A MAIN PROJECT REPORT ON

Blockchain for Enhancing Trust and Privacy in Electronic Know Your Customer

Submitted in partial fulfilment for the award of the degree of

BACHELOR OF TECHNOLOGY

In

Computer Science and Technology

By

G.SRI PRABHU KIRAN (21A81A0612)

K.SAI PRASANTH (21A81A0623)

P.SRINIVASARAO (21A81A0654)

L.T.SRI CHANDRAHAS (21A81A0634)

Sk.SAJID (22A85A0606)

Under the Esteemed Supervision of

Mrs. A NAGA JYOTHI, M.Tech, Asst.Professor

Department of Computer Science and Technology(Accredited by N.B.A.)

SRI VASAVI ENGINEERING COLLEGE(Autonomous)

(Affiliated to JNTUK, Kakinada)

Pedatadepalli, Tadepalligudem-534101, A.P

2024-25

SRI VASAVI ENGINEERING COLLEGE (Autonomous)

Department of Computer Science and Technology

Pedatadepalli, Tadepalligudem

Certificate

This is to certify that the Project Report entitled **"Blockchain for Enhancing Trust and Privacy in Electronic Know Your Customer"** submitted by **G.Sri Prabhu Kiran (21A81A0612), K.Sai Prasanth(21A81A0623),P.Srinivasarao (21A81A0654),L.T.Sri Chandrahas (21A81A0634), Sk.Sajid (22A85A0606)** for the award of the degree of Bachelor of Technology in the Department of Computer Science and Technology during the academic year 2024-2025.

Name of Project Guide

Ms.A.Naga Jyothi M.Tech,

Asst.Professor

Head of the Department

Dr.D. Jaya kumari M.Tech, Ph.D

Professor & HOD

External Examiner

DECLARATION

We hereby declare that the project report entitled "**Blockchain for Enhancing Trust and Privacy in Electronic Know Your Customer**" submitted by us to **Sri Vasavi Engineering College(Autonomous)**, **Tadepalligudem**, affiliated to JNTUK Kakinada in partial fulfilment of the requirement for the award of the degree of B.Tech in Computer Science and Technology is a record of Bonafide project work carried out by us under the guidance of **Ms.A NAGA JYOTHI, M.Tech, Asst. Professor.** We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other degree in this institute or any other institute or University.

Project Associates

G.SRI PRABHU KIRAN (21A81A0612)

K.SAI PRASANTH (21A81A0623)

P.SRINIVASARAO (21A81A0654)

L.T.SRICHANDRAHAS (21A81A0634)

Sk.SAJID (22A85A0606)

ACKNOWLEDGEMENT

First and foremost, we sincerely salute to our esteemed institute **SRI VASAVI ENGINEERING COLLEGE,** for giving us this golden opportunity to fulfill our warm dream to become an engineer.

Our sincere gratitude to our project guide **A.Naga Jyothi, M.Tech, Assistant Professor**, Department of Computer Science and Engineering**,** for her timely cooperation and valuable suggestions while carrying out this project.

We express our sincere thanks and heartful gratitude to **Dr. D. Jaya Kumari**, Professor & Head of the Department of Computer Science and Engineering, for permitting us to do our project.

We express our sincere thanks and heartful gratitude to **Dr. G.V.N.S.R. Ratnakara Rao**, Principal, for providing a favourable environment and supporting us during the development of this project.

Our special thanks to the management and all the teaching and non-teaching staff members, Department of Computer Science and Technology, for their support and cooperation in various ways during our project work. It is our pleasure to acknowledge the help of all those respected individuals.

We would like to express our gratitude to our parents, friends who helped to complete this project.

