

A Blockchain-Based Secure Decentralised Storage System

A report submitted in partial fulfillment of the requirements for the award of a degree of

**Bachelor of Technology
in
Computer Science and Engineering**

by
**G. Shiva Sai Balaji
(21EG505826)**

**K. Rahul
(21EG505835)**

**K. Sree Nailu
(21EG505842)**

Under The Guidance of

Dr. P. Ravinder Rao,
Asst. Professor, Department of CSE



**Department of Computer Science and Engineering
ANURAG UNIVERSITY
Venkatapur (V), Ghatkesar (M), Medchal (D)., T.S-500088
(2023-2024)**

DECLARATION

We hereby declare that the report entitled “**A Blockchain-Based Secure Decentralised Storage System**” submitted for the award of the degree of **Bachelor of Technology (B. Tech)** in Computer Science and Engineering is a record of an original work done by us and the report has not formed the basis for the award of any degree, diploma, associateship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma.

Place: Anurag University, Hyderabad

Date:

G. Shiva Sai Balaji

(21EG505826)

K. Rahul

(21EG505835)

K. Sree Nailu

(21EG505842)

CERTIFICATE

This is to certify that the report entitled “**A Blockchain-Based Secure Decentralised Storage System**” is being submitted by **Mr. G. Shiva Sai Balaji** bearing the Hall Ticket number **21EG505826**, **Mr. K. Rahul** bearing the Hall Ticket number **21EG505835**, **Ms. K. Sree Nailu** bearing the Hall Ticket number **21EG505842** in partial fulfillment for the award of the Bachelor of Technology in Computer Science and Engineering to the Anurag University is a record of bonafide work carried out by them under my guidance and supervision for the academic year 2023 to 2024.

The results embodied in this report have not been submitted to any other University or Institute for the award of any other degree or diploma.

Signature of Supervisor

Dr. P. Ravinder Rao
Assistant Professor
Department of CSE

Signature of Dean

Dr. G. Vishnu Murthy
Dean, CSE

External Examiner

ACKNOWLEDGEMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Dr. P. Ravinder Rao**, Assistant Professor, Dept of CSE for his constant encouragement and inspiring guidance without which this project could not have been completed. His critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. His patience, guidance and encouragement made this project possible.

We would like to express our special thanks to **Dr. V. Vijaya Kumar**, Dean School of Engineering, Anurag University, for his encouragement and timely support in our B. Tech program.

We would like to acknowledge our sincere gratitude for the support extended by **Dr. G. Vishnu Murthy**, Dean, Dept. of CSE, Anurag University. We also express our deep sense of gratitude to **Dr. V V S S S Balaram**, Academic Coordinator, **Dr. Pallam Ravi**, Project Coordinator and Project Review Committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of our project work.

G. SHIVA SAI BALAJI

(21EG505826)

K. RAHUL

(21EG505835)

K. SREE NAILU

(21EG505842)

ABSTRACT

Privacy protection and open sharing are the core of data governance in the AI-driven era. A common data-sharing management platform is indispensable in the existing data-sharing solutions, and users upload their data to the cloud server for storage and dissemination. However, from the moment users upload the data to the server, they will lose absolute ownership of their data, and security and privacy will become a critical issue. Although data encryption and access control are considered up-and-coming technologies in protecting personal data security on the cloud server, they alleviate this problem to a certain extent. However, it still depends too much on a third-party organization's credibility, the Cloud Service Provider (CSP). In this paper, we combined blockchain, ciphertext-policy attribute-based encryption (CP-ABE), and Inter Planetary File System (IPFS) to address this problem to propose a blockchain-based security sharing scheme for personal data named BSSPD. In this user-centric scheme, the data owner encrypts the sharing data and stores it on IPFS, which maximizes the scheme's decentralization. The address and the decryption key of the shared data will be encrypted with CP-ABE according to the specific access policy, and the data owner uses blockchain to publish his data-related information and distribute keys for data users. Only the data user whose attributes meet the access policy can download and decrypt the data. The data owner has fine-grained access control over his data, and BSSPD supports an attribute-level revocation of a specific data user without affecting others. To further protect the data user's privacy, the ciphertext keyword search is used when retrieving data. We analyzed the security of the BSSPD and simulated our scheme on the EOS blockchain, which proved that our scheme is feasible. Meanwhile, we provided a thorough analysis of the storage and computing overhead, which proved that BSSPD has a good performance.

Keywords: IPFS, CSP, CP-ABE, BSSPD

CONTENTS

Title	Pageno
Abstract	v
List of Figures	vii
1.Introduction	1
1.1. Motivation	2
1.2. Problem Definition	3
1.3. Objective of the Project	3
2.Literature Survey	4
3.Analysis	7
3.1. Existing System	7
3.2. Proposed System	7
3.3. System Requirement Specification	7
3.3.1 Software Requirements	7
3.3.2 Hardware Requirements	8
4.Design	9
4.1. System Architecture	10
5.Implementation	15
5.1. Modules	15
5.2. Module Description	16
5.3. Introduction to Technologies Used	17
5.4. Sample code	22
6.Test Cases	36
7.Screenshots	38
8.Conclusion	43
9.Future Enhancement	44
10. Bibliography	45

LIST OF TABLE

S. No	Title	Page No
Fig 6.1	Test cases for the proposed system	37

LIST OF FIGURES

FigureNo	FigureName	Page no
Fig.4.1	Work flow Diagram	9
Fig.4.2	SystemArchitectureDiagram	10
Fig.4.3	Activity Diagram	11
Fig.4.4	Usecase diagram	12
Fig.4.5	Class diagram	13
Fig.4.6	Sequence diagram	13
Fig.4.7	Component diagram	14
Fig.7.1	IPFS server started	38
Fig.7.2	DJANGO server started	38
Fig.7.3	User signup	39
Fig.7.4	user signup process completed	39
Fig.7.5	upload image	40
Fig.7.6	sharing data	40
Fig.7.7	Shared data screen	41
Fig.7.8	Shared data screen	42
Fig.7.9	Logout image	42

1.INTRODUCTION

The rapid evolution of technology, particularly the advent of 5G and Internet of Things (IoT) advancements, has ushered in an era of unprecedented data availability. This surge in data has not only fueled the rapid development of artificial intelligence (AI) but has also brought to the forefront critical concerns regarding data security and privacy. In the contemporary landscape of data governance and sharing, ensuring the confidentiality, integrity, and privacy of personal data has become paramount. This is particularly pertinent given the sensitive nature of data generated by IoT devices, which often encompasses deeply personal information relating to individuals' lives, work, and healthcare.

Traditionally, the prevalent approach to data sharing has involved outsourcing data to centralized cloud servers. While this facilitates collaboration and data management, it also introduces vulnerabilities, especially when dealing with sensitive information. The reliance on centralized entities for data storage and security raises concerns about data ownership, control, and protection against unauthorized access or breaches. The potential consequences of data breaches, particularly when sensitive personal information is involved, can be severe, leading to significant privacy violations and potential harm to individuals.

Against this backdrop, the motivation for the proposed blockchain-based security sharing scheme for personal data (BSSPD) emerges from the pressing need to address these challenges effectively. The BSSPD scheme represents a novel approach to data security and privacy, leveraging cutting-edge technologies such as blockchain, ciphertext-policy attribute-based encryption (CP-ABE), and the InterPlanetary File System (IPFS). By integrating these technologies, the BSSPD scheme aims to provide a secure, decentralized, and privacy-preserving framework for data sharing and management.

The BSSPD scheme is motivated by several critical factors shaping today's technological landscape. Firstly, the proliferation of data generated by IoT devices, coupled with the rapid advancement of AI, underscores the urgent need for robust data security and privacy measures. The abundance of training data for AI systems presents immense opportunities for innovation but also poses significant risks if not adequately protected. Moreover, the sensitive nature of IoT-generated data necessitates heightened security measures to safeguard individuals' privacy and mitigate the risks of data breaches or unauthorized access.

Secondly, the limitations and vulnerabilities associated with traditional centralized data storage solutions highlight the need for alternative approaches that prioritize decentralization and user control. Centralized cloud storage introduces single points of failure and reliance on third-party entities, raising concerns about

data ownership, trust, and security. The BSSPD scheme offers a decentralized alternative that empowers data owners to retain control over their data while ensuring its confidentiality and integrity through encryption and blockchain technology.

Furthermore, the BSSPD scheme is motivated by the growing recognition of the importance of privacy-preserving technologies in today's digital ecosystem. With increasing awareness of data privacy rights and regulatory frameworks such as the General Data Protection Regulation (GDPR), there is a growing demand for solutions that prioritize user privacy and data protection. The BSSPD scheme aligns with these evolving regulatory and societal expectations by offering a comprehensive approach to data security and privacy that prioritizes user control and transparency.

In summary, the proposed blockchain-based security sharing scheme for personal data (BSSPD) addresses the pressing challenges of data security and privacy in contemporary data-sharing environments. By leveraging advanced technologies and decentralized architectures, the BSSPD scheme aims to empower individuals to securely share and manage their data while ensuring confidentiality, integrity, and transparency. Through its innovative approach, the BSSPD scheme seeks to pave the way for a more secure and privacy-preserving digital future.

1.1 Motivation

The motivation for the proposed blockchain-based security sharing scheme for personal data (BSSPD) arises from several critical factors in today's technological landscape. Firstly, the rapid advancement of 5G and Internet of Things (IoT) technologies has led to a surge in data availability, particularly for training artificial intelligence (AI) systems. However, this abundance of data also brings forth challenges in terms of security and privacy, especially considering the sensitive nature of data generated by IoT devices, which often includes personal information related to individuals' lives, work, and healthcare.

Traditionally, many individuals and organizations opt to store and share their data via cloud servers. While this facilitates data management and collaboration, it also raises concerns about data ownership, security, and control. The reliance on centralized cloud service providers (CSPs) for data storage and security introduces vulnerabilities and dependencies on third-party organizations' credibility. To address these challenges, the proposed BSSPD scheme leverages blockchain technology, ciphertext-policy attribute-based encryption (CP-ABE), and the InterPlanetary File System (IPFS) to enhance data security, privacy, and decentralization. By combining these technologies, the scheme aims to empower data owners to encrypt and securely store their sharing data on a decentralized network, thereby reducing reliance on centralized entities and enhancing user control over their data.

1.2 Problem Definition

The principle is free-market between data owners, who need disk space to store/backup private data, and disk space owners, who provide disks for that purpose. File Storage Marketplace is absolutely decentralized, unlike traditional storage. The main purpose of the system is to provide a public, decentralized and secure disk space renting platform.

In Disk Renting System, to log in to the system the admin can log in with a username and password. The admin checks the requests from the user for renting disk space. Admin has the authority to accept or reject the request. And if the request is been accepted by the admin, then the user will be assigned disk space. The admin can check for system manipulations by viewing Transactions, Payments, and Disk usage

1.3 Objective

The main purpose of the system is to provide a public, decentralized and secure disk space renting platform. In Disk Renting System, to log in to the system the admin can log in with a username and password. The admin checks the requests from the user for renting disk space. Admin has the authority to accept or reject the request. And if the request is been accepted by the admin, then the user will be assigned disk space. The admin can check for system manipulations by viewing Transactions, Payments, and Disk usage. The user has to register their account and log in using a username and password. The user can view their total hard disk usage, their total space rented, and their total space unused.

2.LITERATURE SURVEY

1. Title of the paper: A Secure Decentralized Storage Framework on Blockchain.

Authors: Sushmita Ruj, Mohammad Shahriar Rahman, Anirban Basu and Shinsaku Kiyomoto

Journal and Year: 2018 IEEE 32nd International Conference on Advanced Information Networking and Applications 1550-445X/18/\$31.00 ©2018 IEEE DOI10.1109/AINA.2018.00157

Observation: Decentralized storage with Blockchain. It enjoys many features like data integrity, confidentiality and transparency. The transactions are explained in details. The framework is an initial setup and can be extended in many directions to include renting multimedia, apartments and other goods and services.

