everyone including individuals, businesses, governments, and communities that depend on accurate, trustworthy information. But sharing data privately and securely remains difficult, especially with traditional, centralized platforms. Most data-sharing platforms today are centralized, meaning data is stored on a single server or database managed by a third party.
While this setup can be convenient and looks secure, it also brings major risks. Centralized platforms are often targeted for hacking, data leaks, data breach and unauthorized access, putting users information at risk. They also rely on middlemen i.e some central authority controls the data storage, which can increase costs, reduce privacy, and limit users control over their own data. People using these platforms must trust that their data is safe, but unfortunately, it can still be vulnerable to misuse or loss.[1]
This project aims to solve these problems by creating a secure, private, and efficient data-sharing platform that is different from traditional models. By using ideas from blockchain, decentralized storage, and peer-to-peer (P2P) communication, our platform removes the need for central authorities.
Instead, data is stored in a distributed way and shared directly between users, allowing people to have more control over their information, stronger privacy, and fewer costs. With decentralized storage, like IPFS, data is stored in pieces across different locations, which reduces the risk of data being tampered or data loss.[2]
This platform offers a solution for people and organizations who want to share data securely, without involving third parties or facing high transaction fees. Ultimately, this project redefines data sharing by making it a safe, private, and transparent process.
It's a solution that puts user privacy and data security first, helping to create a digital society that is fair, affordable, and secure.

1.1 BLOCKCHAIN

- Blockchain serves as a foundation for secure, trust-based interactions without the need for a central authority. Blockchain is fundamentally a decentralized digital ledger that stores information across numerous computers, making the data secure and resistant to tampering. Rather than depending on a single centralized system like database or server, blockchain ensures that the information shared between users is reliable, authentic, and remains under the user's control. While we're not directly using blockchain technology here, the project uses key principles from blockchain technology, like decentralized storage and direct peer-to-peer sharing, to create a secure and trustworthy environment for data sharing. This helps ensure that users can share their data safely and privately without relying centralized systems.[1]

1.2 INTERPLANETARY FILE SYSTEM (IPFS)

The InterPlanetary File System (IPFS) is a decentralized storage that offers a reliable, distributed way to store and share data across the internet. Unlike traditional storage, where data is stored on a single server, IPFS breaks files into smaller pieces and distributes them across multiple computers, or nodes, all over the world. Each file stored on IPFS is assigned a unique identifier known as a Content Identifier (CID), which is like a digital fingerprint for that specific content. Rather than pointing to a location on a server, the CID directly points to the content itself, allowing users to retrieve the file where it is stored. When a file is added to IPFS, it's divided into chunks and stored across different nodes in the network, making the file accessible from multiple sources. If a node holding part of the file goes offline, the data can still be accessed from other nodes, ensuring high availability. To retrieve a file, users simply use the CID, and IPFS locates the nodes holding that file, allowing it to be downloaded efficiently, often from several sources simultaneously. This decentralized approach means that no single entity controls the user data, giving users greater control and security over their data.[1]

1.3 PEER-TO-PEER (P2P) NETWORK

Using Peer-to-Peer (P2P) network computers can connect directly with each other, instead of going through a central server. In a P2P network, each computer act as both a sender and a receiver, allowing data to to share and receive directly between users. This setup is commonly used for sharing files, media streaming, and decentralized applications, where each user can participate equally without relying on a single authority or host. When users want to share data in a P2P network, they simply send it to the intended recipient's computer. The data doesn't pass through an intermediary server, which makes the transfer faster and often more private. Since there's no central server to store the data, the network is generally more resilient. If one user is not connected to the network, others can still connect and share information. P2P communication will allow users to share data directly with each other, providing faster exchanges and reducing the costs and risks tied to centralized servers. This also means users have more control over their data, as it doesn't pass through or depend on any third-party provider. With P2P, the platform becomes more secure and efficient, supporting private and reliable data sharing that's ideal for a decentralized system[3]

1.4 MOTIVATION OF THE PROJECT

- As digital data sharing increases, there is a pressing demand for secure and private solutions to protect sensitive information. Many systems rely on centralized storage, making them vulnerable to data breaches and unauthorized access , BlockShare uses a decentralized approach, distributing data across a network to enhance security. Traditional storage methods often expose users to dangers like uncontrolled access, where unauthorized people can access sensitive information. This makes it hard for users to feel secure about their data, BlockShare empowers users by providing a decentralized platform where they retain full control over their data. This eliminates reliance on centralized storage, reducing the risk of data breaches and unauthorized access. Users often have limited control over their data, relying on third-party providers to manage access and permissions. BlockShare gives users full control over their data, allowing them to manage permissions independently. Different platforms may provide inconsistent user experiences, making data sharing cumbersome, BlockShare aims to deliver a seamless and intuitive user experience across all devices and interfaces.

CHAPTER 2

LITERATURE SURVEY

2.1 LITERATURE SURVEY

Thong et.al [1] explains blockchain's role in addressing data-sharing challenges, especially issues like data privacy, trust, and scalability. It gives understanding about how blockchain can be used for data sharing. It proposes a comprehensive framework, BlockDaSh, designed for secure and transparent data sharing across sectors such as healthcare, smart grids, and supply chains. The authors provide a detailed survey of blockchain-based data-sharing architectures and applications, emphasizing both benefits and limitations. Paper also identify open research challenges and suggest future research directions to improve the integration of blockchain in data sharing .

Jianping et.al [4] introduces a blockchain-based data-sharing platform for aviation suppliers to enhance transparency and reliability in quality data sharing. It explores about how blockchain can be used in aviation sector for secure data sharing. The proposed platform allows aviation manufacturers to securely share quality data, reducing inefficiencies and risks linked to traditional supply chains. The paper also outlines a layered data architecture, focusing on secure data acquisition, storage, and access, and demonstrates its application in aviation supply management.

Al-Zahrani et.al [3] proposes a subscription-based data-sharing model utilizing blockchain and Data as a Service (DaaS) to ensure secure, consistent, and accessible data exchange. The model addresses issues like data ownership, pricing strategies, and quality control by establishing a recurring revenue system for data providers. This paper propose a subscription-based data-sharing model utilizing blockchain i.e how platform should charge for data sharing utilizing blockchain. Results indicate that this model is both effective and efficient, offering a structured approach to data monetization while maintaining trust and transparency.This paper outlines how the platform can structure pricing and permissions for data sharing through blockchain, creating a transparent environment where data authenticity is preserved and every transaction is auditable.

Jaiman et.al [5] introduces a model for health data sharing that centres on patient consent using blockchain technology. It incorporates two ontologies—Data Use Ontology (DUO) and Automat-able Discovery and Access Matrix (ADA-M)—to match users' consent preferences with data requester intended use, ensuring compliance with consent terms. The model aims to support dynamic, GDPR-compliant data sharing with accountability, providing flexibility in decision-making for both data providers and requester.

Song et.al [6] explores blockchain's role in data sharing and exchange, particularly its advantages over traditional, centralized platforms. By offering decentralization, immutability, and enhanced privacy, blockchain is suited for a range of fields, including IoT, finance, and healthcare. The survey reviews various blockchain-based platforms, addressing architecture, security, and access control, as well as challenges like high transaction costs and privacy protection. The paper also examines methods for secure data marketplaces, highlighting the potential and ongoing limitations of blockchain in large-scale data sharing .

Yang et.al [2] explores the use of network coding to enhance throughput and reliability in peer-to-peer (P2P) file-sharing systems. By encoding and distributing files across multiple peers, the system increases efficiency, allowing any subset of peers with sufficient data to decode the original file. The proposed method, PPFEED, supports dynamic network conditions and shows a 15-20% improvement in throughput compared to traditional P2P networks .

Lu et.al [7] introduces a trust-based mechanism to protect user privacy in P2P data-sharing systems. Using a proxy system, trusted "buddies" act as intermediaries for data requests, ensuring that users' identities and interests remain hidden. The proposed mechanism introduces a proxy system where trusted "buddies" serve as intermediaries for data requests. The system also assesses trust dynamically, adjusting relationships based on past behavior to enhance privacy further while preserving system functionality.

Table 2.1: Comparative Analysis

Year	**Authors**	**Title**	**Methods/Algorithms Used**
2023	Linh T. Nguyen, Lam Duc Nguyen, Thong Hoang, Dilum Bandara, Qin Wang, Qinghua Lu, Xiwei Xu, Liming Zhu, Petar Popovski and Shiping Chen	Blockchain Empowered Trustworthy Data Sharing: Fundamentals, Applications, and Challenges	Reference Architecture, Privacy-Preserving Methods
2023	Pengyong Cao, Guijiang Duan, Jianping Tu, Qimei Jiang, Xianggui Yang and Chen Li	Blockchain Based Process Quality Data Sharing Platform for Aviation Suppliers	SM2 National Encryption Algorithm, Block Structure and Data Packaging
2020	Fahad Ahmad Al-Zahrani	Subscription-Based Data-Sharing Model Using Blockchain and Data as a Service	Pricing Models, Data as a Service (DaaS)
2020	Vikas Jaiman and Visara Urovi	A Consent Model for Blockchain-Based Health Data Sharing Platforms	Data Use Ontology (DUO), Access Control Mechanism
2023	Rui Song, Bin Xiao, Yubo Song, Songtao Guo, and Yuanyuan Yang	A Survey of Blockchain-Based Schemes for Data Sharing and Exchanges	Cryptographic Techniques, Access Control Models
2014	Min Yang and Yuanyuan Yang	Applying Network Coding to Peer-to-Peer File Sharing	Peer-to-Peer File Sharing

Year	Authors	Title	Methods/Algorithms Used
2006	Yi Lu, Weichao Wang, Bharat Bhargava, and Dongyan Xu	Trust-Based Privacy Preservation for Peer-to-Peer Data Sharing	Trust-based Privacy Preservation

CHAPTER 3

PROBLEM DEFINITION AND SCOPE

3.1 PROBLEM STATEMENT

In the digital age, the security and privacy of shared data has become a concern. Traditional data sharing systems often rely on centralized servers, which are susceptible to hacking, data breaches, and unauthorized access. This project aims to develop a blockchain-based secure data sharing platform that leverages the decentralized nature of blockchain technology to enhance data security and privacy.

3.1.1 Goals and objectives

- To understand the basics of blockchain and how blockchain works.
- To understand the security concepts related with blockchain and P2P communication.
- To create a decentralized system to eliminate the need for a central authority.
- To implement a P2P communication system to share data securly.
- To develop a secure data sharing platform utilizing blockchain technology.

3.1.2 Statement of scope

The scope of this project is to develop a secure, decentralized data-sharing platform utilizing peer-to-peer (P2P) communication and IPFS for data storage. Unlike traditional platforms, this solution eliminates the need for a centralized server or blockchain-based smart contracts, allowing users to share data directly and securely. The platform will support efficient data storage through IPFS, where files are given unique content identifiers (CIDs) for simplified retrieval. Users will be able to share data by directly exchanging these CIDs, ensuring that sensitive information is handled with greater privacy and control. The system is designed to handle files up to a practical size limit suitable for typical data-sharing scenarios, with validation to check for file type and size to prevent compatibility issues. Each file uploaded will generate a CID, which acts as the primary input, while the primary output will be a secure retrieval link for recipients to access shared data when online. The input/output flow will be straightforward: users upload files, obtain a CID, and share this directly over the P2P network for secure, decentralized access. This platform focuses on a set of specific capabilities. It includes P2P data sharing without intermediaries, secure storage through IPFS, and enhanced privacy by avoiding centralized control. However, it will exclude features such as smart contract-based access control and advanced access permissions. Instead, the platform emphasizes simplicity and efficiency for secure data sharing without the overhead and complexity of blockchain-based components, making it accessible for various users who need a reliable, privacy-focused data-sharing solution.

3.2 SOFTWARE CONTEXT

The software for this project is designed as a secure data-sharing platform aimed at individuals, businesses, or organizations needing a reliable way to share data or information without relying on centralized systems or servers. It fits within the larger context of privacy-focused communication tools, responding to the growing demand for secure, user-controlled data sharing.
Traditional data-sharing platforms often rely on centralized storage, and centralize systems which can be vulnerable to hacking, data breaches, or unauthorized access. In contrast, our platform uses decentralized storage and peer-to-peer (P2P) communication, making it part of the broader trend toward decentralized technologies. By allowing users to share data directly with one another, the software eliminates the need for intermediaries, offering users greater control and privacy.
The intended business application is to provide a secure, easy-to-use alternative to existing data-sharing solutions for scenarios where sensitive or confidential information is exchanged.
Whether it's individuals sharing personal files, teams collaborating on projects, or organizations handling client information, this platform allows users to retain control over their data while ensuring it is shared securely.

3.3 MAJOR CONSTRAINTS

- Network Dependence: Since the platform depends on a distributed network of nodes, it depends on network stability and speed. Poor connectivity or network disruptions could affect data availability and transmission speed.
- User Authentication and Access Control: Managing user identities securely is critical, as the platform will operate without traditional central control. Secure, reliable methods for authenticating users and controlling access to data must be implemented without compromising user experience.
- Storage Costs and Limitations: Storing data on IPFS or similar decentralized storage systems can incur costs for storing large size data, and there are storage limits for each data block. Efficiently managing and organizing data to minimize storage use and associated costs is a constraint to consider
- Message Queue Reliability: For cases when one user is offline, the platform will store CID if data in a message queue for later delivery. Ensuring the reliability of this queue system, even when network or node issues arise, is necessary for consistent data sharing.

3.4 METHODOLOGIES OF PROBLEM SOLVING AND EFFICIENCY ISSUES

- Decentralized Storage : IPFS (InterPlanetary File System) is chosen for its decentralized storage capability, where data is distributed across a network of nodes. This avoids single points of failure, unlike traditional cloud storage solutions.
- Efficiency Issue : The decentralized nature of IPFS can lead to slower data retrieval under high network traffic or when nodes are unavailable.
- P2P Communication : P2P communication allows direct data sharing between users, eliminating intermediaries and enhancing privacy. This eliminates dependency on a centralized systems, which will provide data privacy.

- Efficiency Issue : In P2P systems, speed of the data transfer can vary based on network conditions and the quality of each participant's connection, which may impact the speed of data sharing.
- Message Queue : To handle situations where data receiver is offline, a message queue can store data references (such as IPFS hashes) for delayed delivery. This ensures that data sharing can resume when the receiver reconnects.
- Efficiency Issues : Maintaining reliable queues in a decentralized environment can be complex, especially when ensuring data is retained long enough for offline users while keeping the process resource-efficient.

3.5 SCENARIO IN WHICH MULTI-CORE, EMBEDDED AND DISTRIBUTED COMPUTING USED

- Distributed Computing: As blockchain-based data-sharing platform relies on decentralized storage and peer-to-peer communication, which are key aspects of distributed computing. The platform uses distributed computing by leveraging a network of multiple independent nodes to store and share data. Instead of relying on a central server, data is divided and stored across various nodes in a peer-to-peer (P2P) network. The platform utilizes distributed architecture to ensure data availability, security, and efficiency in a peer-to-peer network.
- Multi-Core Computing: Multi-Core Computing may be not directly relevant, if platform involves complex computations, such as encryption or hashing, where multi-core processing could improve performance. However, for a data-sharing platform focused on secure transfer, the relevance of Multi-Core Computing may be limited unless there's substantial computational processing.
- Embedded Computing: Embedded Computing is less relevant to this project unless and until platform will eventually integrate with IoT devices or specialized hardware to facilitate data sharing but currently platform does not use such devices.

