

A  
Major Project Report  
On  
**Protecting Data Privacy for Permissioned Blockchains  
Using Identity-Based Encryption**

*Submitted to CMR ENGINEERING COLLEGE(Autonomous)  
HYDERABAD*

*In Partial Fulfillment of the requirements for the Award of Degree of*

**BACHELOR OF TECHNOLOGY  
IN  
COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)**

Submitted By

**G. SAI KIRAN REDDY (218R1A6781)**

**K. SUSHMA (218R1A67A0)**

**K. SIDDU (218R1A6795)**

Under the Esteemed guidance of

**Dr. M. LAXMAIAH**

Head of the Department of CSE (Data Science)



**Department of Computer Science & Engineering  
(Data Science)**

**CMR ENGINEERING COLLEGE**

**UGC AUTONOMOUS**

(Approved by AICTE, NEW DELHI, Affiliated to JNTU,  
Hyderabad)

Kandlakoya, Medchal Road, R.R. Dist. Hyderabad-501 401.

**2024-2025**

# CMR ENGINEERING COLLEGE

## UGC AUTONOMOUS

*(Accredited by NBA, Approved by AICTE NEW DELHI, Affiliated to JNTU, Hyderabad)*

*Kandlakoya, Medchal Road, Hyderabad-501 401*

### Department of Computer Science & Engineering (Data Science)



## CERTIFICATE

This is to certify that the project entitled “**Protecting Data Privacy for Permissioned Blockchains**

**using Identity-Based Encryption**” is a bonafide work carried out by

**G. SAI KIRAN REDDY (218R1A6781)**

**K. SUSHMA (218R1A67A0)**

**K. SIDDU (218R1A6795)**

in partial fulfillment of the requirement for the award of the degree of **BACHELOR OF TECHNOLOGY** in **COMPUTER SCIENCE AND ENGINEERING (DATA SCIENCE)** from CMR Engineering College, affiliated to JNTU, Hyderabad, under our guidance and supervision.

The results presented in this Major project have been verified and are found to be satisfactory. The results embodied in this Major project have not been submitted to any other university for the award of any other degree or diploma.

**Internal Guide**

**Major Project  
Coordinator**

**Head of the  
Department**

**External  
Examiner**

**Dr.M.Laxmaiah**

**Mr. B. KumaraSwamy**

**Dr. M. Laxmaiah**

Professor & HOD  
CSE (Data Science),  
CMREC

Assistant Professor  
CSE (Data Science),  
CMREC

Professor & HOD  
CSE (Data Science),  
CMREC

## **DECLARATION**

This is to certify that the work reported in the present Major project entitled "**Protecting Data Privacy for Permissioned Blockchains using Identity-Based Encryption**" is a record of bonafide work done by us in the Department of Computer Science and Engineering (Data Science), CMR Engineering College, JNTU Hyderabad. The reports are based on the project work done entirely by us and not copied from any other source. We submit our project for further development by any interested students who share similar interests to improve the project in the future.

The results embodied in this Major project report have not been submitted to any other University or Institute for the award of any degree or diploma to the best of our knowledge and belief.

**G. SAI KIRAN REDDY (218R1A6781)**

**K. SUSHMA (218R1A67A0)**

**K. SIDDU (218R1A6795)**

## **ACKNOWLEDGMENT**

We are extremely grateful to **Dr. A. Srinivasula Reddy**, Principal and **Dr. M. Laxmaiah**, HOD, Department of CSE (Data Science), CMR Engineering College for their constant support.

We are extremely thankful to **Dr. M. Laxmaiah**, Professor, Internal Guide, Department of CSE(DS), for his/ her constant guidance, encouragement and moral support throughout the project.

We will be failing in duty if We do not acknowledge with grateful thanks to the authors of the references and other literatures referred in this Project.

We thank **Mr. B. Kumaraswamy**, Associate Professor, CSE(DS) Department, Major Project Coordinator for her constant support in carrying out the project activities and reviews.

We express my thanks to all staff members and friends for all the help and co-ordination extended in bringing out this project successfully in time.

Finally, We are very much thankful to our parents who guided me for every step.

**G. SAI KIRAN REDDY(218R1A6781)**

**K. SUSHMA (218R1A67A0)**

**K. SIDDU (218R1A6795)**

## **ABSTRACT**

Blockchain is an emerging decentralized architecture and distributed public ledger technology underlying Bitcoin, and has recently attracted intensive attention from governments, financial institutions and high-tech enterprises. It is believed that blockchain can improve efficiency, reduce costs and enhance data security, but it is still in the face of serious privacy issues which may hinder the wide application of blockchain. In this project, we present a practical scheme by adding the Identity-Based encryption system, which effectively improves the data privacy for non-transaction applications. Analyses show that our proposal has a high security level which can prevent both disguise and passive attacks, and is functional, effective and practical in many applications for non-transactional scenarios.

# Contents

TOPIC	PAGE NO
ABSTRACT	v
LIST OF FIGURES	vii
<b>1. INTRODUCTION</b>	
1.1. Overview	1
1.2. Research Motivation	2
1.3. Problem Statement	2
1.4. Application	4
<b>2. LITERATURE SURVEY</b>	6
<b>3. EXISTING SYSTEM</b>	
3.1. SVM	10
3.2. SVM working and Drawbacks	11
<b>4. PROPOSED SYSTEM</b>	
4.1. Overview and EDA	13
4.2. Data Preprocessing	13
4.3. Dataset Splitting	14
4.4. KNN and its Advantages	15
<b>5. DIAGRAMS</b>	
5.1. Class Diagram	17
5.2. Use Case Diagram	17
5.3. Sequence Diagram	18
5.4. Activity Diagram	19
<b>6. SOFTWARE ENVIRONMENT</b>	
6.1. What is python and its Advantages and Disadvantages	23
6.2. History of python	23
6.3. Modules used in project	24
6.4. Installation of python	26
<b>7. SYSTEM REQUIREMENTS SPECIFICATIONS</b>	
7.1. Software Requirements	38
7.2. Hardware Requirements	38
<b>8. FUNCTIONAL REQUIREMENTS</b>	
8.1. Output Design and Definition	39
8.2. Input Design , Stages, Types, Media	38

	8.3. User Interface	39
	8.4. Performance Requirements	39
<b>9.</b>	<b>SOURCE CODE</b>	<b>44</b>
<b>10.</b>	<b>RESULTS AND DISCUSSION</b>	
	10.1. Implementation description	47
	10.2. Dataset Description	60
<b>11.</b>	<b>CONCLUSION AND REFERENCES</b>	<b>61</b>

## LIST OF FIGURE

FIG.NO	FIG.NAME	PG.NO
5.1.1	System architecture	9
5.1.2	Flow diagram	9
5.1.3	Dataflow diagram	10
5.2.1	Usecase diagram	12
5.2.2	Class diagram	13
5.2.3	Activity diagram	13
5.2.4	Sequence diagram	14
5.2.5	Collaboration diagram	14
5.2.6	Component diagram	15
5.2.7	Deployment diagram	15





# 1.INTRODUCTION

## 1.1 overview

Blockchain is a distributed public ledger technology in peer-to-peer network characterized by decentralization and distrusting, and it has witnessed a growing interest from different domains and use cases. In general terms, a blockchain is an immutable transaction ledger, maintained within a distributed network of peer nodes. These nodes each maintain a copy of the ledger by applying transactions that have been validated by a consensus protocol, grouped into blocks that include a hash that bind each block to the preceding block. The heart of a blockchain network is a distributed ledger that records all the transactions that take place on the network. In addition, the information recorded to a blockchain is appendonly, using cryptographic techniques that guarantee that once a transaction has been added to the ledger, it cannot be modified. This property of immutability makes it simple to assure that data has not been changed after the fact. Bitcoin cryptocurrency is the first and most widely recognized application of blockchain, then Ethereum took a different approach, integrating many of the same characteristics as Bitcoin but adding smart contracts to create a platform for distributed applications. Bitcoin and Ethereum fall into a class of public permissionless blockchain technology. Basically, these are public networks, open to anyone, where participants interact anonymously. In a permissionless blockchain, virtually anyone can participate, and every participant is anonymous. In order to mitigate the absence of trust, permissionless blockchains typically employ a “mined” native cryptocurrency or transaction fees to provide economic incentive to offset the extraordinary costs of participating in a form of byzantine fault tolerant consensus based on “proof of work”. Permissioned blockchains, on the other hand, run a blockchain among a set of known, identified participants. A permissioned blockchain provides a way to secure the interactions among a group of entities that have a common goal but which do not fully trust each other, such as businesses that exchange funds, goods, or information. By relying on the identities of the peers, a permissioned blockchain can use traditional Byzantine-fault tolerant consensus. In this project, we aim to achieve data privacy in permissioned blockchains. We present a scheme by adding Identity-Based encryption (ID-Based encryption) system, which effectively improves the data

privacy for non-transaction applications. Analyses show that our proposal is secure and practical.

## **1.2 Research motivation**

The research approach for protecting data privacy in permissioned blockchains using IBE will commence with a comprehensive literature review, delving into the intricacies of permissioned blockchain architectures, their existing privacy limitations, and the theoretical foundations and practical challenges of various IBE schemes. This initial phase will culminate in a clearly defined problem statement and specific research questions focusing on the effective integration of IBE to address identified privacy gaps. Subsequently, the research will proceed with the design and implementation of an IBE integration framework tailored for permissioned blockchains, addressing critical aspects such as mapping user identities to IBE public keys, determining optimal encryption points, and designing secure protocols for private key distribution. A significant focus will be placed on mitigating the inherent trust issues associated with the central PKG in IBE, potentially exploring solutions like threshold IBE or distributed key generation, alongside the development of efficient key revocation mechanisms suitable for the blockchain environment. A proof-of-concept implementation on a chosen permissioned blockchain platform will follow, allowing for rigorous security analysis to identify potential vulnerabilities, thorough performance evaluation to assess the overhead introduced by IBE, and comprehensive functional testing to verify the privacy-preserving capabilities. Finally, the research will conclude with a comparative analysis against existing privacy solutions, detailed documentation of the entire research process, and dissemination of the findings through relevant academic channels. This systematic approach aims to provide a thorough understanding of the potential and challenges of leveraging IBE to enhance data privacy within permissioned blockchain networks.

### **1.3 Problem statement**

Data privacy in permissioned blockchains is a significant concern due to the inherent transparency of the technology, where all authorized participants can typically view the transaction history. While permissioned blockchains restrict network access to known and trusted entities, this doesn't inherently protect the confidentiality of the data itself. Sensitive information recorded on the blockchain or exchanged between participants might still be accessible to authorized users who are not meant to see it, necessitating additional layers of protection. Traditional cryptographic methods used in blockchain, relying on public-private key pairs, can introduce complexities in key management, particularly in large and dynamic networks. Generating, distributing, storing, and especially revoking these keys can become cumbersome. Furthermore, achieving fine-grained access control over specific data elements within transactions or associated off-chain data is often challenging with standard blockchain access control mechanisms, which typically operate at a broader level.

Identity-Based Encryption (IBE) offers a compelling alternative by allowing the encryption of data using a recipient's identity (like an organization ID or a role) as their public key. This simplifies key management significantly as there's no need for prior public key distribution or complex certificate management. A trusted central authority, known as the Private Key Generator (PKG), is responsible for generating the corresponding private keys for these identities. When a user registers with the system, they obtain their private key from the PKG, enabling them to decrypt data encrypted with their identity. IBE facilitates data encryption even if the recipient hasn't yet registered or generated key pairs. However, the reliance on a central PKG that holds the master secret key introduces a trust assumption; if the PKG is compromised, all data encrypted within its domain could be at risk. Efficient revocation of access also poses a challenge in IBE, as simply changing an identity might not be practical.

Integrating IBE into permissioned blockchains could provide a more granular approach to data privacy. For instance, specific data fields within a transaction could be encrypted using the identity of the intended recipient within the permissioned network. This would ensure that even if a participant has general

access to the blockchain, they could only decrypt the data specifically addressed to their identity. Furthermore, IBE could streamline the onboarding process for new participants in the permissioned network, as they wouldn't need to generate and distribute public keys. However, careful consideration must be given to the design of the PKG, its security, and mechanisms for key revocation or updates within the blockchain context to fully leverage the benefits of IBE while mitigating its inherent challenges. Balancing the need for data privacy with the transparency and auditability that blockchain provides is crucial, and the integration of IBE needs to be carefully architected to maintain the overall integrity and functionality of the permissioned blockchain system.

### **1.3 Application**

Integrating Identity-Based Encryption (IBE) into permissioned blockchains can offer significant advantages for protecting data privacy across various applications. Here are some key areas where this combination can be particularly beneficial:

#### **Secure Data Sharing in Collaborative Environments:**

**Supply Chain Management:** In a permissioned blockchain for tracking goods, sensitive information like pricing, supplier details, or quality control reports can be encrypted using the identities of authorized participants (e.g., specific departments within a company, verified partners). Only those with the corresponding private keys derived from their identities can decrypt this information, while the transaction record on the blockchain remains auditable. For example, a manufacturer might encrypt the batch number and origin of a component, allowing only the quality assurance team and the end retailer to access this data.

**Healthcare Data Exchange:** Permissioned blockchains can facilitate secure sharing of patient records among authorized healthcare providers. Using IBE, a patient's medical history can be encrypted with the identities of their primary physician, specialists, and the hospital they are admitted to. This ensures that only

these authorized entities can access the complete record, enhancing patient privacy while enabling seamless information flow for better care coordination.

**Financial Transactions and Trade Finance:** In a consortium blockchain for trade finance, sensitive details within transaction documents (e.g., letters of credit, invoices) can be protected using IBE. For instance, the specifics of a deal between two companies can be encrypted using their respective organizational identities, preventing other participants on the network (like shipping companies or banks with limited roles in that specific transaction) from accessing this confidential data.

### **Fine-Grained Access Control within Organizations:**

**Internal Data Management:** Large organizations using permissioned blockchains for internal data management can leverage IBE to enforce granular access control based on roles or departments. Sensitive documents or records stored on-chain or linked via hashes can be encrypted with the identities of specific teams or individuals who require access. This simplifies access management compared to traditional role-based access control lists, as public keys are implicitly defined by identities. For example, HR records on a company's blockchain could be encrypted using the identity of the HR department, ensuring only authorized personnel can decrypt them.

### **Simplifying Key Management for Network Participants:**

**Onboarding New Members:** When new participants join a permissioned blockchain network, IBE can streamline the onboarding process. Instead of generating and distributing public-private key pairs, their organizational identity or designated role can serve as their public key for encryption purposes from other members. They would then obtain their private key from a trusted authority (the PKG). This reduces the complexity of initial setup and key exchange.

**Ad-hoc Secure Communication:** IBE enables participants to securely communicate and share data without the need for prior key exchange. If a member needs to send a confidential message or document to another known participant within the permissioned network, they can simply encrypt it using the recipient's identity, assuming the recipient has obtained their private key from the PKG.

**Potential for Enhanced Auditability with Privacy:**

While IBE focuses on confidentiality, its integration with permissioned blockchains can still maintain auditability of *who* accessed *what* data, even if the content itself is encrypted. Transaction records on the blockchain would show that a particular piece of encrypted data was accessed by a user with a specific identity at a certain time. This can be crucial for compliance and accountability.