2. Title of the paper: Decentralized Storage Space Using Blockchain and IPFS Protocol.

Authors: Piyush Samant, Smita Shinde.

Journal and Year: International Journal for Research in Applied Science & Engineering Technology (IJRASET) ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538 Volume 11 Issue IV Apr 2023- Available at www.ijraset.com.

Observation: IPFS based decentralized storage system named as D-Drive. The proposed system maximise the data security by distributing our data across peer-to-peer network in a decentralized manner. This system uses the IPFS protocol for ensuring the confidentiality of the user's data. Apart from these advantages, it needs certain level of improvement for accuracy and speed. In the proposed system, IPFS protocol is used, but if there is a better system in future, that can also be implemented.

3. Title of the paper: Collective data-sanitization for preventing sensitive information inference attacks in social networks.

AUTHOURS: Zhipeng Cai, Zaobo He, Xin Guan, and Yingshu Li.

Observation: Releasing social network data could seriously breach user privacy. User profile and friendship relations are inherently private. Unfortunately, sensitive information may be predicted out of released data through data mining techniques. Therefore, sanitizing network data prior to release is necessary. In this paper, we explore how to launch an inference attack exploiting social networks with a mixture of non-sensitive attributes and social relationships. We map this issue to a collective classification problem and propose a collective inference model. In our model, an attacker utilizes user profile and social relationships in a collective manner to predict

sensitive information of related victims in a released social network dataset. To protect against such attacks, we propose a data sanitization method collectively manipulating user profile and friendship relations. Besides sanitizing friendship relations, the proposed method can take advantages of various data-manipulating methods. We show that we can easily reduce adversary's prediction accuracy on sensitive information, while resulting in less accuracy decrease on non-sensitive information towards three social network datasets. This is the first work to employ collective methods involving various data-manipulating methods and social relationships to protect against inference attacks in social network.

4. Title of the paper: A private and efficient mechanism for data uploading in smart cyber-physical systems.

AUTHORS: Zhipeng Cai¹, Xu Zheng^{1,2}, Student Member

Observation: To provide fine-grained access to different dimensions of the physical world, data uploading in smart cyber-physical systems suffers novel challenges on both energy conservation and privacy preservation. It is always critical for participants to consume as little energy as possible for data uploading. However, simply pursuing energy efficiency may lead to extreme disclosure of private information, especially when the uploaded contents from participants are more informative than ever. In this paper, we propose a novel mechanism for data uploading in smart cyber-physical systems, which considers both energy conservation and privacy preservation. The mechanism preserves privacy by concealing abnormal behaviors of participants, while still achieves an energy-efficient scheme for data uploading by introducing an acceptable number of extra contents. To derive an optimal uploading scheme is proved to be NP-hard. Accordingly, we propose a heuristic algorithm and analyze its effectiveness. The evaluation results towards a real-world dataset demonstrate that the results obtained through our proposed algorithm is comparable with the optimal ones.

5. Title of the paper: Academic influence aware and multidimensional network analysis for research collaboration navigation based on scholarly big data.

Authors: Zhou, Wei Liang

Observation: Scholarly big data, which is a large-scale collection of academic information, technical data, and collaboration relationships, has attracted increasing attentions, ranging from industries to academic communities. The widespread adoption of social computing paradigm has made it easier for researchers to join collaborative research activities and share academic data more extensively than ever before across the highly interlaced academic networks. In this study, we focus on the academic influence aware and multidimensional network analysis based on the integration of multi-source scholarly big data. Following three basic relations: Researcher-Researcher, Researcher-Article, and Article-Article, a set of measures is

introduced and defined to quantify correlations in terms of activity-based collaboration relationship, specialty-aware connection, and topic-aware citation fitness among a series of academic entities (e.g., researchers and articles) within a constructed multidimensional network model. An improved Random Walk with Restart (RWR) based algorithm is developed, in which the time-varying academic influence is newly defined and measured in a certain social context, to provide researchers with research collaboration navigation for their future works. Experiments and evaluations are conducted to demonstrate the practicability and usefulness of our proposed method in scholarly big data analysis using DBLP and ResearchGate data.

3.ANALYSIS

3.1 Existing System

Cloud storage, although an option, can be expensive and access to the data may have high latency. Since data rests with, and accesses to it are often managed by a third party (the cloud service providers), data security and privacy are major concern.

Drawbacks of Existing System

- Complexity
- Speed
- Reliability
- Scalability

3.2 Proposed System

The proposed method, named BSSPD (Blockchain-Based Security Sharing Scheme for Personal Data), aims to address the issues of data security and privacy in existing data-sharing solutions by leveraging blockchain technology, ciphertext-policy attribute-based encryption (CP-ABE), and the InterPlanetary File System (IPFS). Below are the key components and steps involved in the BSSPD scheme:

Decentralized Storage with IPFS: Data owners encrypt their sharing data and store it on IPFS, maximizing decentralization and ensuring data availability.

Attribute-Based Encryption (ABE): The address and decryption key of the shared data are encrypted using ciphertext-policy attribute-based encryption (CP-ABE) based on specific access policies defined by the data owner.

Fine-Grained Access Control: Data owners have fine-grained control over access to their data, specifying access policies based on attributes.

Only data users whose attributes match the access policy can download and decrypt the shared data.

3.3 System Requirement Specification

Software Requirements

- Operating system: Windows 8 Professional.
- Coding Language: python
- Front-End Technologies: HTML, CSS, JavaScript

Hardware Requirements

- Operating System: Windows Only
- Processor: i5 and above
- Ram: 8gb and above
- Hard Disk: 50 GB in local drive

4. DESIGN

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

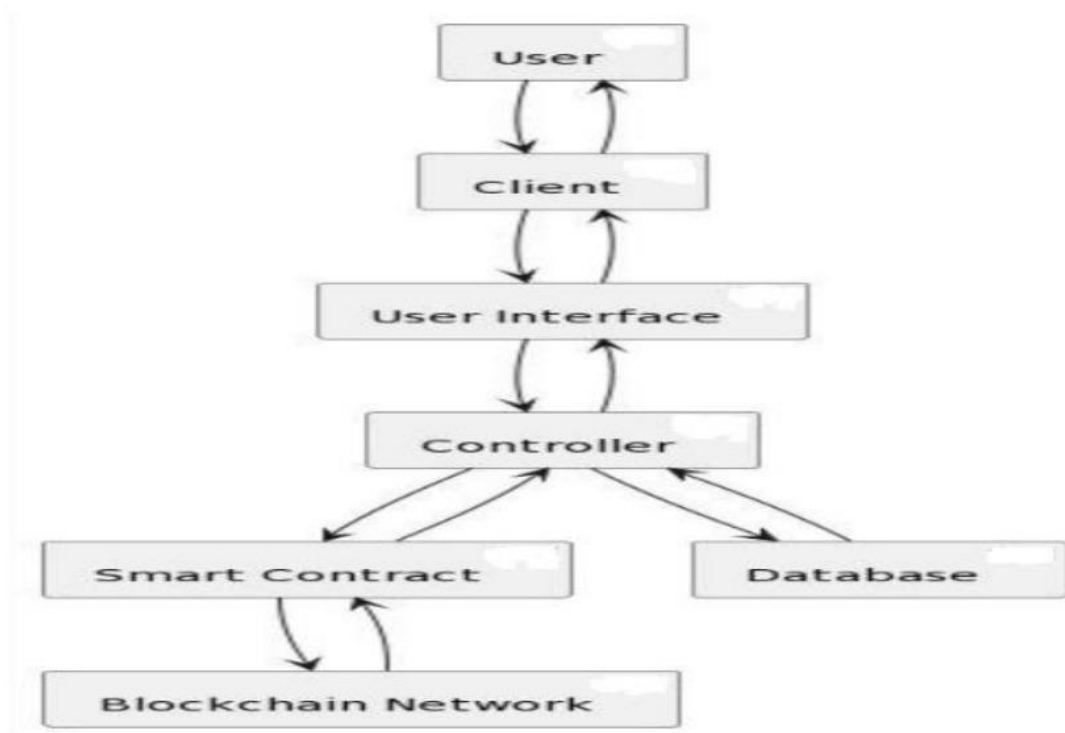


Figure 4.1 Workflow Diagram

ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited.

The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

TECHNICAL FEASIBILITY

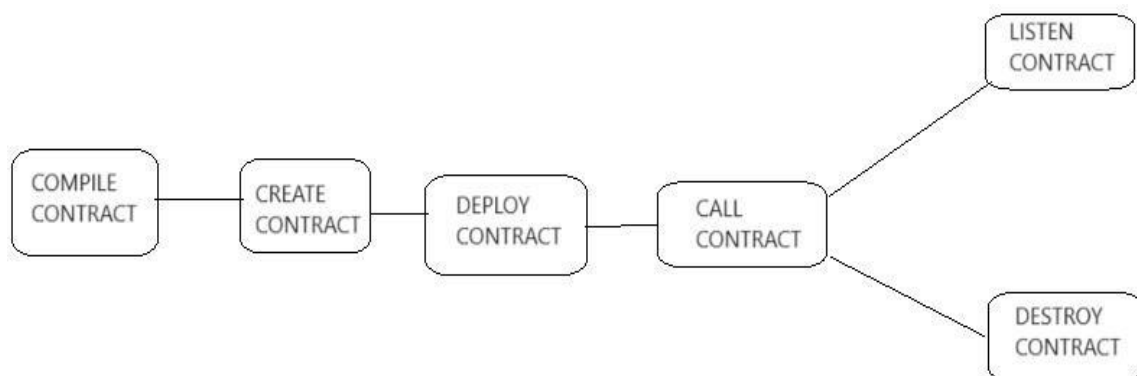
This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the

client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

4.1 ARCHITECTURE DIAGRAM



4.2 System Architecture Diagram

UML stands for Unified Modelling Language. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Metamodel and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modelling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modelling and other non software system. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process.

The UML uses mostly graphical notations to express the design of software projects.

GOALS

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modelling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modelling language.
5. Support higher level development concepts such as collaborations, frameworks, patterns and components.

TYPES OF UML DIAGRAM

Each UML diagram is designed to let developers and customers view a software system from a different perspective and in varying degrees of abstraction. UML diagrams commonly created in visual modelling tools include

A. ACTIVITY DIAGRAM

Activity diagram is another important behavioral diagram in **UML** diagram to describe dynamic aspects of the system. Activity diagram is essentially an advanced version of flow chart that modeling the flow from one activity to another activity.