3.6 OUTCOME

The outcome of this project will be a secure data-sharing platform built on top of blockchain concepts, that enables users to share information directly without relying on central authorities. This platform will address key issues like data privacy, control, and resilience against unauthorized access or data loss.This approach not only enhances privacy and security but also reduces costs by removing the need for intermediaries. Ultimately, this project provides a trustworthy and efficient solution for individuals and organizations seeking to exchange data in a safe, user-controlled environment.

3.7 APPLICATIONS

Secure Data Sharing : Individuals and Organizations can use this platform to securely share sensitive information like personal info, financial records, medical data, legal documents, or customer data without relying on a central authority, reducing risks of data loss and increase data privacy.

3.8 HARDWARE RESOURCES REQUIRED

Table 3.1: Hardware Requirements

Sr. No.	Parameter	Minimum Requirement	Justification
1	CPU Speed	2 GHz	Optimal Performance.
2	RAM	3 GB	Sufficient RAM

3.9 SOFTWARE RESOURCES REQUIRED

Platform :

1. Operating System: Windows/Linux/Mac
2. IDE: Visual Studio Code.
3. Programming Language : JavaScript

CHAPTER 4

SOFTWARE REQUIREMENT SPECIFICATION

4.1 INTRODUCTION

4.1.1 Purpose and Scope of Document

The Software Requirements Specification (SRS) document for the BlockShare Data Sharing Platform outlines the system's purpose, requirements, and functions, serving as a crucial tool for communicating project goals, aligning all stakeholders, managing changes, and guiding the development team in building a secure, decentralized file-sharing solution. This SRS covers both functional and non-functional requirements, such as secure file handling, data protection, and user authentication, as well as details on how the system will integrate with existing infrastructure. By clearly defining these elements, the document ensures that the platform meets stakeholder expectations and provides a reliable method to track performance effectively.

4.1.2 Overview of responsibilities of Developer

The role of developers in building the BlockShare decentralized data-sharing platform is multifaceted, involving a comprehensive range of tasks to ensure the platform's functionality, security, and scalability. Developers are entrusted with creating a user-centric, efficient, and secure environment for data sharing, leveraging blockchain technology and decentralized storage. From analyzing project requirements and translating them into actionable features to rigorously testing the system for quality and security, developers play a pivotal role in aligning the platform with its goal of providing a trustworthy, secure, peer-to-peer data-sharing solution. The following is a summary of the primary responsibilities:

1. Requirement Analysis and Documentation: Understand and document functional and non-functional requirements.

2. System Design: Develop a robust architecture focused on decentralized storage, data security, peer-to-peer communication, message queue and peer-to-peer based data-sharing mechanisms.

3. Core Feature Development: Implement primary features, including secure

user authentication, permission management, data encryption, and peer-to-peer file sharing.

4. Network and Security Implementation: Integrate security protocols and optimize peer-to-peer networking to ensure data privacy and secure data transmission.

5. Testing and Quality Assurance: Perform unit, integration, and security testing to maintain reliability, performance, and data integrity.

6. Performance Optimization and Maintenance: Enhance platform performance, address bugs, and ensure scalability, while also managing ongoing maintenance and updates.

4.2 FUNCTIONAL REQUIREMENTS

4.2.1 System Feature 1(User Authentication)

User authentication is essential, ensuring secure access and protecting sensitive data. Users register with a username, email and password, which are securely stored. The system verifies credentials during login and offers multi-factor authentication (MFA) for added security through a one-time code. Users can also recover forgotten passwords via a secure process. This feature enhances security, empowers user control, and builds trust in the platform.

4.2.2 System Feature 2 (Sending and Retrieving Files)

The ”Sending and Retrieving Files” feature in BlockShare enables users to securely share and access files on a decentralized platform. Users can select and encrypt files for upload to the InterPlanetary File System (IPFS), sharing a unique IPFS content identifier (CID) via a peer-to-peer network powered by libP2P. Recipients can quickly retrieve the shared file by entering the CID into the platform, with access restricted to authorized users only. Users can set permissions to control who can view the files and receive notifications when their files are accessed or downloaded. This feature simplifies the process of sending and retrieving files while maintaining user control and data privacy.

4.2.3 System Feature 3 (Activity Logging and Monitoring)

The ”Activity Logging and Monitoring” feature in BlockShare allows users to view the history of file sharing and retrieval activities. It records details of past interactions, including who accessed the files and when they were shared or retrieved. Users can easily access this information through the platform, providing transparency about file access. Notifications for significant actions, like when files are shared or retrieved, ensure users are kept informed. This feature enhances user trust by offering clear visibility into the activities surrounding their shared data.

4.3 EXTERNAL INTERFACE REQUIREMENTS (IF ANY)

4.3.1 User Interfaces

The BlockShare platform will feature a user-friendly web interface, allowing users to easily navigate the system. The UI will support file uploading, sharing, and accessing functionalities, ensuring a seamless experience.

4.3.2 Hardware Interfaces

The Platform will be accessible on desktops and mobile devices, requiring standard internet connectivity. Minimal hardware requirements include a computer or smartphone with internet access, enabling users to send, retrieve, and manage files.

4.3.3 Software Interfaces

The BlockShare system will utilize IPFS for decentralized file storage and libP2P for peer-to-peer networking, facilitating secure file sharing. Integration with web frameworks will enable smooth user interactions and data management. Additionally, the platform will provide notification services to keep users informed about important activities related to their shared files.

4.3.4 Communication Interfaces

A P2P network in BlockShare enables decentralized file sharing by allowing communication between user devices. Each file-sharing transaction is validated by the participating nodes before being recorded in the system, ensuring secure and reliable data management.

4.4 NONFUNCTIONAL REQUIREMENTS

4.4.1 Performance Requirements

- To provide a seamless user experience, the system must ensure that file uploads and retrievals are completed within few minutes, allowing users to share and access files quickly.
- To keep performance steady during busy times, the platform should be able to manage high loads, ensuring that response times stay consistent even when many users are active.
- BlockShare needs to maintain high uptime, ensuring users can access the service whenever they need it.

4.4.2 Safety Requirements

- To restrict access to data, the platform must implement strong access control measures, ensuring that only authorized users can share or retrieve files.
- To enhance user trust, the platform must clearly communicate its data protection policies and practices, ensuring users understand how their data is safeguarded.
- The platform must clearly share the user data securely via P2P network providing a secure method to share the data.
- To keep the system secure, regular security checks should be done to find and fix any weaknesses.

4.4.3 Security Requirements

- To protect user privacy, the platform should hide user data whenever possible and reduce the sharing of personal information.
- To log all user activities related to file sharing, the system must maintain an audit trail that records who accessed or modified files, providing accountability.
- To control who can see files, the system must have access controls that limit file sharing.

4.4.4 Software Quality Attributes

- Usability : The system should be easy to use for all types of users, with a simple interface that allows them to send and retrieve files without difficulty.
- Reliability : BlockShare must consistently perform its functions without failures, ensuring users can access their files and data when needed.
- Portability : BlockShare must work across different devices and platforms, ensuring users can access it from desktops, laptops, and mobile devices.
- Maintainability : The system should be easy to update and fix, so developers can make improvements or fix problems quickly.

4.5 SYSTEM REQUIREMENTS

4.5.1 Database Requirements

The platform uses IPFS (InterPlanetary File System) for storing files safely and in a decentralized way, keeping file details and user information secure. IPFS is a decentralized storage that offers a reliable, distributed way to store and share data across the internet. To retrieve a file, users simply use the CID, and IPFS locates the nodes holding that file, allowing it to be downloaded efficiently, often from several sources simultaneously. IPFS should support efficient retrieval and linking of files using unique content identifiers (CIDs), enabling quick access to shared files across the network.

4.5.2 Software Requirements(Platform Choice)

- P2P Communication Framework:
  The core of this platform relies on peer-to-peer (P2P) communication to enable secure and direct data sharing between users without relying on central servers. The LibP2P framework is used to facilitate P2P communication. LibP2P is popular P2P framework provides essential tools for establishing and managing P2P networks, including peer discovery, connection management, and secure message transfer, all crucial for a seamless user experience.

- Frontend Frameworks:
  The frontend of the application is developed using React.js. React's component-based structure simplifies the development of a fast, interactive user interface (UI) by allowing developers to create reusable components. For styling, Tailwind CSS is employed to provide a modern, consistent look and feel. Tailwind offers a utility-first approach, enabling rapid UI development while ensuring that design elements are uniform throughout the application. By combining React and Tailwind CSS, the platform delivers an attractive, intuitive interface that users can navigate with ease.

- Backend Frameworks and Libraries:
  In addition to the frontend, certain backend components are essential to support secure data management and storage. A combination of Node.js and Express is used for server-side processing, handling user requests, and managing backend interactions, such as sending and retrieving data from IPFS or adding a IPFS hash to the maessage queue if reciver is offline. This setup provides scalability and robustness for the platform, ensuring it can handle multiple user requests simultaneously.

- Decentralized Storage (IPFS):
  IPFS (InterPlanetary File System) is integrated into the platform to enable

decentralized data storage. When users share data, it is uploaded to IPFS, which assigns a unique content identifier (CID) to each file, ensuring secure storage and retrieval. IPFS helps reduce dependency on a single server and enhances data availability, as files are distributed across a network.

- Version Control and Collaboration:
  For tracking changes, updates, and managing the codebase efficiently, Git is utilized as the version control system. Git allows developers to maintain a history of code changes, making it easier to roll back if necessary and improving collaboration across the team. GitHub's tools help streamline development by allowing team members to review each other's code, report bugs, and document enhancements, ensuring smooth collaboration and continuous improvement of the platform.

- Development Environment and Deployment Tools:
  The code is primarily written and managed in Visual Studio Code (VS Code), a popular development environment known for its user-friendly interface and extensive features. VS Code offers powerful extensions for syntax highlighting, debugging and code formatting, as well as built-in Git integration. These features help improve productivity and efficiency for the development team.

- Testing Tools:
  Testing is a critical step in ensuring the reliability and stability of the platform. Selenium is used for testing the web application to verify that each component functions as expected. Selenium automates browser actions. This helps identify and fix potential bugs before deployment, leading to a better experience for end-users. Additional some another testing libraries or testing tools may be used for unit and integration testing to ensure individual components work seamlessly with one another.

4.5.3 Hardware Requirements

- Processor: A modern multi-core processor (Intel i5 or equivalent AMD Ryzen 5) is recommended to handle the demands of development, testing, and debugging of the application.

- RAM: At least 8GB of RAM is required to manage multiple applications simultaneously, but 16GB or more will provide an improved experience, especially when running development environments and testing IPFS or peer-to-peer communications.
- Storage: A minimum of 256GB SSD storage is recommended for faster data access and quicker application load times. This will help streamline code compilation, dependency management, and test execution.
- Network Connectivity: Stable and high-speed internet (at least 10 Mbps) is necessary for smooth access to IPFS nodes, GitHub repositories, and any external resources required during the development process.

4.6 ANALYSIS MODELS: SDLC MODEL TO BE APPLIED

Why Agile Model?
The Agile model adopts an attitude of flexibility and responsiveness to change, redefining traditional software development. The core tenet of Agile is the division of big, complicated projects into smaller, more manageable work units called iterations or sprints, each of which produces a working piece of software. Agile's adaptability helps teams avoid the danger of investing a lot of effort in features that can become outdated or out of step with changing client or market demands. Agile also promotes an empowered and accountable culture where team members are encouraged to self-organize and have a great deal of autonomy, which raises engagement and productivity. Agile places a strong emphasis on collaboration through techniques like planning sessions, retrospectives, and daily stand-up meetings that foster communication and keep teams focused on their goals. Agile also places a strong emphasis on quality, frequently integrating testing into the development process to identify and fix problems early and provide a more dependable final product. Agile's iterative structure and constant emphasis on quality allow for the incorporation of any necessary modifications without completely reworking the project,

which saves money and time on rework. Users are kept engaged at every stage by Agile's customer-centric methodology, which guarantees that the finished product meets user expectations and adds company value at every iteration. Teams working in Agile environments can pivot quickly and re-prioritize tasks, keeping the project on track while responding effectively to change. With its focus on delivering value consistently and its adaptive planning techniques, Agile not only enhances productivity but also fosters a positive, collaborative work environment. This has led many organizations to adopt Agile practices not only for software development but also for broader project management, recognizing that Agile's principles of adaptability, customer satisfaction, and collaborative problem-solving are universally valuable across industries.[8]. Agile model is represented as shown in the figure 4.1 .

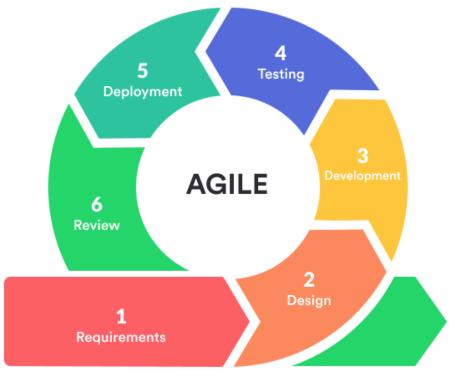


Figure 4.1: Agile Model
[8]

Agile model also focuses on direct customer communication instead of making documentation. Due to these, environment developer team also get a opportunity to engage with the client presents a valuable chance to enhance the project outcome and due to these Customer interaction presents an opportunity to minimize confusion and ensure clarity an back- bone of Agile methodology.

- Agile model steps:

  1. Requirements Collection: The first stage of the project development focused on gathering critical information to define the platform's goals and

functionality. This involves identifying the security features needed to build a secure, user-friendly file-sharing platform. To conduct detailed research to understand user needs, aiming to provide a solution that prioritizes data privacy, user control, and accessibility. Additionally, exploring various pinpoint that capture essential aspects of the file-sharing process, covering stages from file upload to retrieval. Based on this information, carefully prioritizing features and narrowed the project scope to balance functionality with performance. This ensures that platform will meet user expectations while being adaptable across different environments.

2. Design: After a detailed understanding of the requirements, the next phase is design, where these insights are translated into a detailed architecture for the platform. In this phase, various UML diagrams, including Class, Activity, Sequence and Data Flow Diagrams, to understand the relationships between system components. These diagrams play an essential role in visually outlining how data moves through the system, showing each component's role in the file-sharing and retrieval processes. For instance, Sequence Diagrams illustrate the order of interactions, ensuring each step is accounted for, while Class Diagrams define the data structures used across the platform. This visual blueprint not only guides the development team by offering a high-level view of system interactions but also highlights potential integration points with external services, such as IPFS and the LibP2P framework. The design phase also helps identify any technical challenges early on, such as security vulnerabilities or potential scalability issues. By addressing these challenges in advance, it will be helpful to create a resilient and scalable system that can efficiently handle the demands of secure, peer-to-peer data sharing. This phase sets a solid foundation for the development and ensures a streamlined approach to building each part of the platform.