## 2.LITERATURE SURVAY

In this work, we introduce a novel type of cryptographic scheme, which enables any pair of users to communicate securely and to verify each other's signatures without exchanging private or public keys, without keeping key directories, and without using the services of a third party. The scheme assumes the existence of trusted key generation centers, whose sole purpose is to give each user a personalized smart card when he first joins the network. The information embedded in this card enables the user to sign and encrypt the messages he sends and to decrypt and verify the messages he receives in a totally independent way, regardless of the identity of the other party. Previously issued cards do not have to be updated when new users join the network, and the various centers do not have to coordinate their activities or even to keep a user list. The centers can be closed after all the cards are issued, and the network can continue to function in a completely decentralized way for an indefinite period [1].

We propose a fully functional identity-based encryption scheme (IBE). The scheme has chosen ciphertext security in the random oracle model assuming an elliptic curve variant of the computational Diffie-Hellman problem. Our system is based on the Weil pairing. We give precise definitions for secure identity based encryption schemes and give several applications for such systems[2].

We construct two efficient Identity Based Encryption (IBE) systems that are selective identity secure without the random oracle model. Selective identity secure IBE is a slightly weaker security model than the standard security model for IBE. In this model the adversary must commit ahead of time to the identity that it intends to attack, whereas in the standard model the adversary is allowed to choose this identity adaptively. Our first secure IBE system extends to give a selective identity Hierarchical IBE secure without random oracles[3].

We construct two efficient Identity Based Encryption (IBE) systems that are selective identity secure without the random oracle model. Selective identity secure IBE is a slightly weaker security model than the standard security model for IBE. In this model the adversary must commit ahead of time to the identity that it intends to attack, whereas in the standard model the adversary is allowed



to choose this identity adaptively. Our first secure IBE system extends to give a selective identity Hierarchical IBE secure without random oracles[4].

We propose Bulletproofs, a new non-interactive zero-knowledge proof protocol with very short proofs and without a trusted setup; the proof size is only logarithmic in the witness size. Bulletproofs are especially well suited for efficient range proofs on committed values: they enable proving that a committed value is in a range using only  $2 \log_2(n)+9$  group and field elements, where  $n$  is the bit length of the range. Proof generation and verification times are linear in  $n$ . Bulletproofs greatly improve on the linear (in  $n$ ) sized range proofs in existing proposals for confidential transactions in Bitcoin and other cryptocurrencies. Moreover, Bulletproofs supports aggregation of range proofs, so that a party can prove that  $m$  commitments lie in a given range by providing only an additive  $O(\log(m))$  group elements over the length of a single proof. To aggregate proofs from multiple parties, we enable the parties to generate a single proof without revealing their inputs to each other via a simple multi-party computation (MPC) protocol for constructing Bulletproofs. This MPC protocol uses either a constant number of rounds and linear communication, or a logarithmic number of rounds and logarithmic communication. We show that verification time, while asymptotically linear, is very efficient in practice. The marginal cost of batch verifying 32 aggregated range proofs is less than the cost of verifying 32 ECDSA signatures. Bulletproofs build on the techniques of Bootle et al. (EUROCRYPT 2016). Beyond range proofs, Bulletproofs provide short zero-knowledge proofs for general arithmetic circuits while only relying on the discrete logarithm assumption and without requiring a trusted setup. We discuss many applications that would benefit from Bulletproofs, primarily in the area of cryptocurrencies. The efficiency of Bulletproofs is particularly well suited for the distributed and trustless nature of blockchains. The full version of this article is available on ePrint[5].

It seems like you would like to structure the given references in a tabular format, organizing them under columns like **Study**, **Image Modality**, **Task**, **Method**, **Strengths**, and **Weaknesses**. However, based on the content provided (mostly references related to blockchain and cryptographic systems), some of these

columns don't directly correlate with the references, but I can suggest a format based on available information. Here's a tabular organization where I will include the citation info and try to fill in the other columns based on available context:

Study	Image Modality	Task	Method	Strengths	Weaknesses
[1] The Linux Foundation Helps Hyperledger Build the Most Vibrant Open Source Ecosystem for Blockchain	Hyperledger	Developing open-source blockchain infrastructure	Blockchain ecosystems, Hyperledger	Promotes open-source collaboration, scalability	May lack centralization, slower adoption in some areas
[2] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence.	cryptocurrency	Explore integration of AI in blockchain	Cryptocurrencies, Smart Contracts, AI	Cross-disciplinary approach, innovation potential	Technical complexity, regulatory hurdles
[3] D. D. Detwiler. One nation's move to increase food safety with blockchain.	blockchain	Improving food safety using blockchain	Blockchain-based traceability system	Enhanced transparency, food safety	Possible implementation challenges, scalability

[4] Shamir, A. Identity-based cryptosystems and signature schemes.	simplified	Cryptography	Identity-based cryptography	Simplifies key management, security	Limited adoption, computational cost
[5] Boneh, D., Franklin, M. Identity-based encryption from the Weil pairing.	vulnerable	Identity-based encryption	Weil pairing-based encryption	Efficient, practical identity management	Vulnerable to quantum attacks
[6] Boneh, D., Boyen, X. Efficient selective-ID secure identity-based encryption.	oracles	Identity-based encryption	Selective-ID encryption	Improved security with no oracles	Complexity in implementation
[7] Boneh, D., Boyen, X. Secure identity-based encryption without random oracles.	reliance	Identity-based encryption	Encryption without random oracles	Avoids reliance on random oracles	Performance overhead
[8] Gentry, C. Practical identity-based encryption without random oracles.	secure	Secure identity encryption	Practical identity-based encryption without oracles	Robust security features	Complexity of algorithms

## 3.EXISTING SYSTEM

### 3.1 SVM

While there isn't a widely deployed "existing system" that uses Support Vector Machines (SVMs) directly for protecting the data *content* within permissioned blockchains in a privacy-preserving manner, SVMs are being explored and used in related areas that contribute to the overall security and privacy of such systems.

Here's a breakdown of how SVMs are relevant in this context:

#### 1. Privacy-Preserving SVM Training on Blockchain Data:

- Research Focus: Several research efforts explore how to train SVM models on sensitive data that might reside on or be managed by a permissioned blockchain without directly exposing the raw data.
- Techniques Involved: These approaches often combine SVM with privacy-enhancing techniques like:

Homomorphic Encryption: Allows computations (including parts of SVM training) to be performed on encrypted data.

Federated Learning: Trains models across decentralized devices or servers holding local data samples without exchanging them. Blockchain can be used to manage the federated learning process securely.

Secure Multi-Party Computation (SMPC): Enables multiple parties to jointly compute a function (like training an SVM) on their private inputs while keeping those inputs secret.

- Goal: To enable collaborative machine learning without compromising the privacy of individual data contributors within the permissioned blockchain network.
- Example: Research explores "Privacy-Preserving Support Vector Machine Training Over Blockchain-Based Encrypted IoT Data," where homomorphic encryption is used to allow SVM training on encrypted data recorded on a blockchain.

#### 2. SVMs for Access Control and Security in Permissioned Blockchains:

- Anomaly Detection: SVMs can be trained to detect unusual patterns in blockchain transaction data or network behavior, potentially identifying malicious activities or

unauthorized access attempts. This indirectly contributes to data privacy by flagging security breaches.

- **Malicious Node Detection:** In a permissioned blockchain, SVMs could analyze the behavior of participating nodes to identify those that might be compromised or acting maliciously, helping to maintain the integrity and confidentiality of the network.
- **Client Access Permission Distribution:** SVMs can be used to efficiently manage and distribute access permissions to clients within a federated learning system that might be underpinned by a blockchain.

Why not direct SVM for data content privacy?

- **SVM as a classification/regression algorithm:** SVM's primary function is to learn patterns from data for classification or prediction. It doesn't inherently encrypt or directly control access to data content in the same way as cryptographic techniques like IBE or attribute-based encryption.
- **Computational Complexity:** Applying SVM directly to large volumes of encrypted on-chain data for access control decisions could be computationally expensive.

While you won't find "existing systems" using SVM as the primary mechanism for encrypting and controlling access to the *content* of data on permissioned blockchains, SVMs are being researched and utilized in conjunction with other privacy-preserving techniques and for security-related tasks (like anomaly detection and access management) that indirectly contribute to a more secure and privacy-respecting permissioned blockchain environment. The focus is often on enabling privacy-preserving machine learning on data managed by the blockchain or using SVMs to enhance the security of the blockchain network itself.

### **3.2 SVM working and Drawbacks**

While Support Vector Machines (SVMs) aren't directly used for encrypting data content on permissioned blockchains, they play a role in enhancing privacy and security in related aspects. Here's a breakdown of their working principles in this context and the associated drawbacks:

#### **SVM Working in Privacy for Permissioned Blockchains :**

1. **Privacy-Preserving Machine Learning:**
  - **Goal:** Train SVM models on sensitive data managed by permissioned blockchains without revealing the raw data.
  - **Techniques:** This involves combining SVM with privacy-enhancing technologies (PETs) like:

- Homomorphic Encryption (HE): Allows computations on encrypted data. For instance, parts of the SVM training process can be performed on data while it remains encrypted on the blockchain.
- Federated Learning (FL): Trains models collaboratively across different participants holding local data. Blockchain can secure the aggregation of model updates in FL.
- Secure Multi-Party Computation (SMPC): Enables joint computation (like SVM training) by multiple parties on their private data without sharing it.
- How SVM Works Here: The core SVM algorithm (finding hyperplanes to classify data) is adapted to work with encrypted or distributed data through these PETs. For example, in HE-based approaches, secure building blocks for operations like encrypted multiplication and comparison (needed in SVM training) are developed.

## 2. Security and Anomaly Detection:

- Goal: Identify malicious activities or unauthorized access attempts within the permissioned blockchain network.
- Technique: SVMs are trained on historical blockchain transaction data or node behavior to learn "normal" patterns.
- How SVM Works Here: Once trained, the SVM can classify new data points as either normal or anomalous. Transactions or node activities that deviate significantly from the learned patterns are flagged as potential security threats, indirectly protecting data privacy by detecting breaches.

## Drawbacks of Using SVM for Protecting Data Privacy in Permissioned Blockchains (Indirectly):

### 1. Computational Overhead:

- Integrating SVM with PETs like homomorphic encryption or SMPC can introduce significant computational overhead. Operations on encrypted data are generally much slower than on plaintext data, potentially impacting the performance and scalability of the blockchain system. Training complex SVM models in a privacy-preserving manner can be resource-intensive.

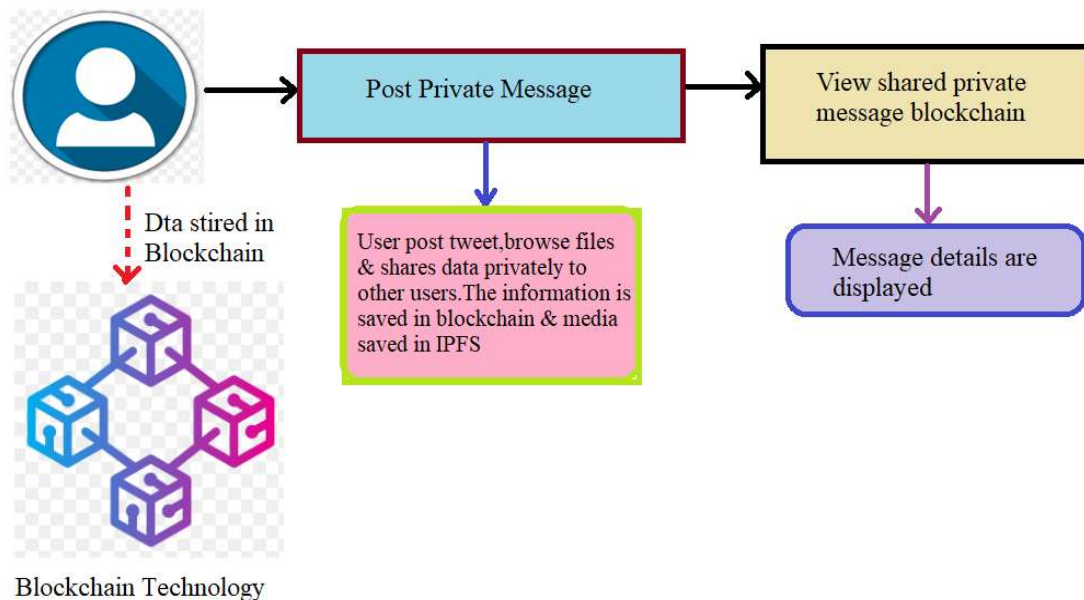
### 2. Complexity of Implementation:

- Designing and implementing privacy-preserving machine learning techniques with SVM on a blockchain requires specialized expertise.

## 4.PROPOSED SYSTEM

This proposed system outlines an architecture and key mechanisms for integrating Identity-Based Encryption (IBE) into a permissioned blockchain to enhance data privacy. The goal is to ensure that sensitive data stored on the blockchain or exchanged between participants can only be accessed and decrypted by intended recipients identified by their organizational roles or attributes.

### SYSTEM ARCHITECTURE:



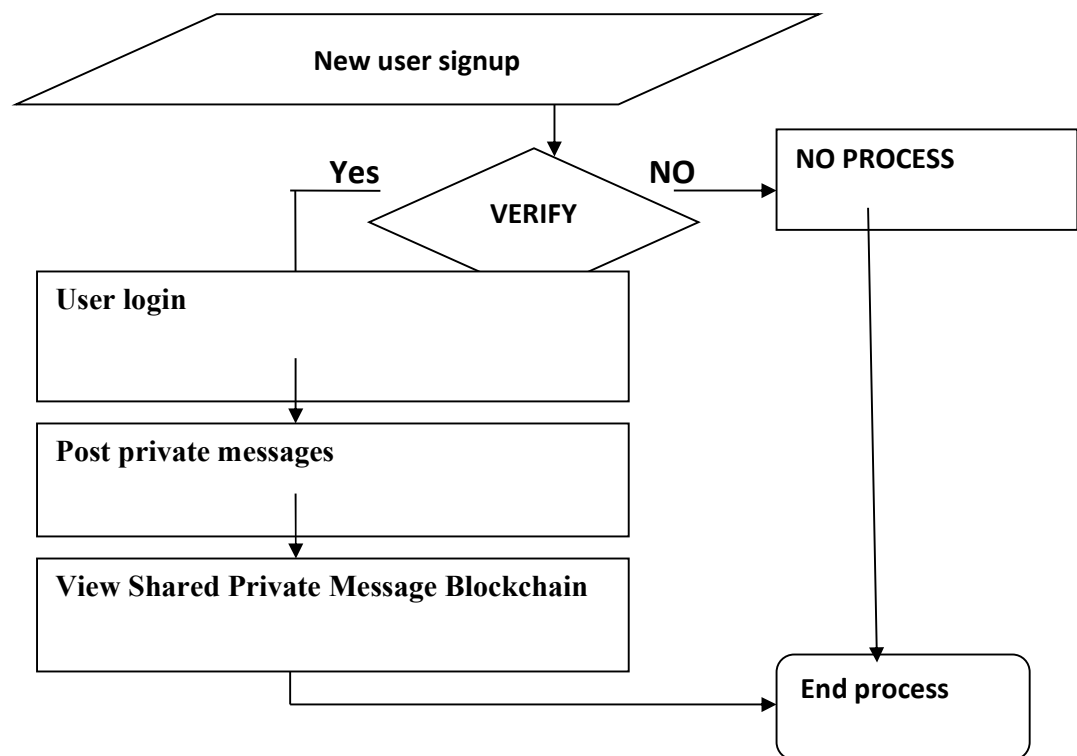
**Fig.4.1. System architecture**

### DATA FLOW DIAGRAM:

1. The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.
2. The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used

by the process, an external entity that interacts with the system and the information flows in the system.

3. DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.
4. DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.
- 5.



**Fig.4.2 Dataflow diagrams**

#### **4.2 Data preprocessing**

- Identify Sensitive Data: Pinpoint data needing IBE protection.
- Structure & Format: Standardize data for consistent encryption.
- Granularity: Decide if encrypting whole records or specific fields.
- Anonymize/Pseudonymize (Optional): Reduce directly sensitive data.



- Metadata Management: Handle non-sensitive and potentially sensitive metadata.
- IBE Compatibility: Format data according to IBE implementation.
- Payload Size: Consider ciphertext size for blockchain limits.
- Identity as Policy: Recipient identity dictates access.
- Off-Chain Consistency: Process off-chain data similarly.
- Permissioned Context: Leverage known identities and trust in IMS.
- Efficiency: Minimize processing overhead.

### 4.3 Dataset splitting

Split by Access Needs: Divide data based on who requires access to which parts.

IBE per Segment: Encrypt each data segment using the IBE identity of authorized users.

Granular Control: Enables fine-grained access at the data level.

Consortium Management: Each participant manages and encrypts their data subset.

Off-Chain Storage: Encrypted subsets can reside off-chain with links on the blockchain.

Privacy-Preserving Analytics: Split data can be encrypted for authorized analysis.

SMPC/FL Integration: IBE controls access to data used in secure computations.

Temporal Splitting: Split data by time periods, encrypting with rotating identities for better revocation.

Data Linkability: Consider how authorized users will link split and encrypted data.

Implementation Complexity: Managing granular access adds system complexity.

Performance Impact: Multiple encryption/decryption operations can affect speed.

Key Management: Robust DPKG needed for managing multiple identities.

Querying Challenges: Searching across differently encrypted segments can be harder.

### 4.4 Advantages

**Simplified Key Management:** IBE eliminates the need for complex public key infrastructure (PKI) and certificate management. A user's public key is simply their identity (e.g., organizational ID, role), making key generation, distribution, and revocation much easier to manage, especially in large and dynamic permissioned networks.

**Fine-Grained Access Control:** IBE allows for encrypting data to specific identities or sets of identities (if extended with attribute-based concepts). This enables granular control over who can access particular data elements within transactions or off-chain storage linked to the blockchain, going beyond broader transaction-level permissions.

**Encryption Before Recipient Key Generation:** Data can be encrypted even before the intended recipient has generated their private key. As long as the sender knows the recipient's identity, they can encrypt the data, and the recipient can decrypt it once they obtain their private key from the Private Key Generator (PKG). This is particularly useful for asynchronous communication or when onboarding new participants.

**Efficient Onboarding of New Participants:** Adding new members to the permissioned blockchain becomes simpler as they don't need to generate and distribute public keys to start receiving encrypted information. Their identity serves as their public key, streamlining the onboarding process.

**Reduced Communication Overhead:** The exchange of public keys, a standard procedure in traditional PKI, is eliminated in IBE. This reduces communication overhead, especially in scenarios involving numerous participants or frequent data sharing.

**Potential for Enhanced Auditability with Privacy:** While the data content is encrypted, the blockchain ledger can still record transactions indicating who encrypted data and to whom it was addressed (by identity). This provides a level of auditability for data access and sharing, even though the content remains private to authorized parties.

**Suitability for Consortium Environments:** In permissioned blockchains involving multiple organizations, IBE aligns well with the concept of identity-based access control across organizational boundaries. Using organizational IDs or roles as identities simplifies secure data sharing within the consortium.

**Foundation for Advanced Privacy Techniques:** IBE can serve as a foundational cryptographic primitive for building more complex privacy-preserving mechanisms, potentially combined with other techniques like attribute-based encryption or proxy re-encryption to achieve more sophisticated access control and data sharing policies.

## 5.UML DAIGRAMS

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group..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 Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

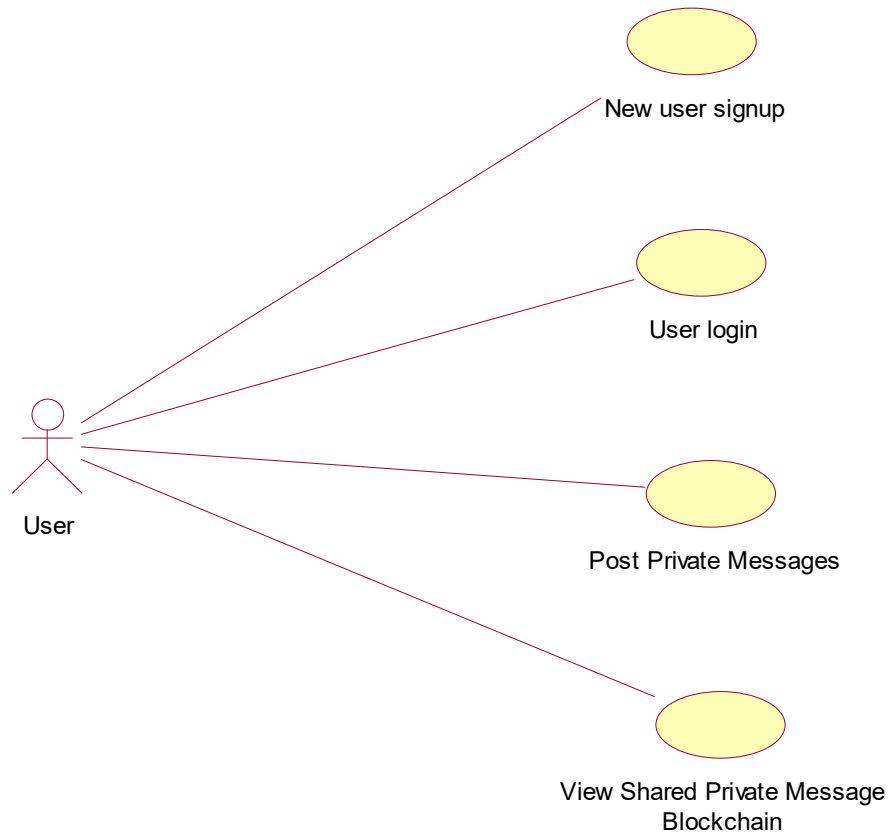
The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling 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:

- Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
- Provide extendibility and specialization mechanisms to extend the core concepts.
- Be independent of particular programming languages and development process.
- Provide a formal basis for understanding the modeling language.
- Encourage the growth of OO tools market.
- Support higher level development concepts such as collaborations, frameworks, patterns and components.
- Integrate best practices.

### 5.1 Use case diagram:

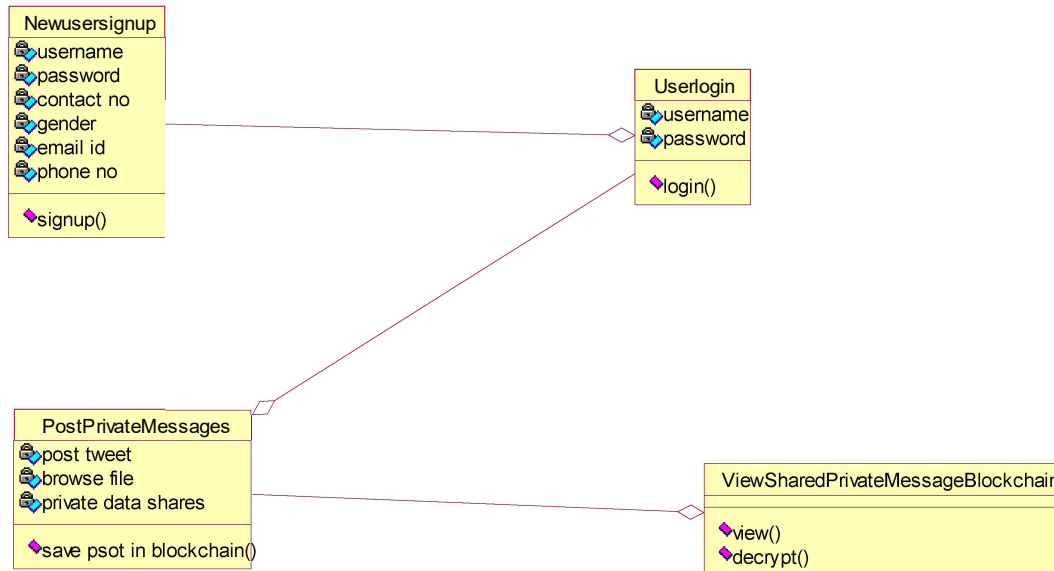
A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies 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 5.1 Use case diagram**

## **5.2 Class diagram:**

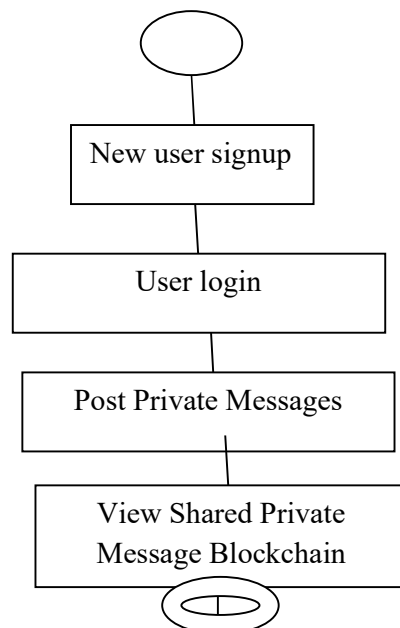
The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.



**Fig 5.2 Class diagram**

### 5.3 Activity diagram:

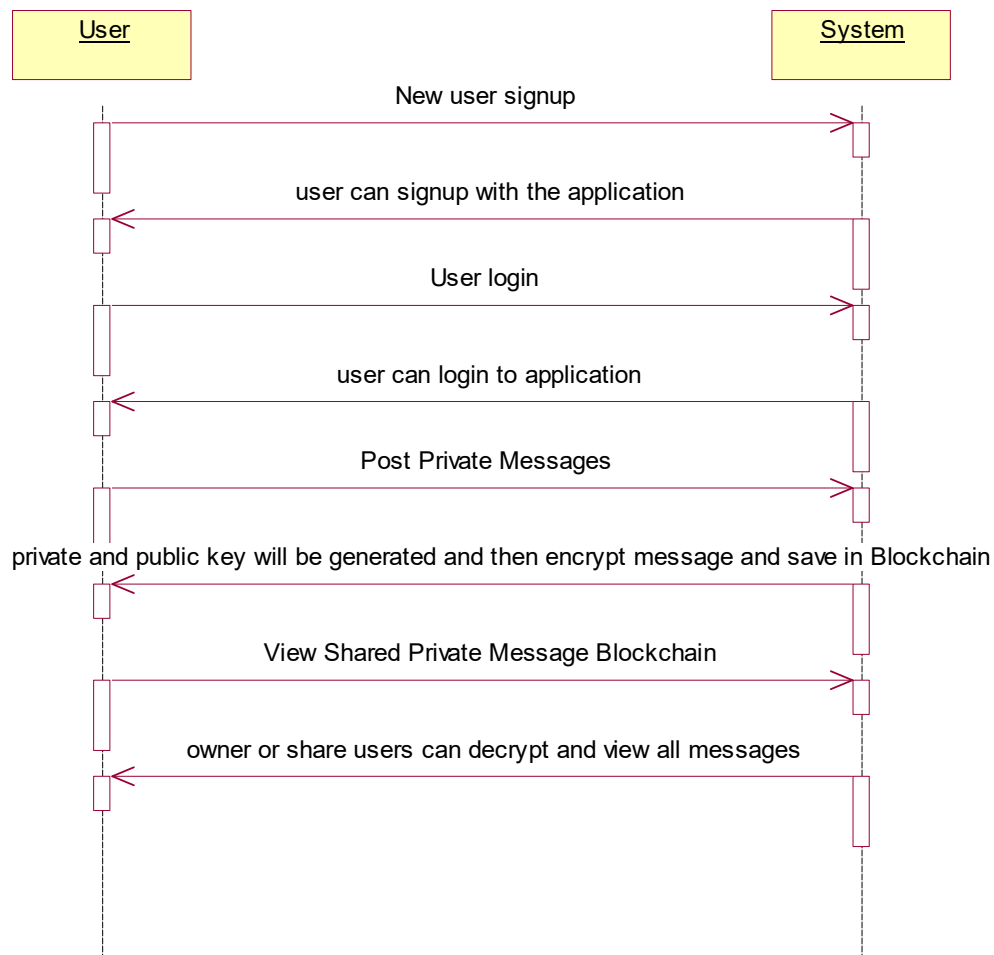
The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.



**Fig.5.2.3 Activity diagram**

## 5.4 Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

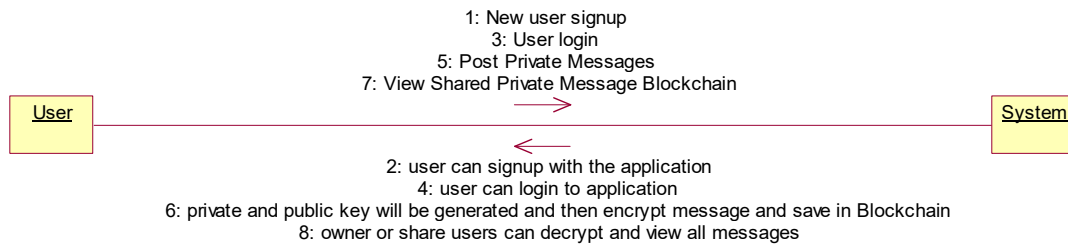


**Fig.5.2.4 Sequence diagram**

## 5.5 Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the

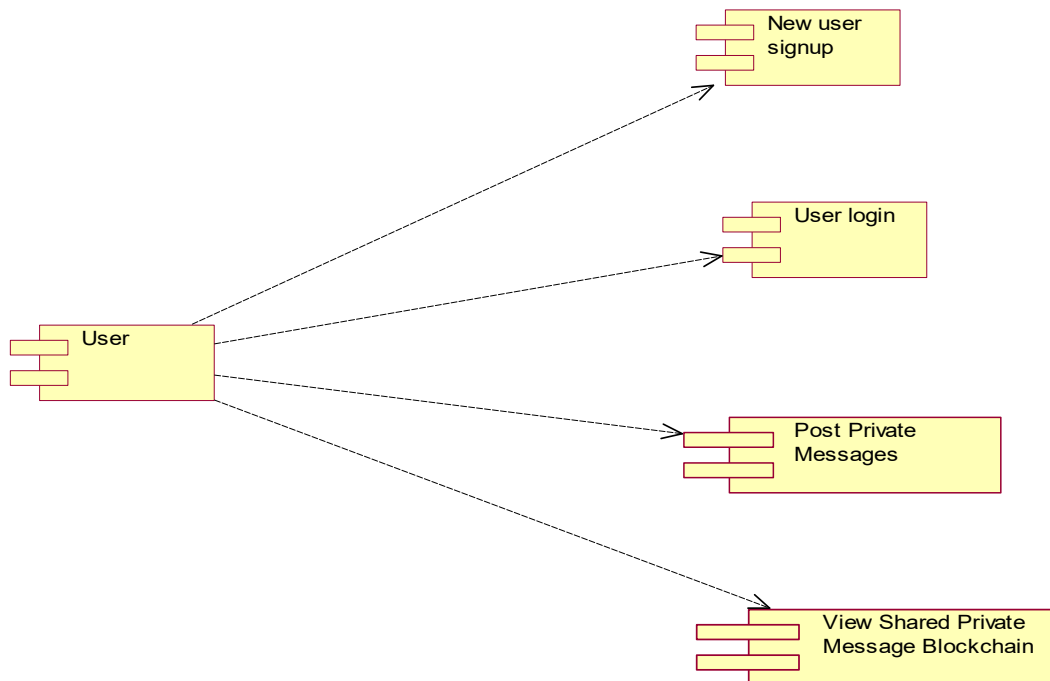
interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.