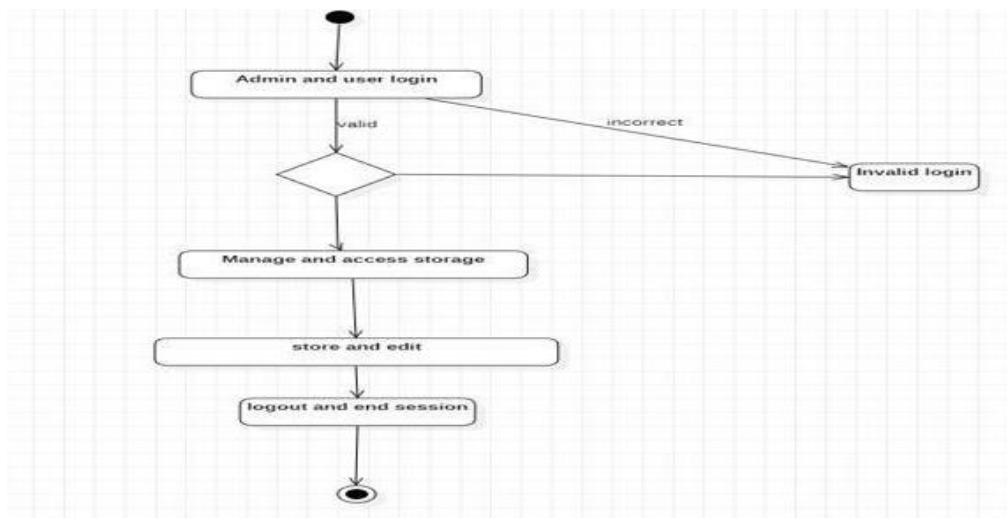


Fig 4.3 Activity Diagram

B. USECASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

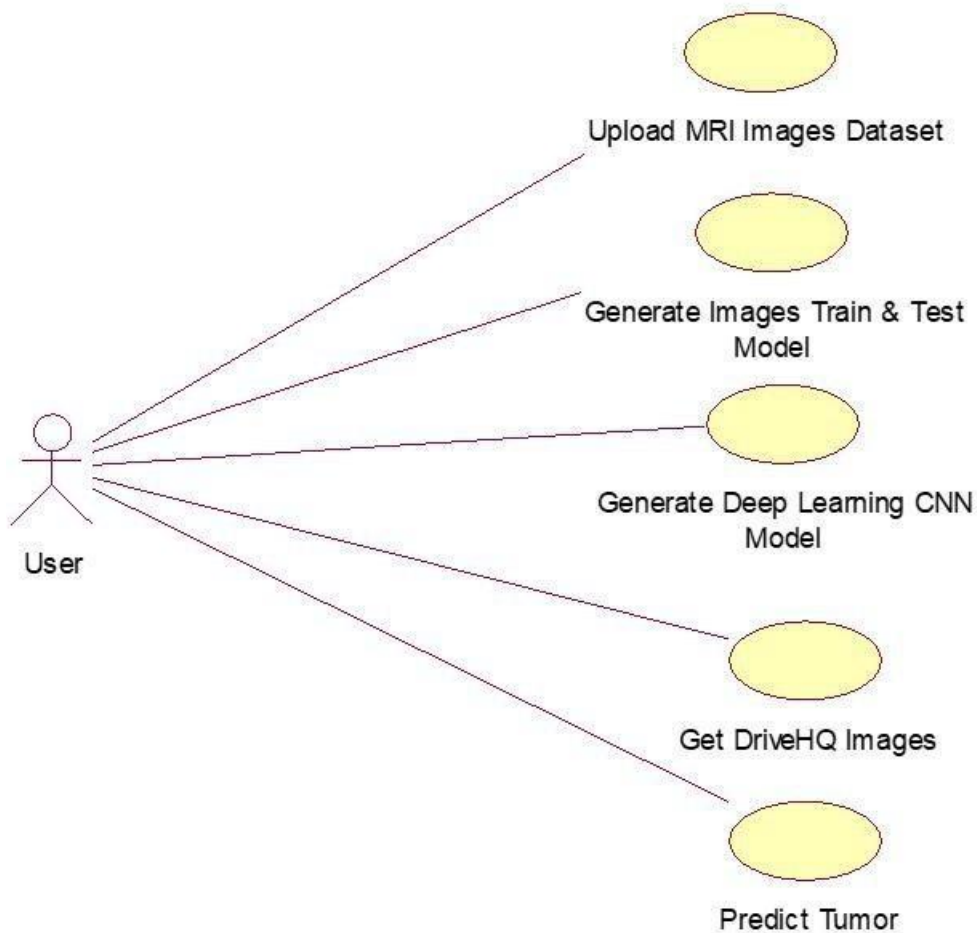


Fig 4.4 Usecase diagram

C. CLASS DIAGRAM

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

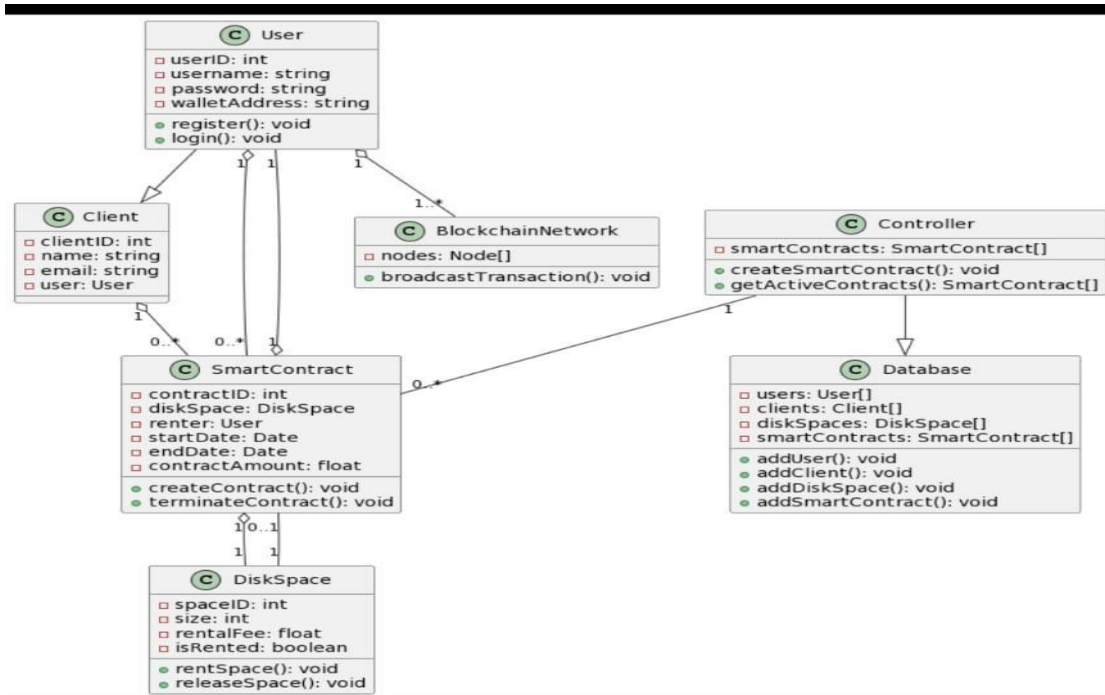


Fig.4.5 Class diagram

D. SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

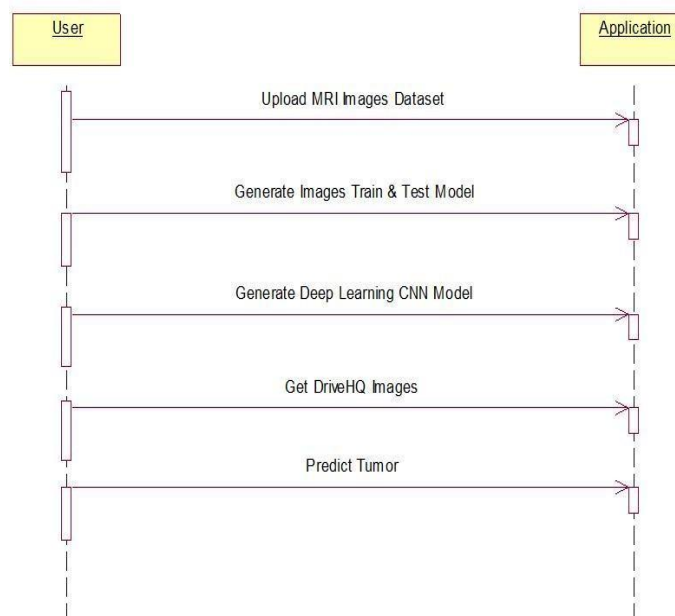


Fig. 4.6 Sequence diagram

E. COMPONENT DIAGRAM

A component diagram breaks down the actual system under development into various levels functionality. Each component is responsible for one clear aim within the entire system and only interacts with other essential elements on a need-to-know basis.

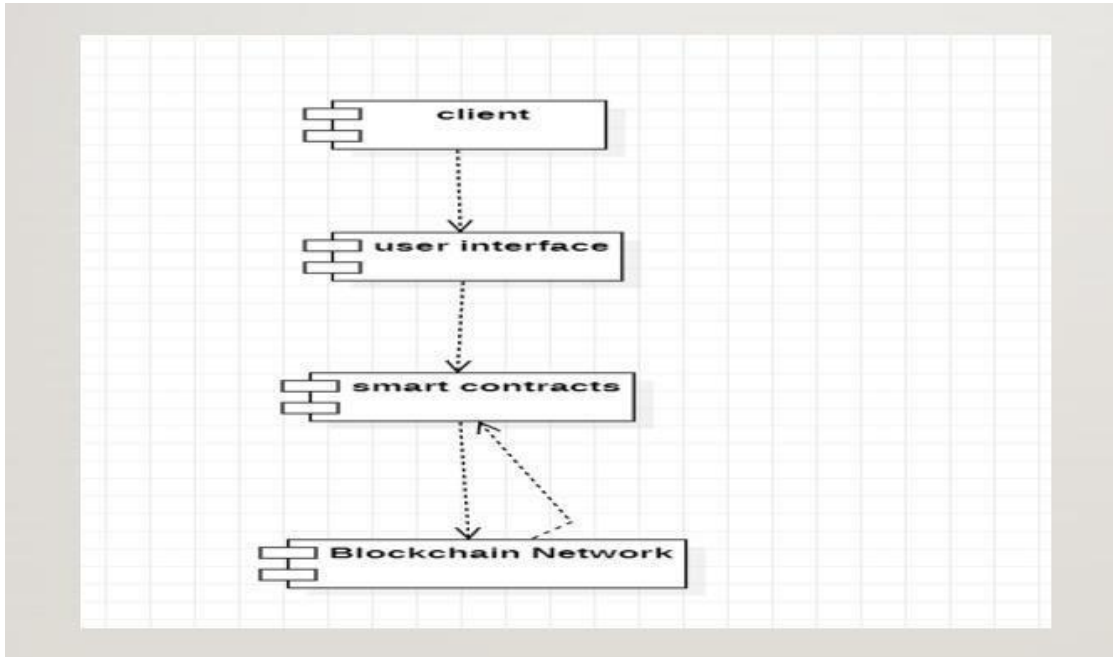


Fig. 4.7 Component diagram

5. IMPLEMENTATION

The implementation of the proposed system involves several key components and technologies to ensure secure and decentralized data sharing.

Firstly, the system utilizes blockchain technology to establish a secure and transparent framework for managing data transactions. Smart contracts are deployed on the blockchain to enforce data sharing agreements and permissions between users.

Secondly, ciphertext-policy attribute-based encryption (CP-ABE) is employed to encrypt and decrypt data based on specific access policies defined by the data owner. This allows for fine-grained access control, ensuring that only authorized users can access sensitive information.

Thirdly, the InterPlanetary File System (IPFS) is utilized for decentralized storage of encrypted data. IPFS enables data to be distributed across a peer-to-peer network, enhancing data availability and resilience while minimizing reliance on centralized servers.

The implementation workflow involves user authentication and authorization, data encryption, storage, and retrieval. Users can register accounts, log in securely, and request access to shared data. Data owners can encrypt their data using CP-ABE and store it on the decentralized IPFS network. Access to shared data is granted based on predefined access policies, ensuring that only authorized users can decrypt and access sensitive information. The system also incorporates mechanisms for monitoring and auditing data transactions on the blockchain, providing transparency and accountability. Additionally, performance metrics such as encryption and decryption overhead are measured and visualized to optimize system efficiency.

5.1. Module

Based on the provided information, the system appears to consist of several modules aimed at implementing a blockchain-based security sharing scheme for personal data (BSSPD). Here's a breakdown of the modules based on the functionalities described:

User Authentication and Management Module: This module handles user authentication, registration, and management. Functionalities include user login, signup, and user profile management.

Data Sharing and Storage Module: This module facilitates the sharing and storage of personal data securely using blockchain and IPFS. Users can upload data, specify access policies using CP-ABE, and securely store the data on the IPFS network.