3. Implementation of Project: The development of the BlockShare system begins in this stage. To securely share, manage and track file sharing ac-

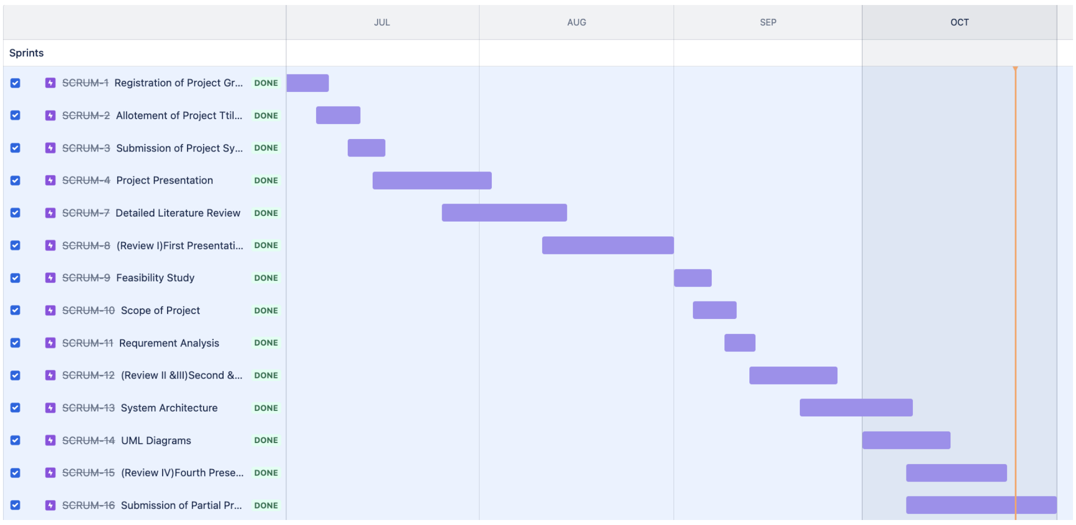
tivities, the team will establish a decentralized system using IPFS for data storage and a P2P network for communication. The Agile methodology is followed to iteratively develop and test components, ensuring data responsiveness and accuracy throughout the implementation process. Initially, the core functionalities will be implemented for file sending and retrieval, gradually adding features like activity logging and monitoring.

4. Testing: Testing is crucial to ensure the functionality and reliability of the system. Throughout this phase, conducting various tests to verify that the system handles data correctly and accurately reflects file sharing activities at every stage. The primary focus of testing will include quality checks on data inputs and outputs, ensuring security measures are effective.

5. Deployment: After thorough testing, the system is ready for deployment. This allows to observe how the file-sharing system performs in practical scenarios and make any adjustments to ensure a smooth working of system. The deployment process guarantees that the system is responsive, fully operational, and equipped to handle broader user adoption.

4.7 SYSTEM IMPLEMENTATION PLAN:

A Gantt chart is a visual project management tool used to plan, schedule, and track the progress of tasks or activities over time. It helps project managers and teams monitor progress, manage time effectively, and ensure that tasks are completed on schedule.Each task is represented by a horizontal bar, with the length of the bar indicating the task's duration, while the position on the timeline shows its start and end dates. Gantt charts help project managers and teams not only to plan and allocate resources but also to identify dependencies between tasks, monitor project progress, and make adjustments in real time as needed. Gantt charts are particularly useful for large, complex projects as they provide a clear overview of each phase and its relationship to other phases. Many Gantt chart tools also include color coding, milestone markers, and progress indicators, which provide visual cues about the status of each task and make it easy to assess overall project health at a glance. Timeline chart of Blockshare is as shown in the figure 4.2 .



Sprints

~~SCRUM-1~~ Registration of Project Gr... DONE
~~SCRUM-2~~ Allotement of Project Ttil... DONE
~~SCRUM-3~~ Submission of Project Sy... DONE
~~SCRUM-4~~ Project Presentation DONE
~~SCRUM-7~~ Detailed Literature Review DONE
~~SCRUM-8~~ (Review I)First Presentati... DONE
~~SCRUM-9~~ Feasibility Study DONE
~~SCRUM-10~~ Scope of Project DONE
~~SCRUM-11~~ Requrement Analysis DONE
~~SCRUM-12~~ (Review II &III)Second &... DONE
~~SCRUM-13~~ System Architecture DONE
~~SCRUM-14~~ UML Diagrams DONE
~~SCRUM-15~~ (Review IV)Fourth Prese... DONE
~~SCRUM-16~~ Submission of Partial Pr... DONE

Figure 4.2: Timeline Chart

Here's a quick overview of a Gantt chart:

1. Task Labels: On the left side, various project tasks (sprints) are listed, such as Registration of Project Group, Project Presentation, Feasibility Study, System Architecture, and UML Diagrams.

2. Timeline: The timeline spans from July to October, showing the time period during which each task was planned and completed.

3. Progress Status: The purple bars indicate the duration and completion of each task. All tasks appear to be marked as "DONE," reflecting their completion.

4. Sequential Tasks: The tasks are completed in a sequence, showing how different stages of the project were scheduled, from the initial registration and project title allotment in July, to later tasks like UML diagrams and submission of the partial project report in October.

CHAPTER 5

METHODOLOGY AND SYSTEM DESIGN

5.1 SYSTEM ARCHITECTURE

An architecture diagram is a visual representation of a system's design. It shows how different parts of the system are organized, connected, and interact with each other. Architecture diagrams are helpful because they provide a clear overview of the system, making it easier to understand how everything works together. Figure 5.1 shows the architecture of the platform.

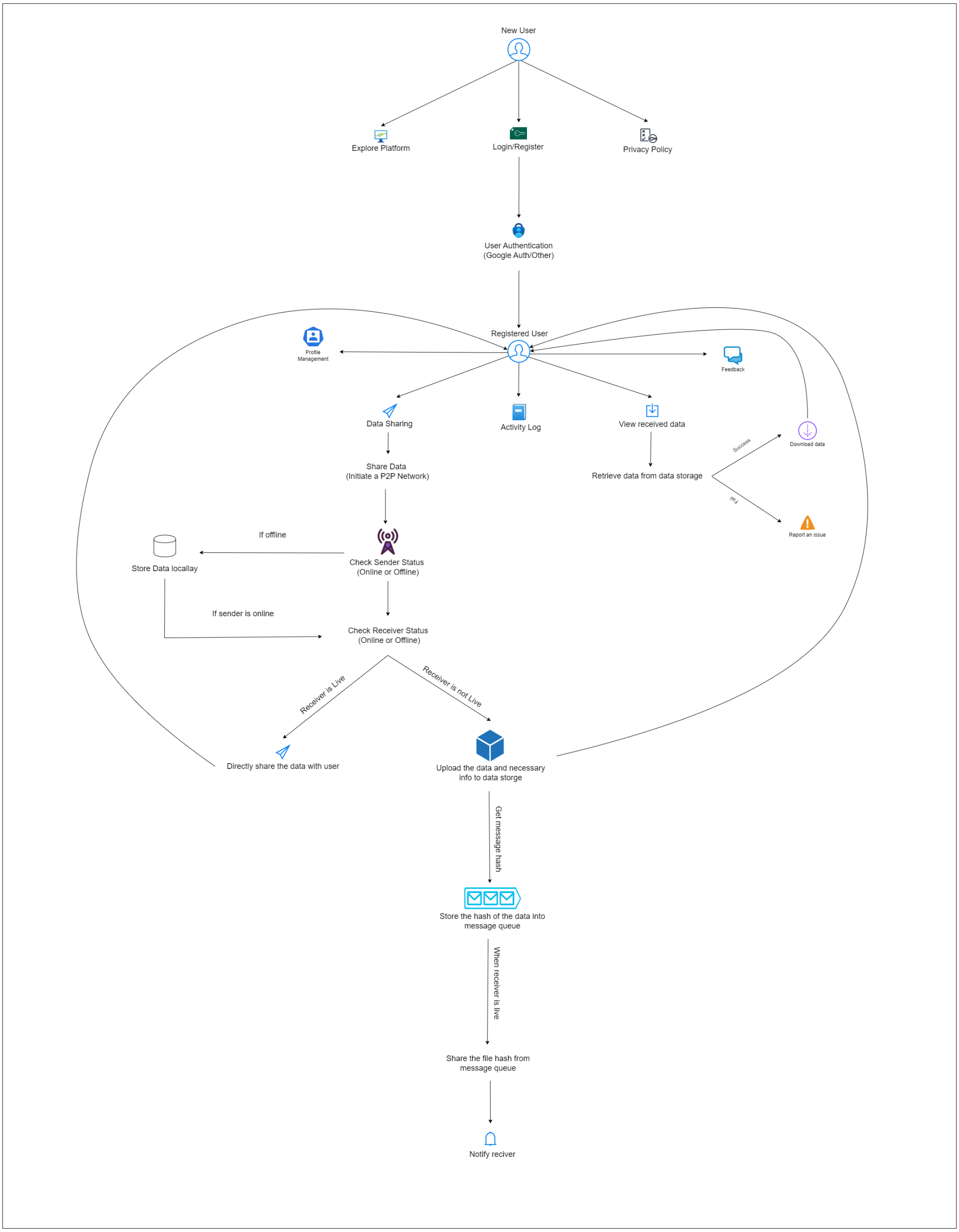

New User
Explore Platform
Login/Register
Privacy Policy
User Authentication (Google Auth/Other)
Registered User
Profile Management
Feedback
Data Sharing
Activity Log
View received data
Download data
Success
Fail
Report an issue
Share Data (Initiate a P2P Network)
Retrieve data from data storage
If offline
Store Data locallay
Check Sender Status (Online or Offline)
If sender is online
Check Receiver Status (Online or Offline)
Receiver is Live
Receiver is not Live
Directly share the data with user
Upload the data and necessary info to data storge
Get message hash
Store the hash of the data into message queue
When receiver is live
Share the file hash from message queue
Notify reciver

Figure 5.1: System Architecture

5.1.1 Platform's Flow

- **New User Flow:**New users can explore the platform, register/login, or view the privacy policy.
- **Registered User Flow:** Registered users have access to profile management, data sharing, activity logs, and can view received data.
- **Data Sharing Flow:** The platform first checks if the sender is online.
  If offline: Data is stored locally until the sender is online.
  If the sender is online, the receiver's status is checked.
  If the receiver is online: Data is shared directly via a P2P network.
  If the receiver is offline: Data is uploaded to IPFS for storage, and the hash of the data is placed in a message queue.
- **Data Retrieval:** Once the receiver receive data it be downloaded from IPFS.

5.2 DATA FLOW DIAGRAMS

5.2.1 DFD Level 0

The Level 0 DFD provides an overview of the entire data-sharing platform's interactions with external entities.

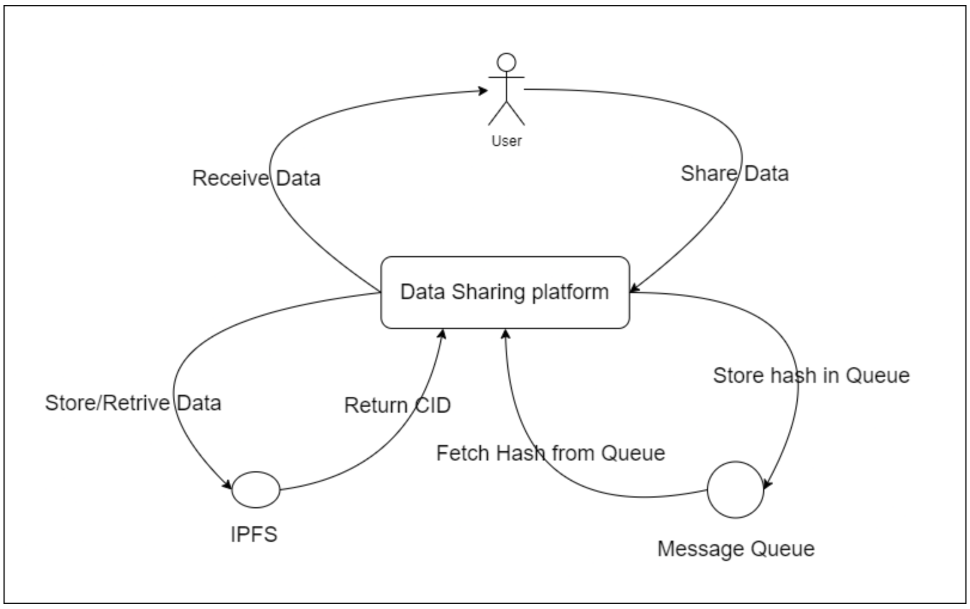


Figure 5.2: DFD Level 0

The user can perform actions such as ”Share Data” and ”Receive Data” through the platform . For sharing data, the platform interacts with IPFS for data storage, obtaining a unique content identifier (CID) to represent the stored file. Additionally, a message queue system manages the transfer of the CID to offline users, ensuring data is accessible once they come online as shown in the figure 5.2 . This high-level diagram illustrates how data flows between the user, the data-sharing platform, IPFS, and the message queue, forming the core of the platform’s functionality.

5.2.2 DFD Level 1

The Level 1 DFD breaks down the platform’s primary functionalities of data sharing and receiving in greater detail.

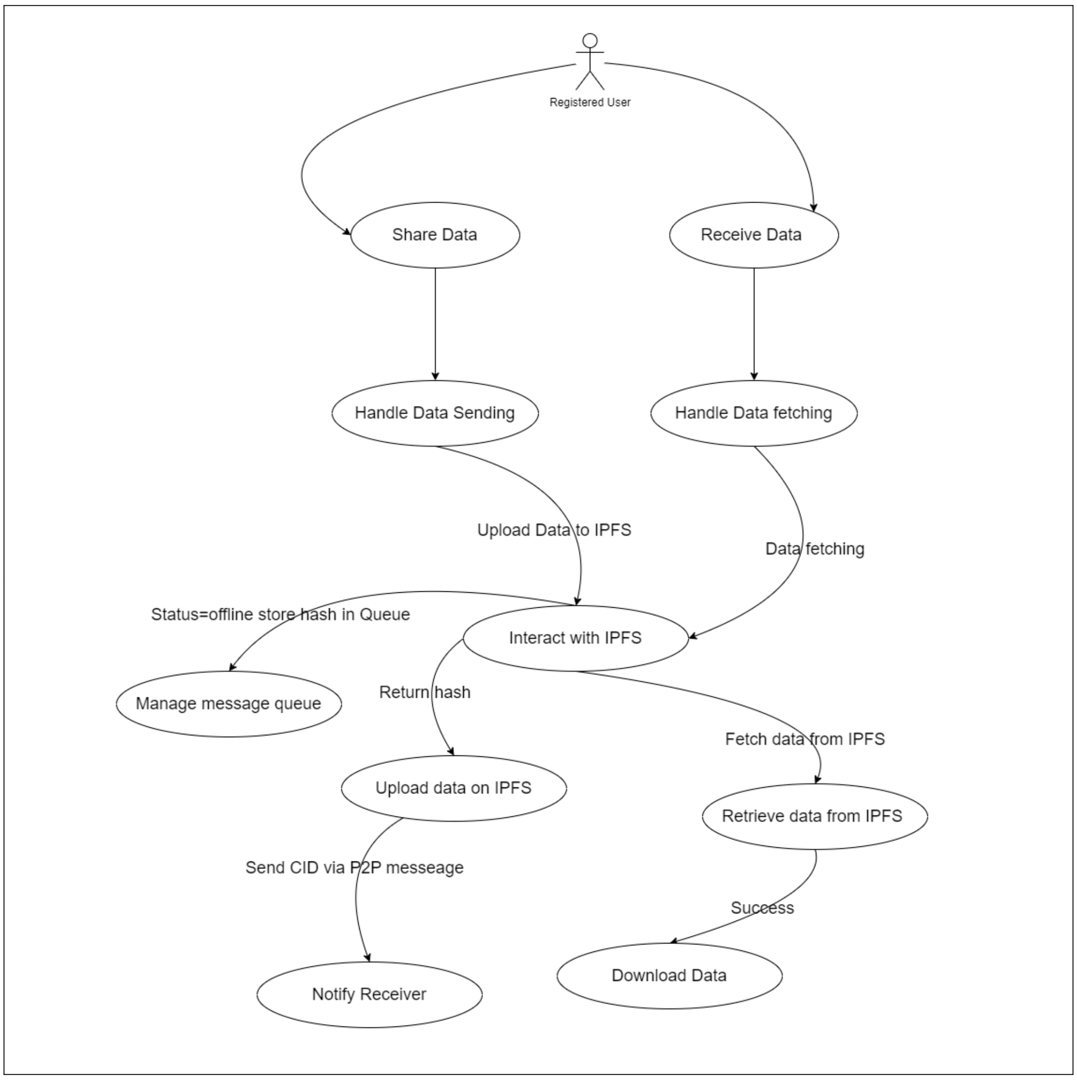


Figure 5.3: DFD Level 1

When a user shares data, the system first handles data storage on IPFS and generates a CID, which is stored and queued for delivery if the recipient is offline. For data retrieval, the user can fetch data directly from IPFS using the received CID. This level highlights essential processes like handling data transmission, managing the message queue, and interacting with IPFS, showing the system's inner workings during data upload and retrieval processes as shown in the figure 5.3 .