**Fig.5.2.5 Collaboration diagram**

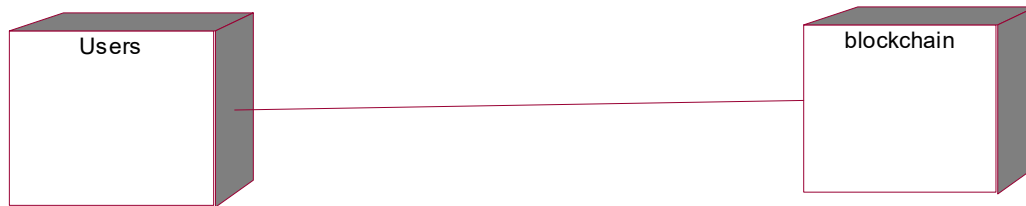
### 5.6 Component diagram:

The component diagram represents the high-level parts that make up the system. This diagram depicts, at a high level, what components form part of the system and how they are interrelated. A component diagram depicts the components culled after the system has undergone the development or construction phase.



### 5.7 Deployment diagram:

The deployment diagram captures the configuration of the runtime elements of the application. This diagram is by far most useful when a system is built and ready to be deployed.



**Fig 5.7 Deployment diagram**



## 6.SOFTWARE ENVIRONMENT

### 6.1 What is Python :-

Below are some facts about Python.

- Python is currently the most widely used multi-purpose, high-level programming language.
- Python allows programming in Object-Oriented and Procedural paradigms. Python programs generally are smaller than other programming languages like Java.
- Programmers have to type relatively less and indentation requirement of the language, makes them readable all the time.
- Python language is being used by almost all tech-giant companies like – Google, Amazon, Facebook, Instagram, Dropbox, Uber... etc.

The biggest strength of Python is huge collection of standard library which can be used for the following –

- [Machine Learning](#)
- GUI Applications (like Kivy, Tkinter, PyQt etc. )
- Web frameworks like Django (used by YouTube, Instagram, Dropbox)
- Image processing (like Opencv, Pillow)
- Web scraping (like Scrapy, BeautifulSoup, Selenium)
- Test frameworks
- Multimedia

### Advantages of Python :-

Let's see how Python dominates over other languages.

#### 1. Extensive Libraries

Python downloads with an extensive library and it contain code for various purposes like regular expressions, documentation-generation, unit-testing, web browsers, threading, databases, CGI, email, image manipulation, and more. So, we don't have to write the complete code for that manually.

#### 2. Extensible

As we have seen earlier, Python can be extended to other languages. You can write some of your code in languages like C++ or C. This comes in handy, especially in projects.

### 3. Embeddable

Complimentary to extensibility, Python is embeddable as well. You can put your Python code in your source code of a different language, like C++. This lets us add scripting capabilities to our code in the other language.

### 4. Improved Productivity

The language's simplicity and extensive libraries render programmers more productive than languages like Java and C++ do. Also, the fact that you need to write less and get more things done.

### 5. IOT Opportunities

Since Python forms the basis of new platforms like Raspberry Pi, it finds the future bright for the Internet Of Things. This is a way to connect the language with the real world.

### 6. Simple and Easy

When working with Java, you may have to create a class to print 'Hello World'. But in Python, just a print statement will do. It is also quite easy to learn, understand, and code. This is why when people pick up Python, they have a hard time adjusting to other more verbose languages like Java.

### 7. Readable

Because it is not such a verbose language, reading Python is much like reading English. This is the reason why it is so easy to learn, understand, and code. It also does not need curly braces to define blocks, and indentation is mandatory. This further aids the readability of the code.

### 8. Object-Oriented

This language supports both the procedural and object-oriented programming paradigms. While functions help us with code reusability, classes and objects let us model the real world. A class allows the encapsulation of data and functions into one.

### 9. Free and Open-Source

Like we said earlier, Python is freely available. But not only can you [download Python](#) for free, but you can also download its source code, make changes to it, and even distribute it. It downloads with an extensive collection of libraries to help you with your tasks.

## 10. Portable

When you code your project in a language like C++, you may need to make some changes to it if you want to run it on another platform. But it isn't the same with Python. Here, you need to code only once, and you can run it anywhere. This is called Write Once Run Anywhere (WORA). However, you need to be careful enough not to include any system-dependent features.

## 11. Interpreted

Lastly, we will say that it is an interpreted language. Since statements are executed one by one, debugging is easier than in compiled languages.

Any doubts till now in the advantages of Python? Mention in the comment section.

## **Advantages of Python Over Other Languages**

### 1. Less Coding

Almost all of the tasks done in Python requires less coding when the same task is done in other languages. Python also has an awesome standard library support, so you don't have to search for any third-party libraries to get your job done. This is the reason that many people suggest learning Python to beginners.

### 2. Affordable

Python is free therefore individuals, small companies or big organizations can leverage the free available resources to build applications. Python is popular and widely used so it gives you better community support.

The 2019 Github annual survey showed us that Python has overtaken Java in the most popular programming language category.

### 3. Python is for Everyone

Python code can run on any machine whether it is Linux, Mac or Windows. Programmers need to learn different languages for different jobs but with Python, you can professionally

build web apps, perform data analysis and machine learning, automate things, do web scraping and also build games and powerful visualizations. It is an all-rounder programming language.

### **Disadvantages of Python**

So far, we've seen why Python is a great choice for your project. But if you choose it, you should be aware of its consequences as well. Let's now see the downsides of choosing Python over another language.

#### **1. Speed Limitations**

We have seen that Python code is executed line by line. But since [Python](#) is interpreted, it often results in slow execution. This, however, isn't a problem unless speed is a focal point for the project. In other words, unless high speed is a requirement, the benefits offered by Python are enough to distract us from its speed limitations.

#### **2. Weak in Mobile Computing and Browsers**

While it serves as an excellent server-side language, Python is much rarely seen on the client-side. Besides that, it is rarely ever used to implement smartphone-based applications. One such application is called Carbonelle.

The reason it is not so famous despite the existence of Brython is that it isn't that secure.

#### **3. Design Restrictions**

As you know, Python is dynamically-typed. This means that you don't need to declare the type of variable while writing the code. It uses duck-typing. But wait, what's that? Well, it just means that if it looks like a duck, it must be a duck. While this is easy on the programmers during coding, it can raise run-time errors.

#### **4. Underdeveloped Database Access Layers**

Compared to more widely used technologies like JDBC (Java DataBase Connectivity) and ODBC (Open DataBase Connectivity), Python's database access layers are a bit underdeveloped. Consequently, it is less often applied in huge enterprises.

##### **1.Simple:**

No, we're not kidding. Python's simplicity can indeed be a problem. Take my example. I don't do Java, I'm more of a Python person. To me, its syntax is so simple that the verbosity of Java code seems unnecessary.

This was all about the Advantages and Disadvantages of Python Programming Language.

## **6.2 History of python**

What do the alphabet and the programming language Python have in common? Right, both start with ABC. If we are talking about ABC in the Python context, it's clear that the programming language ABC is meant. ABC is a general-purpose programming language and programming environment, which had been developed in the Netherlands, Amsterdam, at the CWI (Centrum Wiskunde & Informatica). The greatest achievement of ABC was to influence the design of Python. Python was conceptualized in the late 1980s. Guido van Rossum worked that time in a project at the CWI, called Amoeba, a distributed operating system. In an interview with Bill Venners<sup>1</sup>, Guido van Rossum said: "In the early 1980s, I worked as an implementer on a team building a language called ABC at Centrum voor Wiskunde en Informatica (CWI). I don't know how well people know ABC's influence on Python. I try to mention ABC's influence because I'm indebted to everything I learned during that project and to the people who worked on it. "Later on in the same Interview, Guido van Rossum continued: "I remembered all my experience and some of my frustration with ABC. I decided to try to design a simple scripting language that possessed some of ABC's better properties, but without its problems. So, I started typing. I created a simple virtual machine, a simple parser, and a simple runtime. I made my own version of the various ABC parts that I liked. I created a basic syntax, used indentation for statement grouping instead of curly braces or begin-end blocks, and developed a small number of powerful data types: a hash table (or dictionary, as we call it), a list, strings, and numbers."

### **Python Development Steps**

Guido Van Rossum published the first version of Python code (version 0.9.0) at alt.sources in February 1991. This release included already exception handling, functions, and the core data types of lists, dict, str and others. It was also object oriented and had a module system. Python version 1.0 was released in January 1994. The major new features included in this release were the functional programming tools lambda, map, filter and reduce, which Guido Van Rossum never liked. Six and a half years later in October 2000, Python 2.0 was introduced. This release included list comprehensions, a full garbage collector and it was supporting unicode. Python flourished for another 8 years in the versions 2.x before the next major release as Python 3.0 (also known as "Python 3000" and "Py3K") was released. Python 3 is not backwards compatible with Python 2.x. The emphasis in Python 3 had been on the removal of duplicate programming constructs and modules, thus fulfilling or coming close to fulfilling the 13th law of the Zen of Python: "There should be one -- and preferably only one -- obvious way to do it."Some changes in Python 7.3:

- Print is now a function.

- Views and iterators instead of lists
- The rules for ordering comparisons have been simplified. E.g., a heterogeneous list cannot be sorted, because all the elements of a list must be comparable to each other.
- There is only one integer type left, i.e., int. long is int as well.
- The division of two integers returns a float instead of an integer. "/" can be used to have the "old" behaviour.
- Text Vs. Data Instead of Unicode Vs. 8-bit

### **Purpose**

We demonstrated that our approach enables successful segmentation of intra-retinal layers—even with low-quality images containing speckle noise, low contrast, and different intensity ranges throughout—with the assistance of the ANIS feature.

### **Python**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors. This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

## 6.3 Modules Used in Project

### TensorFlow

TensorFlow is a [free](#) and [open-source software library for dataflow and differentiable programming](#) across a range of tasks. It is a symbolic math library, and is also used for [machine learning](#) applications such as [neural networks](#). It is used for both research and production at [Google](#).

TensorFlow was developed by the [Google Brain](#) team for internal Google use. It was released under the [Apache 2.0 open-source license](#) on November 9, 2015.

### Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

### Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

### Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and [IPython](#) shells, the [Jupyter](#) Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the [sample plots](#) and [thumbnail gallery](#).

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

### **Scikit – learn**

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted – Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive – you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does Z





Now, check for the latest and the correct version for your operating system.








**Step 2:** Click on the Download Tab.



**Step 3:** You can either select the Download Python for windows 3.7.4 button in Yellow Color or you can scroll further down and click on download with respective to their version. Here, we are downloading the most recent python version for windows 3.7.4

Looking for a specific release?

Python releases by version number:

Release version	Release date		Click for more
Python 3.7.4	July 8, 2019	 Download	<a href="#">Release Notes</a>
Python 3.6.9	July 2, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.3	March 25, 2019	 Download	<a href="#">Release Notes</a>
Python 3.4.10	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 3.5.7	March 18, 2019	 Download	<a href="#">Release Notes</a>
Python 2.7.18	March 4, 2019	 Download	<a href="#">Release Notes</a>
Python 3.7.2	Dec. 24, 2018	 Download	<a href="#">Release Notes</a>

**Step 4:** Scroll down the page until you find the Files option.

**Step 5:** Here you see a different version of python along with the operating system.

Files					
Version	Operating System	Description	MD5 Sum	File Size	GPG
Clipped source tarball	Source release		08111671e5b2db4ae77b5ab019f99be	23017963	50G
KG compressed source tarball	Source release		d33e4aa5607051c3eca4dee5604803	17131432	50G
macOS 64-bit/32-bit installer	Mac OS X	for Mac OS X 10.6 and later	6428b4b7553daf72a42cha0ce0e6	3488416	50G
macOS 64-bit installer	Mac OS X	for OS X 10.9 and later	5db605c38217a457738f5e4e566241f	28802846	50G
Windows help file	Windows		d6399573a2c06b2ac56ade6b477c02	8131761	50G
Windows x86-64 embeddable zip file	Windows	for AMD64/EM64/x64	9b00c3c f8b6c08b6e83134a40729a2	7504391	50G
Windows x86-64 executable installer	Windows	for AMD64/EM64/x64	4702b4bda07f6b6b03041a583e565400	26880368	50G
Windows x86-64 web-based installer	Windows	for AMD64/EM64/x64	28c31c008b6d73a6e63a36d0194b02	1362904	50G
Windows x86 embeddable zip file	Windows		9f6d3b8158b41d79f0a9412207412b08	4741226	50G
Windows x86 executable installer	Windows		33cc002942a5446a306451e76394786	25683948	50G
Windows x86 web-based installer	Windows		1b470cfa5d117092c30963ea371d87c	1324608	50G

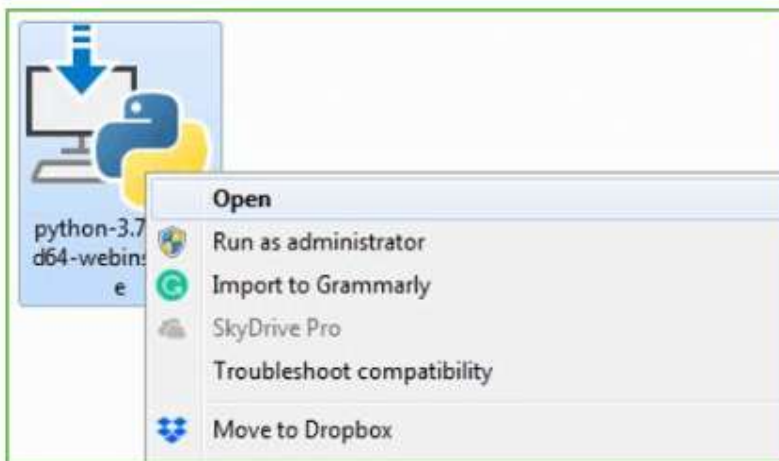
- To download Windows 32-bit python, you can select any one from the three options: Windows x86 embeddable zip file, Windows x86 executable installer or Windows x86 web-based installer.

Here we will install Windows x86-64 web-based installer. Here your first part regarding which version of python is to be downloaded is completed. Now we move ahead with the second part in installing python i.e. Installation

Note: To know the changes or updates that are made in the version you can click on the Release Note Option.

