

A
Major Project Report
on
**BLOCKCHAIN A GAME CHANGER FOR SECURE
TRANSFERRING OF DATA**

Submitted in partial fulfilment of the requirements for the award of the Degree of
Bachelor of Technology

By
Kandula Surya Prakash
(20EG105526)

Narayandas Varun Karthikeya
(20EG105537)

Puppireddy Sai Krishna
(20EG105731)



Under The Guidance Of

M. Sandhya Rani

Assistant 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 “**BLOCKCHAIN A GAME CHANGER FOR SECURE TRANSFERRING OF DATA**” submitted for the award of Bachelor of technology Degree is my original work and the report has not formed the basis for the award of any degree, diploma, associate ship 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

K. Surya Prakash(20EG105526)

N. Varun Karthikeya(20EG105537)

P. Sai Krishna(20EG105731)

CERTIFICATE

This is to certify that the project report entitled “**BLOCKCHAIN A GAME CHANGER FOR SECURE TRANSFERRING OF DATA**” being submitted by **Mr. K. Surya Prakash** bearing the Hall Ticket number **20EG105526**, **Mr. N. Varun Karthikeya** bearing the Hall Ticket number **20EG105537**, **Mr. P. Sai Krishna** bearing the Hall Ticket number **20EG105371** in partial fulfilment of the requirements for the award of the degree 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 presented in this report have been verified and found to be satisfactory. 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

M. Sandhya Rani
Assistant Professor
Department of CSE

Signature of Dean CSE

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 **M. Sandhya Rani, Assistant Professor, Department of Computer Science and Engineering**, Anurag University for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered us towards the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

We would like acknowledge our sincere gratitude for the support extended by **Dr. G. VISHNU MURTHY, Dean, Department of Computer Science and Engineering**, Anurag University. We also express my deep sense of gratitude to **Dr. V. V. S. S. S. BALARAM, Academic Co-ordinator, Dr. Pallam Ravi**, Project Co-ordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated me during the crucial stage our project work.

We would like 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.

Kandula Surya Prakash
(20EG105526)

Narayandas Varun Karthikeya
(20EG105537)

Puppireddy Sai Krishna
(20EG105731)

ABSTRACT

The limitations of centralized file transfer systems in providing the necessary distributed trust and transparency essential for collaborative operations. Recognizing the potential of blockchain technology to address these challenges, the paper proposes a novel blockchain-based secure file transfer system tailored for consortiums.

Key elements of the proposed system include leveraging Hyperledger Fabric, a robust enterprise blockchain framework, for establishing the blockchain network and implementing smart contracts. Additionally, the Inter Planetary File System (IPFS) is employed for decentralized file storage, ensuring data is distributed across the network for enhanced security and availability.

The paper elucidates the workflow for identity management and file-sharing processes within the proposed system, emphasizing the importance of confidentiality, integrity, and availability in facilitating secure information exchange among consortium members.

By offering a comprehensive solution that integrates blockchain technology with decentralized file storage, the proposed system aims to address the inherent limitations of centralized systems while enabling consortiums of organizations to collaborate securely and transparently. This research not only contributes to advancing the field of blockchain applications but also offers practical insights for enhancing collaborative operations among organizations.

INDEX

S.No	Content	Page No.
1.	Abstract	v
2.	List of Figures	viii-ix
3.	Introduction	1-5
	1.1 Overview	1
	1.2 Research Motivation	2
	1.3 Problem Statement	3
	1.4 Applications	4-5
4.	Literature Survey	6-9
5.	System Analysis and Design	10-18
	3.1 Existing System	10-12
	3.2 Proposed System	13-18
6.	UML Diagrams	19-29
	4.1 Use Case Diagram	19-22
	4.2 Class Diagram	23-26
	4.3 Sequence Diagram	26-28
	4.4 Activity Diagram	28-29
7.	Software Environment	30-38
	5.1 Software Overview	30-31
	5.2 Django Web Framework for Python	31-33
	5.3 About MySQL	33-38
8.	System Requirements and Specifications	39
	6.1 Hardware Requirements	39
	6.2 Software Requirements	39
9.	Functional Requirements	40-41
	7.1 Output Design	40
	7.2 Input Design	41
10.	Source Code	42-52
	8.1 Main Code	42-52

11.	Result and Analysis	53-57
12.	Conclusion and Future Enhancement	58-59
	10.1 Conclusion	58
	10.2 Future Enhancement	58-59
13.	References	60

LIST OF FIGURES

S.NO	FIGURE DESCRIPTION	PAGE NO
3.2.1	Proposed System Illustration	13
4.1.1	Use Case Diagram	22
4.2.1	Class Diagram	26
4.3.1	Admin Sequence Diagram	27
4.3.2	User Sequence Diagram	28
4.3.1	Sequence Diagram	25
4.4.1	Activity Diagram	29
5.3.1	Download MySQL	34
5.3.2	Install MySQL Products	35
5.3.3	Full Version and Disk Space for Installing	35
5.3.4	Set Password & Conform Password	36
5.4.1	Install SQLyog and Select Language	36
5.4.2	Accept Terms	36
5.4.3	Select Installation Location	37
5.4.4	Create New MYSQL Connection	37
5.4.5	Create New Database	38
9.1	User Interface	53
9.2	User Registration	53

9.3	User Registration Success	54
9.4	User Login Page	54
9.5	Welcome Page	55
9.6	Entering the Details of Receiver	55
9.7	Upload the File	56
9.8	File Shared Successfully	56
9.9	Receiver Login	56
9.10	File Received	57
9.11	Data Encrypted	57
9.12	Data Decrypted	57

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

A consortium of organizations collaborates and exchanges information to create synergies in their operations. However, centralized systems of file transferring fall short in providing the necessary distributed trust and transparency. To address this, blockchain technology emerges as a solution for secure and transparent file transferring .

This paper proposes a blockchain-based secure file transferring system that can be effectively utilized by a consortium of organizations. Here are the key components:

Hyperledger Fabric: An enterprise blockchain framework is employed for setting up the blockchain network and developing smart contracts. Hyperledger Fabric ensures robustness, scalability, and permissioned access.

Inter Planetary File System (IPFS): This decentralized protocol is used for storing files in a distributed manner. IPFS allows files to be shared across the network without relying on a central server.

Workflow for Identity Management and File Transferring:

Identity management: The system establishes a secure process for managing identities within the consortium. Participants are authenticated and authorized to access specific files.

File sharing: Consortium members can securely exchange files while maintaining confidentiality, integrity, and availability. Blockchain ensures tamper-proof records of file transactions.

In summary, this proposed system leverages blockchain and IPFS to enable secure, transparent, and efficient file sharing among consortium members. It enhances trust, reduces reliance on intermediaries, and ensures data integrity throughout the process.

1.2 Research Motivation

The growing necessity for secure and transparent file-sharing systems amidst the increasing reliance on collaboration and information exchange among organizations. Traditional centralized file-sharing systems often fall short in providing the required levels of trust and transparency, particularly when dealing with sensitive information. This inadequacy poses significant risks to the confidentiality and integrity of shared data.

Blockchain technology presents a promising solution to these challenges by offering a decentralized and immutable ledger that ensures transparency, security, and tamper-resistance. By leveraging blockchain, organizations can establish a distributed network for file-sharing, eliminating the need for a central authority and reducing the risk of single points of failure or manipulation.

The proposed system, which integrates Hyperledger Fabric as the blockchain framework and the Inter Planetary File System for decentralized storage, offers several advantages. Firstly, it enables secure and transparent file-sharing among consortiums of organizations, ensuring that sensitive information remains protected. Secondly, through the implementation of smart contracts on Hyperledger Fabric, the system facilitates efficient management of identities and permissions, thereby controlling access to shared files. Lastly, the use of multi-signatures for usage control adds an extra layer of security, enhancing the overall protection of digital content.

Researching this topic not only addresses a pressing need in the realm of file-sharing but also contributes to advancing the understanding and application of blockchain technology in ensuring data security and integrity in collaborative environments. Furthermore, exploring the feasibility and efficacy of implementing such a system can pave the way for practical solutions that meet the evolving demands of secure information exchange among organizations.

1.3 Problem Statement

With the increasing need for collaboration and information exchange among organizations, there is a demand for secure and transparent file transferring systems. Existing centralized systems of file transferring cannot provide the necessary levels of trust and transparency, especially when it comes to sharing sensitive information (Huang et al., 2020). However, the use of blockchain technology can address these concerns by enabling secure and transparent file transferring in a distributed fashion. By leveraging Hyperledger Fabric as the underlying blockchain framework and the Inter Planetary File System for decentralized storage, this proposed system provides a reliable and efficient solution for secure file transferring among consortiums of organizations. The proposed system allows organizations within a consortium to securely exchange files by leveraging blockchain technology. Through the implementation of smart contracts on Hyperledger Fabric, identities and permissions can be managed efficiently and securely. This ensures that only authorized users can access and share files, providing an added layer of security. Furthermore, the use of multi-signatures for usage control enhances the overall protection of digital content.

1.4 Applications

Blockchain technology offers a promising solution for securely transferring data due to its decentralized and immutable nature. Here are some key applications of blockchain for secure data transfer:

- 1. Secure File Sharing:** Blockchain can be used to create a decentralized file-sharing platform where users can securely exchange files without relying on a central authority. Each transaction is recorded on the blockchain, ensuring transparency and immutability while protecting the confidentiality of shared data through encryption and access control mechanisms.
- 2. Supply Chain Management:** In supply chain management, blockchain can facilitate secure data transfer among multiple stakeholders, including manufacturers, suppliers, distributors, and retailers. By recording transactions and data exchanges on a distributed ledger, blockchain ensures transparency, traceability, and integrity throughout the supply chain, reducing the risk of fraud, counterfeiting, and errors.
- 3. Healthcare Data Exchange:** Blockchain technology can enable secure sharing of electronic health records (EHRs) and other sensitive healthcare data among patients, healthcare providers, and other authorized parties. By storing encrypted patient data on a blockchain network, healthcare organizations can ensure patient privacy, data integrity, and secure access control, while facilitating interoperability and data exchange across different healthcare systems.
- 4. Identity Verification:** Blockchain-based identity verification systems offer a secure and efficient way to verify and authenticate individuals' identities without relying on centralized authorities. By storing digital identities and credentials on a blockchain network, users can control their personal data and

selectively share it with trusted parties, reducing the risk of identity theft and fraud.

- 5. Financial Transactions:** Blockchain technology is widely used in the financial sector for secure and transparent transactions, including cross-border payments, remittances, and asset transfers. By leveraging cryptographic algorithms and distributed consensus mechanisms, blockchain ensures the security, immutability, and integrity of financial transactions, reducing the risk of fraud, manipulation, and unauthorized access.
- 6. Legal and Intellectual Property Management:** Blockchain can be utilized to securely transfer and manage legal documents, contracts, and intellectual property rights. By timestamping and recording transactions on a blockchain ledger, organizations can establish a verifiable and tamper-proof record of ownership, rights, and agreements, reducing disputes and ensuring the integrity of legal and intellectual property assets. By simulating realistic attack scenarios, security teams can identify weaknesses in their systems and infrastructure, allowing them to patch vulnerabilities and strengthen their overall security posture.
- 7. Digital Asset Management:** Blockchain facilitates secure transfer and management of digital assets, including cryptocurrencies, tokens, and non-fungible tokens (NFTs), ensuring ownership, provenance, and liquidity in digital asset markets.

CHAPTER 2

LITERATURE SURVEY

[1] S. Nakamoto, “**bitcoin: A peer-to-peer electronic cash system,**” 2008.

A purely peer-to-peer version of electronic cash would allow online payments to be sent directly from one party to another without going through a financial institution. Digital signatures provide part of the solution, but the main benefits are lost if a trusted third party is still required to prevent double-spending. We propose a solution to the double-spending problem using a peer-to-peer network. The network timestamps transactions by hashing them into an ongoing chain of hash-based proof-of-work, forming a record that cannot be changed without redoing the proof-of-work. The longest chain not only serves as proof of the sequence of events witnessed, but proof that it came from the largest pool of CPU power. As long as a majority of CPU power is controlled by nodes that are not cooperating to attack the network, they'll generate the longest chain and outpace attackers. The network itself requires minimal structure. Messages are broadcast on a best effort basis, and nodes can leave and rejoin the network at will, accepting the longest proof-of-work chain as proof of what happened while they were gone. We have proposed a system for electronic transactions without relying on trust. We started with the usual framework of coins made from digital signatures, which provides strong control of ownership, but is incomplete without a way to prevent double-spending. To solve this, we proposed a peer-to-peer network using proof-of-work to record a public history of transactions that quickly becomes computationally impractical for an attacker to change if honest nodes control a majority of CPU power. The network is robust in its unstructured simplicity. Nodes work all at once with little coordination. They do not need to be identified, since messages are not routed to any particular place and only need to be delivered on a best effort basis. Nodes can leave and rejoin the network at will, accepting the proof-of-work chain as proof of what happened while they were gone. They vote with their CPU power, expressing their acceptance of valid blocks by working on extending them and rejecting invalid blocks by refusing to work on them. Any needed rules and incentives can be enforced with this consensus mechanism.