5.2.3 DFD Level 2

The Level 2 DFD dives deeper into the technical steps involved in sharing and receiving data. It starts with user authentication, followed by data selection for sharing.

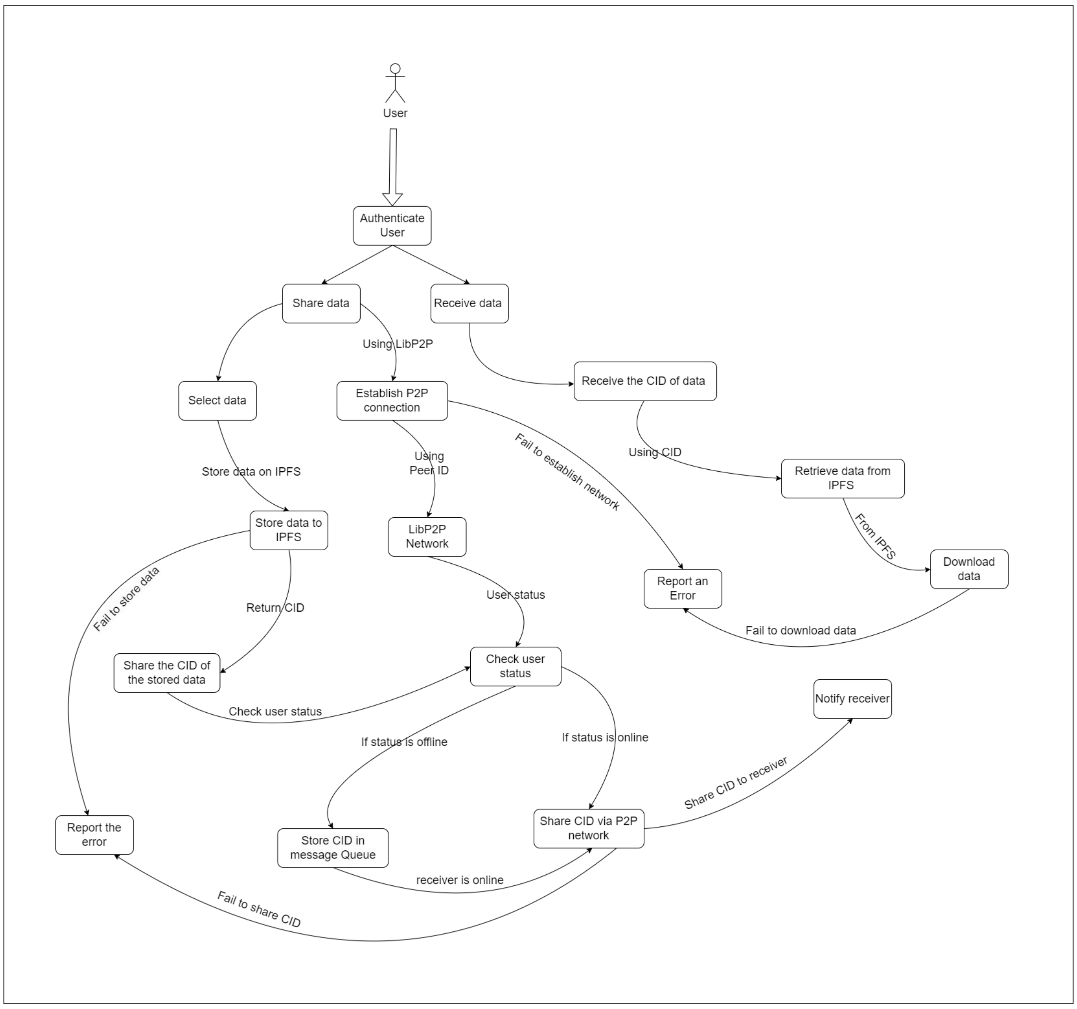


Figure 5.4: DFD Level 2

Once the P2P connection is established (using protocols like LibP2P), the sender's and receiver's statuses are checked. If both are online, data is directly shared via P2P; if not, the data is uploaded to IPFS, and the CID is stored in the message queue for future access. The retrieval process involves checking the CID in the message queue, fetching data from IPFS, and notifying the user upon successful download as shown in the figure 5.4 . This level provides a detailed view of each process step, error handling, and interactions within the platform, covering every detail from user authentication to data download and error reporting.

5.3 ENTITY RELATIONSHIP DIAGRAMS

An Entity-Relationship (ER) Diagram is a visual representation used to model the structure of a database. It defines the relationships among entities (objects or concepts) within a system, illustrating how data is connected and organized.

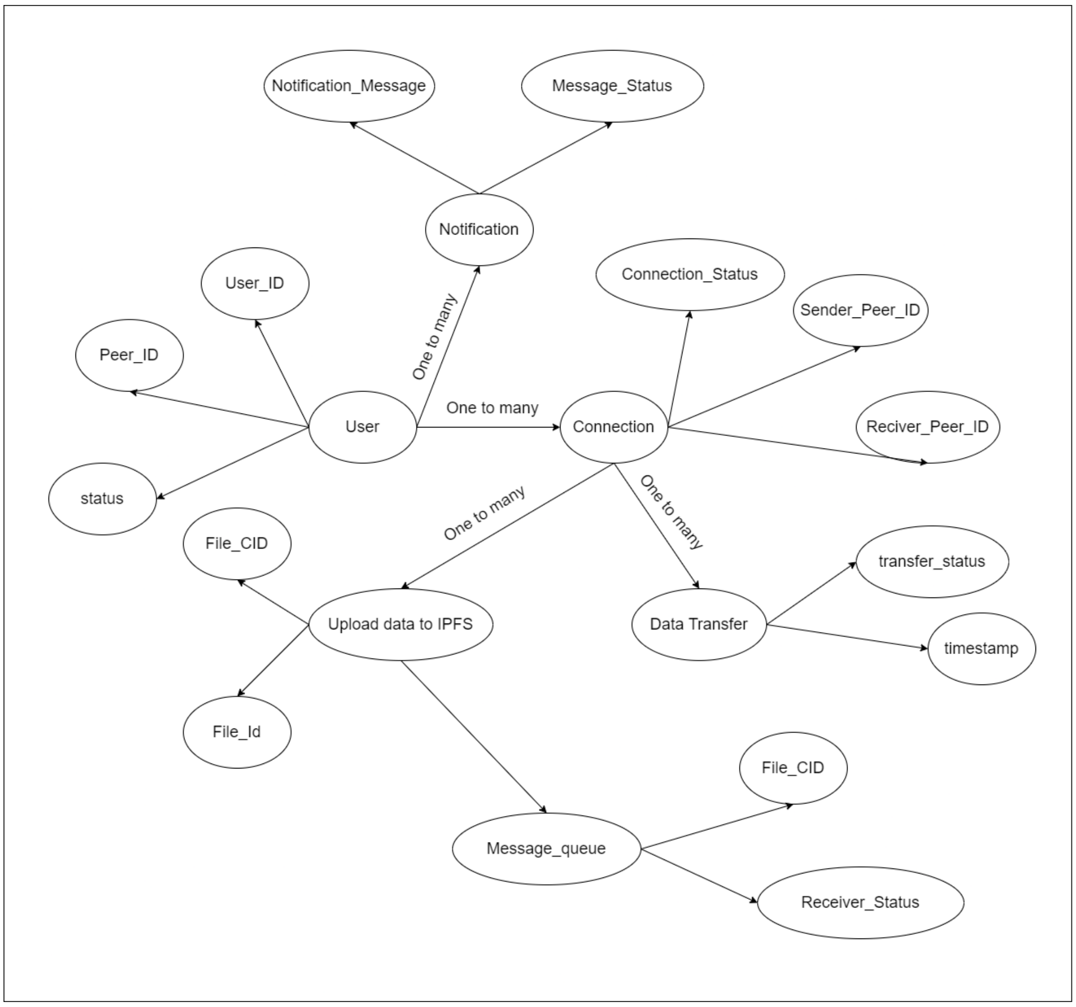


Figure 5.5: Entity Relationship Diagram

Figure 5.5 shows the ER diagram for platform. The primary entities involved are User, Connection, Notification, Upload Data to IPFS, Data Transfer, and Message Queue. The User entity represents individuals in the platform, uniquely identified by a User ID and connected to other users through Peer ID. Each user can have multiple Connections with other peers, allowing them to share data. The Connection entity includes attributes such as Sender Peer ID and Receiver Peer ID, which denote the peers involved, along with a Connection Status that indicates whether the connection is active or inactive.

Users can receive various Notifications containing a Notification Message and Message Status, alerting them about new updates or changes. Users are also associated with the Upload Data to IPFS entity, which handles data uploads to the InterPlanetary File System (IPFS), where each uploaded file is represented by a unique File CID (Content Identifier) and File ID. This entity establishes a one-to-many relationship with Data Transfer, indicating that multiple transfers may occur for each uploaded file.

The Data Transfer entity is responsible for tracking file sharing between users. It records details like the transfer status, which shows the current state of the data transfer and a timestamp to log the time of transfer. Finally, the Message Queue entity manages messages queued for delivery, linked to both the File CID and the Receiver Status to track whether the message has been successfully received or is pending. Overall, the diagram highlights a layered structure where users can establish connections, receive notifications, upload files, transfer data, and queue messages within a decentralized platform.

5.4 UML DIAGRAMS

5.4.1 Class Diagram

Class objects in a model system are represented visually by class types in a class diagram. In a class diagram, each class is depicted with its attributes and methods, helping to show the data it holds and the actions it can perform. This visual representation is utilized in the UML to show the dependencies, relationships, and interactions between classes, making it easier to understand how different parts of the system communicate and work together. Class diagrams are essential for designing complex systems, as they provide a clear overview of how objects are organized and help identify any structural issues early in the development process. Figure 5.6 shows the class diagram for platform.

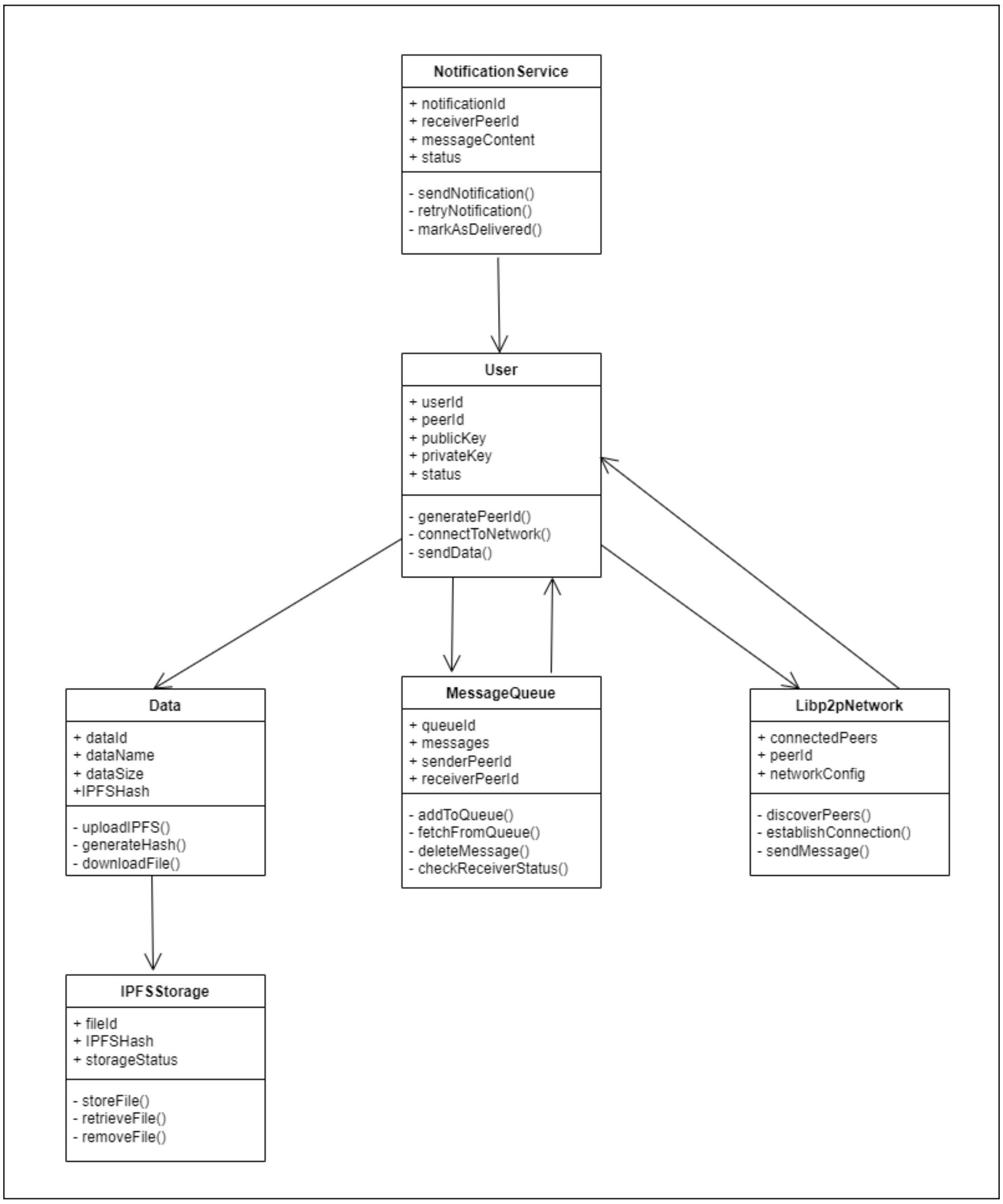


Figure 5.6: Class Diagram

- **Notification Service Class:**

  This class manages notifications sent to users. It includes attributes like notificationId, receiverPeerId, messageContent and status to track notification details. The class has methods for sending notifications (sendNotification()), retrying failed notifications (retryNotification()) and marking notifications as delivered (markAsDelivered()).