Includes functionalities for encrypting and decrypting data using CP-ABE.

Access Control Module: Provides fine-grained access control mechanisms based on attributes defined by data owners. Ensures that only authorized users with matching attributes can access and decrypt shared data.

Blockchain Integration Module: Handles interactions with the blockchain network for storing and retrieving shared data and access control policies. Includes functionalities for reading and writing data to the blockchain, as well as querying blockchain records.

IPFS Integration Module: Integrates with the InterPlanetary File System (IPFS) for decentralized storage of shared data. Allows users to store and retrieve data securely using IPFS.

User Interface Module: Provides a user-friendly interface for users to interact with the system. Includes web pages for login, signup, data sharing, viewing shared messages, and other system functionalities.

Data Encryption and Decryption Module: Implements cryptographic operations for encrypting and decrypting data using CP-ABE. Ensures that shared data remains confidential and can only be accessed by authorized users.

System Monitoring and Performance Module: Monitors system performance and generates reports on encryption/decryption overhead, transaction times, etc. Provides insights into the efficiency and reliability of the system.

5.2 Module Description:

The proposed system, named Blockchain-Based Security Sharing Scheme for Personal Data (BSSPD), encompasses a comprehensive framework designed to revolutionize the landscape of data security and privacy in contemporary data-sharing environments. Comprising several key components and leveraging cutting-edge technologies such as blockchain, ciphertext-policy attribute-based encryption (CP-ABE), and the InterPlanetary File System (IPFS), the BSSPD scheme aims to address the pressing challenges associated with centralized data storage and sharing solutions.

At its core, the BSSPD module facilitates decentralized storage through the utilization of IPFS, a distributed peer-to-peer hypermedia protocol. This decentralized approach to data storage ensures maximum availability and resilience while minimizing the risks associated with single points of failure inherent in traditional centralized storage solutions. By encrypting and storing data on IPFS, the BSSPD module enables data owners to retain control over their information, mitigating concerns related to data ownership, security, and privacy. Furthermore, the BSSPD scheme incorporates the principles of ciphertext-policy attribute-based encryption (CP-ABE) to enforce fine-grained access control over shared data. With CP-ABE, data owners can define access policies based on specific attributes, such as user roles or organizational affiliations. Only users whose attributes match the access policy defined by the data owner can access and decrypt the shared data, ensuring confidentiality and integrity while preserving user privacy.

The module also prioritizes usability and user experience, offering a streamlined interface for both data owners and users. Through a user-friendly dashboard, data owners can easily manage access policies, monitor data usage, and track transactions within the system. Similarly, data users benefit from a seamless experience, with intuitive features for accessing and downloading shared data based on their authorized attributes. Additionally, the BSSPD module integrates blockchain technology to enhance data security, transparency, and immutability. By recording transactions and access control policies on a distributed ledger, the blockchain ensures tamper-proof audit trails and accountability throughout the data-sharing process. Any changes or updates to access policies are transparently recorded on the blockchain, providing a verifiable history of data access and usage.

In terms of system architecture, the BSSPD module follows a modular and scalable design, allowing for seamless integration with existing data management systems and applications. The modular architecture facilitates interoperability and extensibility, enabling organizations to customize and tailor the BSSPD framework to meet their specific security and privacy requirements. Moreover, the BSSPD module incorporates robust authentication and authorization mechanisms to authenticate users and enforce access control policies effectively. User authentication is performed using secure login credentials, while access control policies are enforced through cryptographic techniques and smart contracts deployed on the blockchain.

In terms of deployment, the BSSPD module can be implemented across various environments, including cloud-based infrastructures, on-premises servers, and edge computing devices. The flexible deployment options ensure compatibility with diverse organizational setups and enable seamless integration with existing IT ecosystems.

5.3 Introduction to the technologies used:

Implementation of the proposed system involves integrating various technologies to achieve the desired functionality.

1. Blockchain Technology:

Blockchain is a decentralized and distributed digital ledger technology that records transactions across multiple computers in a network. It consists of a chain of blocks, where each block contains a list of transactions and a unique cryptographic hash of the previous block. This design ensures the immutability and security of data stored on the blockchain. At its core, blockchain operates on a consensus mechanism, where network participants validate and confirm transactions through cryptographic algorithms. Once a transaction is verified, it is added to a block and linked to the previous block in a chronological order, forming a continuous chain of blocks. This process creates a transparent and tamper-proof record of transactions, as altering any data in a block would require changing the entire chain, making it computationally infeasible and highly secure.

Blockchain technology is most commonly associated with cryptocurrencies like Bitcoin and Ethereum, where it serves as the underlying infrastructure for recording and verifying transactions. However, its applications extend far beyond digital currencies. Blockchain has found use cases in various industries such as supply chain management, healthcare, finance, and government, where data security, transparency, and traceability are paramount.

One of the key advantages of blockchain is its decentralized nature, eliminating the need for intermediaries like banks or government institutions to facilitate transactions. This decentralization enhances trust among participants, reduces transaction costs, and accelerates the transfer of assets or information.

Moreover, blockchain offers enhanced security features through its consensus mechanisms, cryptographic hashing, and data encryption. This makes it highly resistant to hacking, fraud, and unauthorized tampering, ensuring the integrity and confidentiality of stored data.

2. InterPlanetary File System (IPFS):

IPFS, which stands for InterPlanetary File System, is a decentralized and distributed protocol designed to create a peer-to-peer network for storing and sharing data in a secure, efficient, and censorship-resistant manner. Unlike traditional web protocols that rely on centralized servers, IPFS leverages a network of interconnected nodes, allowing users to access and retrieve content directly from other users' devices. The core concept of IPFS revolves around content addressing, where each piece of data is uniquely identified by its content hash rather than its location. This means that identical data will have the same hash regardless of where it is stored, enabling efficient content distribution and redundancy elimination.

IPFS also employs a distributed hash table (DHT) to facilitate content discovery and retrieval. When a user requests a piece of content, the DHT helps locate the nodes that store that content, enabling direct peer-to-peer file transfer without relying on centralized servers.

One of the key advantages of IPFS is its resilience to censorship and data manipulation. Since content is distributed across multiple nodes in the network, there is no single point of failure or control. Additionally, IPFS supports data versioning and integrity verification through cryptographic hashes, ensuring that retrieved content matches its original state.

IPFS has gained popularity in various applications, including decentralized file storage, content delivery networks (CDNs), and blockchain-based platforms. Its decentralized nature, efficient content addressing, and robust security features make it a powerful protocol for building decentralized and resilient applications in the modern web ecosystem.

3. Ciphertext-Policy Attribute-Based Encryption (CP-ABE):

Ciphertext-Policy Attribute-Based Encryption (CP-ABE) is an advanced encryption technique used to secure data in a way that grants access based on attributes specified by the data owner or administrator. In CP-ABE, access policies are defined using attributes, such as user roles, affiliations, or other characteristics, rather than specific user identities. This approach provides a more flexible and dynamic method of access control, especially in complex systems with varying levels of data sensitivity and diverse user groups.

In CP-ABE, data is encrypted with a policy that specifies the attributes required for decryption. Users or entities are assigned attributes, and they can decrypt the data if their attributes match the access policy. This attribute-based access control model allows for fine-grained access control, where different users with different attribute sets can access different parts of the encrypted data based on their permissions.

CP-ABE is particularly useful in scenarios where access control needs to be dynamic and adaptable, such as in healthcare systems where different medical personnel require access to different patient records based on their roles and responsibilities. It also finds applications in secure data sharing environments, IoT networks, and cloud computing, where maintaining data confidentiality while allowing efficient and controlled access is crucial.

4. Truffle:

Truffle refers to a sophisticated development framework designed specifically for Ethereum blockchain applications. It serves as a comprehensive suite of tools that significantly simplifies and enhances the process of creating, testing, and deploying smart contracts on the Ethereum network. At its core, Truffle provides developers with a seamless environment for writing Solidity smart contracts, the programming language used for Ethereum smart contracts. It streamlines the compilation of these contracts into bytecode, making them deployable on the Ethereum blockchain. This functionality is crucial for developers as it ensures their smart contracts are compatible with the Ethereum network's requirements and standards.

One of Truffle's standout features is its built-in testing framework, which allows developers to create automated tests for their smart contracts. These tests simulate various scenarios and interactions with the smart contract, helping developers identify and rectify any issues or vulnerabilities before deploying the contract on the live Ethereum network. This proactive approach to testing enhances the security, reliability, and overall quality of Ethereum smart contracts developed using Truffle.

Furthermore, Truffle offers integration with other essential tools and libraries in the Ethereum ecosystem, such as Ganache for local blockchain simulation, Web3.js for interacting with Ethereum nodes, and Drizzle for frontend development of decentralized applications (DApps). This seamless integration extends Truffle's capabilities and empowers developers to build end-to-end Ethereum applications efficiently and effectively.

In essence, Truffle plays a pivotal role in accelerating the development lifecycle of Ethereum blockchain applications by providing developers with a robust set of tools, testing capabilities, and integration options, thereby facilitating the creation of secure, reliable, and scalable smart contracts and DApps on the Ethereum network.

5. Python Programming Language:

Python is a high-level programming language known for its simplicity, versatility, and readability. It was created by Guido van Rossum and first released in 1991, designed with an emphasis on code readability and a clear, expressive syntax. Python is widely used across various domains, including web development, data science, artificial intelligence, scientific computing, and automation. One of Python's key strengths is its readability and simplicity, making it accessible to both beginners and experienced programmers. Its syntax is clean and easy to understand, using indentation to define code blocks instead of curly braces or keywords, which enhances code readability and reduces the potential for errors.

Python's versatility is evident in its extensive standard library, which provides a wide range of modules and functions for tasks such as file handling, networking, data manipulation, and more. Additionally, Python has a vast ecosystem of third-party libraries and frameworks, such as Django for web development, NumPy and Pandas for data analysis, TensorFlow and PyTorch for machine learning, and Flask for building APIs. Another notable feature of Python is its cross-platform compatibility, allowing developers to write code once and run it on multiple operating systems without modification. This portability makes Python a popular choice for developing applications that need to run on different platforms seamlessly.

Python's readability, versatility, extensive libraries, cross-platform compatibility, and active community support make it a powerful and preferred language for a wide range of programming tasks, from simple scripts to complex software development projects.

6.HTML, CSS, JavaScript:

HTML, CSS, and JavaScript are fundamental technologies used in web development to create and design interactive and visually appealing websites and web applications. Each of these technologies serves a specific purpose and works together to enhance the user experience and functionality of web pages.

HTML (Hypertext Markup Language):

HTML is the standard markup language used to structure and create the content of web pages. It provides a set of tags and elements that define the structure of a webpage, including headings, paragraphs, lists, links, images, tables, and more. HTML tags are enclosed in angle brackets $\langle \rangle$ and are used to indicate the beginning and end of elements.

CSS (Cascading Style Sheets):

CSS is a style sheet language used to control the presentation and layout of HTML elements on a webpage. It allows developers to apply styles, such as colors, fonts, spacing, borders, and positioning, to HTML elements, making the webpage visually appealing and responsive across different devices.

CSS works by selecting HTML elements using selectors and applying styles to them. Styles can be defined inline within HTML elements, in a separate CSS file, or within a `<style>` block in the HTML document.

JavaScript :

JavaScript is a high-level programming language used to add interactivity, functionality, and behavior to web pages. It enables developers to create dynamic elements, handle user events, manipulate the DOM (Document Object Model), perform calculations, validate forms, and make asynchronous requests to servers (AJAX).

JavaScript code is typically embedded within HTML pages using `<script>` tags or included in separate .js files. It can interact with HTML elements, modify their content and styles, respond to user actions (like clicks and keystrokes), and create interactive features such as sliders, dropdown menus, pop-ups, and more.