[2] Naz M, Al-zahrani FA, Khalid R, Javaid N, Qamar AM, Afzal MK, Shafiq M. A secure data sharing platform using blockchain and interplanetary file system. Sustainability. 2019 Dec 10;11(24):7054.

In a research community, data sharing is an essential step to gain maximum knowledge from the prior work. Existing data sharing platforms depend on trusted third party (TTP). Due to the involvement of TTP, such systems lack trust, transparency, security, and immutability. To overcome these issues, this paper proposed a blockchain-based secure data sharing platform by leveraging the benefits of interplanetary file system (IPFS). A meta data is uploaded to IPFS server by owner and then divided into n secret shares. The proposed scheme achieves security and access control by executing the access roles written in smart contract by owner. Users are first authenticated through RSA signatures and then submit the requested amount as a price of digital content. After the successful delivery of data, the user is encouraged to register the reviews about data. These reviews are validated through Watson analyzer to filter out the fake reviews. The customers registering valid reviews are given incentives. In this way, maximum reviews are submitted against every file. In this scenario, decentralized storage, Ethereum blockchain, encryption, and incentive mechanism are combined. To implement the proposed scenario, smart contracts are written in solidity and deployed on local Ethereum test network. The proposed scheme achieves transparency, security, access control, authenticity of owner, and quality of data. In simulation results, an analysis is performed on gas consumption and actual cost required in terms of USD, so that a good price estimate can be done while deploying the implemented scenario in real set-up. Moreover, computational time for different encryption schemes are plotted to represent the performance of implemented scheme, which is shamir secret sharing (SSS). Results show that SSS shows the least computational time as compared to advanced encryption standard (AES) 128 and 256.

[3] Liu J, Li X, Ye L, Zhang H, Du X, Guizani M. BPDS: A blockchain based privacy-preserving data sharing for electronic medical records. In 2018 IEEE Global Communications Conference (GLOBECOM) 2018 Dec 9 (pp. 1-6). IEEE.

Electronic medical record (EMR) is a crucial form of healthcare data, currently drawing a lot of attention. Sharing health data is considered to be a critical approach to improve the quality of healthcare service and reduce medical costs. However, EMRs are fragmented across decentralized hospitals, which hinders data sharing and puts patients' privacy at risks. To address these issues, we propose a blockchain based privacy-preserving data sharing for EMRs, called BPDS. In BPDS, the original EMRs are stored securely in the cloud and the indexes are reserved in a tamper-proof consortium blockchain. By this means, the risk of the medical data leakage could be greatly reduced, and at the same time, the indexes in blockchain ensure that the EMRs can not be modified arbitrarily. Secure data sharing can be accomplished automatically according to the predefined access permissions of patients through the smart contracts of blockchain. Besides, the joint-design of the CP-ABE-based access control mechanism and the content extraction signature scheme provides strong privacy preservation in data sharing. Security analysis shows that BPDS is a secure and effective way to realize data sharing for EMRs.

[4] Satapathy U, Mohanta BK, Panda SS, Sobhanayak S, Jena D. A secure framework for communication in internet of things application using hyperledger based blockchain. In 2019 10th international conference on computing, communication and networking technologies (ICCCNT) 2019 Jul 6 (pp. 1-7). IEEE.

In the era of the Internet of Things(IoT) smart devices are connected with wire or wireless way. The IoT devices are capable of sensing the environment and has the ability to transmit that information to the next level. The application area of IoT is Smart city, Smart transportation,

Healthcare sector, Agriculture, Monitoring environment. Each of these applications, lots of information are share or transmit among different devices. In the information sharing system

among devices, lots of security and privacy challenges exist like data leakage, data modification, device identity. In this paper, authors firstly identify the communication protocols used in IoT application and given their working principle. Secondly, challenges exist in IoT and corresponding Blockchain solution approach are explained, Lastly, the authors proposed a secure architecture based on open Blockchain which can

solve some of the challenges in IoT applications. It is anticipated that Blockchain will make far-reaching changes in IoT applications in near future. We have addressed the vulnerabilities occur in secure communication in an IoT application by integrating it with Blockchain. In this paper, we proposed an architecture to communicate securely in IoT Applications using Hyperledger based Blockchain. In IoT application the data sent by various devices stored in a centralized database making it vulnerable for security breaches. Also the authenticity of the sender could not be verified properly making it open for security threats. In this paper we have proposed a secure architecture based on open Blockchain (Hyperledger) for IoT applications. Malicious actor detection will be easy as every node aware of all other node in the Hyperledger network. Concluding it guarantees the security measures for non-repudiation, privacy and scalability in an IoT application.

CHAPTER 3

SYSTEM ANALYSIS AND DESIGN

3.1 EXISTING SYSTEM

In order to preserve the privacy for traceable encryption in blockchain, previous works proposed a system in which authenticity and non-repudiation of digital content is guaranteed. The problem tackled by authors is the secret key of user, which when shared with other entities does not hold the specific information of user. In case the shared key is corrupted or abused, it makes difficult to analyse the source of secret key. Moreover, leakage of confidential information in access control is a bottleneck for existing systems. Therefore, previous works have integrated the privacy protection algorithm such as attribute based encryption (ABE) to secure the secret keys.

In the realm of blockchain-based secure data transfer, maintaining privacy while ensuring traceability and authenticity of digital content is of paramount importance. Previous research efforts have focused on addressing the challenge of preserving privacy while guaranteeing the authenticity and non-repudiation of digital content. One key issue tackled by researchers is the safeguarding of users' secret keys, which are essential for encryption and decryption processes. However, when these keys are shared with other entities for collaborative purposes, ensuring that they do not reveal specific user information becomes critical. In scenarios where shared keys are compromised or misused, it becomes challenging to trace the source of the leaked key, potentially undermining the security of the entire system.

Additionally, leakage of confidential information in access control mechanisms poses a significant bottleneck for existing systems. Unauthorized access to sensitive data can lead to breaches of privacy and confidentiality, posing serious risks to individuals and organizations alike. To address these challenges, previous studies have integrated privacy protection algorithms, such as attribute-based encryption (ABE), into blockchain-based systems. ABE allows for fine-grained access control by encrypting data with attributes rather than specific keys, ensuring that only users with the necessary attributes can decrypt and access the data. By incorporating ABE and similar privacy-preserving techniques, researchers aim to enhance the security and privacy of

blockchain-based data transfer systems while enabling efficient and secure collaboration among multiple parties.

Limitations

Despite the promising applications of blockchain technology for secure data transfer with privacy protection, there are several limitations that researchers and practitioners should be aware of:

- 1. Scalability:** One of the major limitations of blockchain technology is scalability. As the size of the blockchain network grows and the number of transactions increases, the system may encounter performance bottlenecks, such as slow transaction processing times and high network congestion. This can hinder the efficiency of data transfer operations, especially in high-volume environments.
- 2. Resource Intensity:** Blockchain networks require significant computational resources and energy consumption to validate and record transactions on the blockchain. This resource intensity can limit the scalability and accessibility of blockchain-based data transfer systems, particularly for users with limited computational capabilities or access to reliable internet infrastructure.
- 3. Storage and Bandwidth Requirements:** Storing large volumes of data on the blockchain can pose challenges in terms of storage capacity and bandwidth requirements. As more data is added to the blockchain, the storage and bandwidth requirements for network participants increase, potentially leading to higher operational costs and reduced network performance.
- 4. Privacy Trade-offs:** While blockchain offers transparent and immutable transaction records, achieving privacy protection can be challenging. Despite the integration of privacy-preserving techniques like attribute-based encryption (ABE), there may still be limitations in ensuring complete anonymity and confidentiality, especially in public blockchain networks where transaction data is visible to all participants.

- 5. Regulatory and Legal Considerations:** Blockchain-based data transfer systems may face regulatory and legal challenges, particularly concerning data privacy, security, and compliance with existing regulations, such as the General Data Protection Regulation (GDPR). Ensuring compliance with regulatory requirements and navigating legal complexities can be complex and time-consuming.
- 6. User Adoption and Interface Complexity:** The complexity of blockchain technology and associated tools may present barriers to user adoption, particularly for non-technical users. Improving user interfaces and enhancing usability are essential for promoting widespread adoption of blockchain-based data transfer systems.
- 7. Interoperability:** Achieving interoperability between different blockchain networks and legacy systems remains a challenge. Lack of standardization and compatibility between blockchain platforms can hinder seamless data transfer and integration with existing IT infrastructure.
- 8. Security Risks:** While blockchain is often touted for its security features, it is not immune to security risks and vulnerabilities. Threats such as 51% attacks, smart contract vulnerabilities, and cryptographic weaknesses can undermine the security of blockchain-based data transfer systems, necessitating robust security measures and continuous monitoring.

Addressing these limitations requires ongoing research and development efforts to improve the scalability, efficiency, privacy, security, and usability of blockchain technology for secure data transfer. Collaborative initiatives between researchers, industry stakeholders, policymakers, and regulatory bodies are essential for addressing these challenges and unlocking the full potential of blockchain in safeguarding data privacy and security

3.2 PROPOSED SYSTEM

3.2.1 OVERVIEW

Figure 3.2.1 Describes the proposed system illustration.

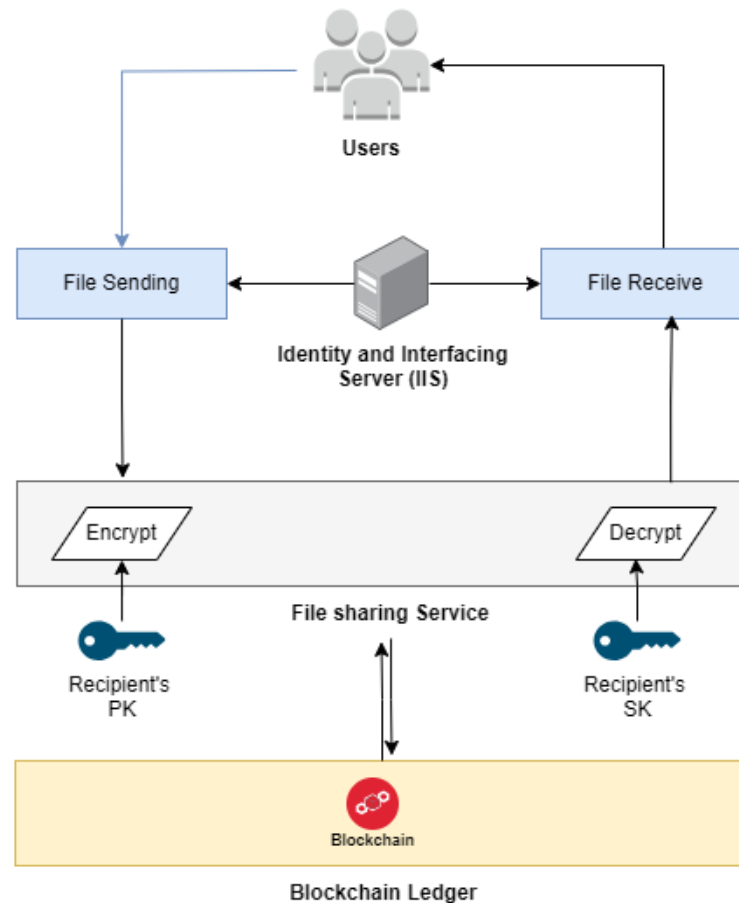


Fig 3.2.1 Proposed System Illustration

The In a consortium of organizations, a number of organizations can share data in the form files and synergies their operations. A blockchain network created among multiple organizations, each of the organizations will host Identity and Interfacing Server (IIS), Smart contract, and blockchain ledger. IIS maintains the identity details in identity database and is also the interfacing point with the smart contract. A smart contract is a program, which contains the business logic of the proposed file transferring mechanism, is installed on each of the organizations. The blockchain ledger maintains transactions in the form of blocks. The following Figure illustrates the high-level view of the proposed system.

3.2.2 PROPOSED SYSTEM MODULES

Identity and Interfacing Server

For sharing files among the users of the participant or organizations, the concerned users need to be registered with the blockchain through the smart contract. A key pair (pk_i , ski) is generated by the end user application. The private key ski is kept with the user machine and the public key along with user details like name, email, organization, password, etc is sent to the IIS. IIS admin verifies the user identity registration request. Upon successful verification the hash of the password is written to the identity database.