- **User Class:**

  The User class represents an individual user in the network, identified by attributes such as userId, peerId, publicKey, privateKey and status. The methods in this class include generatePeerId() to create a unique peer ID, connectToNetwork() to join the network and sendData() to transmit data to others.

- **Data Class:**

  This class handles data-related activities, storing information such as dataId, dataName, dataSize and IPFSHash (unique identifier for IPFS storage). It includes methods to upload data to IPFS (uploadIPFS()), generate a hash for data verification (generateHash()) and download files (downloadFile()).

- **IPFSStorage Class:**

  This class manages storage on IPFS, identified by attributes like fileId, IPFSHash and storageStatus. The class provides methods for storing files on IPFS (storeFile()), retrieving files (retrieveFile()) and removing files (removeFile()). IPFSStorage: This class manages storage on IPFS, identified by attributes like fileId, IPFSHash and storageStatus. The class provides methods for storing files on IPFS (storeFile()), retrieving files (retrieveFile()) and removing files (removeFile()).

- **MessageQueue Class:** The MessageQueue class is responsible for storing messages to be delivered to peers. It contains attributes such as queueId, messages, senderPeerId and receiverPeerId. The methods include addToQueue() to queue a new message, fetchFromQueue() to retrieve a message, deleteMessage() to remove a message and checkReceiverStatus() to verify if the recipient has received the message.

- **Libp2pNetwork:** This class facilitates network connections between users. It has attributes connectedPeers, peerId and networkConfig to manage network settings. The methods include discoverPeers() to find available peers, establishConnection() to set up a connection and sendMessage() to transmit messages to other peers.

5.4.2 Object Diagram

An object diagram is a type of diagram used in UML (Unified Modeling Language) that shows a snapshot of instances of classes (objects) in a system at a specific point in time. It represents a static structure, showing how objects interact, their relationships, and their states at a particular moment. Figure 5.7 shows the object diagram for the platform.

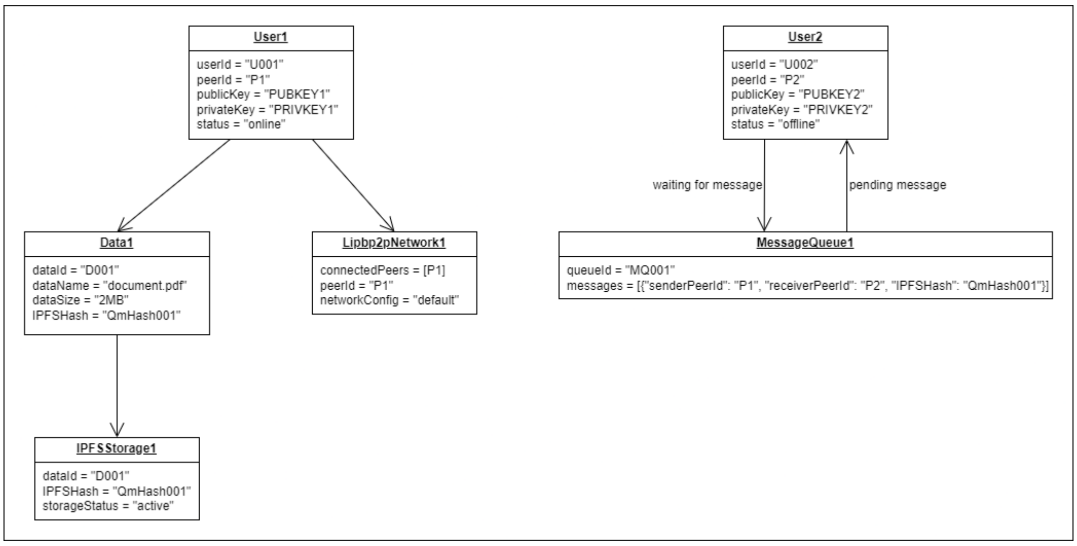


Figure 5.7: Object Diagram

In this scenario, User1 and User2 are two users on the network. User1, who has attributes like userId ”U001”, peerId ”P1”, publicKey ”PUBKEY1”, privateKey ”PRIVKEY1” and status ”online”, is actively participating in the network, while User2, identified by userId ”U002”, peerId ”P2”, publicKey ”PUBKEY2”, privateKey ”PRIVKEY2” and status ”offline”, is currently waiting to receive a message.
Data1 represents a specific file uploaded by User1 to the network. This file, named ”document.pdf”, is 2MB in size and has a unique IPFS identifier QmHash001, which allows it to be stored on the IPFS network. IPFSStorage1 represents the storage location of Data1 on IPFS, where it is tracked with fileId ”D001” and marked with a storageStatus of ”active,” signifying that the file is successfully stored and accessi-

ble. The Libp2pNetwork1 object handles the network connection for User1. With peerId ”P1” and a networkConfig set to ”default”, Libp2pNetwork1 enables User1 to connect to the network and interact with other peers. Currently, connectedPeers includes only User1's own peerId, indicating readiness to communicate.
The MessageQueue1 object stores messages awaiting delivery. With a queueId ”MQ001”, it contains a message from User1 (sender peerId ”P1”) intended for User2 (receiver peerId ”P2”). The message references Data1 via its IPFSHash ”QmHash001”, allowing User2 to access the file once online.

5.4.3 Use Case Diagram

An illustration of a user's potential interactions with a technology is called a use case diagram. Either circles or ellipses are used to symbolize the use cases. Figure 5.8 shows the use case diagram for the platform.

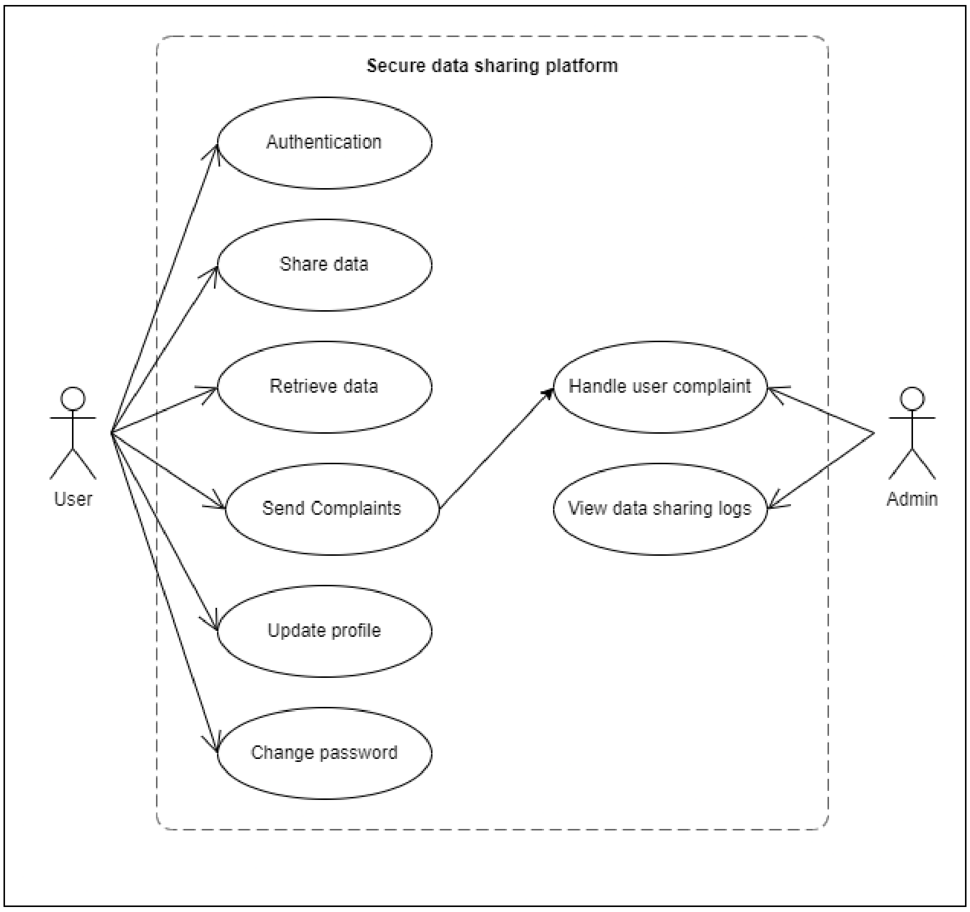


Figure 5.8: Use Case Diagram

- **User:** This actor represents a person who uses the platform to share and retrieve data, manage their profile and send complaints.
- **Admin** This actor represents a person with administrative privileges who manages complaints and monitors data-sharing logs.
- **Authentication:** The User must authenticate (log in) to access the platform's features securely.
- **Share Data:** The User can upload or share their data with others on the platform.
- **Retrieve Data:** The User can retrieve or download data shared by others.
- **Update Profile:** The User can update their personal details on the platform, such as name, email or other relevant information.
- **Change Password:** The User can change their password for security purposes.

5.4.4 Sequence Diagram

A sequence diagram is a type of diagram that shows how different parts of a system interact over time. It captures the flow of messages or events between components, such as users, servers, or databases, in a step-by-step order.
Sequence diagrams are useful because they make it easier to understand how a specific process or function works. IT provides a clear, visual flow of actions, helping developers and team members see the order of operations and identify any gaps or inefficiencies. This makes sequence diagrams a valuable tool for planning and troubleshooting workflows within a system.The figure 5.9 shows the sequence diagram of our platform

- **User1 wants to share the files:** User1 wants to share a file with User2, so the platform then establishes the P2P connection between user1 and user2 if user 2 is online then user1 can directly share a data with user2 via P2P connection.

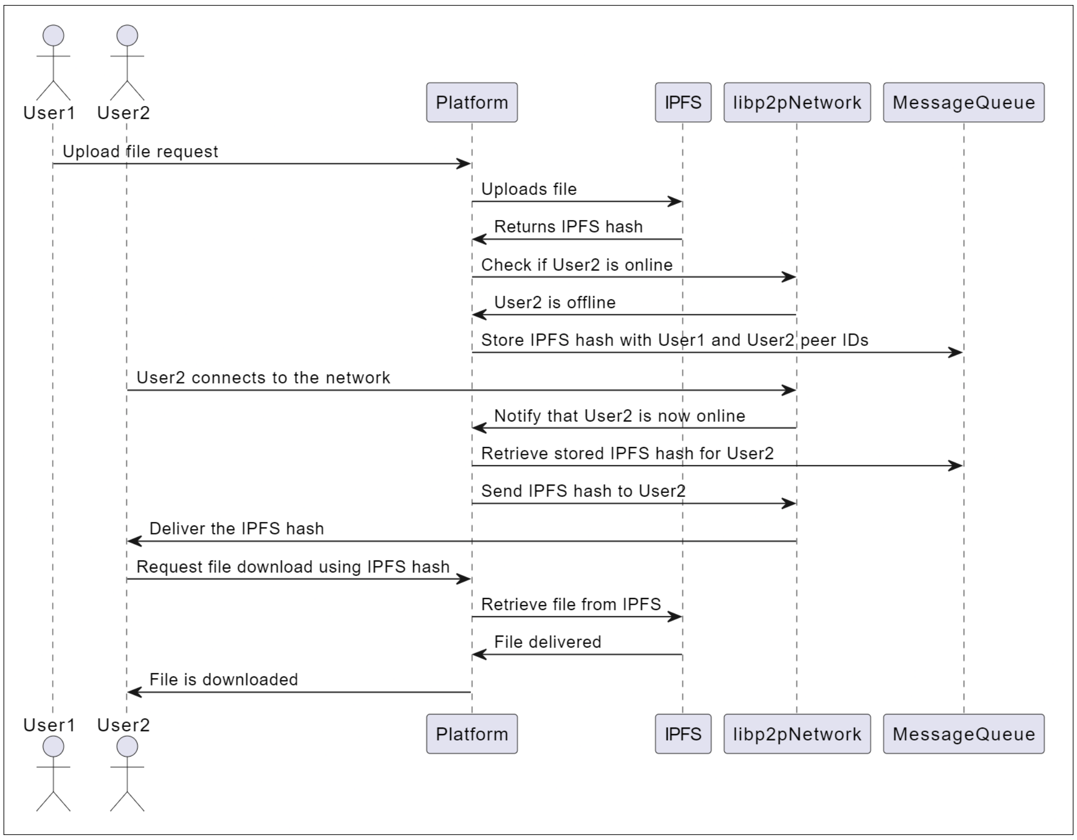


Figure 5.9: Sequence Diagram

- **User2 is offline:** If User2 is offline, the platform uploads the data on IPFS and saves the IPFS hash in a temporary storage area (a message queue) along with User1 and User2's IDs. This ensures the data can be sent as soon as User2 connects to the network.
- **User2 connects:** When User2 eventually comes online, the platform retrieves the stored IPFS hash and sends it to User2. This hash allows User2 to view or download the data.

5.4.5 Activity Diagram

An Activity Diagram is a type of UML (Unified Modeling Language) diagram that represents the workflow or the sequence of activities in a system. It is used to model the dynamic aspects of a system and describe the flow of control or data between activities, actions, and decisions within the system. It visually outlines the sequence of actions and conditions. Figure 5.10 shows the activity diagram for the platform.