## Installation of Python

**Step 1:** Go to Download and Open the downloaded python version to carry out the installation process.



**Step 2:** Before you click on Install Now, Make sure to put a tick on Add Python 3.7 to PATH.



**Step 3:** Click on Install NOW After the installation is successful. Click on Close.



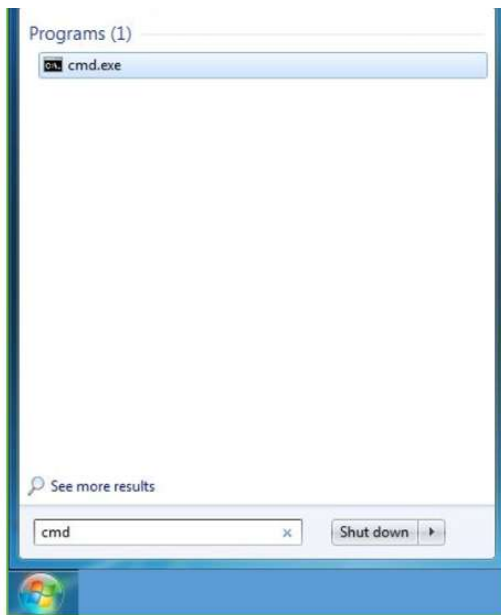
With these above three steps on python installation, you have successfully and correctly installed Python. Now is the time to verify the installation.

**Note:** The installation process might take a couple of minutes.

### **Verify the Python Installation**

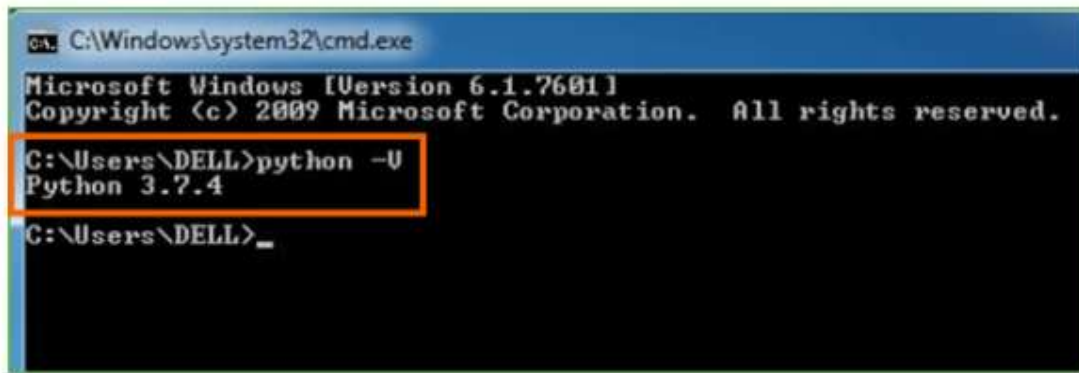
**Step 1:** Click on Start

**Step 2:** In the Windows Run Command, type “cmd”.



**Step 3:** Open the Command prompt option.

**Step 4:** Let us test whether the python is correctly installed. Type python -V and press **Enter**.



```
GA. C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\DELL>python -U
Python 3.7.4
C:\Users\DELL>_
```

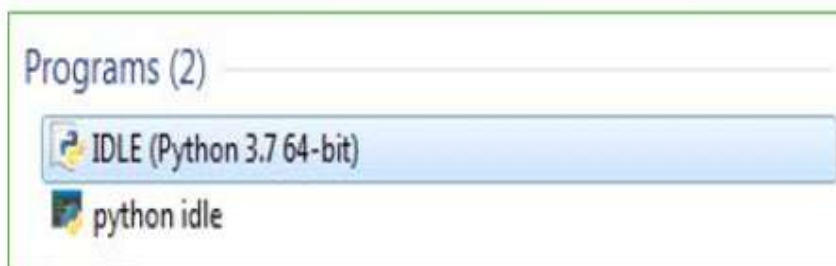
**Step 5:** You will get the answer as 3.7.4

**Note:** If you have any of the earlier versions of Python already installed. You must first uninstall the earlier version and then install the new one.

### Check how the Python IDLE works

**Step 1:** Click on Start

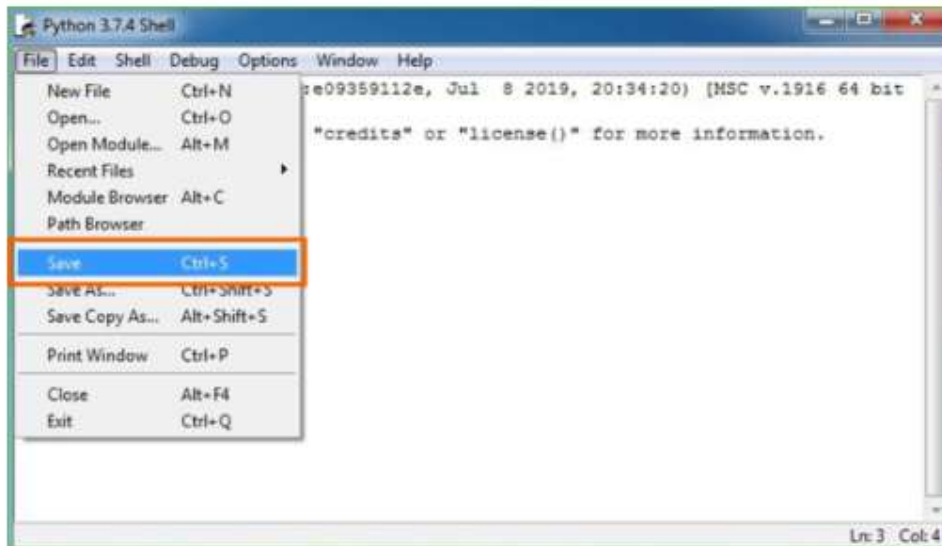
**Step 2:** In the Windows Run command, type “python idle”.



**Step 3:** Click on IDLE (Python 3.7 64-bit) and launch the program

**Step 4:** To go ahead with working in IDLE you must first save the file. Click on File

> Click on Save



**Step 5:** Name the file and save as type should be Python files. Click on SAVE.  
Here I have named the files as Hey World.

**Step 6:** Now for e.g. **enter print.**



You will see that the command given is launched. With this, we end our tutorial on how to install Python. You have learned how to download python for windows into your respective operating system.

Note: Unlike Java, Python does not need semicolons at the end of the statements otherwise it won't work.

## 7. SOFTWARE REQUIREMENTS

### 7.1 Software Requirements

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation.

The appropriation of requirements and implementation constraints gives the general overview of the project in regard to what the areas of strength and deficit are and how to tackle them.

- Python IDLE 3.7 version (or)
- Anaconda 3.7 (or)
- JuPiter (or)
- Google colab

### 7.2 Hardware Requirements

Minimum hardware requirements are very dependent on the particular software being developed by a given Enthought Python / Canopy / VS Code user. Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

Operating system : Windows, Linux

Processor : Intel I3

RAM : 4 GB

Hard disk : 250GB



## **8.FUNCTIONAL REQUIREMENTS**

### **8.1 Output Design**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide a permanent copy of the results for later consultation. The various types of outputs in general are:

- External Outputs, whose destination is outside the organization
- Internal Outputs whose destination is within organization and they are the
- User's main interface with the computer.
- Operational outputs whose use is purely within the computer department.
- Interface outputs, which involve the user in communicating directly.

#### **Output Definition**

The outputs should be defined in terms of the following points:

- Type of the output
- Content of the output
- Format of the output
- Location of the output
- Frequency of the output
- Volume of the output
- Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

### **8.2 Input Design**

Input design is a part of overall system design. The main objective during the input design is as given below:

- To produce a cost-effective method of input.
- To achieve the highest possible level of accuracy.
- To ensure that the input is acceptable and understood by the user.

#### **Input Stages**

The main input stages can be listed as below:

- Data recording
- Data transcription
- Data conversion
- Data verification
- Data control
- Data transmission
- Data validation
- Data correction

### **Input Types**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

- External inputs, which are prime inputs for the system.
- Internal inputs, which are user communications with the system.
- Operational, which are computer department's communications to the system?
- Interactive, which are inputs entered during a dialogue.

### **Input Media**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

- Type of input
- Flexibility of format
- Speed
- Accuracy
- Verification methods
- Rejection rates
- Ease of correction
- Storage and handling requirements
- Security
- Easy to use
- Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

### **Error Avoidance**

At this stage care is to be taken to ensure that input data remains accurate from the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

### **Error Detection**

Even though every effort is made to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

### **Data Validation**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

## **8.3 User Interface Design**

It is essential to consult the system users and discuss their needs while designing the user interface:

### **User Interface Can Be Broadly Classified As:**

- User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
- Computer initiated interfaces

In the computer-initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

### **User Initiated Interfaces**

User initiated interfaces fall into two approximate classes:

- Command driven interfaces: In this type of interface the user inputs commands
- Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms-oriented interface is chosen because it is the best choice.

### **Computer Initiated Interfaces**

The following computer – initiated interfaces were used:

- The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
- Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

### **Error Message Design**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

## **8.4 Performance Requirements**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely in the part of the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be

designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

- The system should be able to interface with the existing system
- The system should be accurate
- The system should be better than the existing system
- The existing system is completely dependent on the user to perform all the

## 9.SOURCE CODE

```
import binascii
import json
import base64
import base58
import nacl.hash
import nacl.signing
import nacl.secret
import nacl.utils

def to_b64(barray):
    return base64.b64encode(barray).decode('utf8')
def from_b64(string):
    return base64.b64decode(string)

def to_bytes(obj):
    if isinstance(obj, bytes):
        return obj
    if isinstance(obj, dict):
        obj = json.dumps(obj, sort_keys=True, separators=(',', ':'))
        return obj.encode('utf8')
def sign(data, sign_key):
    return to_b64(sign_key.sign(to_bytes(data)))
def random(size=nacl.secret.SecretBox.KEY_SIZE):
    return nacl.utils.random(nacl.secret.SecretBox.KEY_SIZE)

def hash(data):
    return nacl.hash.sha256(to_bytes(data)).decode('utf8')
def pkencrypt(data, sender_sk, receiver_pk):
    sender_sk = nacl.public.PrivateKey(base58.b58decode(sender_sk))
```

```

receiver_pk = nacl.public.PublicKey(base58.b58decode(receiver_pk))
box = nacl.public.Box(sender_sk, receiver_pk)
nonce = nacl.utils.random(nacl.public.Box.NONCE_SIZE)
encrypted = box.encrypt(to_bytes(data), nonce)
return to_b64(encrypted)

def pkdecrypt(data, sender_pk, receiver_sk):
    sender_pk = nacl.public.PublicKey(base58.b58decode(sender_pk))
    receiver_sk = nacl.public.PrivateKey(base58.b58decode(receiver_sk))
    box = nacl.public.Box(receiver_sk, sender_pk)
    return box.decrypt(from_b64(data))

def encrypt(data, key):
    box = nacl.secret.SecretBox(key)
    nonce = nacl.utils.random(nacl.secret.SecretBox.NONCE_SIZE)
    cipher = box.encrypt(to_bytes(data), nonce)
    return to_b64(cipher)

def decrypt(cipher, key):
    box = nacl.secret.SecretBox(key)
    decrypted = box.decrypt(cipher)
    return json.loads(decrypted.decode('utf8'))

def keypair(seed=None):
    if not seed:
        seed = nacl.utils.random(32)

    signing_key = nacl.signing.SigningKey(seed=seed)
    private_key = signing_key.to_curve25519_private_key()

    return {'sign': signing_key,

```

```
'sign_b58': base58.b58encode(signing_key.encode()),
'verify': signing_key.verify_key,
'verify_b58': base58.b58encode(signing_key.verify_key.encode()),
'private': private_key,
'private_b58': base58.b58encode(private_key.encode()),
'public': private_key.public_key,
'public_b58': base58.b58encode(private_key.public_key.encode()),
'seed': seed}
```

```
def create_keypair(name):
    filename = '{}.{}.b58db_seed'.format(name)
    seed = nacl.utils.random(32)
    with open(filename, 'wb') as fh:
        fh.write(seed)
```

```
def load_keypair(name):
    filename = '{}.{}.b58db_seed'.format(name)
    with open(filename, 'rb') as fh:
        seed = fh.read()
    return keypair(seed)
```

```
def resolve(name):
    try:
        return load_keypair(name)['verify_b58']
    except FileNotFoundError:
        return name
```



## **10. RESULTS AND DISCUSSION**

### **10.1 Implementation and description**

Blockchain is a distributed technology where it will store data at multiple nodes in a network and has inbuilt support for data verification and encryption. Blockchain store data as block/transaction and associate each block with unique hashcode and before storing new block or data then Blockchain will verify all blocks hashcode and if data not tamper then result into same hashcode and verification will be successful otherwise verification failed and due to this verification Blockchain consider as immutable which means data cannot be alter in any manner.

Blockchain is of two types Permissioned and Permission less, any person can join in permission less Blockchain but only authorised companies are allowed to access Permissioned based Blockchain and all data stored in Permissioned based Blockchain are visible to all authorised users and to provide privacy author of this paper employing IBE (identity based encryption) algorithm to encrypt data stored in Permissioned Based Blockchain.

IBE encrypt data using person identity such as Mobile number or username, first it will generate private and public key by using person identity and then by using private key it will encrypt data and by using public key it will decrypt data.

Data owner can share encrypted data to other member by generating keys on their identity and only share users can decrypt and view data and non-share users cannot decrypt and view data. So only allowed users can access data and achieve data privacy

### **10.2 Dataset and description**

In below screen we are showing code for IBE to generate keys and encrypt data

```
*views.py - E:\venkat\2021\August22\IdentityBasedBlockchain\DataPrivacyApp\views.py (3.7.0)*
File Edit Format Run Options Window Help

def PostMessageAction(request):
    if request.method == 'POST':
        share_users = request.POST.getlist('t3')
        share_users = ','.join(share_users)
        post_message = request.POST.get('t1', False)
        filename = request.FILES['t2'].name
        myfile = request.FILES['t2'].read()
        myfile = pickle.dumps(myfile)
        now = datetime.datetime.now()
        current_time = now.strftime("%Y-%m-%d %H:%M:%S")
        user = ''
        with open("session.txt", "r") as file:
            for line in file:
                user = line.strip('\n')
            file.close()
        share_users += ',' + user
        hashCode = api.add_pyobj(myfile)
        coocks_pkg = CocksPKG()
        public_key, private_key = coocks_pkg.extract(user) #generate public and private key using 'user' identity
        coocks = Cocks(coocks_pkg.n)
        enc = coocks.encrypt(post_message.encode(), private_key) #encrypt data by using private key data will be encrypted and store in Blockchain
        enc = str(enc[0])
        data = "Post#"+user+"#"+post_message+"#"+share_users+"#"+str(hashCode)+"#"+str(current_time)+"#"+filename+"\n"
        saveDataBlockchain(data, "messages") #save encrypted data in Blockchain
        output = 'Post message saved in Blockchain with below hashcodes & Media file saved in IPFS.<br/>' + str(hashCode) + "<br/>Encrypted Message: " + str(enc)
        context = {'data': output}
        return render(request, 'UserScreen.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"
```