Blockchain Ledger

IIS sends the user registration request to the smart contract with public key pk_i and user details. The smart contract generates the blockchain identifier $BCID_i$ and inserts $BCID_i$, userdetails, pk_i into blockchain ledger. The blockchain update status(success/failure) and $BCID_i$ is sent to IIS. IIS sends the user identity registration status and $BCID_i$ to the end user application.

File Sharing

Once the user is registered with blockchain, he can securely share files with any other registered user. The user logs in using the user authentication process. The user selects the file to be shared and specifies the file receivers. The end user application generates the symmetric key, K and encrypts file, M to be shared using K . The encrypted file M is uploaded to the IPFS distributed storage. IPFS returns content ID of the uploaded file, M_{cid} to the end user application. The end user application generates file identifier, M_{fid} for the unique identification of the file. The end user application requests the public keys of (R_1 , R_2 , ... R_n) i.e., the receivers with whom the file is to be shared.

Feasibility study

A feasibility analysis evaluates the project's potential for success; therefore, perceived objectivity is an essential factor in the credibility of the study for potential investors and lending institutions.

Technical Feasibility

Technical resources need for project Development

- Windows family Operating System
- Python 3.6 Technology

- Vs Code
- Mysql
- SQLyog

Economic Fesibility

Cost/ benefits analysis of the project as over project is academic project we will not have only basic cost for learning of the technologies

Operational Feasibility

This assessment involves undertaking a study to analyze and determine whether and how well the organization's needs can be met by completing the project. Operational feasibility studies also examine how a project plan satisfies the requirements identified in the requirements analysis phase of system development.

Functional Requirements

- Admin Login
- User Login
- User Signup
- View Block Chain

NON- Functional Requirements

- Serviceability requirement
- Manageability requirement
- Recoverability requirement
- Security requirement

3.2.3 Applications

1. **Supply Chain Management:** Blockchain enables secure and transparent transfer of data across the supply chain, ensuring authenticity, traceability, and accountability of products from manufacturers to consumers.

- 2. Digital Identity Management:** Blockchain-based identity management systems offer secure and decentralized solutions for verifying and managing digital identities, reducing the risk of identity theft and fraud in online transactions.
- 3. Cross-Border Payments:** Blockchain facilitates secure and cost-effective cross-border payments by eliminating intermediaries and providing transparent and immutable transaction records.
- 4. Healthcare Data Exchange:** Blockchain enables secure sharing and exchange of electronic health records (EHRs) and other sensitive healthcare data among patients, healthcare providers, and insurers, ensuring privacy, integrity, and interoperability.
- 5. Legal and Intellectual Property Management:** Blockchain facilitates secure transfer and management of legal documents, contracts, and intellectual property rights, ensuring authenticity, ownership, and protection against infringement.
- 6. Decentralized Finance (DeFi):** Blockchain powers decentralized finance applications such as lending, borrowing, and trading, enabling secure and permissionless financial transactions without relying on traditional financial intermediaries.
- 7. Real Estate Transactions:** Blockchain streamlines and secures real estate transactions by providing transparent and tamper-proof records of property ownership, transfers, and contracts.
- 8. Voting Systems:** Blockchain can be used to develop secure and transparent voting systems, ensuring the integrity and confidentiality of votes while preventing tampering and fraud.
- 9. Digital Asset Management:** Blockchain facilitates secure transfer and management of digital assets, including cryptocurrencies, tokens, and non-fungible tokens (NFTs), ensuring ownership, provenance, and liquidity in digital asset markets.

10. Document and File Sharing: Blockchain-based platforms allow secure and transparent sharing of documents and files, ensuring authenticity, integrity, and confidentiality of shared data among users.

3.2.4 Advantages

- 1. Security:** Blockchain offers enhanced security through cryptographic algorithms and decentralized consensus mechanisms, ensuring that data transferred over the network is protected from unauthorized access, tampering, and fraud.
- 2. Transparency:** Transactions recorded on the blockchain are transparent and immutable, providing a tamper-proof audit trail of data transfers. This transparency increases trust among participants and reduces the risk of disputes or discrepancies.
- 3. Decentralization:** Blockchain operates on a decentralized network of nodes, eliminating the need for a central authority or intermediary to facilitate data transfers. This decentralization enhances resilience, reliability, and censorship resistance of the network.
- 4. Efficiency:** Blockchain streamlines and automates data transfer processes, reducing the need for manual intervention and administrative overhead. Smart contracts, in particular, enable self-executing agreements, automating tasks and improving operational efficiency.
- 5. Cost-Effectiveness:** By eliminating intermediaries and reducing administrative costs associated with traditional data transfer methods, blockchain offers cost-effective solutions for secure data transfer, especially for cross-border transactions and supply chain management.
- 6. Immutable Recordkeeping:** Once data is recorded on the blockchain, it cannot be altered or deleted, ensuring the integrity and authenticity of transferred data.

This immutable recordkeeping enhances trust and accountability among participants.

- 7. Resilience to Downtime:** Blockchain networks are highly resilient to downtime and disruptions due to their decentralized architecture. Even if some nodes fail or go offline, the network remains operational, ensuring continuous data transfer and availability.
- 8. Enhanced Privacy:** Blockchain offers privacy-enhancing features such as encryption and zero-knowledge proofs, enabling secure data transfer while protecting sensitive information from unauthorized disclosure.
- 9. Innovation and Scalability:** Blockchain technology continues to evolve, driving innovation in secure data transfer solutions and enabling scalability to support growing volumes of data and users. This scalability ensures that blockchain-based systems can adapt to changing business needs and technological advancements.

CHAPTER 4

UML DIAGRAMS

4.1 USE CASE DIAGRAM

The System Design Document describes the system requirements, operating environment, system and subsystem architecture, files and database design, input formats, output layouts, human-machine interfaces, detailed design, processing logic, and external interfaces.

Global Use Case Diagrams:

Identification of actors:

Actor: Actor represents the role a user plays with respect to the system. An actor interacts with, but has no control over the use cases.

Graphical representation:



An actor is someone or something that:

Interacts with or uses the system.

- Provides input to and receives information from the system.
- Is external to the system and has no control over the use cases.

Actors are discovered by examining:

- Who directly uses the system?
- Who is responsible for maintaining the system?
- External hardware used by the system.
- Other systems that need to interact with the system.

Questions to identify actors:

- Who is using the system? Or, who is affected by the system? Or, which groups need help from the system to perform a task?
- Who affects the system? Or, which user groups are needed by the system to perform its functions? These functions can be both

main functions and secondary functions such as administration.

- Which external hardware or systems (if any) use the system to perform tasks?
- What problems does this application solve (that is, for whom)?
- And, finally, how do users use the system (use case)? What are they doing with the system?

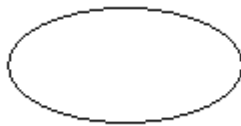
The actors identified in this system are:

- a. System Administrator**
- b. Customer**
- c. Customer Care**

Identification of use cases:

Use case: A use case can be described as a specific way of using the system from a user's (actor's) perspective.

Graphical representation:



A more detailed description might characterize a use case as:

- Pattern of behaviour the system exhibits
- A sequence of related transactions performed by an actor and the system
- Delivering something of value to the actor

Use cases provide a means to:

- capture system requirements
- communicate with the end users and domain experts
- test the system

Use cases are best discovered by examining the actors and defining what the actor will be able to do with the system.

Guide lines for identifying use cases:

- For each actor, find the tasks and functions that the actor should be able to perform or that the system needs the actor to perform. The use case should represent a course of events that leads to clear goal
- Name the use cases.

- Describe the use cases briefly by applying terms with which the user is familiar.

This makes the description less ambiguous

Questions to identify use cases:

- What are the tasks of each actor?
- Will any actor create, store, change, remove or read information in the system?
- What use case will store, change, remove or read this information?
- Will any actor need to inform the system about sudden external changes?
- Does any actor need to inform about certain occurrences in the system?
- What use cases will support and maintains the system?

Flow of Events

A flow of events is a sequence of transactions (or events) performed by the system. They typically contain very detailed information, written in terms of what the system should do, not how the system accomplishes the task. Flow of events are created as separate files or documents in your favorite text editor and then attached or linked to a use case using the Files tab of a model element.

A flow of events should include:

- When and how the use case starts and ends
- Use case/actor interactions
- Data needed by the use case
- Normal sequence of events for the use case
- Alternate or exceptional flows

Construction of Use case diagrams:

Use-case diagrams graphically depict system behaviour (use cases). These diagrams present a high level view of how the system is used as viewed from an outsider's (actor's) perspective. A use-case diagram may depict all or some of the use cases of a system.

A use-case diagram can contain:

- actors ("things" outside the system)
- use cases (system boundaries identifying what the system should do)
- Interactions or relationships between actors and use cases in the system including the associations, dependencies, and generalizations.

Relationships in use cases:

1. Communication:

The communication relationship of an actor in a use case is shown by connecting the actor symbol to the use case symbol with a solid path. The actor is said to communicate with the use case.

2. Uses:

A Uses relationship between the use cases is shown by generalization arrow from the use case.

3. Extends:

The extend relationship is used when we have one use case that is similar to another use case but does a bit more. In essence it is like subclass.



Fig 4.1.1 Use Case Diagram

4.2 CLASS DIAGRAM

Identification of analysis classes:

A class is a set of objects that share a common structure and common behavior (the same attributes, operations, relationships and semantics). A class is an abstraction of real-world items.

There are 4 approaches for identifying classes:

- a. Noun phrase approach:
- b. Common class pattern approach.
- c. Use case Driven Sequence or Collaboration approach.
- d. Classes , Responsibilities and collaborators Approach

1. Noun Phrase Approach:

The guidelines for identifying the classes:

- Look for nouns and noun phrases in the usecases.
- Some classes are implicit or taken from general knowledge.
- All classes must make sense in the application domain; Avoid computer implementation classes – defer them to the design stage.
- Carefully choose and define the class names After identifying the classes we have to eliminate the following types of classes:
- Adjective classes.

2. Common class pattern approach:

The following are the patterns for finding the candidate classes:

- Concept class.
- Events class.
- Organization class
- Peoples class
- Places class
- Tangible things and devices class.

3. Use case driven approach:

We have to draw the sequence diagram or collaboration diagram. If there is need for some classes to represent some functionality then add new classes which perform those functionalities.

4. CRC approach:

The process consists of the following steps:

- Identify classes' responsibilities (and identify the classes)
- Assign the responsibilities
- Identify the collaborators.

Identification of responsibilities of each class:

The questions that should be answered to identify the attributes and methods of a class respectively are:

- a. What information about an object should we keep track of?
- b. What services must a class provide?

Identification of relationships among the classes:

Three types of relationships among the objects are:

Association: How objects are associated?

Super-sub structure: How are objects organized into super classes and sub classes?

Aggregation: What is the composition of the complex classes?

Association:

The **questions** that will help us to identify the associations are:

- a. Is the class capable of fulfilling the required task by itself?
- b. If not, what does it need?
- c. From what other classes can it acquire what it needs?

Guidelines for identifying the tentative associations:

- A dependency between two or more classes may be an association. Association often corresponds to a verb or prepositional phrase.
- A reference from one class to another is an association. Some associations are implicit or taken from general knowledge.

Some common association patterns are:

Location association like part of, next to, contained in.....

Communication association like talk to, order to

We have to eliminate the unnecessary association like implementation associations, ternary or n-ary associations and derived associations.

Super-sub class relationships:

Super-sub class hierarchy is a relationship between classes where one class is the parent class of another class (derived class). This is based on inheritance.

Guidelines for identifying the super-sub relationship, a generalization are

1. **Top-down:**

Look for noun phrases composed of various adjectives in a class name. Avoid excessive refinement. Specialize only when the sub classes have significant behavior.

2.Bottom-up:

Look for classes with similar attributes or methods. Group them by moving the common attributes and methods to an abstract class. You may have to alter the definitions a bit.

3.Reusability:

Move the attributes and methods as high as possible in the hierarchy.

4. Multiple inheritances:

Avoid excessive use of multiple inheritances. One way of getting benefits of multiple inheritances is to inherit from the most appropriate class and add an object of another class as an attribute.

Aggregation or a-part-of relationship:

It represents the situation where a class consists of several component classes. A class that is composed of other classes doesn't behave like its parts. It behaves very differently. The major properties of this relationship are transitivity and anti symmetry. The **questions** whose answers will determine the distinction between the part and whole relationships are:

- Does the part class belong to the problem domain?
- Is the part class within the system's responsibilities?
- Does the part class capture more than a single value?(If not then simply include it as an attribute of the whole class)
- Does it provide a useful abstraction in dealing with the problem domain?