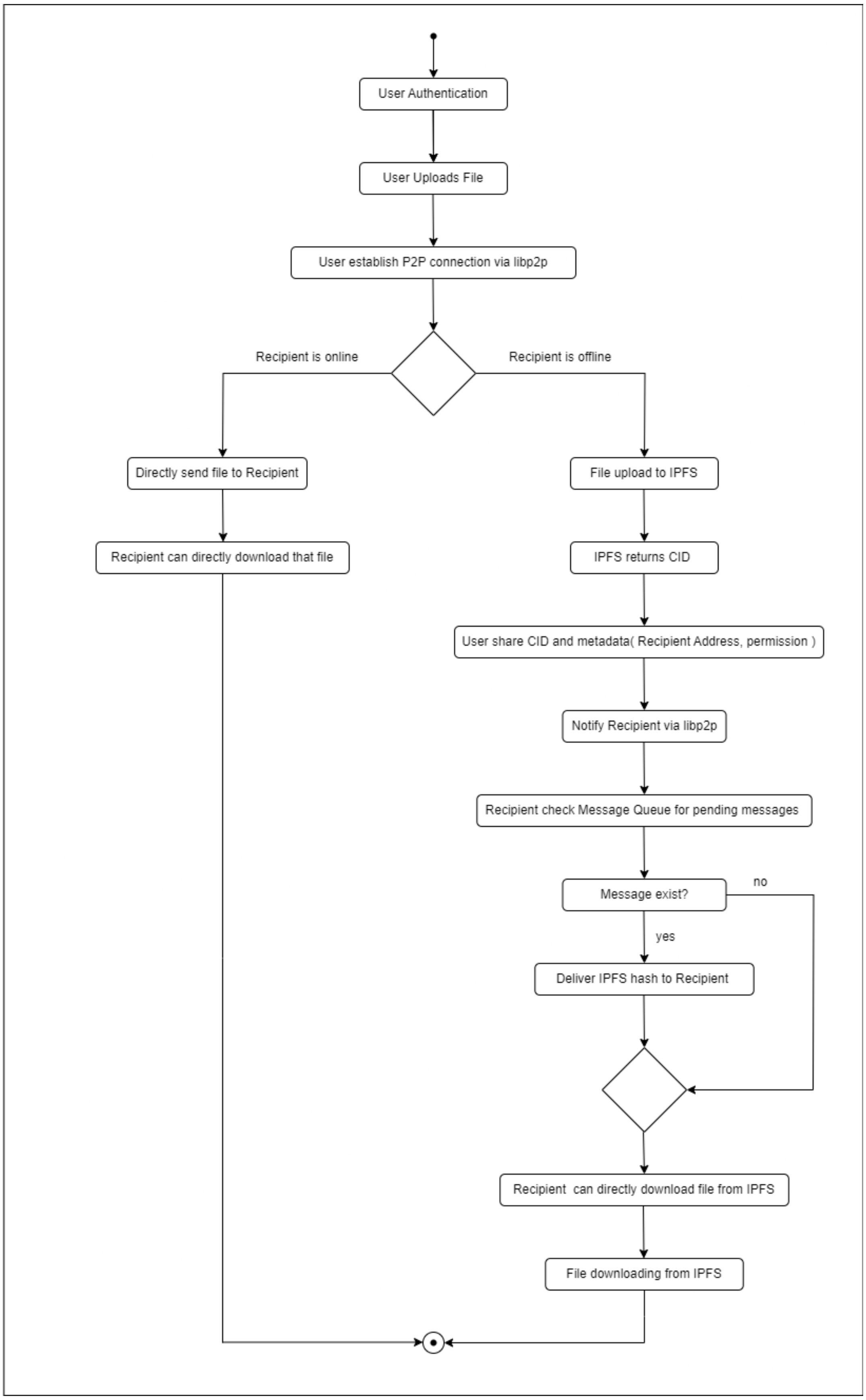

User Authentication
User Uploads File
User establish P2P connection via libp2p
Recipient is online
Recipient is offline
Directly send file to Recipient
Recipient can directly download that file
File upload to IPFS
IPFS returns CID
User share CID and metadata(Recipient Address, permission)
Notify Recipient via libp2p
Recipient check Message Queue for pending messages
Message exist?
no
yes
Deliver IPFS hash to Recipient
Recipient can directly download file from IPFS
File downloading from IPFS

Figure 5.10: Activity Diagram

- **User Authentication:** The user first logs into the platform. Authentication ensures only authorized users can access and use the system.
- **P2P Connection Check:** The platform tries to establish a peer-to-peer (P2P) connection via libp2p to check if the recipient is currently online if online then P2P connection will be established between users.
- **If the Recipient is Online:** If the recipient is online, the file is directly sent to them and they can download it immediately. For this firstly the P2P connection will be established between sender and receiver.
- **If the Recipient is Offline:** If the recipient is offline, the file is uploaded to IPFS (a decentralized storage system) by sender. IPFS generates a unique CID (Content Identifier), which acts as an address for the uploaded file. This hash then will be stored in a message queue and shared with receiver when receiver comes online.
- **Retrieve the received data:** When sender sends the data to the recipient then recipient can directly view or download the file from IPFS. The process ends once the file download is complete.

CHAPTER 6

SOFTWARE IMPLEMENTATION

The implementation of the Secure Data Sharing Platform is done using modern web technologies, peer-to-peer (P2P) communication, and decentralized file storage mechanisms. This project focuses on creating a secure, scalable, and efficient system where users can exchange data without relying on centralized servers. The platform handles real-time as well as offline file sharing using a combination of front-end, back-end, and decentralized storage tools.

The system is implemented in a modular format, allowing each component (frontend, backend, P2P, and database) to function independently while still maintaining a cohesive interaction for the user. Core aspects like file sharing, session persistence, and offline delivery are handled seamlessly using Peer.js and IPFS.

6.1 TECHNOLOGY DETAILS USED IN THE PROJECT

Frontend Technologies

- **React.js**: A powerful JavaScript library used to create the responsive and dynamic user interface for the platform. It enables smooth component rendering, event handling, and real-time user interactions.
- **Ant Design**: A React-based UI library used for creating a clean, professional, and consistent user experience. It offers ready-made components like modals, forms, buttons, and tables.

Backend Technologies

- **Node.js**: A fast, event-driven JavaScript runtime used to handle backend operations such as user registration, message queuing, and database communication.
- **Express.js**: A lightweight web framework for Node.js used to create RESTful APIs. It manages routing and serves as the bridge between the frontend, database, and P2P logic.

Peer-to-Peer Communication

- **Peer.js**: A JavaScript library used for building WebRTC-based P2P data sharing. It facilitates real-time file transfer between users without relying on a central server.
- The platform assigns each user a unique Peer ID for direct communication, improving speed and privacy.

Decentralized File Storage

- **IPFS (InterPlanetary File System)**: A distributed file system used to store and retrieve shared files. IPFS returns a unique hash for each file, ensuring immutability, decentralization, and content integrity.

Database

- **MongoDB Atlas**: A cloud-hosted NoSQL database used to store user details, peer IDs, and message queue data. It enables fast reads/writes and supports the polling system that checks for offline messages.

Offline File Sharing (Message Queue System)

- Implemented using two MongoDB collections:
  - **MessageQueue**: Stores pending file transfers (IPFS hash, sender, receiver).
  - **OnlineUsers**: Tracks which users are currently online.
- A polling script runs every 30 seconds to check and deliver files to online users.

Additional Tools

- **Dotenv**: Used to manage environment variables securely for database connections and other sensitive configurations.
- **Axios & Fetch API**: Used on the frontend to send HTTP requests to the Express backend.

CHAPTER 7

PROJECT ESTIMATION, SCHEDULE AND TEAM STRUCTURE

7.1 PROJECT COST

Cost Estimation using COCOMO'81 Model

To estimate the development effort and cost of our Blockchain-Based Secure Data-Sharing Platform, we employed the **Constructive Cost Model (COCOMO'81)**. This model is effective in estimating the number of person-months, development time, and cost based on the size of the project measured in lines of code (LOC).

Development Mode: Semi-Detached

Our project falls under the *Semi-Detached* mode because it involves a moderately complex system with a development team possessing intermediate experience in full-stack and blockchain development.

Technologies Used:

- **Frontend:** React.js, Ant Design
- **Backend:** Node.js, Express.js
- **P2P Communication:** Peer.js (WebRTC)
- **Decentralized Storage:** IPFS
- **Database:** MongoDB Atlas
- **Tools:** Dotenv, Axios

Project Details:

- **Project Title:** Blockchain-Based Secure Data-Sharing Platform
- **Project Duration:** July 2024 – April 2025 (10 months)
- **Team Members:** 4
- **Estimated LOC:** 5000 (i.e., 5 KLOC)
- **Cost per Person-Month:** Rs. 9600

COCOMO'81 Equations Used:

- **Effort:** $E = a \times (KLOC)^b$
- **Development Time:** $D = c \times (E)^d$
- **People Required:** $N = \frac{E}{D}$
- **Cost:** $C = E \times RateperPM$

Semi-Detached Mode Constants:

- $a = 3.0, \quad b = 1.12$
- $c = 2.5, \quad d = 0.35$

Step-by-Step Estimation:

1. **Effort (E):**

   $E = 3.0 \times (5)^{1.12} = 3.0 \times 6.164 = 18.49 person - months$

2. **Development Time (D):**

   $D = 2.5 \times (18.49)^{0.35} = 2.5 \times 2.63 = 6.6 months$

3. **Team Size (N):**

   $N = \frac{18.49}{6.6} = 2.8 \approx 3 persons$

4. **Actual Effort:**

   Actual team size = 4 members, Duration = 10 months

   Total effort = $4 \times 10 = 40 person - months$

5. **Cost Estimation:**

   $C = 40 \times 9600 = Rs.384,000$

Summary Table:

Aspect	COCOMO Estimate	Actual Project Data
Effort (Person-Months)	18.49	40
Duration (Months)	6.6	10
Team Size	3	4
Estimated Cost	Rs. 177,504	Rs. 384,000

Table 7.1: Comparison of Estimated and Actual Project Metrics

The COCOMO model provided a theoretical benchmark indicating the project could be completed by 3 people over approximately 6.6 months at an estimated cost of Rs. 177,504. However, to ensure quality, security, testing, and complete documentation, our actual implementation involved 4 members over 10 months, resulting in an actual cost of Rs. 384,000.

7.2 Project Schedule

The project was divided into two major phases with intermediate milestones and deliverables. It was developed by a four-member team from July 2024 to April 2025.

- **Project Start (July 2024):**
  - Team formation, topic finalization, and technology stack selection.
  - Planning of scope, objectives, deliverables, and tools.
- **First Phase Completion (November 2024):**
  - Setup of development environment and version control.
  - Designed architecture for peer-to-peer communication and offline messaging.
  - Developed and tested user registration, IPFS upload, and Peer.js communication.
- **Second Phase Completion (April 2025):**
  - Complete integration of message queue, polling mechanism, and offline delivery.
  - Final implementation of frontend with real-time updates.
  - Extensive testing, debugging, UI improvement, and documentation.
  - Final review and evaluation.

7.3 Team Structure

The development team was composed of four Computer Engineering students, each taking responsibility for key modules of the project. The team collaborated using GitHub for version control and followed Agile-inspired weekly sync-ups to ensure progress.

Table 7.2: Team Structure and Responsibilities

Team Member	**Role**	**Responsibilities**
Abhijit Rajaram Sahane	Leader and Developer	Project Management, Fullstack Implementation, Database Management
Rohit Nivrutti Shinde	Developer	Backend Development, AI ChatBot development, API Development
Saani Sayyad Shahid	Developer	Cloud Implementation, Testing, Deployment
Sairaj Chandrakant Kankate	Developer	Maintenance, Testing, Documentation

CHAPTER 8

SOFTWARE TESTING AND VALIDATION

8.1 TYPE OF TESTING

To ensure the security, accuracy, and functionality of the Blockchain-Based Secure Data-Sharing Platform, various testing methods were implemented. The platform enables peer-to-peer (P2P) data exchange and offline file sharing using IPFS and MongoDB message queues. Below are the testing types used:

- **(a) Unit Testing**
  Objective: Test individual components such as peer ID registration, file upload/download, and MongoDB interactions.
  Tools Used: Jest, Mocha, Chai

- **(b) Integration Testing**
  Objective: Ensure seamless interaction between frontend, backend, IPFS, and MongoDB.
  Tools Used: Postman, Jest, Supertest

- **(c) System Testing**
  Objective: Verify the complete system behavior under real-world conditions.
  Tools Used: Manual Testing, Browser DevTools, MongoDB Compass

- **(d) Offline Messaging & File Queue Testing**
  Objective: Ensure offline file hashes are queued in MongoDB and correctly retrieved upon recipient login.
  Tools Used: MongoDB Atlas, Mongoose, Cron Jobs

- **(e) User Acceptance Testing (UAT)**
  Objective: Validate end-user experience and functionality.
  Tools Used: Manual Testing, User Surveys

- **(f) Performance Testing**
  Objective: Evaluate system performance under load and concurrent usage.
  Tools Used: Apache JMeter, Chrome Lighthouse

- **(g) Security Testing**
  Objective: Validate access control, data confidentiality, and secure session

handling.

Tools Used: OWASP ZAP, Postman, Manual Exploit Attempts

8.2 TEST CASES

8.2.0.1 Manual Testing Template

Test Case	**Input**	**Expected Output**	**Result**
Register New Peer	Username, Email, User ID	Fixed Peer ID generated and stored in localStorage	Pass
Upload File to IPFS	File input from frontend	IPFS hash is displayed and logged	Pass
Send File to Offline User	Receiver Peer ID, IPFS hash	MongoDB stores hash and receiver ID	Pass
Retrieve Queued File	Receiver logs in	File downloaded from IPFS, queue cleared	Pass
Real-Time Sharing	Sender and Receiver online	File transferred instantly without loss	Pass
Offline Data Handling	Sender online, Receiver offline	File delivered once receiver logs in	Pass
Unauthorized Access Attempt	Wrong Peer ID retrieval request	Access denied or file not found	Pass
Direct MongoDB Manipulation	Unauthenticated API request	Server returns unauthorized error	Pass

Table 8.1: Manual Testing Summary

8.2.0.2 Automated Testing Results

- **Unit Testing**
  - Peer ID registration module passed

  - IPFS hash generation validated
  - MongoDB queuing logic confirmed

- **Integration Testing**
  - Peer-to-peer transfer simulation successful
  - MongoDB integration verified
  - File retrieval triggers hash deletion

- **Performance Testing**
  - Simulated 50 concurrent users uploading files without delay
  - MongoDB access time within acceptable range (200ms)

- **Security Testing**
  - Unauthorized retrieval blocked
  - Session data protected in localStorage
  - Direct database access denied via API

8.3 RISK MANAGEMENT

8.3.0.1 Risk Identification

- **Data Breach:** Unauthorized access to stored IPFS hashes or MongoDB queue data.
- **Peer ID Collision:** Users receiving duplicate or incorrect Peer IDs.
- **System Downtime:** IPFS or backend service unavailability.
- **Data Loss:** Failure to retrieve files or accidental queue deletions.
- **Performance Bottlenecks:** Latency during concurrent uploads or downloads.
- **User Session Issues:** Loss of user state due to improper localStorage handling.

8.3.0.2 Risk Analysis

Risk	**Likelihood**	**Impact**	**Mitigation Strategy**
Data Breach	Medium	High	Use authenticated endpoints and sanitize all inputs.
Peer ID Collision	Low	Medium	Generate Peer ID post OTP verification to ensure uniqueness.
System Downtime	Medium	High	Implement fallback mechanisms and monitor uptime.
Data Loss	Low	High	Backup MongoDB data and confirm before queue deletion.
Performance Bottlenecks	High	Medium	Use load balancers and optimize MongoDB queries.
Session Issues	Medium	Medium	Validate localStorage on page load and implement fallback flows.

Table 8.2: Risk Analysis and Mitigation

CHAPTER 9

RESULT AND ANALYSIS

The Blockchain-Based Secure Data-Sharing Platform was successfully designed, implemented, and tested to achieve secure, decentralized, and efficient peer-to-peer (P2P) file exchange. The system also ensures offline file delivery using IPFS for storage and MongoDB for message queuing. This section presents the key outcomes and insights derived from the testing and deployment of the system.

9.1 KEY RESULTS

- **Real-Time Peer-to-Peer File Transfer:** The platform effectively supports instant file sharing between two online users using PeerJS. Unique Peer IDs are generated and stored for consistent identification. File transfers were observed to complete in less than 2 seconds for files under 5MB in normal network conditions.
- **Offline File Delivery using IPFS and MongoDB:** When the receiver is offline, the file is uploaded to IPFS and the corresponding hash is queued in MongoDB with the receiver's Peer ID. Upon the recipient's login, the system retrieves and delivers the file successfully. Post-delivery, the queued entry is automatically deleted, ensuring storage hygiene.
- **User Session Management:** The platform uses browser `localStorage` to retain user registration details like Peer ID and username. This allows session persistence and avoids repeated registrations, providing a seamless user experience.
- **Security Enforcement:** Unauthorized attempts to retrieve queued file hashes were effectively denied. Only the designated recipient was able to access and download the IPFS content. Additionally, all API endpoints interacting with MongoDB were protected, ensuring secure data operations.
- **Performance and Load Handling:** Stress testing showed that the system could handle concurrent uploads and downloads from up to 20 simulated users without significant delays. IPFS and MongoDB performed reliably under the imposed load conditions.