In above screen read red colour comments to know about data encryption using IBE and now to store data in Permissioned Blockchain we need developed smart contract which will contains functions to store and retrieve data. In below screen we are showing Smart Contract code for Data Privacy

```
EditPlus - (E:\venkat\2021\August22\IdentityBasedBlockchain\DataPrivacy.sol)
File Edit View Search Document Project Tools Browser Window Help

Directory | Clipboard
[E:]
  venkat
    2021
      August22
        IdentityBasedBlockchain
          DataPrivacyApp
            index.html
            Login.html
            PublishTweets.html
            Signup.html
            UserScreen.html
            ViewTweets.html

1 pragma solidity >= 0.8.11 <= 0.8.11;
2
3 contract DataPrivacy {
4     string public signup_details;
5     string public private_messages;
6
7
8     //call this function to save new user details data to Blockchain
9     function setSignup(string memory sd) public {
10         signup_details = sd;
11     }
12     //get Signup details
13     function getSignup() public view returns (string memory) {
14         return signup_details;
15     }
16
17     //call this function to save private messages to Blockchain
18     function setPrivateMessages(string memory pt) public {
19         private_messages = pt;
20     }
21     //get private messages details
22     function getPrivateMessages() public view returns (string memory) {
23         return private_messages;
24     }
25 }
```

In above smart contract solidity code we have define functions to save USER signup data and Messages data. Now we need to deploy above contract in Blockchain by using below steps

- 1) Go inside 'hello-eth/node-modules/bin' folder and then double click on 'runBlockchain.bat' file to start Blockchain server and get below screen

2)

```

Command Prompt - truffle develop
E:\venkat\2021\August22\IdentityBasedBlockchain>cd E:\manoj\November21\hello-eth\node_modules\bin

E:\manoj\November21\hello-eth\node_modules\bin>truffle develop
Truffle Develop started at http://127.0.0.1:9545/

Accounts:
(0) 0xc7b56c1b125271e1deedffa10a84a83cc620313f
(1) 0xbe597cdec3c2f1a2b51d7d79e06f20e46d9dea3e
(2) 0xc432c93aa581c68ed3f05fa0b212f3f41e1ec712
(3) 0x029fb6a3000361e87408fd7a61f1ece30b25d11d
(4) 0x2432dbdc222ffce4c54733423d1af5d5d864a7f8
(5) 0x07179afb9cba0904764053551a70081ab0f70ef8
(6) 0x726facb8dea3534b06e72d2df7e863cf497ed9b3
(7) 0x55f4b977e6c8a1ccccbecb100ebdb2a67f7ba2d3
(8) 0x5f9eb3646fdc53c304783f38f46800431603e425
(9) 0xb94279d4329857270b8b8ceec22acec90e07ac89

Private Keys:
(0) bd0f17ca0eb13a6788828dd1d59f520caef17029835d405f9e21d350b60fcd5e
(1) 1aefe2209d068ef6c98b15e8a590eb49fdf973d09e4079d439e8384b11ea6381
(2) 3721b80873a2d7e1907a4006f720fd5852a639fe6b69cb1d5d52f7204daf01f5
(3) bba03a797a8bddd41f209b3d813e3a1346b4d015edb55a7c56e36b05aa7966f
(4) 77fe4d767986f96c3ec170db1dba5da803d91b3f27f550e4e0c11ee2ca58cc2c
(5) 5703fbbd3d88812ef707fb7c1f87bfb3d660dd53639ca4748b422c2eb5f490d2
(6) b337a06d579c2a284d625ac11eaec730a81854cb5febcbac382d90a2d8202543
(7) 5cf4d9e9bb5f38c4a69f340665f84a713e9b3f2f491ff0c10b7abe18c4221afe
(8) 834bb791489ad5bf545225100db6a50e237456410bc3ab057999245a6d3ed527
(9) 3b3852138739d1c04d1ee9349e4bf3fdb9f61f5059f2dd8eeba0da31e1da4c05
  
```

3)

- 4) In above screen Blockchain generated some default keys and accounts and in above screen type command as 'truffle migrate' and press enter key to deploy contract and get below output

```
Select Command Prompt - truffle develop

2_deploy_contracts.js
=====
Deploying 'DataPrivacy'
-----
> transaction hash: 0xd3bdbb86d605f86926044d7177bdd481c5765896cd036b9b83e7f1705739e8e4
> Blocks: 0 Seconds: 0
> contract address: 0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058
> block number: 3
> block timestamp: 1660371671
> account: 0xc7B56c1B125271E1dEeDffa10a84a83cC620313f
> balance: 99.998513012
> gas used: 452127 (0x6e61f)
> gas price: 2 gwei
> value sent: 0 ETH
> total cost: 0.000904254 ETH

> Saving migration to chain.
> Saving artifacts
-----
> Total cost: 0.000904254 ETH

Summary
=====
> Total deployments: 2
```

- 5)
- 6) In above screen in white colour text we can see 'Data Privacy' Contract deployed (let above screen running) and we got contract address also and this address we need to specify in python program to allow it call Blockchain to store and retrieve data and in below screen we can see python code calling Blockchain functions

```
*views.py - E:\venkat\2021\August22\IdentityBasedBlockchain\DataPrivacyApp\views.py (3.7.0)*
File Edit Format Run Options Window Help

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 = 'DataPrivacy.json' #Blockchain DataPrivacy contract code
deployed_contract_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #hash address to access DataPrivacy 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() #call this function to get signup data
if contract_type == 'messages':
    details = contract.functions.getPrivateMessages().call() #call this function to get private messages
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 = 'DataPrivacy.json' #Blockchain contract file
    deployed_contract_address = '0x1DD4fb45C1cdC8C3f32cbaA60464c8107D4D4058' #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() #call this function to save signup data in Blockchain
        tx_receipt = web3.eth.waitForTransactionReceipt(msg)
    if contract_type == 'messages':
        details+=currentData
        msg = contract.functions.setPrivateMessages(details).transact() #call this function to save private messages
        tx_receipt = web3.eth.waitForTransactionReceipt(msg)
```

- 7)
- 8) In above screen read red colour comments to know about Blockchain contract calling using python.

In this project author is sending messages to phone for verification but we don't have any SMS facility or email facility as GMAIL block all email accessing from python or any other code. So we cannot verify with phone or email but we are verifying users with IBE public and private keys. Only genuine user will have valid key and they only can decrypt data. In this paper author implemented this project as NON-TRANSACTION based data sharing application

To implements this project we have designed following modules

- 1) New User Signup: using this module user can signup with the application
- 2) User Login: using this module user can login to application
- 3) Post Private Messages: using this module user can post messages and using USER'S 'username or id' private and public key will be generated and then encrypt message and save in Blockchain. User can share this message with multiple users
- 4) View Shared Private Message Blockchain: using this module message owner or share users can decrypt and view all messages and non-sharing users cannot view or decrypt message.

While sharing data user may upload images also but Blockchain can store only text data not images so to store images we are using IPFS file server.

## **Results and description**

To run project first double click on 'Start\_IPFS.bat' to start IPFS file server and get below output

```
C:\Windows\system32\cmd.exe
E:\venkat\2021\August22\IdentityBasedBlockchain>ipfs init
initializing IPFS node at C:\Users\Admin\.ipfs
generating 2048-bit RSA keypair...done
peer identity: QmUmwAMgjFGZXPQqyEyLAKTN8w2QVytDqNJzpBuXscs656
to get started, enter:

  ipfs cat /ipfs/QmS4ustL54uo8FzR9455qaxZwuMiUhyvMcX9Ba8NUHUvVv/readme

E:\venkat\2021\August22\IdentityBasedBlockchain>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/QmUmwAMgjFGZXPQqyEyLAKTN8w2QVytDqNJzpBuXscs656
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
```