There are three types of aggregation relationships. They are:

Assembly:

It is constructed from its parts and an assembly-partsituation physically exists.

Container:

A physical whole encompasses but is not constructed from physical parts.

Collection member:

A conceptual whole encompasses parts that may be physical or conceptual. The container and collection are represented by hollow diamonds but composition is represented by solid diamond.

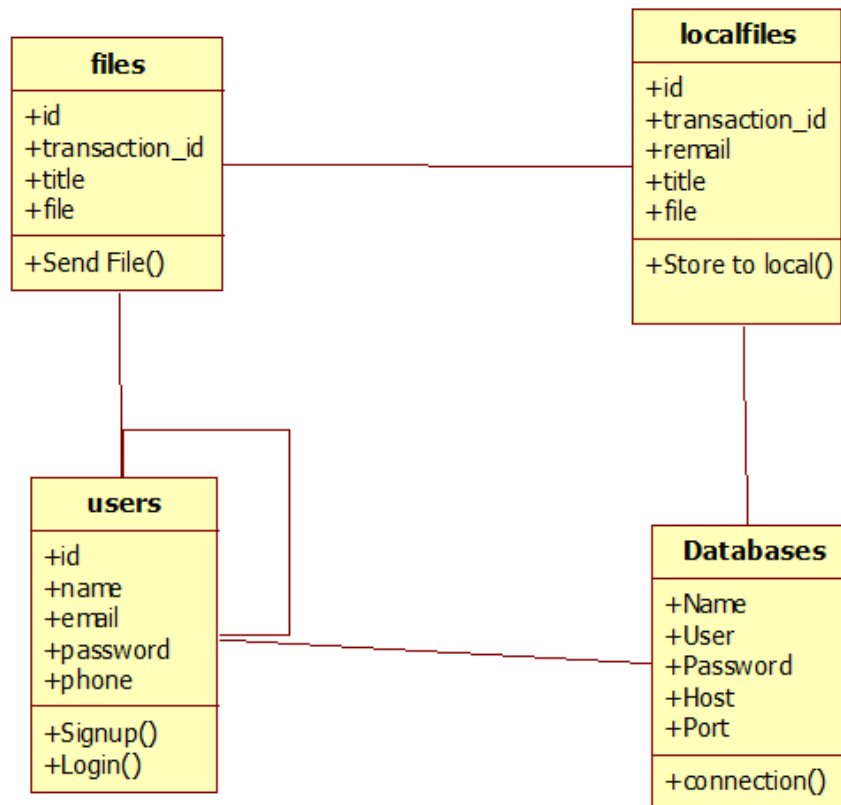


Fig 4.2.1 Class Diagram

4.3 SEQUENCE DIAGRAM

A sequence diagram is a graphical view of a scenario that shows object interaction in a time-based sequence what happens first, what happens next. Sequence diagrams establish the roles of objects and help provide essential information to determine class responsibilities and interfaces.

There are two main differences between sequence and collaboration diagrams: sequence diagrams show time-based object interaction while collaboration diagrams show how objects associate with each other. A sequence diagram has two dimensions: typically, vertical placement represents time and horizontal placement represents different objects.

Object:

An object has state, behavior, and identity. The structure and behavior of similar objects are defined in their common class. Each object in a diagram indicates some instance of a class. An object that is not named is referred to as a class instance.

The object icon is similar to a class icon except that the name is underlined:

An object's concurrency is defined by the concurrency of its class.

Message:

A message is the communication carried between two objects that trigger an event. A message carries information from the source focus of control to the destination focus of control. The synchronization of a message can be modified through the message specification. Synchronization means a message where the sending object pauses to wait for results.

Link:

A link should exist between two objects, including class utilities, only if there is a relationship between their corresponding classes. The existence of a relationship between two classes symbolizes a path of communication between instances of the classes: one object may send messages to another. The link is depicted as a straight line between objects or objects and class instances in a collaboration diagram. If an object links to itself, use the loop version of the icon.

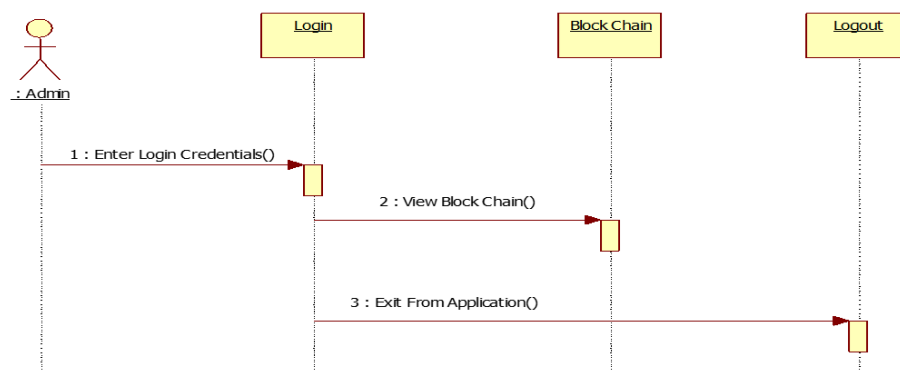


Fig 4.3.1 Admin Sequence Diagram

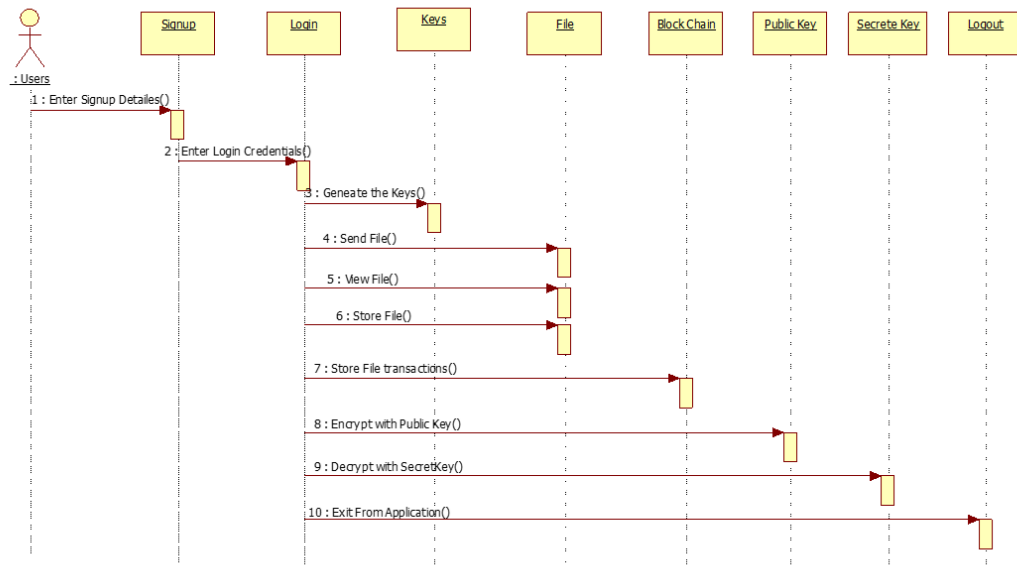


Fig 4.3.2 User Sequence Diagram

4.4 ACTIVITY DIAGRAM

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions. An activity diagram shows the flow of control from activity to activity. An activity is an ongoing execution within a state machine. It is essentially a flowchart modelling the dynamic aspects of the system.

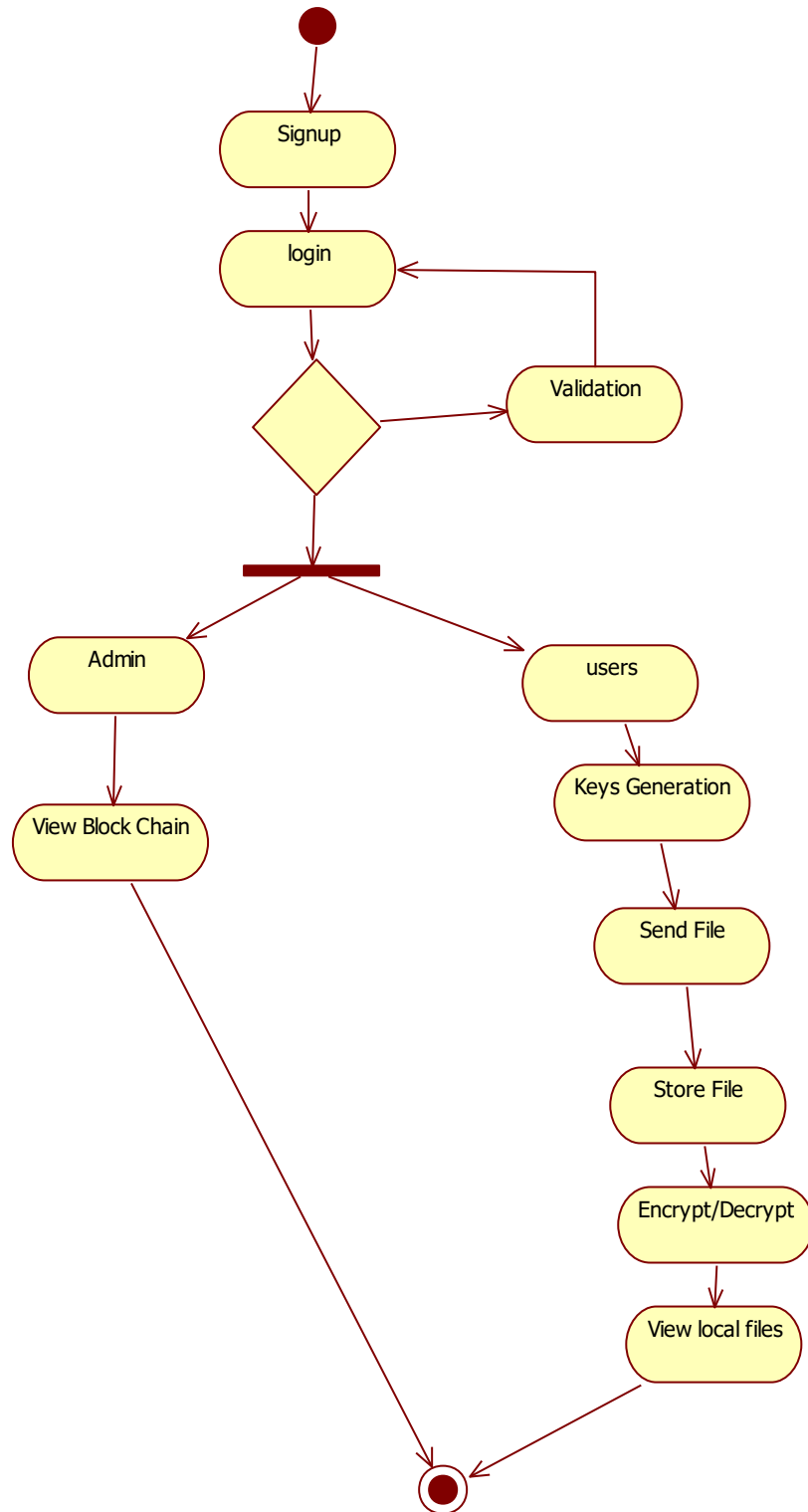


Fig 4.4.1 Activity Diagram

CHAPTER 5

SOFTWARE ENVIRONMENT

5.1 SOFTWARE OVERVIEW

History of Python

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands. Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

Input as CSV File

Reading data from CSV (comma separated values) is a fundamental necessity in Data Science. Often, we get data from various sources which can get exported to CSV format so that they can be used by other systems. The Panadas library provides features using which we can read the CSV file in full as well as in parts for only a selected group of columns and rows.

The CSV file is a text file in which the values in the columns are separated by a comma. Let's consider the following data present in the file named input.csv. You can create this file using windows notepad by copying and pasting this data. Save the file as input.csv using the save As All files (*.*) option in notepad.

```
import pandas as pd
data= pd.read_csv('path/input.csv')
print(data)
```

Operations using NumPy

NumPy is a Python package which stands for 'Numerical Python'. It is a library consisting of multidimensional array objects and a collection of routines for processing of array.

Using NumPy, a developer can perform the following operations –

- Mathematical and logical operations on arrays.

- Fourier transforms and routines for shape manipulation.
- Operations -related to linear algebra. NumPy has in-built functions for linear algebra and random number generation.

Key Features of Pandas

- Fast and efficient DataFrame object with default and customized indexing.
- Tools for loading data into in-memory data objects from different file formats.
- Data alignment and integrated handling of missing data.
- Reshaping and pivoting of date sets.
- Label-based slicing, indexing and subsetting of large data sets.
- Columns from a data structure can be deleted or inserted.
- Group by data for aggregation and transformations.
- High performance merging and joining of data.
- Time Series functionality.