9.2 ANALYSIS

- **Functionality:** All primary modules—Peer ID registration, file sharing, hash queuing, and file retrieval—performed as intended during both manual and automated tests.
- **Reliability:** The system consistently maintained file integrity and delivery accuracy during real-time and offline scenarios.
- **Scalability:** The use of decentralized IPFS nodes and a cloud-hosted MongoDB Atlas database ensures the system can scale horizontally to support more users without degrading performance.
- **Security:** The platform avoids centralized file storage, and implements recipient-based access controls to prevent unauthorized data access. Sensitive data is protected using frontend-only session handling and backend input validation.
- **Usability:** User feedback collected during acceptance testing reflected that the interface is intuitive, fast, and efficient for data sharing and retrieval tasks.

9.3 SUMMARY

The Blockchain-Based Secure Data-Sharing Platform meets its design objectives of providing secure and decentralized data exchange. It addresses both online and offline sharing challenges without relying on blockchain or smart contracts. The outcomes of comprehensive testing validate the platform's robustness, scalability, and practical usability, making it suitable for real-world deployment scenarios.

CHAPTER 10

ADVANTAGES, LIMITATIONS AND APPLICATION

10.1 ADVANTAGES

- Decentralization : BlockShare leverages IPFS and a P2P network, reducing reliance on centralized servers and enhancing data security.
- Enhanced Security : By using blockchain technology and decentralized storage, BlockShare minimizes risks of data breaches and unauthorized access.
- Transparency : Users can track file sharing activities and access logs, fostering trust and accountability within the network.
- Cost Savings : Reduces costs by eliminating the need for middlemen and expensive servers.
- Easy File Sharing : Users can quickly and easily share files with others without complicated processes.
- Increased Trust: By using peer-to-peer communication for sharing data it will increase the trust of the user on system

10.2 LIMITATIONS

- Dependency on Internet Connectivity : BlockShare requires a stable internet connection for users to send and retrieve files, which may be a challenge in areas with poor connectivity.
- Performance with Large Files : While IPFS is designed for efficient file storage, very large files may take longer to upload or retrieve compared to traditional methods.
- Network Overload : If too many users access the network simultaneously, it could lead to slower response times and performance issues.
- Energy Consumption: Running a decentralized network with IPFS nodes may lead to increased energy consumption compared to centralized storage, as multiple nodes need to store and share files to ensure redundancy.

10.3 APPLICATIONS

- Secure File Sharing: Users can securely share files without relying on centralized servers, reducing the risk of data breaches and ensuring greater privacy. This is useful for individual users and organizations that need to share sensitive data with clients or partners.
- Healthcare Data Exchange: BlockShare enables healthcare providers to securely share patient records or research files, ensuring data integrity and compliance with privacy regulations without centralized servers.
- Education and Learning Platforms: Academic institutions and training centers can use BlockShare for sharing course materials, research papers, and project files securely among students and faculty.
- Cross-Organization Collaborations: BlockShare can serve companies working across boundaries, enabling them to share data securely without exposure to external threats, offering a traceable and verifiable data-sharing solution.
- Government Data Exchange: BlockShare can be used by government agencies to securely share sensitive information, such as tax records or census data, ensuring that access is restricted and data integrity is maintained across agencies.
- Legal Document Sharing: Law firms and legal departments can use BlockShare to exchange confidential case files, contracts, and legal documents with clients or within legal teams, ensuring privacy and traceability.

CHAPTER 11

SUMMARY AND CONCLUSION

11.1 SUMMARY

BlockShare uses IPFS and a peer-to-peer network to let users securely share files without relying on a central server, keeping data private and controlled. Files are stored on IPFS and shared through unique identifiers, allowing users to manage access easily. The platform also includes an activity log so users can see past sharing and retrieving actions, providing full transparency and making it easy to track file history. By removing the need for third-party control, BlockShare gives users full ownership of their data, ensuring both security and privacy. With an intuitive interface, it supports seamless file management for personal and business use, enabling efficient collaboration without compromising data integrity. Additionally, BlockShare ensures fast file retrieval and sharing, even in a decentralized environment, by leveraging the power of the peer-to-peer network. This results in a scalable and robust solution for a wide range of file-sharing needs, while keeping operational costs low by eliminating reliance on central servers.BlockShare promotes scalability and reliability, making it a suitable choice for a wide range of file-sharing needs, from small-scale personal use to large organizational file exchanges. It also helps reduce operational costs by eliminating the reliance on central servers and minimizing the risks of data breaches or downtime. By enhancing security through encryption and user-controlled access, BlockShare ensures that sensitive information remains safe while giving users full control over their data. In essence, BlockShare offers a modern, secure, and cost-effective solution for file sharing that aligns with the growing demand for privacy and decentralized systems in today's digital world.

11.2 CONCLUSION

BlockShare aims to improve data privacy, control and security in file-sharing by using IPFS and peer-to-peer networks instead of centralized servers. It ensures secure file exchanges by providing unique identifiers for sharing while allowing users full ownership and control over their data. Users can verify file authenticity and track sharing history through activity logging, enhancing transparency and reliability. By addressing common challenges like data breaches and third-party control, Block-

Share builds trust among users, ensuring that personal and business files are shared securely. This approach supports a shift toward secure, user-centered file sharing, empowering users to manage their digital exchanges with confidence.

Ultimately, BlockShare supports a shift towards a more secure, user-centered file-sharing ecosystem. It enables individuals and businesses alike to manage their digital exchanges with confidence, ensuring that their data remains private, protected, and under their control. By providing a reliable, transparent, and decentralized solution, BlockShare sets the foundation for a future where users can confidently share files and collaborate without compromising their privacy or security.

CHAPTER 12

REFERENCES

[1] Thong Hoang Dilum Bandara Qin Wang Qinghua Lu Xiwei Xu Liming Zhu Petar Popovski Linh T. Nguyen, Lam Duc Nguyen and Shiping Chen. Blockchain empowered trust worthy data sharing: Fundamentals, applications, and challenges. *NA.*, 1:40, 2023.

[2] Yi Lu, Weichao Wang, Bharat Bhargava, and Dongyan Xu. Trust-based privacy preservation for peer-to-peer data sharing. *IEEE Transactions*, 36:498–502, 2006.

[3] Al-Zahrani Fahad Ahmad. Subscription-based data-sharing model using blockchain and data as a service. *IEEE Access*, 8:115966–115981, 2020.

[4] Jianping Tu Qimei Jiang Xianggui Yang Pengyong Cao, Guijiang Duan and Chen Li. Blockchain based process quality data sharing platform for aviation suppliers. *IEEE Access*, 11:19007–19023, 2024.

[5] Vikas Jaiman and Visara Urovi. A consent model for blockchain-based health data sharing platforms. *IEEE Access*, 8(1):143734–143745, 2020.

[6] Rui Song, Bin Xiao, Yubo Song, Songtao Guo, and Yuanyuan Yang. A survey of blockchain-based schemes for data sharing and exchange. *IEEE Transactions on Big Data*, 9:1477–1495, 2023.

[7] Min Yang and Yuanyuan Yang. Applying network coding to peer-to-peer file sharing. *IEEE Transactions*, 63:1938–1950, 2014.

[8] Asana. What is agile methodology? (a beginner's guide). 2024. Accessed: October 26, 2024.

ANNEXURE A

PROBLEM STATEMENT FEASIBILITY

A.1 PROBLEM STATEMENT

In today's digital world, data privacy and security while sharing the data is becoming increasingly critical, yet traditional centralized data-sharing platforms often fail to provide adequate solutions. These centralized platforms are prone to data breaches, data misuse, data loss and unauthorized access, leaving users vulnerable and often relies on costly, centralized services. Additionally, users have limited control over who can access their data and must place trust in third-party servers, which can lead to data loss or manipulation. This project aims to address these issues by creating a decentralized, peer-to-peer data-sharing platform that eliminates the need for central authorities. This platform will enable users to share data securely, privately and directly with each other, leveraging concepts like distributed storage and P2P communication to ensure data remains safe and accessible. By providing an alternative to traditional data-sharing methods, our solution is designed to empower users with better control over their data, reduce risks associated with centralization and offer a more secure, efficient way to exchange information.

A.2 FEASIBILITY

A feasibility analysis is necessary to ascertain whether a project is feasible and doable in available resources like time, money and technical capabilities. It ensures that resources are used efficiently and risks are reduced to a minimum by assisting in the identification of possible obstacles and evaluating the possibility of successful implementation.

The feasibility of this project centers on creating a secure, decentralized data-sharing platform that leverages existing technologies like IPFS for distributed storage and peer-to-peer (P2P) networking for direct data exchange. Implementing this platform is technically feasible, as both IPFS and P2P technologies are well-established, open-source solutions that support secure, resilient data-sharing which increase technical feasibility. Additionally, by eliminating the need for a central authority or centralize servers, the project aims to lower operational costs, which increases financial feasibility. The platform's design is also user-friendly, allowing individuals and organizations to share data without complex setup. Increasing demand for privacy and data control, this approach aligns with market needs, making it a valuable solution. Overall, the project is feasible in terms of technology, cost and usability and has the potential to provide a secure, efficient alternative for data-sharing in both individual and organizational contexts.

- Technical Feasibility: The project uses well-established decentralized technologies, such as IPFS for data storage and P2P communication for data transfer. These technologies are accessible and robust, making them ideal for implementing a secure data-sharing platform without needing a central authority.
- Operational Feasibility: Users will be able to easily share data directly without requiring technical expertise, improving usability and adoption. The system is designed to run on a decentralized network, minimizing maintenance and operational demands.
- Financial Feasibility: By reducing reliance on third-party servers and avoiding central infrastructure costs, the platform offers a cost-effective solution for data-sharing. The use of open-source technologies further reduces development and maintenance expenses.
- Market Feasibility: The demand for secure, decentralized solutions is growing as users and organizations prioritize data security and privacy. The project meets these needs by offering a safer, transparent alternative to traditional data-sharing models.

ANNEXURE B

DETAILS OF THE PAPERS REFERRED

[1] S Linh T. Nguyen, Lam Duc Nguyen, Thong Hoang, Dilum Bandara, Qin Wang, Qinghua Lu, Xiwei Xu, Liming Zhu, Petar Popovski and Shiping Chen. Blockchain Empowered Trust worthy Data Sharing: Fundamentals, Applications, and Challenges.NA, 1:40, 2023.
This explains blockchain's role in addressing data-sharing challenges, especially issues like data privacy, trust, and scalability. It gives understanding about how blockchain can be used for data sharing. It proposes a comprehensive framework, BlockDaSh, designed for secure and transparent data sharing across sectors such as healthcare, smart grids, and supply chains. The authors provide a detailed survey of blockchain-based data-sharing architectures and applications, emphasizing both benefits and limitations. Paper also identify open research challenges and suggest future research directions to improve the integration of blockchain in data sharing .

[2] P. Cao, G. Duan, J. Tu, Q. Jiang, X. Yang and C. Li, Blockchain-Based Process Quality Data Sharing Platform for Aviation Suppliers, in IEEE Access, vol. 11, pp. 19007-19023, 2023.
Author introduces a blockchain-based data-sharing platform for aviation suppliers to enhance transparency and reliability in quality data sharing. It explores about how blockchain can be used in aviation sector for secure data sharing. The proposed platform allows aviation manufacturers to securely share quality data, reducing inefficiencies and risks linked to traditional supply chains. The paper also outlines a layered data architecture, focusing on secure data acquisition, storage, and access, and demonstrates its application in aviation supply management.

[3] A. Al-Zahrani, Subscription-Based Data-Sharing Model Using Blockchain and Data as a Service, in IEEE Access, vol. 8, pp. 115966-115981, 2020.
Paper proposes a subscription-based data-sharing model utilizing blockchain and Data as a Service (DaaS) to ensure secure, consistent, and accessible data exchange. The model addresses issues like data ownership, pricing strategies, and quality control by establishing a recurring revenue system for data providers. This paper propose a subscription-based data-sharing model uti-

lizing blockchain i.e how platform should charge for data sharing utilizing blockchain. Results indicate that this model is both effective and efficient, offering a structured approach to data monetization while maintaining trust and transparency.This paper outlines how the platform can structure pricing and permissions for data sharing through blockchain, creating a transparent environment where data authenticity is preserved and every transaction is auditable.

[4] V. Jaiman and V. Urovi, A Consent Model for Blockchain-Based Health Data Sharing Platforms, in IEEE Access, vol. 8, pp. 143734-143745, 2020,
This paper introduces a model for health data sharing that centres on patient consent using blockchain technology. It incorporates two ontologies—Data Use Ontology (DUO) and Automat-able Discovery and Access Matrix (ADA-M)—to match users' consent preferences with data requester intended use, ensuring compliance with consent terms. The model aims to support dynamic, GDPR-compliant data sharing with accountability, providing flexibility in decision-making for both data providers and requester.

[5] R. Song, B. Xiao, Y. Song, S. Guo and Y. Yang, A Survey of Blockchain-Based Schemes for Data Sharing and Exchange, in IEEE Transactions on Big Data, vol. 9, no. 6, pp. 1477-1495, Dec. 2023.
Author explores blockchain's role in data sharing and exchange, particularly its advantages over traditional, centralized platforms. By offering decentralization, immutability, and enhanced privacy, blockchain is suited for a range of fields, including IoT, finance, and healthcare. The survey reviews various blockchain-based platforms, addressing architecture, security, and access control, as well as challenges like high transaction costs and privacy protection. The paper also examines methods for secure data marketplaces, highlighting the potential and ongoing limitations of blockchain in large-scale data sharing
.