Project Associates

G.SRI PRABHU KIRAN (21A81A0612)

K.SAI PRASANTH (21A81A0623)

P.SRINIVASARAO (21A81A0654)

L.T.SRI CHANDRAHAS (21A81A0634)

Sk.SAJID (22A85A0606)

ABSTRACT

The e-KYC Trust Block is a blockchain-based electronic Know Your Customer (e-KYC) system designed to enhance security, privacy, and trust in digital identity verification. Traditional e-KYC solutions often face challenges related to centralized storage, complex key management, and data vulnerabilities. To overcome these issues, this system integrates Ciphertext-Policy Attribute-Based Encryption (CP-ABE) for fine-grained access control, and utilizes the Interplanetary File System (IPFS) for secure, decentralized document storage. Smart contracts are employed to enforce user consent and ensure auditability. By combining symmetric and public-key encryption, the system minimizes communication overhead while maintaining data confidentiality. Experimental results demonstrate that the proposed system is efficient, scalable, and secure, making it a robust solution for privacy-preserving e-KYC processes.

INDEX

CHAPTER-1

INTRODUCTION

1.1 Introduction :

Electronic-Know Your Customer (e-KYC) is a service that banks or financial institutions (FIs) provide for authentication and identity verification electronically. It enhances cost efficiency and customer satisfaction. The e-KYC system enables FIs to verify customer identities and retrieve KYC data for individuals and corporations. Financial institutions can implement e-KYC using off-the-shelf software or by developing their own systems, deploying them either on-premise or in the cloud. Due to the growing trend of outsourcing, most enterprises prefer cloud-based solutions.

Cloud-based e-KYC systems offer efficient and flexible authentication compared to host-based systems, which cause traffic bottlenecks and single points of failure. However, security and privacy remain concerns, as customer data stored on the cloud might be accessed by public tenants or cloud service providers (CSPs). To address this, banks and FIs implement encryption alongside CSP authentication features. Before uploading, e-KYC data is encrypted, and decryption keys must be securely managed. Secure key sharing, revocation, and re-generation are critical yet underexplored challenges in cloud-based e-KYC.

Blockchain technology has gained traction across industries, including banking and finance, for enhancing transparency, trustworthiness, and cost efficiency. A blockchain-based e-KYC system can decentralize authentication, improving traceability and security. Smart contracts automate processes, enhancing usability and programmability. Despite advancements, existing blockchain-based KYC solutions lack strong non-repudiation mechanisms for client consent, overlook privacy in smart contract transactions, and provide limited customer access to update credentials.

To address these gaps, this paper proposes a blockchain-based e-KYC system integrating cloud Interplanetary File System (IPFS) and cryptographic protocols. A smart contract ensures digitally signed user consent stored immutably on the blockchain. Symmetric encryption with public key encryption secures credential files, while cipher text policy attribute-based encryption (CP-ABE) protects transactions. CP-ABE enables fine-grained access control, allowing multiple FIs to access encrypted transactional data based on predefined policies. A policy update algorithm ensures efficient re-encryption, while a synchronization mechanism updates e-KYC data across participating banks and FIs.

1.2 Objective :

The primary objective of a **Blockchain-Based e-KYC System for Secure Identity Verification** is to develop a decentralized and secure KYC platform that ensures user data protection, enables efficient identity verification, and maintains data integrity and confidentiality while addressing key challenges in traditional e-KYC processes.

- **Enhance Identity Verification Security** – Implement blockchain-based authentication and cryptographic techniques to prevent unauthorized access and fraud.
- **Ensure Data Privacy and Confidentiality** – Encrypt customer identity data before uploading it to the cloud, preventing unauthorized access by third parties, including cloud service providers.
- **Enable Secure and Transparent Data Sharing** – Utilize smart contracts and cipher text policy attribute-based encryption (CP-ABE) to enforce controlled access to encrypted KYC data across financial institutions.
- **Facilitate Customer Consent Management** – Develop a smart contract mechanism to digitally sign and store user consent immutably on the blockchain, ensuring non-repudiation and compliance with privacy regulations.
- **Improve System Traceability and Auditing** – Leverage blockchain's immutable ledger to maintain a transparent and tamper-proof record of KYC transactions for auditing and regulatory compliance.
- **Optimize Key Management and Access Control** – Implement a secure cryptographic protocol for key sharing, revocation, and re-generation to enable fine-grained access control in multi-bank environments.
- **Allow Seamless Data Updates and Synchronization** – Enable customers to update their KYC credentials securely, with changes broadcasted and synchronized across financial institutions via blockchain.
- **Enhance Efficiency and Scalability** – Design a high-performance, scalable system for fast and secure identity verification while minimizing operational costs and complexities.