5.2 DJANGO WEB FRAMEWORK FOR PYTHON

Introduction to Django

If you go to the Web site djangoproject.com using your Web browser or, depending on the decade in which you're reading this destined-to-be-timeless literary work, using your cell phone, electronic notebook, shoe, or any Internet-superceding contraption you'll find this explanation: "Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design." That's a mouthful or eyeful or pixelful, depending on whether this book is being recited, read on paper or projected to you on a Jumbotron, respectively. Let's break it down. Django is a high-level Python Web framework... A high-level Web framework is software that eases the pain of building dynamic Web sites. It abstracts common problems of Web development and provides shortcuts for frequent programming tasks. For clarity, a dynamic Web site is one in which pages aren't simply HTML documents sitting on a server's filesystem somewhere. In a dynamic Web site, rather, each page is generated by a computer program a so-called "Web application" that you, the Web developer, create. A Web application may, for instance, retrieve records from a database or take

some action based on user input. A good Web framework addresses these common concerns:

- It provides a method of mapping requested URLs to code that handles requests. In other words, it gives you a way of designating which code should execute for which URL. For instance, you could tell the framework, “For URLs that look like `/users/joe/`, execute code that displays the profile for the user with that username.”
- It makes it easy to display, validate and redisplay HTML forms. HTML forms are the primary way of getting input data from Web users, so a Web framework had better make it easy to display them and handle the tedious code of form display and redisplay (with errors highlighted).
- It converts user-submitted input into data structures that can be manipulated conveniently. For example, the framework could convert HTML form submissions into native data types of the programming language you’re using.
- It helps separate content from presentation via a template system, so you can change your site’s look-and-feel without affecting your content, and vice-versa.
- It conveniently integrates with storage layers such as databases but doesn’t strictly require the use of a database.
- It lets you work more productively, at a higher level of abstraction, than if you were coding against, say, HTTP. But it doesn’t restrict you from going “down” one level of abstraction when needed.
- It gets out of your way, neglecting to leave dirty stains on your application such as URLs that contain `“.aspx”` or `“.php”`.
- It is never too late to start **learning** and it would be a shame to miss an opportunity to learn a tutorial or course that can be so useful as **Django Web framework for Python** especially when it is free! You do not have to register for expensive classes and travel from one part of town to another to take classes. All you need to do is download the course and open the PDF file. This specific program is classified in the Web programming category where you can find some other similar courses.
- Thanks to people (like you?) Who share their knowledge, you can discover the extent of our being selected to easily learn without spending a fortune! Django Web framework for Python. is available for free by its author. But also many other tutorials are accessible just as easily!

- Computer PDF guide you and allow you to save on your studies.
- **Django Web framework for Python.** help on the contact form if problems
- Computer PDF is also courses for training in **html & html5, css & css3, javascript, php, asp, j2ee, ajax, jquery, node.js, angularjs** and many others IT.
- You should come see our Web programming documents. You will find your happiness without trouble !
- The latest news and especially the best tutorials on your favorite topics, that is why Computer PDF is number 1 for courses and tutorials for download in pdf files - Django Web framework for Python. and Web programming!
- Download other tutorials for advice on Django Web framework for Python. you will see! We will do everything to help you!
- And you dear surfers what you need? The best **course and tutorial, and how to learn and use Django Web framework for Python.** of course!

5.3 About MySQL:

MySQL is a relational database management system (RDBMS)¹ that runs as a server providing multi-user access to a number of databases. The SQL phrase stands for Structured Query Language. Free-software-open source projects that require a full-featured database management system often use MySQL. For commercial use, several paid editions are available, and offer additional functionality. Applications which use MySQL databases include: TYPO3, Joomla, WordPress, phpBB, Drupal and other software built on the LAMP software stack. MySQL is also used in many high-profile, large-scale World Wide Web products, including Wikipedia, Google , Facebook, and Twitter.

MySQL is the world's most popular open source database software, with over 100 million copies of its software downloaded or distributed throughout it's history. With its superior speed, reliability, and ease of use, MySQL has become the preferred choice for Web, Web 2.0, SaaS, ISV, Telecom companies and forward-thinking corporate IT Managers because it eliminates the major problems associated with downtime, maintenance and administration for modern, online applications.

Many of the world's largest and fastest-growing organizations use MySQL to save time and money powering their high-volume Web sites, critical business systems, and packaged software including industry leaders such as Yahoo!, Alcatel-Lucent, Google, Nokia, YouTube, Wikipedia, and Booking.com.

The flagship MySQL offering is MySQL Enterprise, a comprehensive set of production-tested software, proactive monitoring tools, and premium support services available in an affordable annual subscription.

MySQL is a key part of LAMP (Linux, Apache, MySQL, PHP / Perl / Python), the fast-growing open source enterprise software stack. More and more companies are using LAMP as an alternative to expensive proprietary software stacks because of its lower cost and freedom from platform lock-in.

MySQL was originally founded and developed in Sweden by two Swedes and a Finn: David Axmark, Allan Larsson and Michael "Monty" Widenius, who had worked together since the 1980's. More historical information on MySQL is :