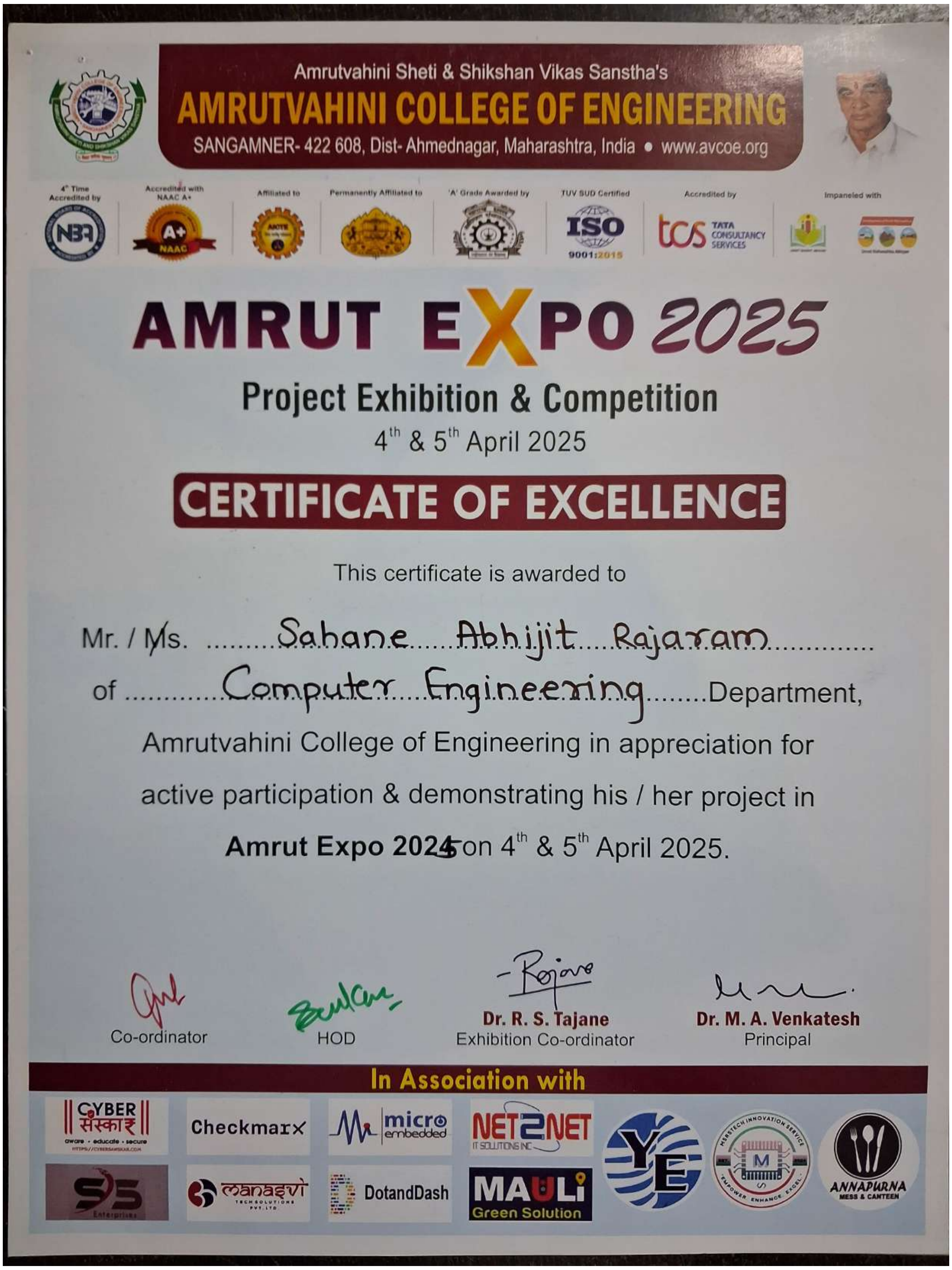
[6] M. Yang and Y. Yang, Applying Network Coding to Peer-to-Peer File Sharing, IEEE Transactions, vol. 63, no. 8, pp. 1938-1950, Aug. 2014.

This paper explores the use of network coding to enhance throughput and reliability in peer-to-peer (P2P) file-sharing systems. By encoding and distributing files across multiple peers, the system increases efficiency, allowing any subset of peers with sufficient data to decode the original file. The proposed method, PPFEED, supports dynamic network conditions and shows a 15-20% improvement in throughput compared to traditional P2P networks

[7] Yi Lu, Weichao Wang, B. Bhargava and Dongyan Xu, Trust-based privacy preservation for peer-to-peer data sharing, in IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, vol. 36, no. 3, pp. 498-502, May 2006.
This paper introduces a trust-based mechanism to protect user privacy in P2P data-sharing systems. Using a proxy system, trusted ”buddies” act as intermediaries for data requests, ensuring that users' identities and interests remain hidden. The system also assesses trust dynamically, adjusting relationships based on past behavior to enhance privacy further while preserving system functionality.

ANNEXURE C

AWARDS/PARTICIPATION IN PROJECT COMPETITION/EXHIBITION



Amrutvahini Sheti & Shikshan Vikas Sanstha's

AMRUTVAHINI COLLEGE OF ENGINEERING

SANGAMNER- 422 608, Dist- Ahmednagar, Maharashtra, India • www.avcoe.org

AMRUT EXPO 2025

Project Exhibition & Competition

4th & 5th April 2025

CERTIFICATE OF EXCELLENCE

This certificate is awarded to

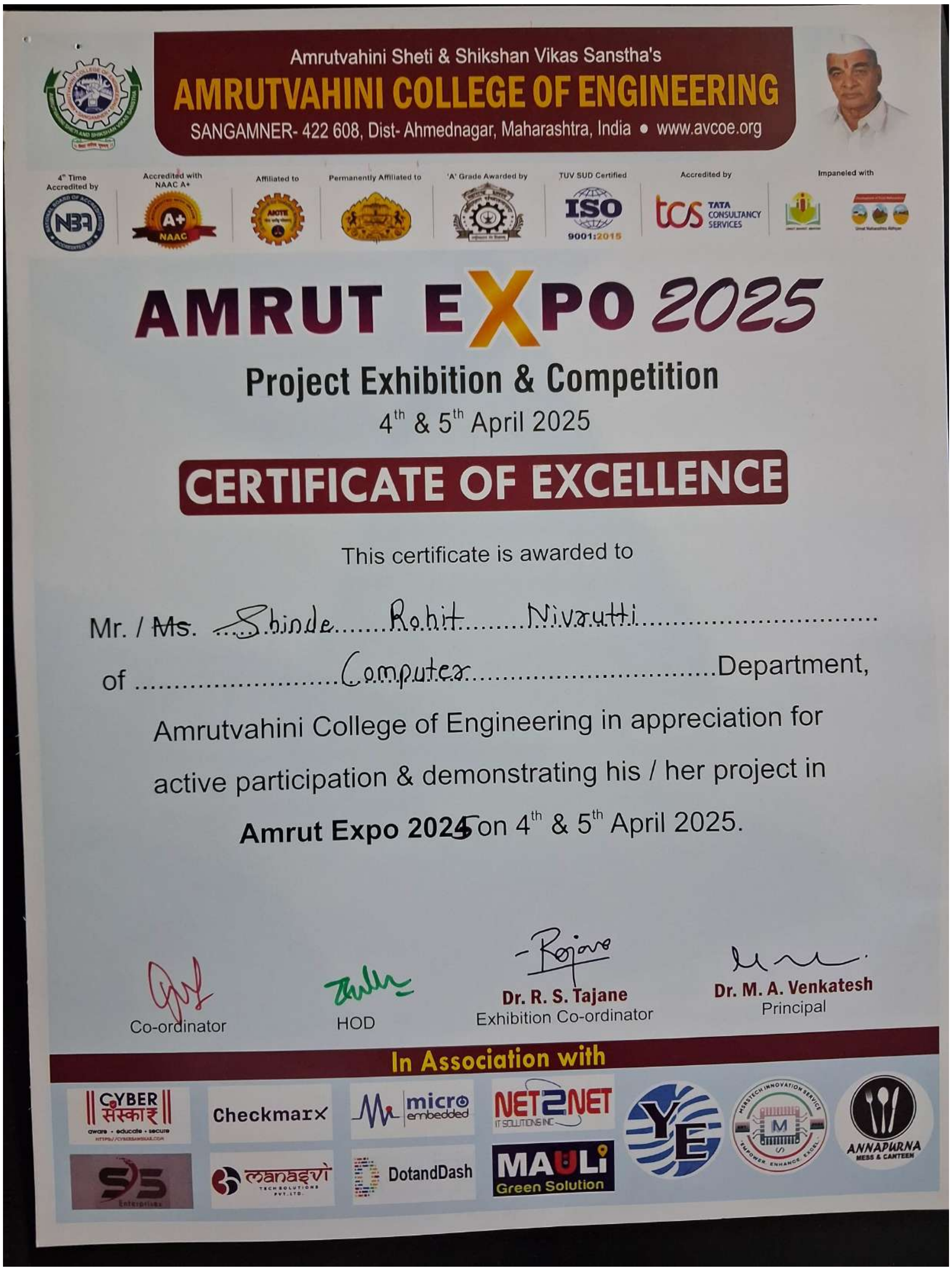
Mr. / ~~Ms.~~ Sahane Abhijit Rajaram

of Computer Engineering Department,

Amrutvahini College of Engineering in appreciation for active participation & demonstrating his / her project in **Amrut Expo 2025** on 4th & 5th April 2025.

[illegible]	[illegible]	[illegible]	[illegible]
Co-ordinator	HOD	**Dr. R. S. Tajane**	**Dr. M. A. Venkatesh**
		Exhibition Co-ordinator	Principal

In Association with



Amrutvahini Sheti & Shikshan Vikas Sanstha's

AMRUTVAHINI COLLEGE OF ENGINEERING

SANGAMNER- 422 608, Dist- Ahmednagar, Maharashtra, India • www.avcoe.org

4th Time Accredited by NBA | Accredited with NAAC A+ | Affiliated to AICTE | Permanently Affiliated to | 'A' Grade Awarded by | TUV SUD Certified ISO 9001:2015 | Accredited by TCS Tata Consultancy Services | Impaneled with

AMRUT EXPO 2025

Project Exhibition & Competition

4th & 5th April 2025

CERTIFICATE OF EXCELLENCE

This certificate is awarded to

Mr. / ~~Ms.~~ Shinde Rohit Nivrutti

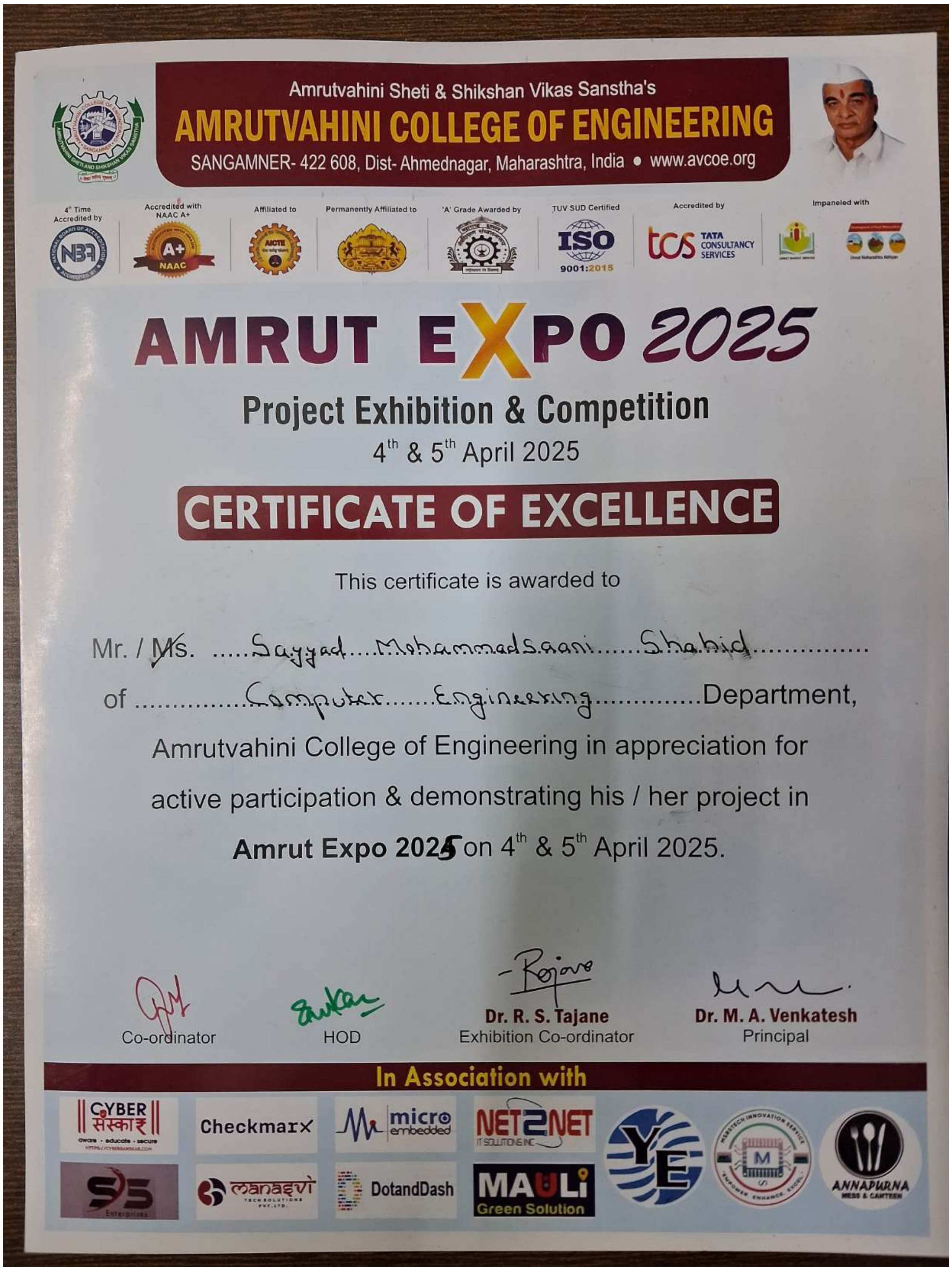
of Computer Department,

Amrutvahini College of Engineering in appreciation for active participation & demonstrating his / her project in **Amrut Expo 2025** on 4th & 5th April 2025.

[illegible] Co-ordinator | [illegible] HOD | Dr. R. S. Tajane Exhibition Co-ordinator | Dr. M. A. Venkatesh Principal

In Association with

Cyber संस्कार aware • educate • secure | Checkmarx | micro embedded | NET2NET IT Solutions Inc | YE | M | Annapurna Mess & Canteen | S5 Enterprises | manasvi Tech Solutions Pvt. Ltd. | DotandDash | MAULi Green Solution



Amrutvahini Sheti & Shikshan Vikas Sanstha's

AMRUTVAHINI COLLEGE OF ENGINEERING

SANGAMNER- 422 608, Dist- Ahmednagar, Maharashtra, India • www.avcoe.org

4th Time Accredited by | Accredited with NAAC A+ | Affiliated to | Permanently Affiliated to | 'A' Grade Awarded by | TUV SUD Certified ISO 9001:2015 | Accredited by TATA CONSULTANCY SERVICES | Impaneled with

AMRUT EXPO 2025

Project Exhibition & Competition

4th & 5th April 2025

CERTIFICATE OF EXCELLENCE

This certificate is awarded to

Mr. / ~~Ms.~~ ....Sayyad....Mohammadsaani.....Shahid.............

of ....Computer......Engineering.............Department,

Amrutvahini College of Engineering in appreciation for active participation & demonstrating his / her project in **Amrut Expo 202~~4~~5** on 4th & 5th April 2025.

[illegible] Co-ordinator

[illegible] HOD

[illegible] **Dr. R. S. Tajane** Exhibition Co-ordinator

[illegible] **Dr. M. A. Venkatesh** Principal

In Association with

CYBER संस्कार · Checkmarx · micro embedded · NET2NET IT SOLUTIONS INC · YE · M · ANNAPURNA MESS & CANTEEN · SS Enterprises · manasvi TECH SOLUTIONS PVT. LTD. · DotandDash · MAULI Green Solution

ANNEXURE D

PLAGIARISM REPORT FOR THIS REPORT

All must attach certificate/report of Plagiarism issued by Turnitine Software. Percentage of Similarity should not be more than 20% . Prepare separate file including Abstract, All Chapters and conclusion for Plagiarism Checking.

ANNEXURE E

ANY OTHER DOCUMENTATION EVIDENCES RELATED TO PROJECT

All must attach certificate/report of Plagiarism issued by Turnitine Software. Percentage of Similarity should not be more than 20%