7. Truffle Migrate: In the context of Ethereum blockchain development using Truffle, "truffle migrate" refers to the command used to deploy smart contracts onto the blockchain. When developers write and compile their Solidity smart contracts using Truffle, the next step is to deploy these contracts onto the Ethereum network. The "truffle migrate" command initiates this deployment process, which involves sending transactions to the Ethereum network to create and deploy the compiled bytecode of the smart contracts. This command is essential for making the smart contracts functional and accessible on the Ethereum blockchain, enabling interactions and transactions as intended by the developers.

8. Solidity:

Solidity is a high-level programming language used for writing smart contracts on blockchain platforms, with Ethereum being its primary application environment. Smart contracts are self-executing agreements with predefined conditions written in code. Solidity enables developers to define these contracts' logic and behavior, including rules for interactions and transactions within a blockchain network.

The language is specifically designed to target the Ethereum Virtual Machine (EVM), which executes smart contracts on the Ethereum blockchain. Solidity syntax is similar to JavaScript and is designed to be user-friendly for developers familiar with traditional programming languages. Solidity's key features include data types, control structures, functions, inheritance, and libraries, allowing developers to create complex and secure smart contracts. It also supports error handling, event logging, and gas optimization for efficient contract execution.

Solidity plays a crucial role in the decentralized application (dApp) ecosystem by enabling developers to create reliable and secure smart contracts that automate and enforce digital agreements on blockchain networks.

9. Web3.py:

Web3.py is a Python library that serves as an interface for interacting with the Ethereum blockchain using Python code. It provides developers with tools and functionalities to create, deploy, and interact with smart contracts, as well as to manage Ethereum accounts and transactions programmatically. One of the key features of Web3.py is its ability to connect Python applications to Ethereum nodes, allowing developers to access blockchain data, query smart contract states, and send transactions directly from their Python code. This makes it easier for Python developers to integrate Ethereum functionality into their applications without having to write low-level blockchain code.

Web3.py also supports Ethereum's JSON-RPC API, enabling developers to perform a wide range of blockchain operations such as reading and writing data to smart contracts, querying blockchain state, and monitoring events. Additionally, it provides utilities for handling Ethereum units, encoding and decoding data, and managing blockchain accounts securely. Web3.py simplifies Ethereum blockchain development in Python, offering a high-level interface and comprehensive toolset for building Ethereum-powered applications.

10. IPFS API:

The IPFS API refers to the Application Programming Interface (API) provided by the InterPlanetary File System (IPFS), a distributed and peer-to-peer protocol for storing and sharing hypermedia content on a decentralized network. The IPFS API allows developers to interact with the IPFS network programmatically, enabling them to access and manipulate files, publish content, and manage nodes within the IPFS ecosystem. With the IPFS API, developers can perform a variety of operations such as adding files to the IPFS network, retrieving files based on their content identifiers (CID), fetching data from other IPFS nodes, and managing IPFS nodes' configuration and connectivity. The API provides a standardized set of methods and endpoints that developers can use to integrate IPFS functionality into their applications, making it easier to leverage the benefits of decentralized and distributed file storage and sharing.

IPFS API serves as a bridge between applications and the IPFS network, empowering developers to create decentralized and resilient data storage and sharing solutions with ease.

5.4 Sample code

```
from django.shortcuts import render
from django.template import RequestContext
from django.contrib import messages
from django.http import HttpResponse
```

```

import datetime
import ipfsapi
import os
import json
from web3 import Web3, HTTPProvider
from django.core.files.storage import FileSystemStorage
import pickle
from ecies.utils import generate_eth_key, generate_key
from ecies import encrypt, decrypt
import time
import matplotlib.pyplot as plt
import numpy as np

api = ipfsapi.Client(host='http://127.0.0.1', port=5001)
global details, username
global enc_time, dec_time

#function to generate public and private keys for CP-ABE algorithm
def CPABEGenerateKeys():
    if os.path.exists("pvt.key"):
        with open("pvt.key", 'rb') as f:
            private_key = f.read()
        f.close()
        with open("pri.key", 'rb') as f:
            public_key = f.read()
        f.close()
        private_key = private_key.decode()
        public_key = public_key.decode()
    else:
        secret_key = generate_eth_key()
        private_key = secret_key.to_hex() # hex string
        public_key = secret_key.public_key.to_hex()

```

```

    with open("pvt.key", 'wb') as f:
        f.write(private_key.encode())
    f.close()
    with open("pri.key", 'wb') as f:
        f.write(public_key.encode())
    f.close()
    return private_key, public_key

#CP-ABE will encrypt data using plain text and public key
def CPABEEncrypt(plainText, public_key):
    cpabe_encrypt = encrypt(public_key, plainText)
    return cpabe_encrypt

#CP-ABE will decrypt data using private key and encrypted text
def CPABEDecrypt(encrypt, private_key):
    cpabe_decrypt = decrypt(private_key, encrypt)
    return cpabe_decrypt

def readDetails(contract_type):
    global details
    details = ""
    print(contract_type+"=====")
    blockchain_address = 'http://127.0.0.1:9545' #Blockchain connection IP
    web3 = Web3(HTTPProvider(blockchain_address))
    web3.eth.defaultAccount = web3.eth.accounts[0]
    compiled_contract_path = 'BlockchainSecureSharing.json' #Blockchain Secure
    Shared Data contract code
    deployed_contract_address = '0x26e3039c500Cfa3201869460371f1897e8BdC35e'
    #hash address to access Shared Data contract
    with open(compiled_contract_path) as file:
        contract_json = json.load(file) # load contract info as JSON
        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its
        functions

```

```

file.close()

contract = web3.eth.contract(address=deployed_contract_address,
abi=contract_abi) #now calling contract to access data

if contract_type == 'signup':
    details = contract.functions.getSignup().call()
if contract_type == 'attribute':
    details = contract.functions.getAttribute().call()
print(details)

def saveDataBlockchain(currentData, contract_type):
    global details
    global contract
    details = ""
    blockchain_address = 'http://127.0.0.1:9545'
    web3 = Web3(HTTPProvider(blockchain_address))
    web3.eth.defaultAccount = web3.eth.accounts[0]
    compiled_contract_path = 'BlockchainSecureSharing.json' #Blockchain contract
    file
    deployed_contract_address = '0x26e3039c500Cfa3201869460371f1897e8BdC35e'
    #contract address

    with open(compiled_contract_path) as file:
        contract_json = json.load(file) # load contract info as JSON
        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its
        functions
    file.close()

    contract = web3.eth.contract(address=deployed_contract_address,
abi=contract_abi)

    readDetails(contract_type)
    if contract_type == 'signup':
        details+=currentData
        msg = contract.functions.setSignup(details).transact()
        tx_receipt = web3.eth.waitForTransactionReceipt(msg)
    if contract_type == 'attribute':
        details+=currentData

```



```

msg = contract.functions.setAttribute(details).transact()
tx_receipt = web3.eth.waitForTransactionReceipt(msg)

def index(request):
    if request.method == 'GET':
        return render(request, 'index.html', {})

def Login(request):
    if request.method == 'GET':
        return render(request, 'Login.html', {})

def Signup(request):
    if request.method == 'GET':
        return render(request, 'Signup.html', {})

def SharedData(request):
    if request.method == 'GET':
        global username
        readDetails('signup')
        arr = details.split("\n")

        status = '<tr><td><font size="'
color="black">Choose&nbsp;Shared&nbsp;Users</b></td><td><select name="t3"
multiple>'

        for i in range(len(arr)-1):
            array = arr[i].split("#")
            if array[1] != username:
                status += '<option value="'+array[1]+'">'+array[1]+'</option>'

        status += "</select></td></tr>"

        context= {'data1':status}

        return render(request, 'SharedData.html', context)

def ViewSharedMessages(request):
    if request.method == 'GET':
        global enc_time, dec_time, username
        dec_time = 0

```

```

        strdata = '<table border=1 align=center width=100%><tr><th><font size=""
color="black">Data Owner</th><th><font size="" color="black">Shared
Message</th>'

        strdata+='<th><font size="" color="black">IPFS Image Address</th><th><font
size="" color="black">Shared Image</th>'

        strdata+='<th><font size="" color="black">Shared Date Time</th><th><font
size="" color="black">Shared Data Users</th></tr>'

    for root, dirs, directory in os.walk('static/tweetimages'):

        for j in range(len(directory)):

            os.remove('static/tweetimages/'+directory[j])

    readDetails('attribute')
    arr = details.split("\n")
    start_times = time.time()
    for i in range(len(arr)-1):
        array = arr[i].split("#")
        share_user = array[6].split(",")
        if array[0] == 'post' and (username in share_user or username == array[1]):
            content = api.get_pyobj(array[3])
            private_key, public_key = CPABEGenerateKeys()
            decrypted = CPABEDecrypt(content, private_key)
            content = pickle.loads(decrypted)
            with open("BlockchainSecurityApp/static/shareimages/"+array[5], "wb") as
file:

                file.write(content)

            file.close()

            strdata+='<tr><td><font size=""
color="black">'+str(array[1])+'</td><td><font size=""
color="black">'+array[2]+'</td><td><font size=""
color="black">'+str(array[3])+'</td>'

            strdata+='<td><img src=static/shareimages/'+array[5]+' width=200
height=200></img></td>'

            strdata+='<td><font size="" color="black">'+str(array[4])+'</td>'
            strdata+='<td><font size="" color="black">'+str(array[6])+'</td>'

    end_times = time.time()
    dec_time = end_times - start_times

```

```

context= {'data':strdata}
return render(request, 'ViewSharedMessages.html', context)

```

```

def LoginAction(request):
    if request.method == 'POST':
        global username
        username = request.POST.get('t1', False)
        password = request.POST.get('t2', False)
        readDetails('signup')
        arr = details.split("\n")
        status = "none"
        for i in range(len(arr)-1):
            array = arr[i].split("#")
            if array[1] == username and password == array[2]:
                status = "Welcome "+username
                break
        if status != 'none':
            file = open('session.txt','w')
            file.write(username)
            file.close()
            context= {'data':status}
            return render(request, 'UserScreen.html', context)
        else:
            context= {'data':'login failed'}
            return render(request, 'Login.html', context)

```

```

def SharedDataAction(request):
    if request.method == 'POST':
        global enc_time, username
        post_message = request.POST.get('t1', False)

```

```

share = request.POST.getlist('t3')
share = ','.join(share)
filename = request.FILES['t2'].name
start = time.time()
myfile = request.FILES['t2'].read()
myfile = pickle.dumps(myfile)
private_key, public_key = CPABEGenerateKeys()
cpabe_encrypt = CPABEEncrypt(myfile, public_key)
now = datetime.datetime.now()
current_time = now.strftime("%Y-%m-%d %H:%M:%S")
user = username
hashcode = api.add_pyobj(cpabe_encrypt)

data =
"post#" + user + "#" + post_message + "#" + str(hashcode) + "#" + str(current_time) + "#" + filename + "#" + share + "\n"

end = time.time()
enc_time = end - start
saveDataBlockChain(data, "attribute")

output = 'Shared Data saved in Blockchain with below hashcodes & Image file saved in IPFS.<br/>' + str(hashcode)

context= {'data':output}

return render(request, 'SharedData.html', context)

```

```
def SignupAction(request):
```

```
    if request.method == 'POST':
```

```
        global details
```

```
        username = request.POST.get('t1', False)
```

```
        password = request.POST.get('t2', False)
```

```
        contact = request.POST.get('t3', False)
```

```
        gender = request.POST.get('t4', False)
```

```
        email = request.POST.get('t5', False)
```

```
        address = request.POST.get('t6', False)
```

```

output = "Username already exists"
readDetails('signup')
arr = details.split("\n")
status = "none"
for i in range(len(arr)-1):
    array = arr[i].split("#")
    if array[1] == username:
        status = username+" already exists"
        break
if status == "none":
    details = ""
    data =
"signup#" + username + "#" + password + "#" + contact + "#" + gender + "#" + email + "#" + address + "\n"
    saveDataBlockchain(data,"signup")
    context = {"data": "Signup process completed and record saved in Blockchain"}
    return render(request, 'Signup.html', context)
else:
    context = {"data": status}
    return render(request, 'Signup.html', context)

def Graph(request):
    if request.method == 'GET':
        global username
        global enc_time, dec_time
        height = [enc_time, dec_time]
        bars = ('CP-ABE Encryption Time', 'CP-ABE Decryption Time')
        y_pos = np.arange(len(bars))
        plt.bar(y_pos, height)
        plt.xticks(y_pos, bars)
        plt.title("Uploading, Encryption & Decryption Overhead Graph")
        plt.show()