Download MySQL Installer from Office website

<http://dev.mysql.com/downloads/installer/>

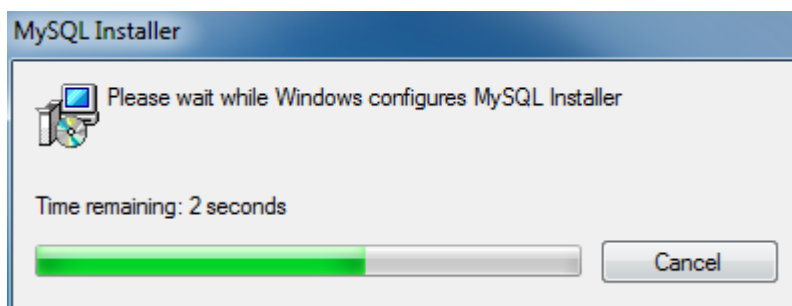


Fig 5.3.1 Download MySQL

Install MySQL Step 1: Windows configures MySQL Installer

Select Install MySQL Products

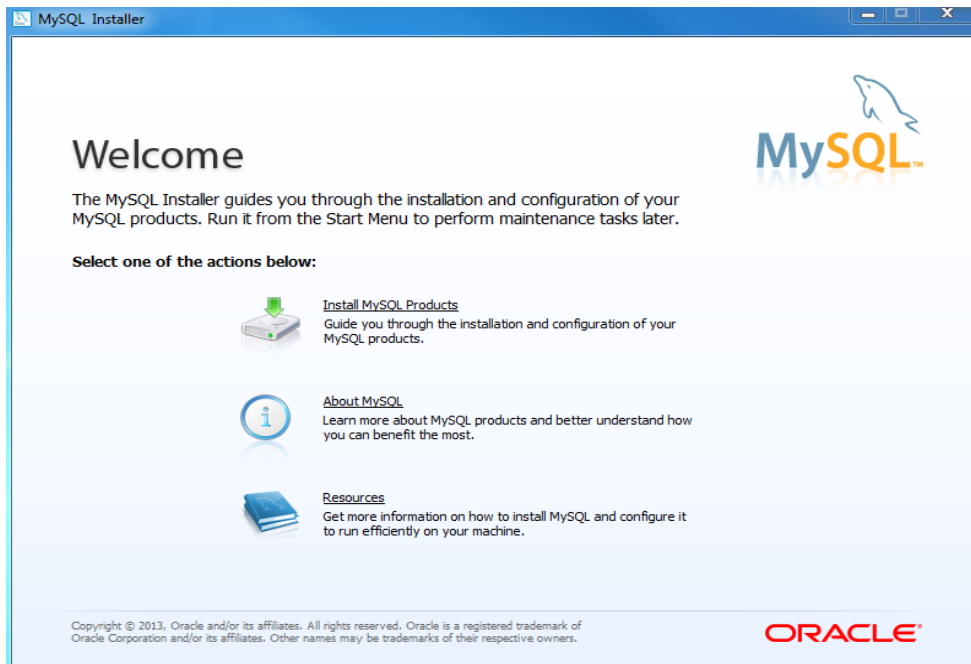


Fig 5.3.2 Install MySQL Products

Select Full version and disk space for installing

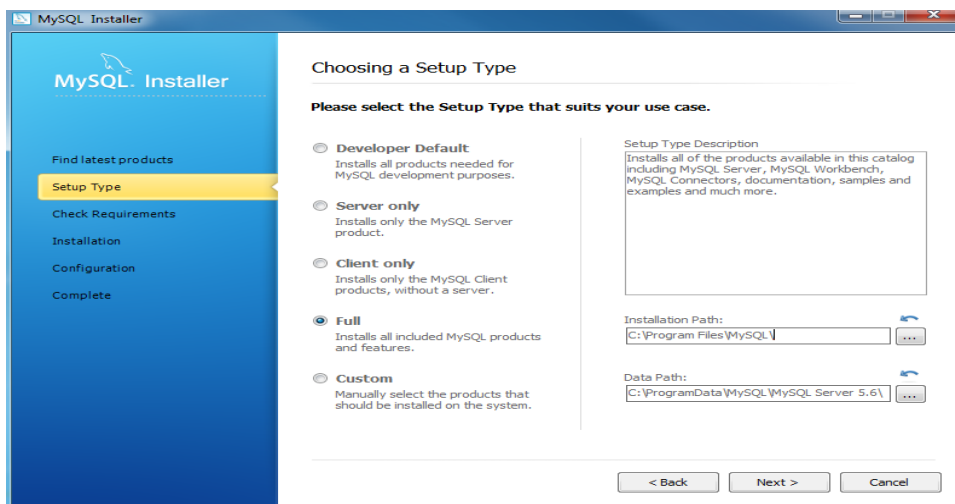


Fig 5.3.3 Full Version and Disk Space for Installing

Set Password & Conform password

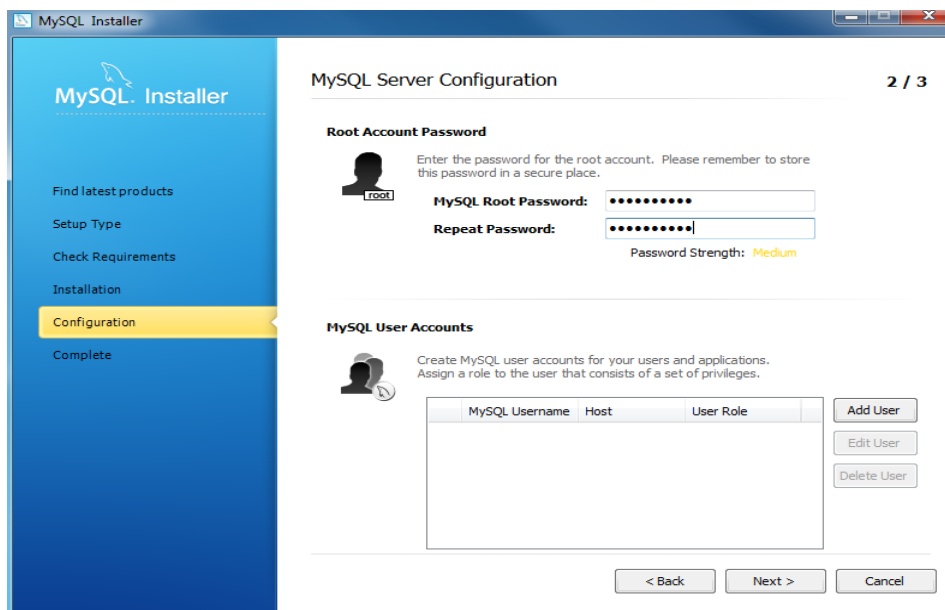


Fig 5.3.4 Set Password & Conform Password

5.4 Install SQLyog community edition

Select the Language

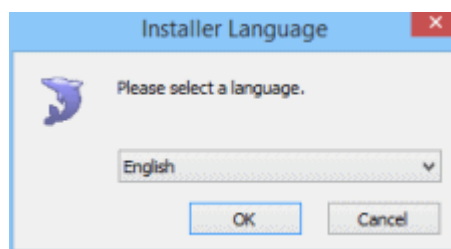


Fig 5.4.1 Install SQLyog and Select Language

Accept terms of the software

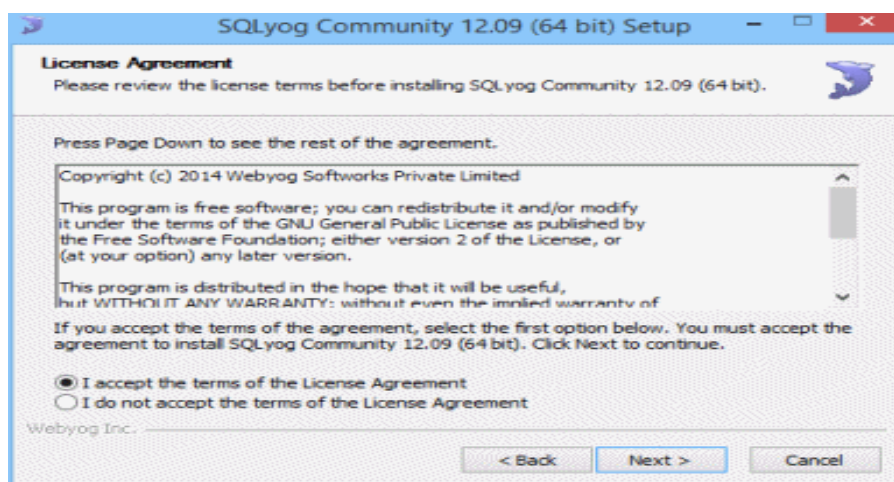


Fig 5.4.2 Accept Terms

Selecting the installing location in our system

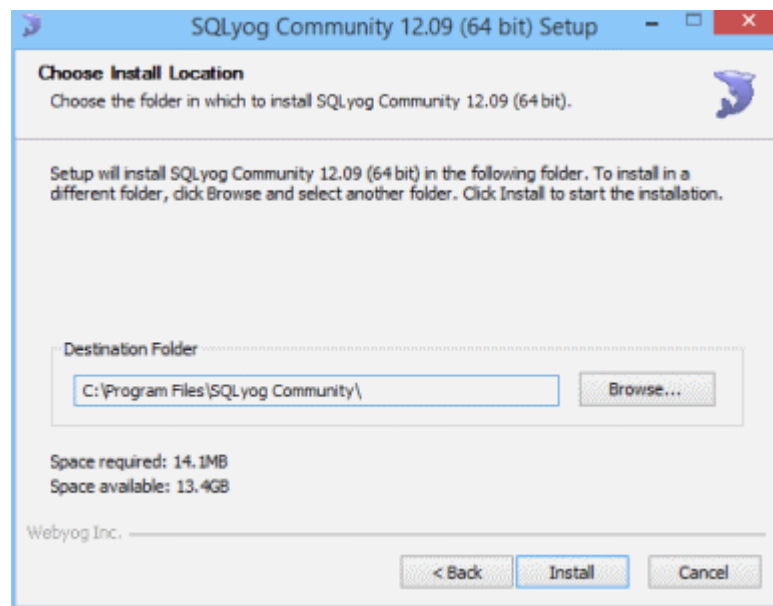


Fig 5.4.3 Select Installation Location

Create a New MYSQL Connection

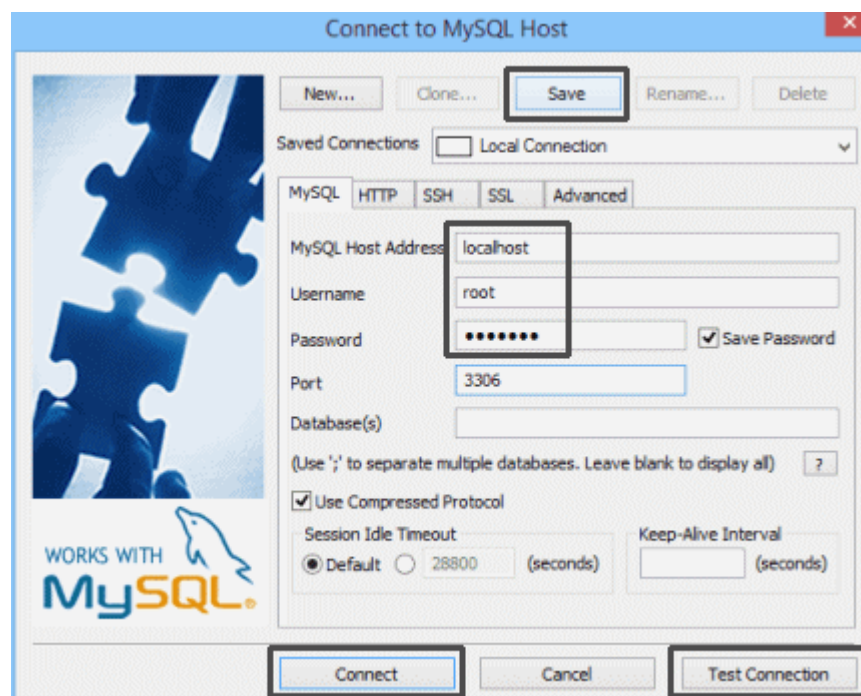


Fig 5.4.4 Create New MYSQL Connection

Create any New Database

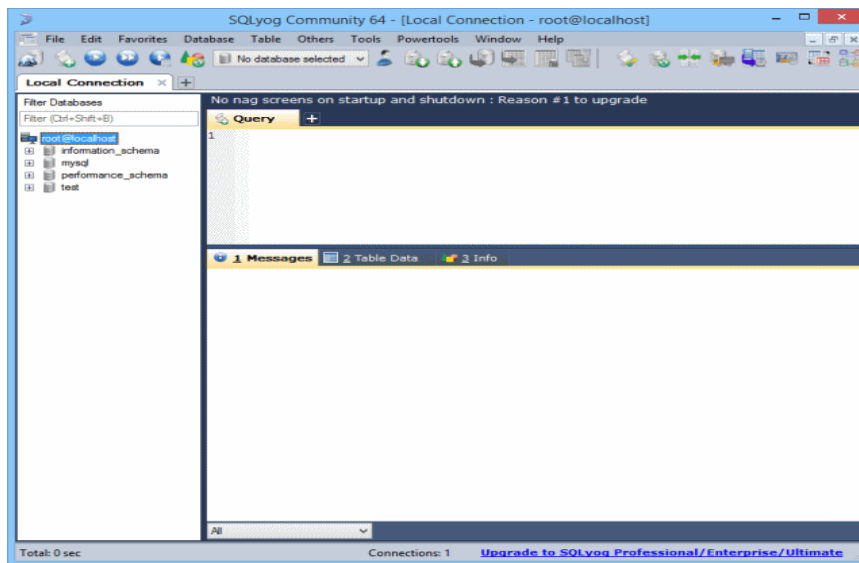


Fig 5.4.5 Create New Database

Project Database Tables

```
USE `filesharing`;  
  
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0 */;  
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;  
  
CREATE TABLE `webapp_files` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `transaction_id` varchar(149) NOT NULL,  
  `title` varchar(449) NOT NULL,  
  `file` longblob NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=7 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
  
/*Table structure for table `webapp_localfiles` */  
  
DROP TABLE IF EXISTS `webapp_localfiles`;  
  
CREATE TABLE `webapp_localfiles` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `transaction_id` varchar(149) NOT NULL,  
  `remai` varchar(149) NOT NULL,  
  `title` varchar(449) NOT NULL,  
  `file` longtext NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;  
  
/*Table structure for table `webapp_users` */  
  
DROP TABLE IF EXISTS `webapp_users`;  
  
CREATE TABLE `webapp_users` (  
  `id` int NOT NULL AUTO_INCREMENT,  
  `name` varchar(149) NOT NULL,  
  `email` varchar(149) NOT NULL,  
  `password` varchar(149) NOT NULL,  
  `phone` varchar(149) NOT NULL,  
  PRIMARY KEY (`id`)  
) ENGINE=InnoDB AUTO_INCREMENT=8 DEFAULT CHARSET=utf8mb4 COLLATE=utf8mb4_0900_ai_ci;
```

CHAPTER 6

SYSTEM REQUIREMENT AND SPECIFICATION

6.1 HARDWARE REQUIREMENTS:

Hardware requirements specify the physical components needed for a computer system to run software effectively. This includes items such as the processor (CPU), memory (RAM), storage (hard drive or solid-state drive), graphics processing unit (GPU), and input/output devices (keyboard, mouse, monitor, etc.). Hardware requirements may vary depending on the specific software or application being used, and they are typically outlined by the software developer to ensure optimal performance.

RAM	4 GB Minimum
Processor	i3 Minimum
Hard disk	500 GB HDD Min

6.2 SOFTWARE REQUIREMENTS:

Software requirements outline the necessary software components or environments required to run a particular application or system. This includes the operating system (such as Windows, macOS, Linux, etc.), as well as any additional software dependencies or libraries required by the application. Software requirements may also specify minimum versions or configurations needed to ensure compatibility and functionality. These requirements are essential for users to install and run the software successfully on their computer systems.

Technology	Python 3.6
Operating System	Windows Family
IDE	VS Code
Technology	Python, Django
Database Server	MySQL
Front Design Technology	HTML, CSS, JS

CHAPTER 7

FUNCTIONAL REQUIREMENTS

Functional requirements are essential specifications that define what a system should do, outlining its capabilities and behaviours.

7.1 Output Design

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

7.2 Input Design

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

OBJECTIVES

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

CHAPTER 8

SOURCE CODE

8.1 Main Code :

```
from web3 import Web3
from solcx import compile_standard
from solcx import install_solc

install_solc("0.8.0")
import json

# Connect to Ganache
ganache_url = "http://127.0.0.1:7545" # Update with your Ganache URL
web3 = Web3(Web3.HTTPProvider(ganache_url))

# Account information
account_1 = "0x4a919ea205b266d611a47Ef82Cef913DBB880001" # Update with
your Ganache account address
private_key =
"b151d16948eda51c21b7f81084558176ac02949e776db1d83a23c135719b033f"
# Update with your Ganache account private key

# Compile the solidity contract
def compile_contract():
    with open("./contracts/DataStorage.sol", "r") as file:
        data_storage_file = file.read()

    compiled_sol = compile_standard(
        {
            "language": "Solidity",
            "sources": {"DataStorage.sol": {"content":
data_storage_file}},
            "settings": {
                "outputSelection": {
                    "*": {"*": ["abi", "metadata", "evm.bytecode",
"evm.sourceMap"]}}
            },
        },
        solc_version="0.8.0",
    )
    return compiled_sol
```

```

# Deploy the contract
def deploy_contract(compiled_sol):
    bytecode =
compiled_sol["contracts"]["DataStorage.sol"]["DataStorage"]["evm"]["byt
ecode"]["object"]
    abi =
compiled_sol["contracts"]["DataStorage.sol"]["DataStorage"]["abi"]

    # Get nonce
    nonce = web3.eth.getTransactionCount(account_1)

    # Build transaction
    transaction = {
        "from": account_1,
        "gas": 2000000,
        "gasPrice": web3.toWei("50", "gwei"),
        "nonce": nonce,
        "data": bytecode,
    }

    # Sign transaction
    signed_transaction = web3.eth.account.sign_transaction(transaction,
private_key)

    # Send transaction
    tx_hash =
web3.eth.send_raw_transaction(signed_transaction.rawTransaction)

    # Get transaction receipt
    tx_receipt = web3.eth.wait_for_transaction_receipt(tx_hash)

    return web3.eth.contract(address=tx_receipt.contractAddress,
abi=abi)

# Main function
def main():
    compiled_sol = compile_contract()
    contract = deploy_contract(compiled_sol)

    # Store data
    contract.functions.addData("sajid24x7@gmail.com",
"1").transact({"from": account_1})
    contract.functions.addData("sajid24x7@gmail.com",
"2").transact({"from": account_1})
    contract.functions.addData("sajid24x7@gmail.com",
"3").transact({"from": account_1})
    #contract.functions.addData("name1", "Jane").transact({"from":
account_1})

```

```

# Retrieve data
data = contract.functions.getData("sajid24x7@gmail.com").call()
print("Retrieved Data:", data)

if __name__ == "__main__":
    main()

```

```

import mimetypes
import os
import pickle

from django.shortcuts import render
from django.http import HttpResponse, request
from .models import *

import matplotlib.pyplot as plt;
import numpy as np
import numpy
from django.shortcuts import render, redirect
from PIL import ImageTk, Image
from PIL import Image

from .DateTime import getdate

account_1 = "0x4a919ea205b266d611a47Ef82Cef913DBB880001" # Update with
your Ganache account address
private_key =
"b151d16948eda51c21b7f81084558176ac02949e776db1d83a23c135719b033f"
# Update with your Ganache account private key
from .Blockchain import compile_contract, deploy_contract
compiled_sol = compile_contract()
contract = deploy_contract(compiled_sol)

def homepage(request):

    return render(request, 'index.html')