CHAPTER-2

LITERATURE SURVEY

1.Federated Learning for Privacy-Preserving KYC (2024)

This paper explores the use of federated learning to enable privacy-preserving KYC processes without centralizing sensitive data. It leverages differential privacy techniques to ensure data protection while allowing financial institutions to verify identities securely. The proposed method enhances privacy and security in KYC processes while reducing risks associated with centralized data storage.

2. Enhancing KYC with Behavioural Biometrics in Cloud Environments (2024)

This paper investigates the use of behavioural biometrics and machine learning to strengthen KYC verification. By analysing user behaviour patterns, the system provides an additional layer of security for identity verification, particularly for mobile users. The approach ensures data owner control while improving authentication accuracy and reducing fraud risks in cloud-based KYC solutions.

3.Enhancing Privacy in e-KYC Systems Using Zero-Knowledge Proofs (2023)

This study demonstrates how zero-knowledge proofs (ZKPs) can improve privacy in e-KYC systems by allowing identity verification without revealing sensitive data. It employs data anonymization techniques to enhance security and protect users' personal information. The proposed solution effectively reduces the risk of identity theft while maintaining compliance with privacy regulations.

4.A Blockchain-Based Approach for Secure and Efficient Verification (2022)

This paper proposes a blockchain-based system for secure and efficient identity verification. It utilizes smart contracts and data encryption to ensure transparency and prevent fraudulent activities. By decentralizing the verification process, the system improves security, enhances trust, and eliminates reliance on third-party intermediaries.

5. Blockchain Technology for Secure and Transparent e-Governance Systems (2021)

This research explores the potential of blockchain in e-governance for secure and transparent identity verification. By leveraging blockchain’s decentralized nature, the system enhances security and reduces the risks associated with centralized identity management. The proposed approach strengthens transparency and trust in e-governance systems.

6. A Privacy-Preserving e-KYC System (2020)

This paper presents a privacy-preserving e-KYC system that employs cryptographic techniques, including zero-knowledge proofs, to protect user data during identity verification.

CHAPTER-3

SYSTEM STUDY AND ANALYSIS

3.1 Problem Statement :

Traditional e-KYC (electronic Know Your Customer) systems face significant security and privacy challenges, particularly in handling sensitive user data. Existing blockchain-based e-KYC solutions primarily focus on identity verification and document management but fail to ensure **privacy preservation and efficient key management**. Many approaches rely on smart contracts and distributed ledgers but lack robust encryption mechanisms, making them vulnerable to data exposure. Furthermore, the absence of fine-grained access control and user consent mechanisms results in security risks, as unauthorized entities may gain access to sensitive credentials. Additionally, existing systems do not provide adequate **traceability and auditability** of cryptographic operations, leading to potential misuse of KYC data.

This work proposes a **Privacy-Preserving e-KYC Framework** leveraging **Ciphertext-Policy Attribute-Based Encryption (CP-ABE)** and blockchain technology. By integrating **fine-grained access control, dynamic key management, and user consent-based data sharing**, this approach ensures that sensitive KYC credentials remain secure while granting access only to authorized entities. The system enhances **data privacy, integrity, and auditability** by recording access policies and cryptographic operations on a blockchain ledger. Furthermore, it introduces **traceability features** to detect and mitigate unauthorized key abuses, ensuring a more **secure and privacy-compliant** e-KYC process.