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

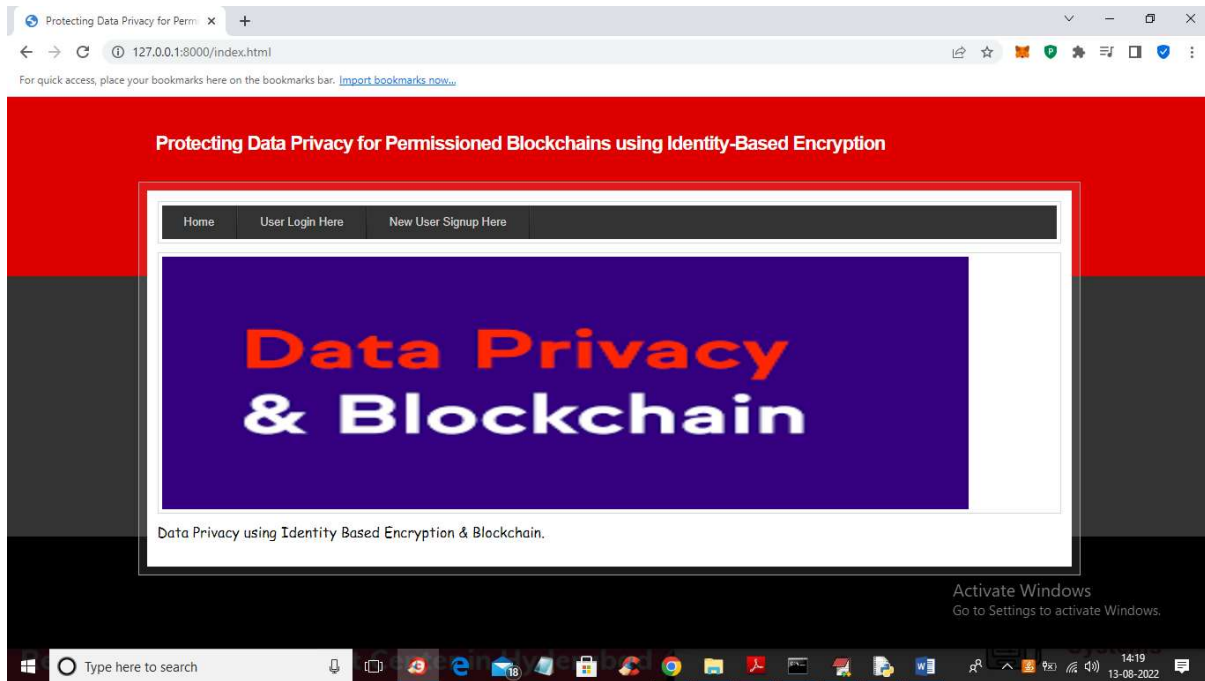
```
C:\Windows\system32\cmd.exe
E:\venkat\2021\August22\IdentityBasedBlockchain>python manage.py runserver
Performing system checks...

System check identified no issues (0 silenced).

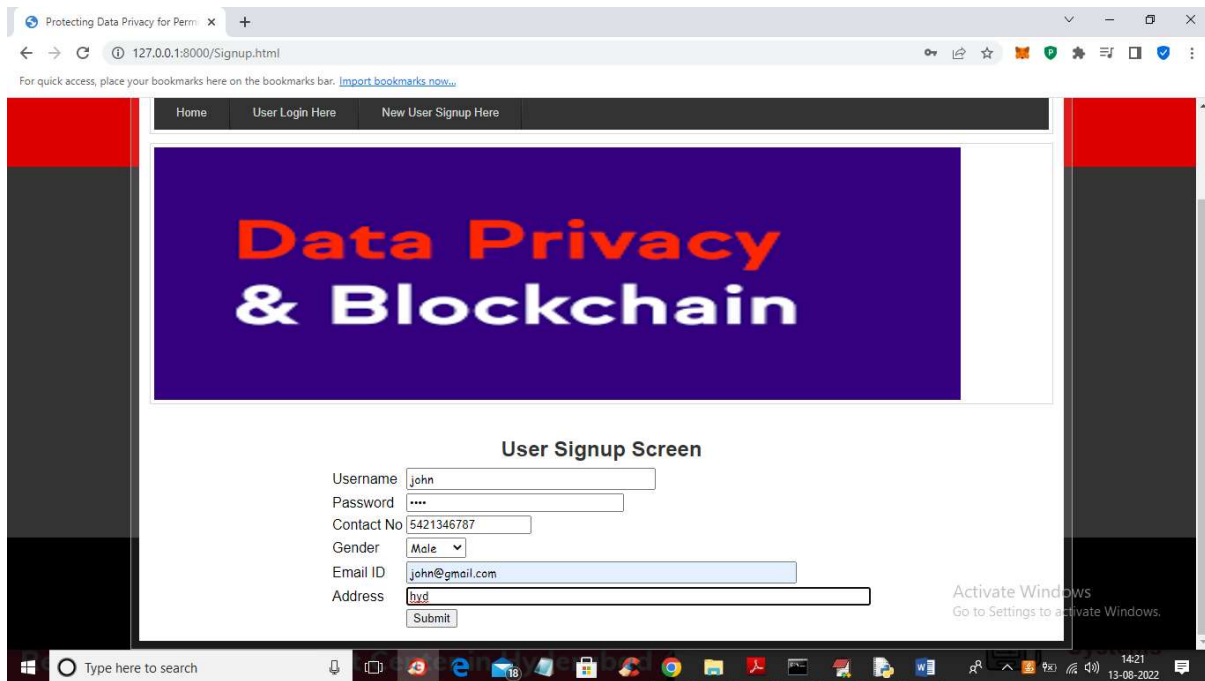
You have 15 unapplied migration(s). Your project may not work properly until you apply the migrations for app(s): admin, auth, contenttypes, sessions.
Run 'python manage.py migrate' to apply them.
August 13, 2022 - 14:18:36
Django version 2.1.7, using settings 'DataPrivacy.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

In above screen python 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 home page

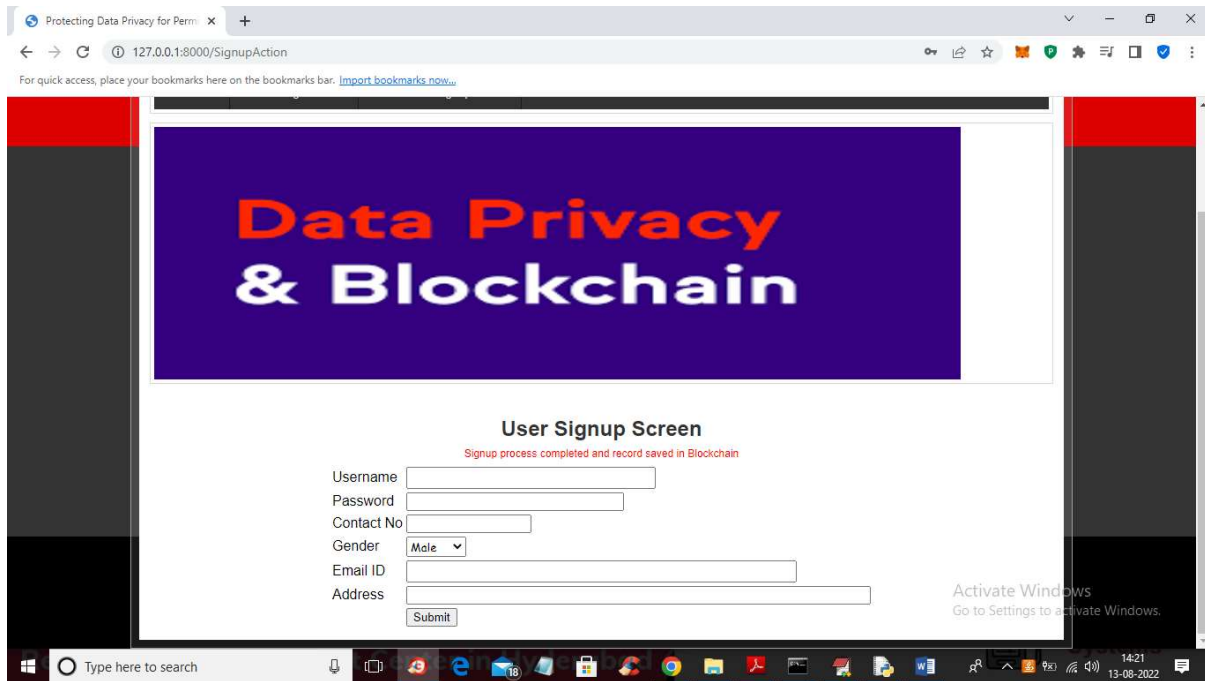




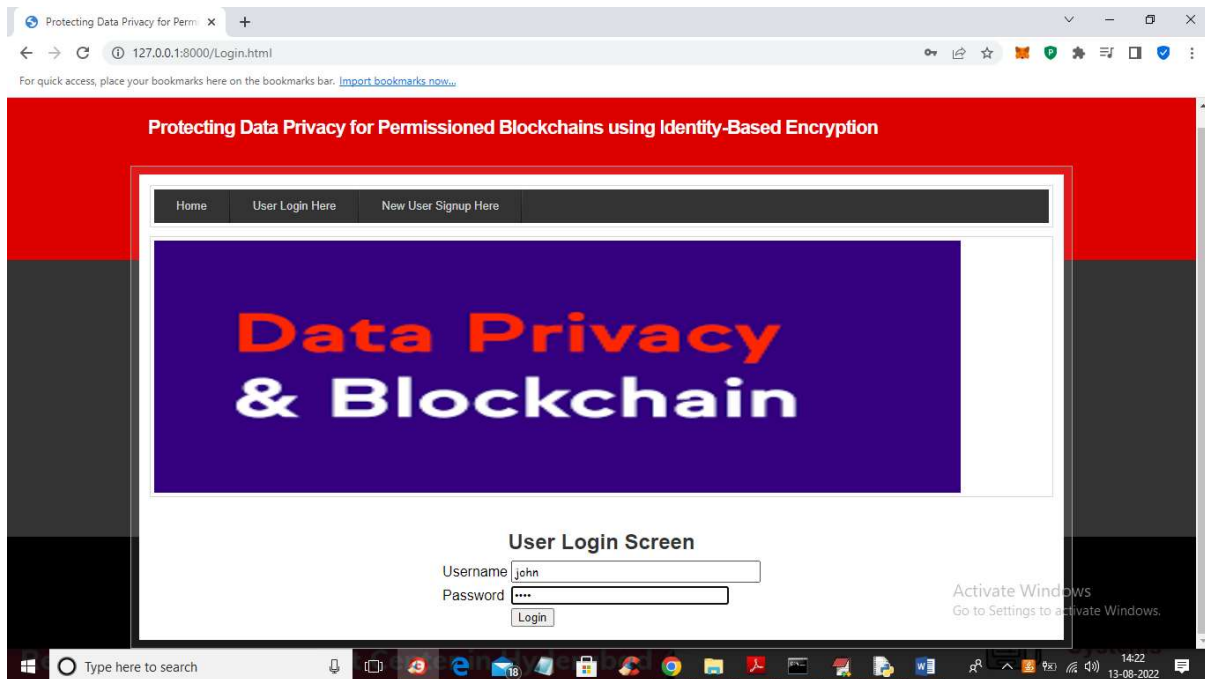
In above screen click on 'New User Signup Here' link to signup user



In above screen user is entering signup details and press submit button to store details in Blockchain and get below output

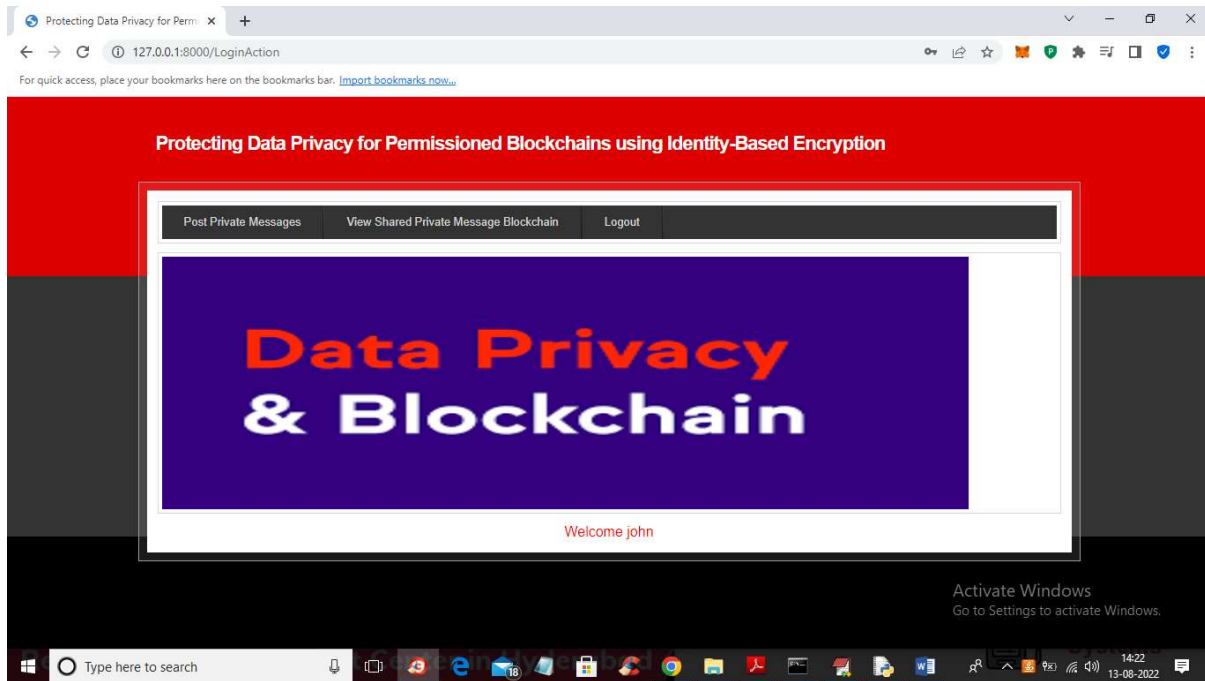


In above screen user signup completed and details saved in Blockchain and now click on ‘User Login’ link to get below screen

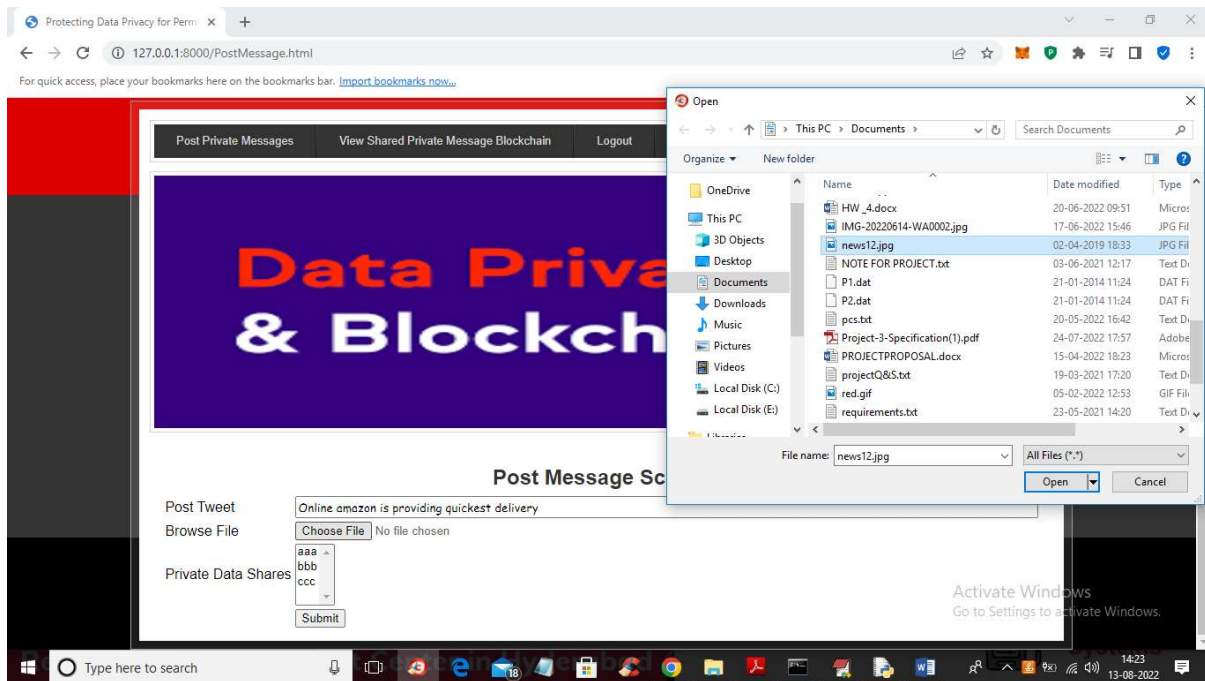


In above screen user is login and after login will get below screen





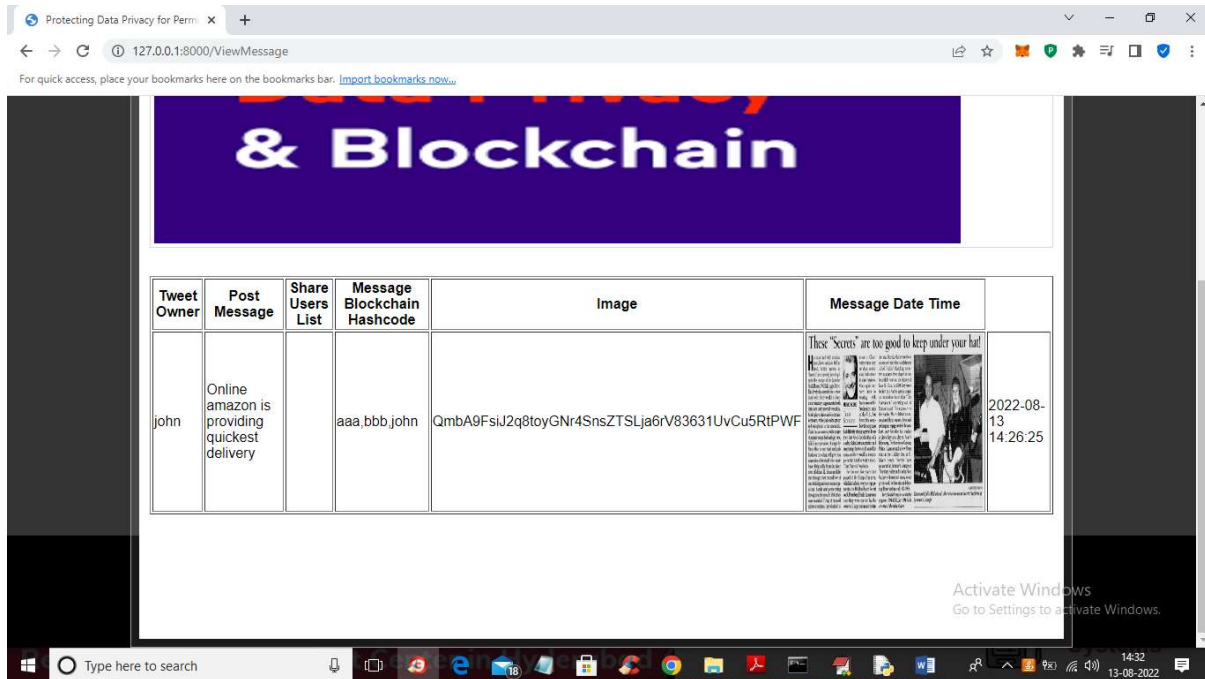
In above screen user can click on ‘Post Private Messages’ link to upload message




In above screen user type some message and then uploading image and then select list of users to share with and you can select multiple users by holding CTRL key like below screen

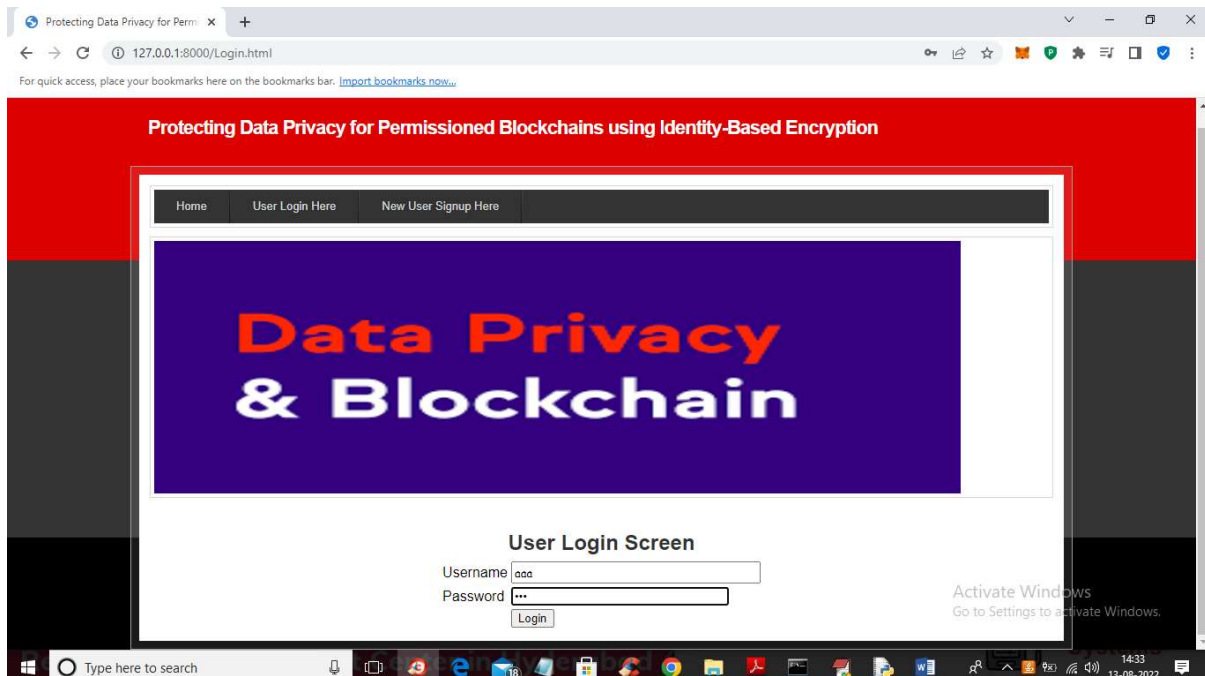


In above screen we can see message in red colour as POST MESSAGE saved in Blockchain and with Hashcode and we can see IBE encrypted message and now click on 'View Shared Private Message Blockchain' link to view message in decrypted format



Tweet Owner	Post Message	Share Users List	Message Blockchain Hashcode	Image	Message Date Time
john	Online amazon is providing quickest delivery	aaa,bbb,john	QmbA9FsiJ2q8toyGNr4SnsZTSLja6rV83631UvCu5RtPWF		2022-08-13 14:26:25

In above screen use can view decrypted message with image and hashcode and this user has shared post with user 'aaa' and now we login as 'aaa' and check message



Protecting Data Privacy for Permissioned Blockchains using Identity-Based Encryption

Home User Login Here New User Signup Here

# Data Privacy & Blockchain

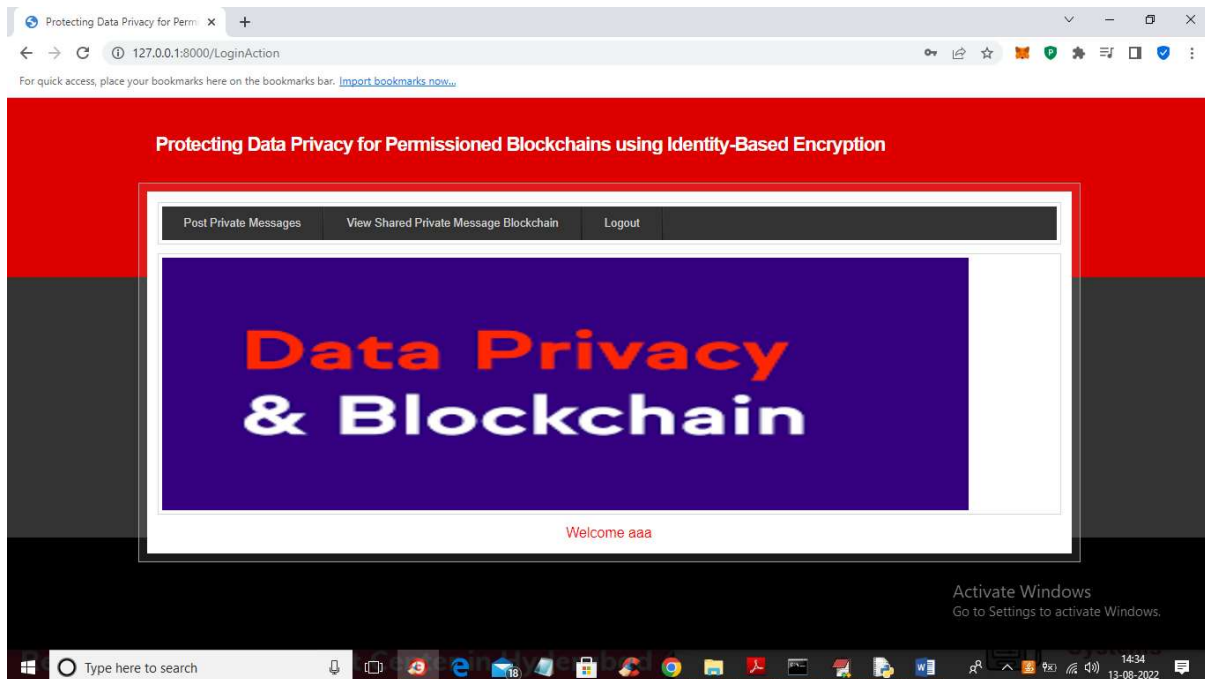
User Login Screen

Username: aaa

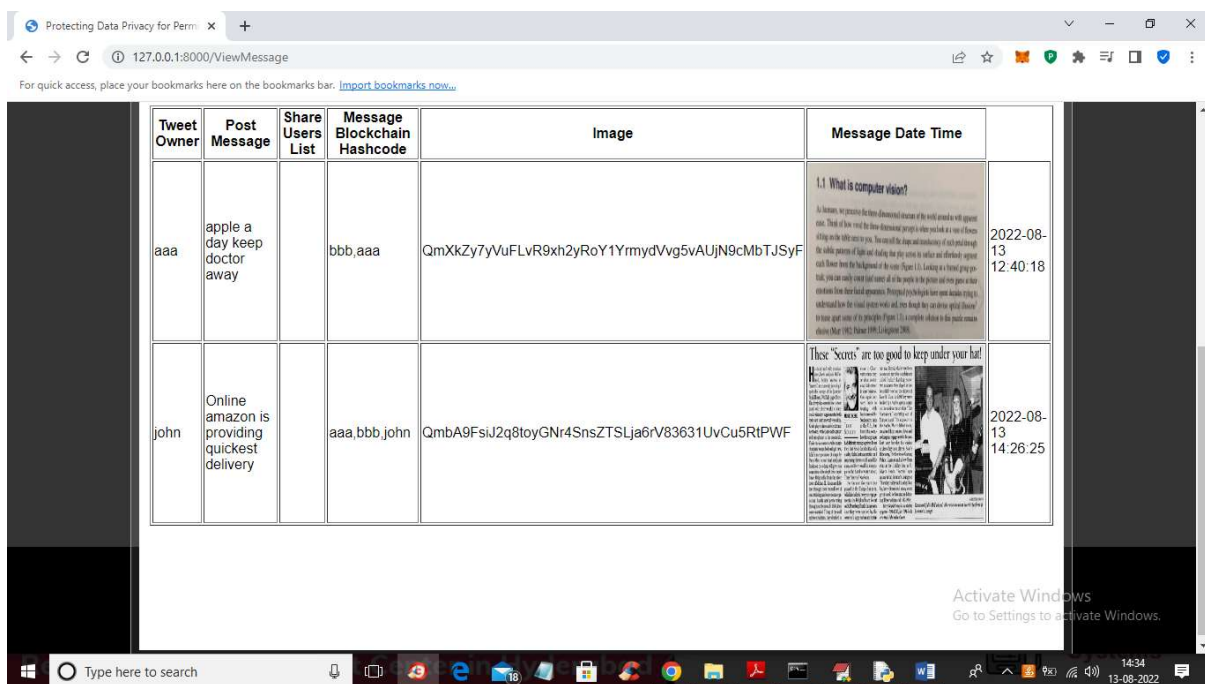
Password: ...

Login

In above screen user 'aaa' is login and after login will get below screen



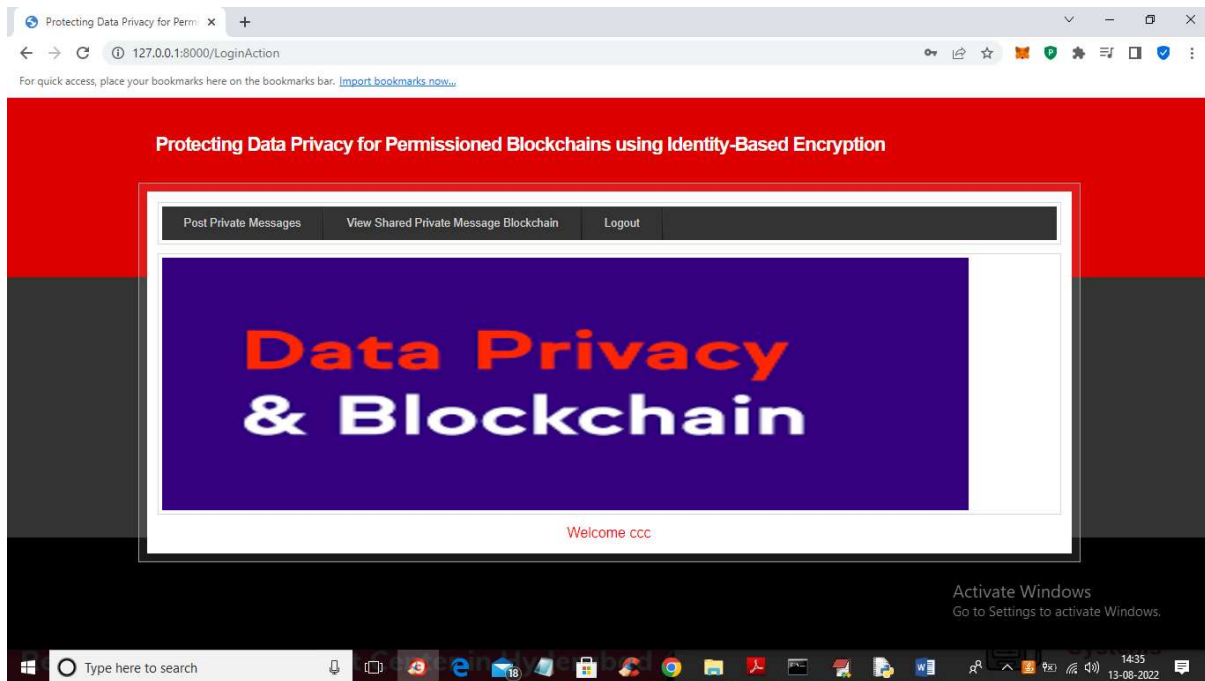
In above screen click on 'View Shared Message' link to get below output



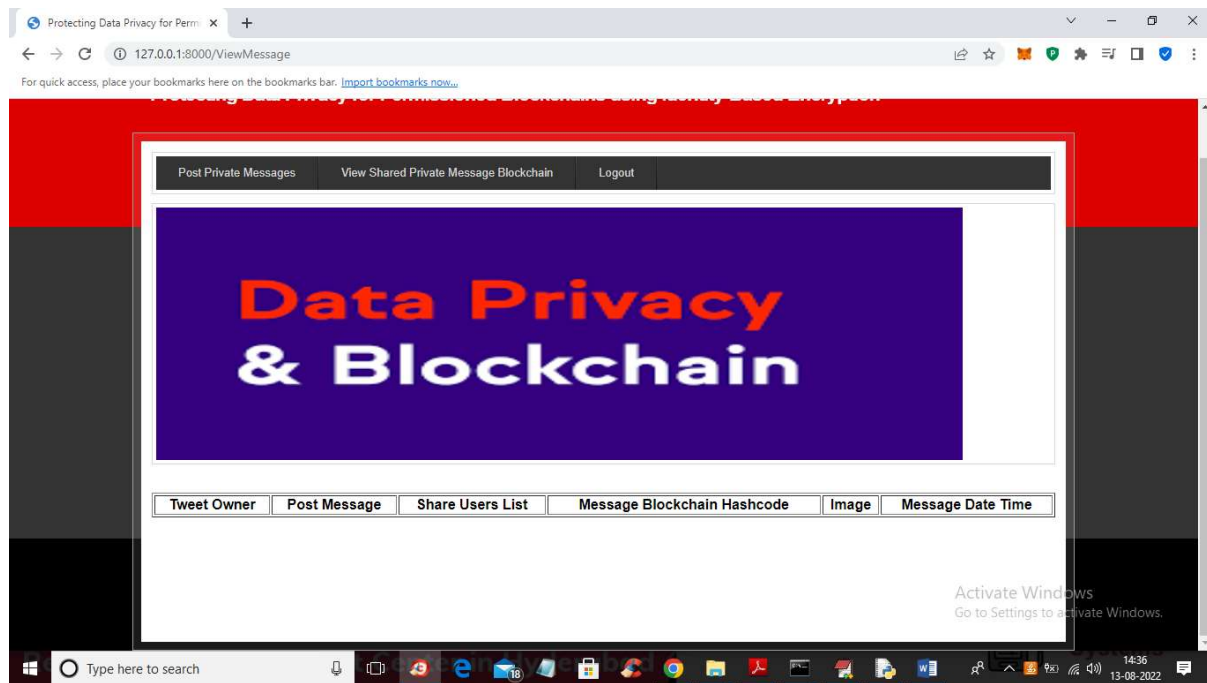
In above screen user aaa can view all his and shared messages and now we will login as user 'ccc' and check message as this user has no sharing permission



In above screen user 'ccc' is login and after login will get below output



In above screen user can click on 'View Shared Message' link to view messages



In above screen we can see user CCC has no share permission so he cannot decrypt and view messages and privacy will be achieved

## 11.CONCLUSION AND FUTURE SCOPE

We have proposed an improved delicately scheme on top of non-transactional cases in permissioned blockchain to improve the privacy. Our scheme can hide the information by encrypting the plaintext into the ciphertext, without using advanced technologies such as ring signature, homomorphic encryption and zero-knowledge proofs. Our scheme not only avoids the complicated certificate management and issuance in the traditional PKI system, but has a high security level which can prevent both disguise and passive attacks, and is functional, effective and practical for applications. This scheme provides an inspiring way to achieve delicate confidentiality of the transactions in many applications for non-transactional scenarios.