```

```
context = {"data":"Welcome "+username}  
return render(request, 'UserScreen.html', context)
```

HTML Code:

```
{% load static %}  
<html>  
<head>  
<title>BSSPD</title>  
<meta http-equiv="content-type" content="text/html; charset=utf-8" />  
<link href="{% static 'style.css' %}" rel="stylesheet" type="text/css" />  
<script language="javascript">  
    function validate(formObj)  
    {  
        if(formObj.t1.value.length==0)  
        {  
            alert("Please Enter username");  
            formObj.t1.focus();  
            return false;  
        }  
        if(formObj.t2.value.length==0)  
        {  
            alert("Please Enter password");  
            formObj.t2.focus();  
            return false;  
        }  
  
        if(formObj.t3.value.length==0)  
        {  
            alert("Please Enter Contact No");  
            formObj.t3.focus();  
            return false;  
        }  
    }  
</script>
```

```

    }
    if(isNaN(formObj.t3.value)){
        alert("Contact No must be numeric");
        formObj.t3.focus();
        return false;
    }
    if(formObj.t5.value.length==0)
    {
        alert("Please Enter Email Id");
        formObj.t5.focus();
        return false;
    }
    var filter = /^[a-zA-Z0-9_\.\\-]+\@[a-z]+\.(com)+$/;
    if (!filter.test(formObj.t5.value)) {
        window.alert('Please enter valid email address');
        formObj.t5.focus();
        return false;
    }
    if(formObj.t6.value.length==0)
    {
        alert("Please Enter Address");
        formObj.t6.focus();
        return false;
    }
    return true;
}
</script>
<script language="javascript" type="text/javascript" src="datetimepicker.js">
</script>
</head>
<body>
<div class="main">

```

```

<div class="main_resize">

  <div class="header">

    <div class="logo">

      <h1><span>BlockStore: A Secure Decentralized Storage Framework On
Blockchain</span><small></small></h1>

    </div>

  </div>

  <div class="content">

    <div class="content_bg">

      <div class="menu_nav">

        <ul>

          <li><a href="{% url 'index' %}">Home</a></li>

          <li><a href="{% url 'Login' %}">User Login Here</a></li>

          <li><a href="{% url 'Signup' %}">New User Signup Here</a></li>

        </ul>

      </div>

      <div class="hbg"></div>

      <center>

<form name="f1" method="post" action="{% url 'SignupAction' %}" onsubmit="return
validate(this);">

{% csrf_token %}<br/>

<h2><b>User Signup Screen</b></h2>

      <font size="" color="red"><center>{{ data|safe }}</center></font>

      <table align="center" width="30%" >

        <tr><td><font size=" color=black>Username</b></td>

          <td><input type="text" name="t1" style="font-family: Comic
Sans MS" size="30"></td></tr>

        <tr><td><font size=" color=black>Password</b></td><td><input
name="t2" type="password" size="30"></td></tr>

        <tr><td><font size=" color=black>Contact&nbsp;No</b></td>

          <td><input name="t3" type="Text" size="15"></td></tr>

```



```

        <tr><td><font size=" color=black>Gender</b></td><td><select
name="t4" style="font-family: Comic Sans MS">
        <option value="Male">Male</option>
        <option value="Female">Female</option>
        </select>
        </td></tr>
        <tr><td><font size="
color=black>Email&nbsp;ID</b></td><td><input type="text" name="t5"
style="font-family: Comic Sans MS" size=50/></td></tr>
        <tr><td><font size=" color=black>Address</b></td><td><input name="t6"
type="Text" size="70"></td></tr>
        <tr><td></td><td><input type="submit"
value="Submit"></td>
        </table>
        </div>
        </div>
    </body>
</html>

```

Solidity Code:

```
pragma solidity >= 0.8.11 <= 0.8.11;
```

```

contract BlockchainSecureSharing {
    string public user_registration;
    string public attribute_manage

    //call this function to save new user details data to Blockchain
    function setSignup(string memory ur) public {
        user_registration = ur;
    }
    //get Signup details
    function getSignup() public view returns (string memory) {
        return user_registration;
    }
}

```

```

//call this function to save attribute details to Blockchain
function setAttribute(string memory am) public {
    attribute_manage = am;
}
//get attributes details
function getAttribute() public view returns (string memory) {
    return attribute_manage;
}

constructor() public {
    user_registration="";
    attribute_manage="";
}
}

```

6. TESTCASES

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product. It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

TYPES OF TESTS

Unit testing

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application. It is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process.

Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfactory, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

Functional testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

- Valid Input : identified classes of valid input must be accepted.
- Invalid Input : identified classes of invalid input must be rejected.
- Functions : identified functions must be exercised.
- Output : identified classes of application outputs must be exercised.
- Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special testcases. In addition, systematic pertaining to identify Business process flows; data fields, predefined processes, and successive processes be considered for testing. Before functional testing is complete, additional tests are identified and the value of current tests is determined.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the enduser. It also ensures that the system meets the functional requirements.

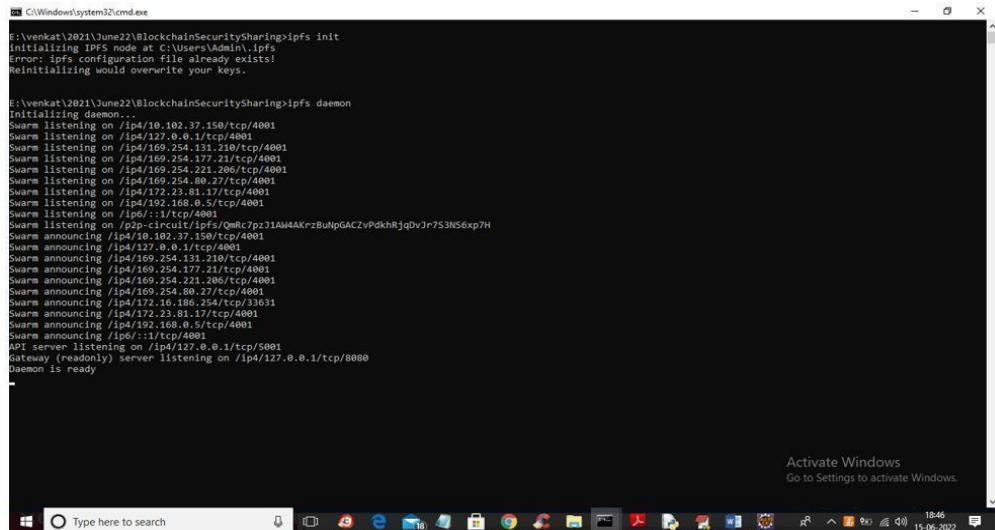
Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Table 6.1 Test cases for the proposed system

TEST ID	TEST DESCRIPTION	TEST DESIGN	EXPECTED OUTPUT	ACTUAL OUTPUT	STATUS
1	Insert signup information	Save the information	Signup process completed	Signup process completed	passed
2	Login using registered information	Verify and login the user	Login successful	Login successful	passed
3	Upload photo	Save photo in blockchain	Photo saved and displayed	Photo saved and displayed	passed
4	Download photo	Download photo from blockchain	Photo downloaded	Photo downloaded	passed
5	Login without registration	Login failed	Invalid login	Invalid login	passed

7. SCREENSHOTS

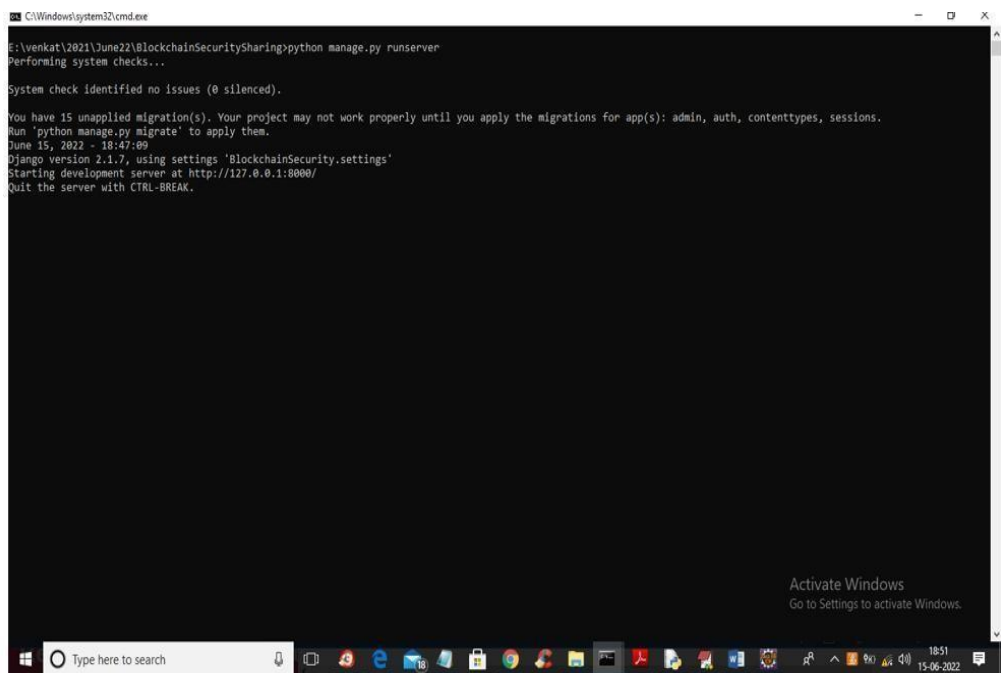
To run project first double click on 'Start_IPFS.bat' file to start IPFS server and get below screen



```
C:\Windows\system32\cmd.exe
E:\venkat\2021\June22\BlockchainSecuritySharing>ipfs init
Initializing IPFS node at C:\Users\Admin\.ipfs
Error: ipfs configuration file already exists!
Reinitializing would overwrite your keys.

E:\venkat\2021\June22\BlockchainSecuritySharing>ipfs daemon
Initializing daemon...
Swarm listening on /ip4/10.102.37.150/tcp/4001
Swarm listening on /ip4/127.0.0.1/tcp/4001
Swarm listening on /ip4/169.254.131.210/tcp/4001
Swarm listening on /ip4/169.254.177.21/tcp/4001
Swarm listening on /ip4/169.254.221.206/tcp/4001
Swarm listening on /ip4/169.254.80.27/tcp/4001
Swarm listening on /ip4/172.23.81.17/tcp/4001
Swarm listening on /ip4/192.168.0.5/tcp/4001
Swarm listening on /ip6:::1/tcp/4001
Swarm listening on /p2p-circuit/ipfs/QmKc7p2J1Am4AKrzBUNpGACZvPdKhrJgDv3r753NS5xp7H
Swarm announcing /ip4/10.102.37.150/tcp/4001
Swarm announcing /ip4/127.0.0.1/tcp/4001
Swarm announcing /ip4/169.254.131.210/tcp/4001
Swarm announcing /ip4/169.254.177.21/tcp/4001
Swarm announcing /ip4/169.254.221.206/tcp/4001
Swarm announcing /ip4/169.254.80.27/tcp/4001
Swarm announcing /ip4/172.23.81.17/tcp/4001
Swarm announcing /ip4/192.168.0.5/tcp/4001
Swarm announcing /ip6:::1/tcp/4001
API server listening on /ip4/127.0.0.1/tcp/5001
Gateway (readonly) server listening on /ip4/127.0.0.1/tcp/8080
Daemon is ready
```