```

```

def home(request):
    return render(request, 'index.html')

def signuppge(request):
    if request.method == 'POST':
        email = request.POST['email']

        d = users.objects.filter(email__exact=email).count()
        if d > 0:
            return render(request, 'signup.html', {'msg': "email
Already Registered"})

        else:

            password = request.POST['password']
            phone = request.POST['phone']
            name = request.POST['name']

            from .HashCode import convert
            hashid=convert(email)

            from .GenerateKeys import generate
            generate(hashid)

            d = users(name=name, email=email, password=password,
phone=phone)
            d.save()

            return render(request, 'signup.html', {'msg': "Register
Success, You can Login.."})

        else:

            return render(request, 'signup.html')

def userloginaction(request):
    if request.method=='POST':
        uid=request.POST['uid']
        pass_word=request.POST['pwd']
        d=users.objects.filter(email__exact=uid).filter(password__exact
=pass_word).count()

        if d>0:
            d=users.objects.filter(email__exact=uid)
            request.session['email']=uid

```

```

        request.session['name']=d[0].name
        return render(request, 'user_home.html',{'data': d[0]})
    else:
        return render(request, 'user.html',{'msg':"Login Fail"})
else:
    return render(request, 'user.html')

def adminloginaction(request):
    if request.method == 'POST':
        uid = request.POST['uid']
        pwd = request.POST['pwd']

        if uid == 'admin' and pwd == 'admin':
            request.session['adminid'] = 'admin'

            return render(request, 'admin_home.html')

        else:
            return render(request, 'admin.html', {'msg': "Login Fail"})

    else:
        return render(request, 'admin.html')

def adminhomedef(request):
    if "adminid" in request.session:
        uid = request.session["adminid"]
        return render(request, 'admin_home.html')

    else:
        return render(request, 'admin.html')

def adminlogoutdef(request):
    try:
        del request.session['adminid']
    except:
        pass
    return render(request, 'admin.html')

def userlogoutdef(request):
    email= request.session["email"]
    del request.session['email']
    return render(request, 'user.html')

```

```

def userhomedef(request):
    if "email" in request.session:
        email=request.session["email"]
        d=users.objects.filter(email__exact=email)

        return render(request, 'user_home.html',{'data': d[0]})

    else:
        return redirect('n_userlogout')

def viewprofilepage(request):
    if "email" in request.session:
        uid=request.session["email"]
        d=users.objects.filter(email__exact=uid)
        return render(request, 'viewpprofile.html',{'data': d[0]})

    else:
        return render(request, 'user.html')

from .HashCode import convert
from .GenerateKeys import getkeys

def fileupload(request):
    if request.method == 'POST':

        user = request.POST['user']

        remail,rname = user.split('|')

        rh=convert(remail)

        pk, sk=getkeys(rh)

        return render(request, 'uploadfile2.html', {'pk':pk,
'remail':remail, 'rname':rname})

    else:
        email = request.session["email"]
        data=users.objects.all().exclude(email=email)
        return render(request, 'uploadfile.html',{'data':data})

def fileupload2(request):
    if request.method == 'POST':

        from .DateTime import getdate

```



```

ts=getdate()

file = request.POST['file']
rname = request.POST['rname']
remail = request.POST['remail']
import random
tid=random.randrange(111111,999999)

file2 = 'Data/' + file

title = request.POST['title']
name = request.session["name"];
email = request.session["email"]

f = open(file2, "r")
dt = f.read()
rh=convert(remail)
pk, sk=getkeys(rh)
from .GenerateKeys import encrypt

edata=encrypt(pk, dt)

d=files(transaction_id=tid, sender_email=email,
recipient_email=remail, title=title, file=edata)
d.save()

#bc.add_block([{'transaction_id':tid,'sender_email':email,'sender_name':name,'recipient_email':remail, 'timestamp':ts }])

return render(request, 'user_home.html', {'msg': 'File has
shared !! '})

else:
    pass

def viewfiles(request):
    if "email" in request.session:
        email = request.session["email"]
        d = localfiles.objects.filter(remail=email)

        return render(request, 'viewfiles.html', {'data': d})

    else:

```

```

        return render(request, 'user.html')

def viewfile(request, op):
    if "email" in request.session:
        d = files.objects.filter(id=op)

        return render(request, 'viewfile.html', {'d': d[0]})

    else:
        return render(request, 'user.html')

def updateprofile(request):
    if request.method == 'POST':
        name = request.POST["name"]
        phone = request.POST['phone']
        email = request.session["email"]
        users.objects.filter(email=email).update(name=name,
phone=phone)
        return render(request, 'user_home.html',{'msg':'Profile Updated
!!'}) )

    else:
        email = request.session["email"]
        d = users.objects.filter(email=email)

        return render(request, 'updateprofile.html', {'data': d[0]})

def updatepwd(request):
    if request.method == 'POST':
        newpwd = request.POST["newpwd"]
        old = request.POST['old']
        email = request.session["email"]
        d=users.objects.filter(email=email).filter(password=old).count(
)

        if d>0:
            users.objects.filter(email=email).update(password=newpwd)
            return render(request, 'user_home.html',{'msg':'Password
Updated !!'}) )

    else:
        return render(request, 'updatepwd.html')

```



```

        rh=convert(remail)
        pk, sk=getkeys(rh)

        from .GenerateKeys import decrypt
        text=decrypt(sk, encdata)

        print(text, '*****')

        return render(request, 'viewifile2.html', {'text':text,
'd':d[0]})

    else:
        return render(request, 'user.html')

def filesave(request):
    if request.method == 'POST':

        file = request.POST['data']
        name = request.POST['name']
        email = request.POST['email']
        tid = request.POST['id']

        title = request.POST['title']
        remail = request.session["email"]

        d=localfiles(transaction_id=tid, title=title, file=file,
remail=remail)
        d.save()

        return render(request, 'user_home.html', {'msg': 'File has
stored !! '})

    else:
        pass

def lviewfile(request, op):
    if "email" in request.session:
        d = localfiles.objects.filter(id=op)

```

```
        return render(request, 'lviewfile.html', {'d': d[0]})

    else:
        return render(request, 'user.html')

def delete(request, op):
    if True:
        d = localfiles.objects.filter(id=op)
        d.delete()

        return redirect('viewfiles')
```

CHAPTER 9

RESULTS AND ANALYSIS

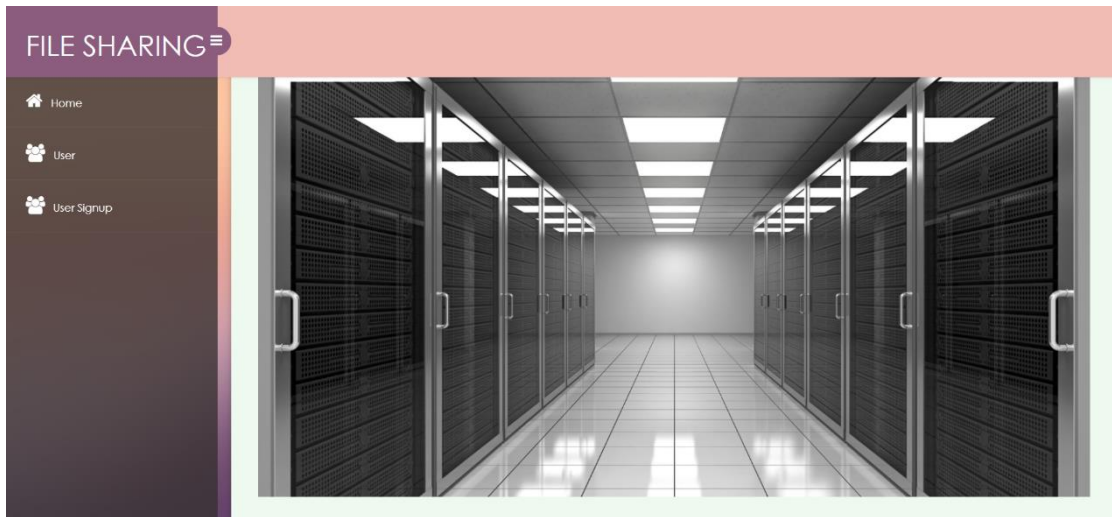
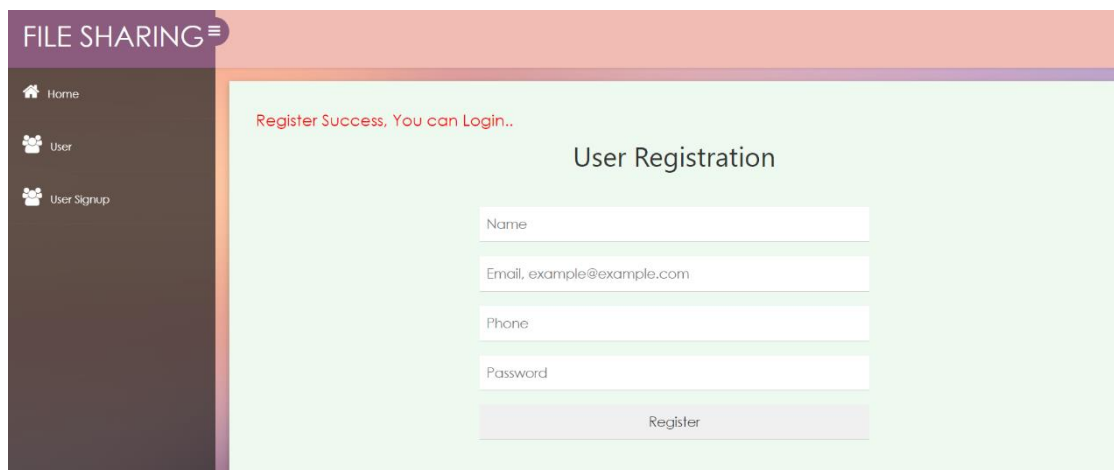


Fig 9.1 User Interface

A screenshot of the 'User Registration' page within the same application. The sidebar and header are identical to the previous image. The main content area is a light green box containing the title 'User Registration' and a form with four input fields: 'Name', 'Email, example@example.com', 'Phone', and 'Password'. Below these fields is a grey 'Register' button. At the bottom of the page, there is a footer with the text '© 2024-25. All rights reserved.' and a small blue box on the left containing the text '127.0.0.1:8000/u_loginaction'.

Fig 9.2 User Registration



The screenshot shows a web application interface for a file sharing system. On the left is a dark purple sidebar with the text "FILE SHARING" and a menu icon, followed by three items: "Home" with a house icon, "User" with a person icon, and "User Signup" with a person and plus icon. The main content area has a light green background. At the top, a red message says "Register Success, You can Login..". Below this, the heading "User Registration" is centered. There are four input fields: "Name", "Email, example@example.com", "Phone", and "Password". A grey "Register" button is at the bottom of the form.

FILE SHARING

Home

User

User Signup

Register Success, You can Login..

User Registration

Name

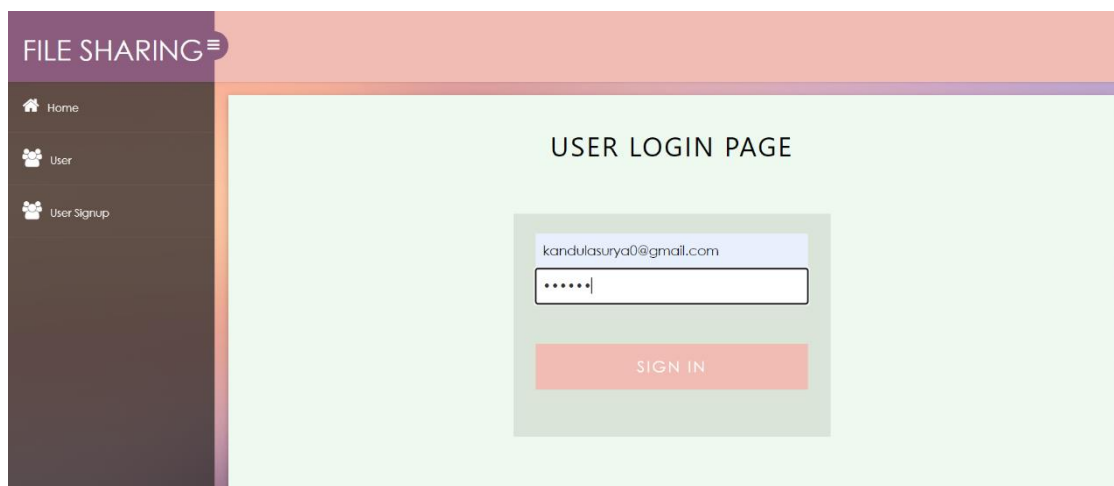
Email, example@example.com

Phone

Password

Register

Fig 9.3 User Registration Success



The screenshot shows the same web application interface as Fig 9.3. The sidebar is identical. The main content area has a light green background. The heading "USER LOGIN PAGE" is centered. Below it, there is a login form with two fields: an email field containing "kandulasurya0@gmail.com" and a password field with masked characters "*****". A red "SIGN IN" button is at the bottom of the form.