### **Future Scope**

The future of data privacy in permissioned blockchains using IBE holds substantial promise, with ongoing research likely to yield more efficient and secure revocable IBE schemes and threshold IBE implementations that enhance trust in decentralized settings. The development of standardized integration frameworks for popular permissioned blockchain platforms will be crucial for wider adoption, potentially leading to modular privacy solutions incorporating IBE. Hybrid approaches combining IBE with other Privacy-Enhancing Technologies like ZKPs and SMPC, alongside its use for securing off-chain data with on-chain access control, will offer more comprehensive privacy solutions. Specific industries such as healthcare, supply chain, and finance are poised to benefit significantly from IBE's ability to provide fine-grained, identity-based access control. Advancements in key management infrastructure, including the use of secure enclaves for PKG operations and decentralized key management services, will further bolster the security and practicality of IBE in blockchain environments. While challenges related to trust in key generation, implementation complexity, performance overhead, and regulatory compliance need to be addressed, the future trajectory indicates a significant role for IBE in bolstering data privacy within permissioned blockchain networks.



## REFERENCES

- [1] The Linux Foundation Helps Hyperledger Build the Most Vibrant Open Source Ecosystem for Blockchain. <http://www.linuxfoundation.org/>.
- [2] S. Omohundro. Cryptocurrencies, smart contracts, and artificial intelligence. *AI Matters*, 1(2):19C21, Dec. 2014.
- [3] D. D. Detwiler. One nations move to increase food safety with blockchain. <https://www.ibm.com/blogs/blockchain/2018/02/one-nationsmove-to-increase-food-safety-with-blockchain/>, 2018. [Online; accessed 1-May-2018].
- [4] Shamir, A. Identity-based cryptosystems and signature schemes. In: Blakely, G.R., Chaum, D. (eds.) *CRYPTO 1984*. LNCS, vol. 196, pp. 47C53. Springer, Heidelberg (1985)
- [5] Boneh, D., Franklin, M. Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) *CRYPTO 2001*. LNCS, vol. 2139, pp. 213C229. Springer, Berlin, Germany (2001)
- [6] Boneh, D., Boyen, X. Efficient selective-ID secure identity based encryption without random oracles. In: Cachin, C., Camenisch, J.L. (eds.) *EUROCRYPT 2004*. LNCS, vol. 3027, pp. 223C238. Springer, Berlin, Germany (2004)
- [7] Boneh, D., Boyen, X. Secure identity based encryption without random oracles. In: Franklin, M. (ed.) *CRYPTO 2004*. LNCS, vol. 3152, Springer, Berlin, Germany (2004).
- [8] Gentry, C. Practical identity-based encryption without random oracles. In: Vaudenay, S. (ed.) *EUROCRYPT 2006*. LNCS, vol. 4004, pp. 445C464. Springer, Berlin, Germany (2006).
- [9] Labs, Shen Noether Mrl. Ring confidential transactions. 2016.
- [10] R. S. Wahby, I. Tzialla, A. Shelat, J. Thaler and M. Walfish. DoublyEfficient zkSNARKs Without Trusted Setup. 2018 IEEE Symposium on Security and Privacy (SP), San Francisco, CA, 2018, pp. 926-943.
- [11] B. Bnz J. Bootle D. Boneh A. Poelstra P. Wuille G. Maxwell. Bulletproofs: Efficient range proofs for confidential transactions”, IEEE S&P May 2018.



- [12] A. Chiesa E. Tromer M. Virza. Cluster computing in zero knowledge, EUROCRYPT Apr. 2015.
- [13] A. Chiesa M. A. Forbes N. Spooner. A zero knowledge sumcheck and its applications. CoRR abs1704.02086 2017.
- [14] T. P. Pedersen et al. Non-interactive and information-theoretic secure verifiable secret sharing. in Crypto, vol. 91, pp. 129C140, Springer, 1991.
- [15] P. Paillier et al. Public-key cryptosystems based on composite degree residuosity classes. in Eurocrypt, vol. 99, pp. 223C238, Springer, 1999.