FIG 7.1: In above screen IPFS server started and now double click on 'runServer.bat' file to start python DJANGO server and get below screen



```
C:\Windows\system32\cmd.exe
E:\venkat\2021\June22\BlockchainSecuritySharing>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
June 15, 2022 - 18:47:09
Django version 2.1.7, using settings 'BlockchainSecurity.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

FIG 7.2: python DJANGO server started and now open browser and enter URL as 'http://127.0.0.1:8000/index.html' and press enter key to get below screen.

In above screen click on 'New User Signup Here' link to add new user to Blockchain

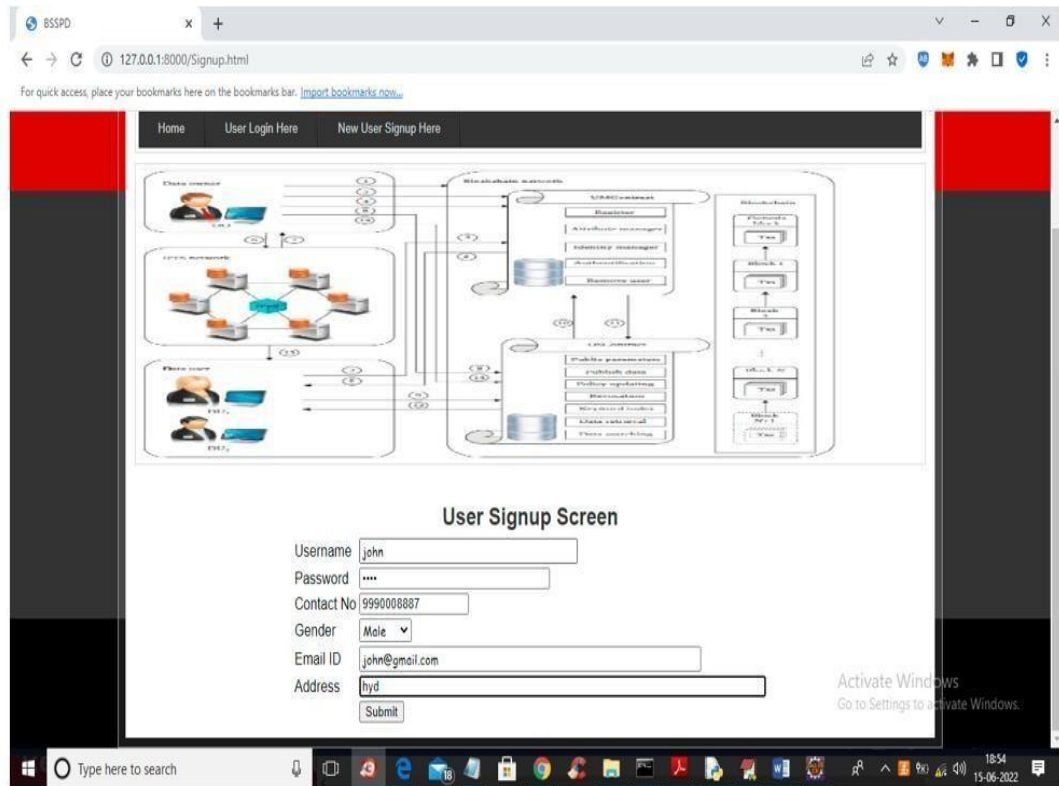


FIG 7.3: In above screen user is signup and press button to get below output.

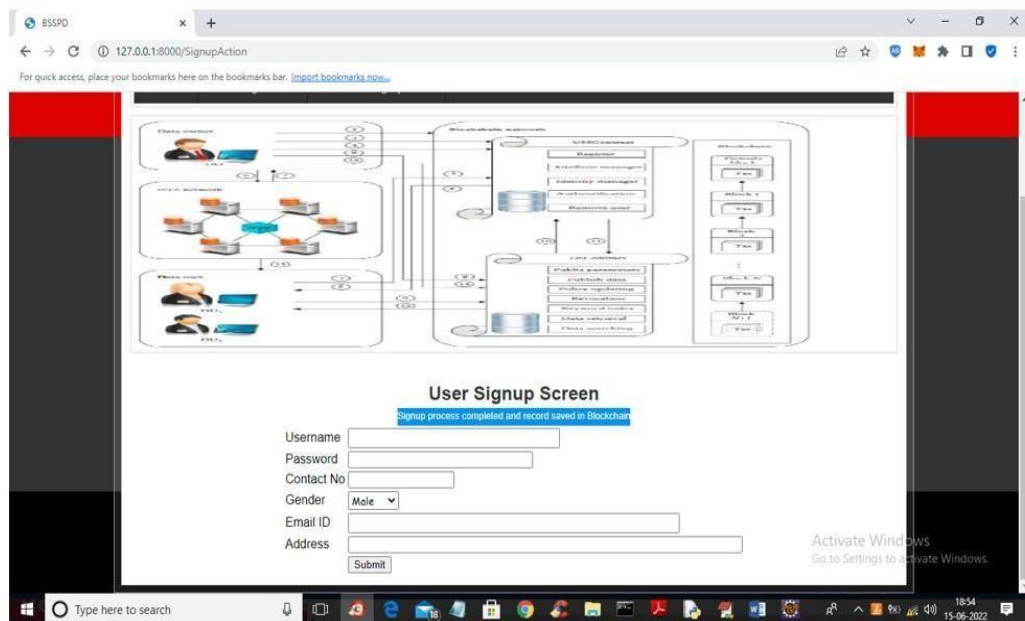


FIG 7.4: user signup process completed and similarly you can add any number of users and now click on 'User Login Here' link to get below login screen.
In above screen user is login and press button to get below output

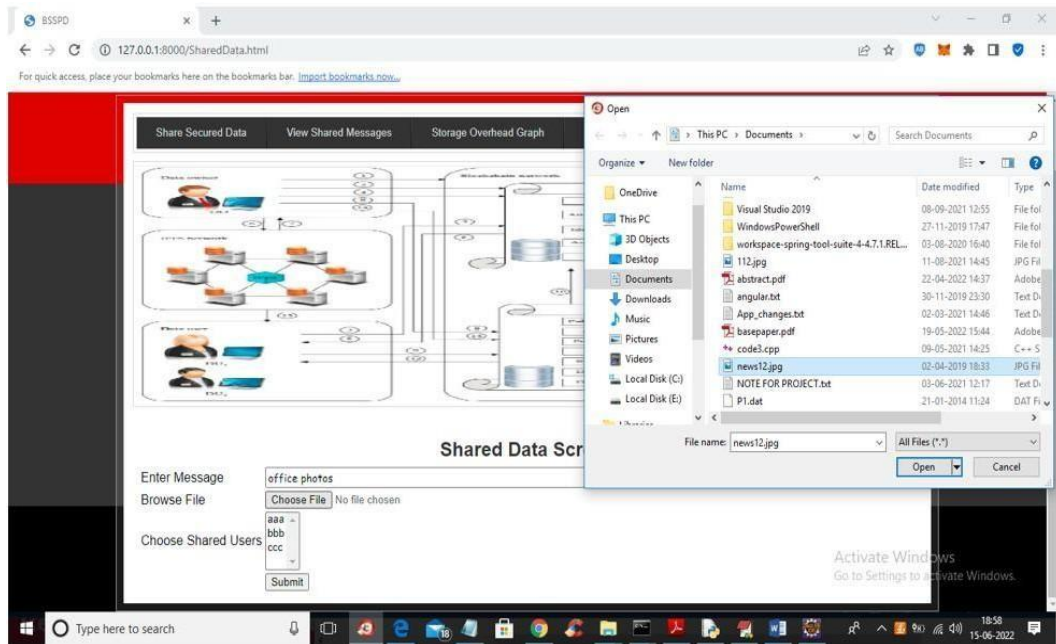


FIG 7.5: user can enter some message and then upload image and by holding CTRL KEY you can select names of users with whom you want to share this data and press button to get below output.

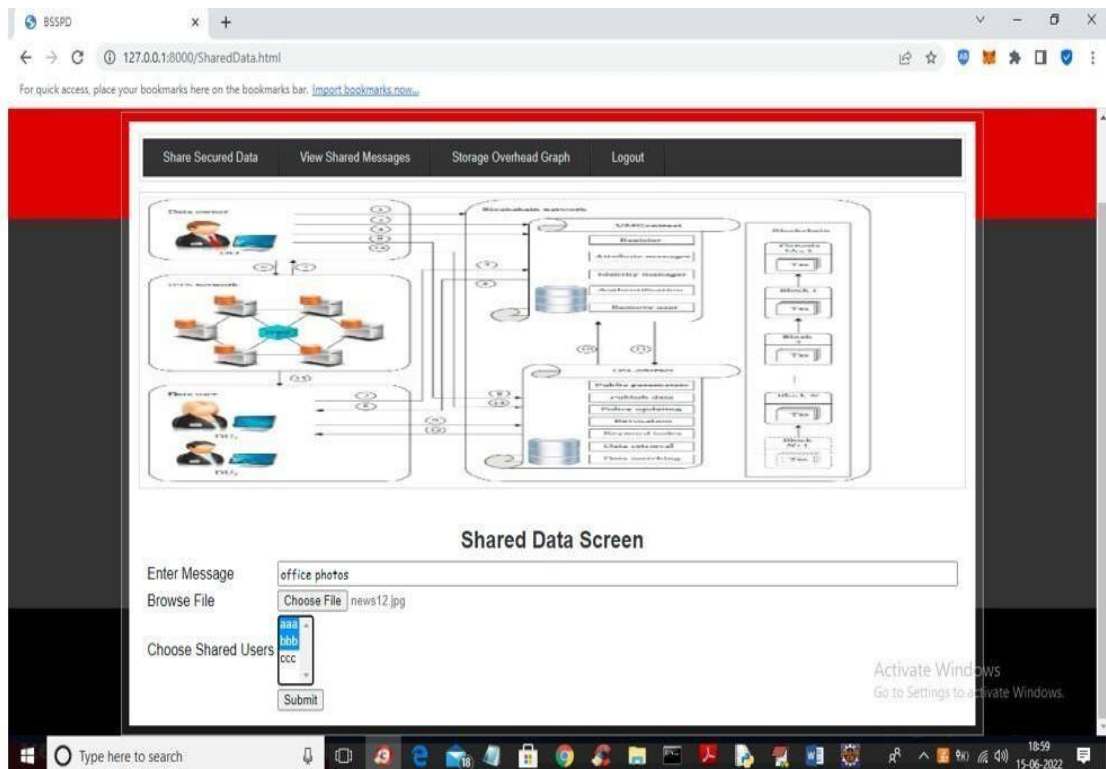


FIG 7.6: 'John' is sharing data with user 'aaa' and 'bbb' and both users can decrypt and view data but user 'ccc' cannot view it.

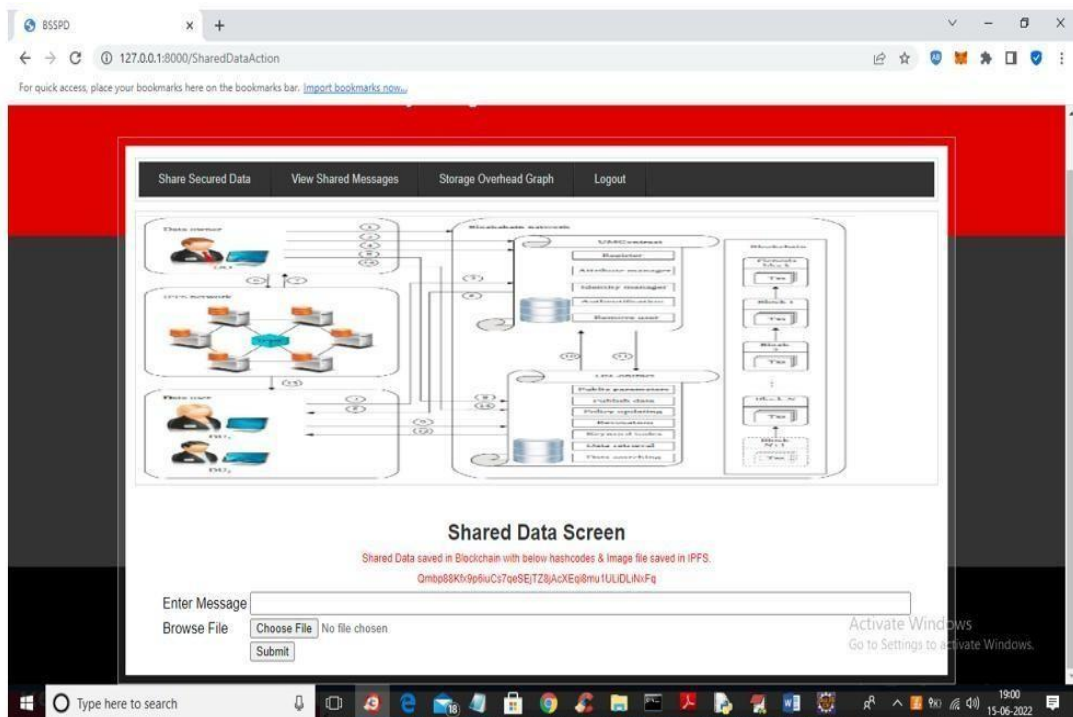


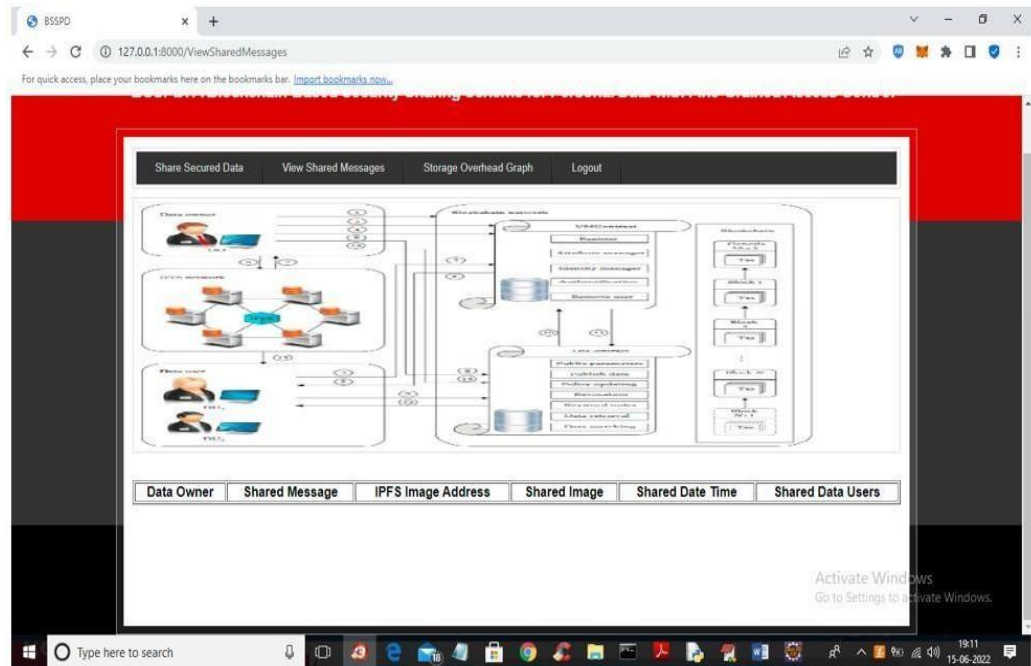
FIG 7.7: we can see sharing attributes stored at Blockchain and images and decryption keys stored at IPFS and now click on ‘View Shared Messages’ link to view own messages and other users shared messages so ‘John’ is the data owner so he can view his own upload and others shared data.

For quick access, place your bookmarks here on the bookmarks bar: [import bookmarks now...](#)

Data Owner	Shared Message	IPFS Image Address	Shared Image	Shared Date Time	Shared Data Users
aaa	school photos	QmYVDEdDhQpWvXVok638fJvesHy5Xkp3yK3PB2uwr52TuL		2022-06-15 17:28:15	bbb
john	office photos	Qmbp88Kt9p6iuc57qeSEjTZ8jAcXEqi8mu1ULiDLiNxFq	These 'Secrets' are too good to keep under your hat	2022-06-15 18:59:20	aaa,bbb
aaa	test data	QmYVBwQio8LMb5Le1fBCDcwYQqvqV3XCPBDatgySs7wtB	These 'Secrets' are too good to keep under your hat	2022-06-15 19:04:25	bbb

Activate Windows
Go to Settings to activate Windows.

FIG 7.8: screen ‘bbb’ can view shared data from aaa and john and now logout and login as ‘ccc’ and nobody shared data with ‘ccc’ so he cannot access any data



8. CONCLUSION

The proposed blockchain-based security sharing scheme for personal data (BSSPD) addresses the critical challenges of data security and privacy in contemporary data-sharing environments, especially in the context of rapid advancements in technologies like 5G, Internet of Things (IoT), and artificial intelligence (AI).

The introduction highlights the significance of data security and privacy in the era of big data and AI, emphasizing the sensitivity of personal data generated by IoT devices and the potential risks associated with centralized cloud storage. Motivated by these challenges, the BSSPD scheme leverages blockchain technology, ciphertext-policy attribute-based encryption (CP-ABE), and the InterPlanetary File System (IPFS) to ensure decentralized, secure, and privacy-preserving data sharing.

The literature survey provides insights into related research, including decentralized storage frameworks, data sanitization techniques, and efficient mechanisms for data uploading in cyber-physical systems. These studies underscore the importance of leveraging advanced technologies to mitigate privacy risks and ensure secure data handling.

The analysis section evaluates the existing system's drawbacks, such as complexity, speed, reliability, and scalability, and proposes the BSSPD scheme as a solution. It outlines the key components and steps involved in the proposed scheme, including decentralized storage with IPFS, attribute-based encryption for fine-grained access control, and the use of blockchain for ensuring data integrity and transparency.

The system requirement specification delineates the software and hardware requirements for implementing the BSSPD scheme, ensuring compatibility and performance efficiency.

The design phase elaborates on the architecture of the proposed system, incorporating UML diagrams to depict the system's components and their interactions. It assesses the economic, technical, and social feasibility of the system, ensuring its viability and acceptance among users.

The implementation section presents the code snippets and functionalities for user authentication, data sharing, encryption, and decryption, integrating blockchain, CP-ABE, and IPFS technologies.