FILE SHARING

Home

User

User Signup

USER LOGIN PAGE

kandulasurya0@gmail.com

SIGN IN

Fig 9.4 User Login Page

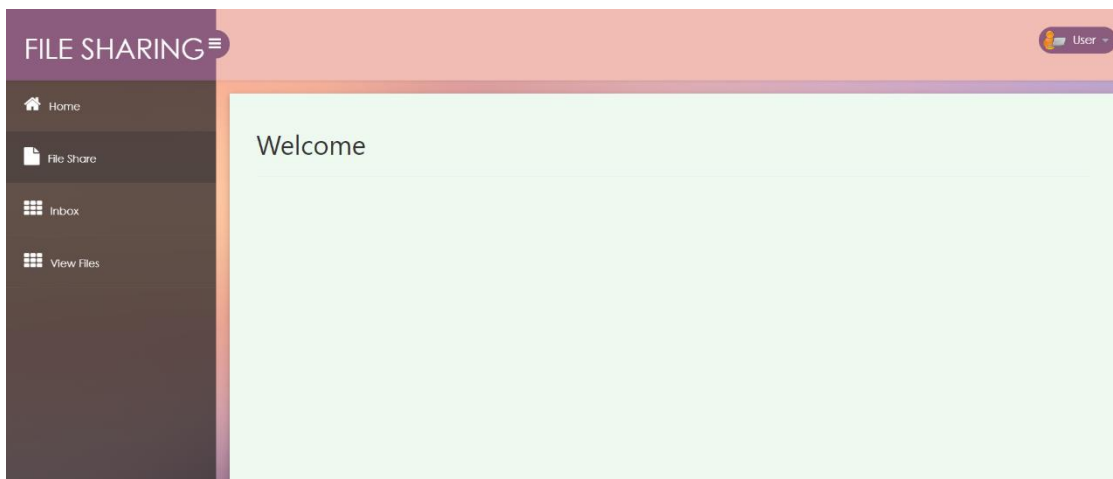


Fig 9.5 Welcome Page

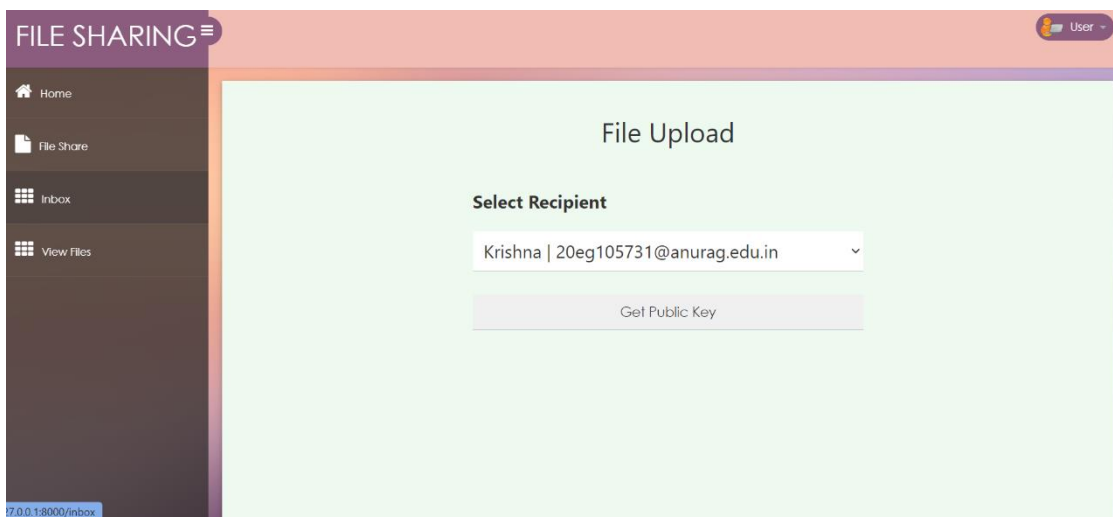


Fig 9.6 Entering the Details of Receiver

The screenshot shows a web application titled "FILE SHARING" in the top left corner. The top right corner features a "User" profile icon. A dark sidebar on the left contains navigation links: "Home", "File Share", "Inbox", and "View Files". The main content area, titled "File Upload", is light green and contains the following elements:

- A "Select File" section with a "Choose File" button and the text "manage.py".
- A text input field containing "IMP".
- A label "Public key of the recipient" above a text input field containing "Public RSA key at 0x28345D3CCC0".
- A large "Encrypt and Send" button at the bottom.

Fig 9.7 Upload the File

The screenshot shows the same "FILE SHARING" application. The sidebar and top navigation remain the same. The main content area is light green and displays a "Welcome" message. Below the welcome message, it says "File has shared !!".

Fig 9.8 File Shared Successfully

The screenshot shows a "USER LOGIN PAGE" with a light green background. In the center, there is a light gray box containing the following elements:

- A text input field with the email address "20eg105731@anurag.edu.in".
- A password input field represented by a series of dots ".....".
- A large red "SIGN IN" button at the bottom.

Fig 9.9 Receiver Login

View Files

Transaction Id	Sender	Sender Email	
502296	kandulasurya0@gmail.com	kandulasurya0@gmail.com	View File
502296	kandulasurya0@gmail.com	kandulasurya0@gmail.com	View File

Fig 9.10 File Received

View File	
Title	IMP FILE
File Data	<pre> {\xd3\xdb\xaby\xa3\xc7\xe1\1r\x93\x896\xbc\x00\x1d}\x97\xe8\xab\xe7\x0b_gg. (; \x9a\x94k\x04\xeb\xef:\xf8\xbeM-\xd98\xa4w\xc2\x3b\x8a1\x7fs\xfe\x8d\x94\x9f\x6\xca x\xab\x3c2\x90#,N\xbf\xe0*t\x9:\xcdoa5D\xe7^&:\xf7\xfaF \xb9Wx\x3e3W\x01&\xe8yWu\x92\ xf3\ \xe2\x7f8m\xc4/\x93\x76\x83\x91\1n\xbc\xf5\x5c\x5b3\xefc\x0c%LaF\x98\xf6\rj\x8e\x7c\x0 e\x18-\1xd4\xc3\x8d#\xdcf\x3c3\x1d1\xe8\x1d1\x1e\x97P>\xde\xbc^+K\xbeU\xe0\xbaz\xe6.\x9fA Y\x94\xec\xccox92\xfd\x99\x0c\x116\x88\x0c\x14-\18\x5d\xe9-\15d\xba\x7ff\xaa^1\xees(d\xfa\x1d\1\xbb\x3c1\x1d3m\xe4\x95\x04\x87? \xeb\xcc\x9e'\xbf\xab\xbf\xad\xaa\xcaZpTvp\x8d\x00+\xaf\x7f\xfa\xec\x6M18\x2d\x82\x cclv2v3v4v72v14v15v17v26v6v5v6v1v71v11E1B1v84v12Cv1v9v1v2v1v2v1v8Cv1d9v4b </pre>
Secret Key	Private RSA key at 0x1FA45861828
Decrypt	

Fig 9.11 Data Encrypted

View File

Title	IMP FILE
File Data	<pre>/* SQLyog Community Edition- MySQL GUI v6.07 Host - 5.5.30 : Database - /* SQLyog Community Edition- MySQL GUI v6.07 Host - 5.5.30 : Database - /* SQLyog Community Edition- MySQL GUI v6.07 Host - 5.5.30 : Database - /* SQLyog Community Edition- MySQL GUI v6.07 Host - 5.5.30 : Database -</pre>
	Save this file to your account? <input type="button" value="Yes"/>

Fig 9.12 Data Decrypted

CHAPTER 10

CONCLUSION AND FUTURE ENHANCEMENT

10.1 CONCLUSION

In conclusion, the proposed system offers a comprehensive solution for secure file transfer within a consortium of organizations, leveraging blockchain technology to enhance security and transparency. It ensures the safe transfer of files among participating organizations, safeguarding them from unauthorized access or tampering throughout the transmission process. Blockchain integration plays a crucial role by securely storing the content ID of each shared file, making it resistant to tampering or manipulation, thereby ensuring the authenticity and integrity of the files. The system guarantees confidentiality, integrity, and availability of shared files, ensuring that only authorized parties can access the files, they remain unaltered and accurate, and are accessible whenever needed. Files are encrypted end-to-end, adding an extra layer of security to the file transfer process, protecting them from unauthorized access even if intercepted during transmission. The use of distributed storage further enhances data resilience and availability, with encrypted files stored using the InterPlanetary File System (IPFS) and file metadata stored on the blockchain ledger. Implemented with the Hyperledger Fabric framework, tailored for enterprise use cases, the system benefits from its open-source nature, providing tools and infrastructure for building secure and scalable blockchain applications. Additionally, testing using Hyperledger Caliper ensures the system's performance under various conditions.

10.2 FUTURE SCOPE

The future of blockchain for secure data transfer holds significant promise, with numerous advancements and applications anticipated:

Interoperability stands out as a crucial development, aiming to enhance seamless data and asset transfer across diverse blockchain networks. This advancement would bolster efficiency and decrease adoption barriers.

Scalability is another focal point for future blockchain evolution. With the maturation of blockchain technology, scalable solutions are imperative to manage the escalating volume of data transactions. Techniques like sharding, sidechains, and layer 2 solutions are poised to improve scalability, enabling networks to process more transactions without compromising security or decentralization.

Privacy enhancements are expected in future blockchain systems to safeguard sensitive data. Advanced privacy features such as zero-knowledge proofs, homomorphic encryption, and secure multiparty computation will likely be integrated to facilitate data sharing and verification while preserving privacy.

The convergence of blockchain with emerging technologies like IoT, AI, and edge computing holds immense potential for creating robust and versatile applications. Blockchain's secure data transfer capabilities can be leveraged to manage and transfer data generated by IoT devices, ensuring its integrity and authenticity.

Regulatory compliance will become increasingly vital as blockchain technology gains wider acceptance. Future solutions may incorporate features such as smart contracts and digital identities to ensure compliance with regulations and standards across various industries and jurisdictions.

Addressing concerns about the environmental impact of blockchain operations, sustainability will be a key focus for future developments. Efforts may include adopting more energy-efficient consensus mechanisms and implementing initiatives to offset carbon emissions associated with blockchain activities.

In summary, the future of blockchain for secure data transfer is poised for continued innovation and evolution. Advancements in interoperability, scalability, privacy, integration with other technologies, regulatory compliance, and sustainability are set to drive transformative applications across diverse industries.

CHAPTER 11

REFERENCES

- [1] S. Nakamoto, "bitcoin: A peer-to-peer electronic cash system," 2008.
- [2] "Hyperledger Fabric Documentation, Release main",
<https://readthedocs.org/projects/hlf/downloads/pdf/latest/>, accessed on Dec 2022
- [3] Naz M, Al-zahrani FA, Khalid R, Javaid N, Qamar AM, Afzal MK, Shafiq M. A secure data sharing platform using blockchain and interplanetary file system. Sustainability. 2019 Dec 10;11(24):7054.
- [4] Liu J, Li X, Ye L, Zhang H, Du X, Guizani M. BPDS: A blockchain based privacy-preserving data sharing for electronic medical records. In 2018 IEEE Global Communications Conference (GLOBECOM) 2018 Dec 9 (pp. 1-6). IEEE.
- [5] Satapathy U, Mohanta BK, Panda SS, Sobhanayak S, Jena D. A secure framework for communication in internet of things application using hyperledger based blockchain. In 2019 10th international conference on computing, communication and networking technologies (ICCCNT) 2019 Jul 6 (pp. 1-7). IEEE.
- [6] YSari L, Sipos M. FileTribe: blockchain-based secure file-sharing on IPFS. In European Wireless 2019; 25th European Wireless Conference 2019 May 2 (pp. 1-6). VDE.
- [7] Benet J. Ipfs-content addressed, versioned, p2p file system. arXiv preprint arXiv:1407.3561. 2014 Jul 14.
- [8] "Hyperledger Caliper Documentation, Getting Started",
<https://hyperledger.github.io/caliper/v0.5.0/getting-started/>, accessed on Dec 2022
- [9] "Apache Couch Database Documentation: Release 3.3.0",
<https://docs.couchdb.org/en/latest/pdf/>, accessed on Dec 2022
- [10] Buterin, Vitalik. "A next-generation smart contract and decentralized application platform." white paper 3, no. 37 (2014): 2-1.