In BSSPD scheme offers a comprehensive solution to the challenges of data security and privacy in contemporary data-sharing environments. By leveraging advanced technologies and decentralized architectures, it empowers users to securely share and manage their data while ensuring confidentiality, integrity, and transparency.

9. FUTURE ENHANCEMENTS

Future enhancements for the proposed blockchain-based security sharing scheme for personal data (BSSPD) could focus on several key areas to further improve its effectiveness and adaptability in evolving technological landscapes. Firstly, integration with emerging blockchain advancements such as sharding and sidechains could enhance scalability and throughput, enabling the BSSPD to handle larger volumes of data transactions efficiently. Additionally, incorporating advanced cryptographic techniques beyond CP-ABE, such as homomorphic encryption or zero-knowledge proofs, could provide even stronger privacy guarantees while maintaining data usability. Moreover, enhancing interoperability with other decentralized technologies and platforms could enable seamless data sharing across different ecosystems while maintaining security and privacy standards. Furthermore, the adoption of decentralized identity solutions could strengthen user authentication and access control mechanisms, further enhancing the security posture of the BSSPD. Lastly, continuous research and development efforts should focus on optimizing user experience, reducing overhead, and ensuring compatibility with emerging technologies to ensure the BSSPD remains at the forefront of secure and privacy-preserving data sharing solutions.

10. BIBLIOGRAPHY

- [1] X. Zhang, J. Grannis, I. Baggili, and N. L. Beebe, “Frameup: An incriminatory attack on Storj: A peer to peer blockchain enabled distributed storage system,” *Digit. Investig.*, vol. 29, pp. 28–42, 2019, doi: [10.1016/j.diin.2019.02.003](https://doi.org/10.1016/j.diin.2019.02.003).
- [2] F. Shawn and J. L. Wilkinson, “Metadisk: Blockchain-based decentralized file storage application,” *Liq. Cryst.*, vol. 14, no. 2, pp. 573–580, 2014.
- [3] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, “Storj a peer-to-peer cloud storage network,” IPFS, U.K., Tech. Rep., 2016, pp. 1–37.
- [4] I. Vakiliinia, S. Vakiliinia, S. Badsha, E. Arslan, and S. Sengupta, “Pooling approach for task allocation in the blockchain based decentralized storage network,” in *Proc. 15th Int. Conf. Netw. ServiceManage. (CNSM)*, Oct. 2019, pp. 1–6, doi: [10.23919/CNSM46954.2019.9012719](https://doi.org/10.23919/CNSM46954.2019.9012719).
- [5] A. M. Girgis, O. Ercetin, M. Nafie, and T. ElBatt, “Decentralized coded caching in wireless networks: Trade-off between storage and latency,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017, pp. 2443–2447.
- [6] Z. Kong, S. A. Aly, and E. Soljanin, “Decentralized coding algorithms for distributed storage in wireless sensor networks,” *IEEE J. Sel. Areas Commun.*, vol. 28, no. 2, pp. 261–267, Feb. 2010, doi: [10.1109/JSAC.2010.100215](https://doi.org/10.1109/JSAC.2010.100215).
- [7] C. Cai, X. Yuan, and C. Wang, “Towards trustworthy and private keyword search in encrypted decentralized storage,” in *Proc. IEEE Int. Conf. Commun. (ICC)*, Paris, France, Jul. 2017, doi: [10.1109/ICC.2017.7996810](https://doi.org/10.1109/ICC.2017.7996810).
- [8] N. Z. Benisi, M. Aminian, and B. Javadi, “Blockchain-based decentralized storage networks: A survey,” *J. Netw. Comput. Appl.*, vol. 162, Jul. 2020, Art. no. 102656, doi: [10.1016/j.jnca.2020.102656](https://doi.org/10.1016/j.jnca.2020.102656).
- [9] A. Shah, N. Sheoran, and S. Gupta, “Decentralized storage network with smart contract incentivisation candidate’s declaration,” Bachelor Technol. Comput. Sci. Eng., Tech. Rep., 2018.

- [10] M. Aloqaily, O. Bouachir, A. Boukerche, and I. A. Ridhawi, “Design guidelines for blockchain- assisted 5G-UAV networks,” *IEEE Netw.*, vol. 35, no. 1, pp. 64–71, Jan. 2021.
- [11] M. Firdaus and K. H. Rhee, “On blockchain-enhanced secure data storage and sharing in vehicular edge computing networks,” *Appl. Sci.*, vol. 11, no. 1, pp. 1–21, Jan. 2021, doi: [10.3390/app11010414](https://doi.org/10.3390/app11010414).
- [12] N. Lipusch, “Initial coin offerings—A paradigm shift in funding disruptive innovation,” *SSRN Electron. J.*, vol. 139, pp. 1–21, Mar. 2018, doi: [10.2139/ssrn.3148181](https://doi.org/10.2139/ssrn.3148181).
- [13] S. Yang, A. Hareedy, R. Calderbank, and L. Dolecek, “Topology-aware cooperative data protection in blockchain-based decentralized storage networks,” in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2020, pp. 1-6