3.2 Existing System :

Existing blockchain-based e-KYC systems focus primarily on identity verification and credential management but fail to ensure **privacy preservation and efficient key management**. Many current solutions rely on **smart contracts and permissioned blockchains** to facilitate document sharing and verification, yet they lack robust encryption mechanisms to **protect sensitive KYC data**. While some approaches use **Hyperledger Fabric** for decentralized KYC storage, they do not provide strong privacy guarantees, as data remains accessible to multiple entities within the network. Additionally, solutions based on **Interplanetary File System (IPFS) and blockchain** store KYC documents off-chain, but without encryption, making them susceptible to unauthorized access and tampering

Attribute-Based Encryption (ABE) has been explored to enhance **data security and access control**, yet traditional ABE schemes introduce **high computational overhead**, making them inefficient for large-scale e-KYC implementations. Furthermore,

existing systems **lack real-time user revocation**, posing risks when a compromised identity needs immediate deactivation. While some models integrate **traceable attribute-based encryption (TABE-DAC)** to track key abuses, they still require **complex policy updates** that reduce efficiency. The absence of a **privacy-preserving framework** that ensures fine-grained access control, efficient encryption, and secure key management highlights the need for a **more secure and scalable e-KYC solution**.

3.3 Limitations of the Existing System :

Lack of Privacy-Preserving Encryption: Existing blockchain-based e-KYC systems do not implement robust privacy-preserving techniques, leaving sensitive user data vulnerable. Many solutions store KYC documents on-chain or in external repositories like IPFS without proper encryption, making them susceptible to unauthorized access and data breaches.

Inefficient Access Control: Current approaches rely on static permission models that lack fine-grained access control. Users have limited control over how their KYC data is shared among financial institutions, and access policies cannot be dynamically updated, leading to inefficient identity verification processes.

High Computational Overhead: Many solutions integrate Attribute-Based Encryption (ABE) and Zero-Knowledge Proofs (ZKP) to enhance security, but these techniques introduce heavy computational requirements, making them impractical for real-time identity verification, especially for resource-constrained devices.

Lack of Immediate User Revocation: When a user's identity or KYC credentials need to be revoked or updated, existing systems fail to enforce immediate access restrictions. This delayed revocation increases the risk of unauthorized access and fraudulent activities, as compromised credentials may remain valid for extended periods**.**

3.4 Proposed System :

The proposed **Privacy-Preserving e-KYC Framework using Blockchain** enhances security, efficiency, and user control in digital identity verification. It integrates **Zero-Knowledge Proofs (ZKP)** and **Ciphertext-Policy Attribute-Based Encryption (CP-ABE)** to enable privacy-preserving authentication while ensuring fine-grained access control.

Unlike traditional blockchain-based KYC models, this system allows **users to encrypt their KYC credentials** before storage, preventing unauthorized access by intermediaries or

financial institutions.The framework supports **efficient identity verification** through **decentralized access control mechanisms** while enabling **immediate user revocation** to mitigate security risks. **Blockchain-integrated key management** ensures tamper-proof identity validation without relying on a central authority. By leveraging smart contracts for access permissions and revocation, this system enhances transparency, reduces computational overhead, and ensures scalable, privacy-preserving KYC verification.

3.5 Advantages of the Proposed System :

- End-to-End Client-Side Encryption
- Fine-Grained Access Control
- Immediate User Revocation
- Offline Encryption Support
- Outsourced Decryption for Performance Optimization
- Cross-Platform and Plugin-Free
- Enhanced Security Model

3.6 Functional Requirements :

- Users must register and log in securely with role-based access control to restrict operations based on user roles.
- Data must be encrypted before uploading and decrypted only by authorized users using end-to-end client-side encryption.
- Cipher text-Policy Attribute-Based Encryption (CP-ABE) should be implemented to enable fine-grained access control for secure data sharing.
- Immediate user revocation must be supported to prevent revoked users from decrypting previously shared data.
- Offline encryption should allow users to encrypt data without an active internet connection and synchronize once online.

- Outsourced decryption should offload complex tasks to a secure cloud server while ensuring the final decryption step occurs on the client side.
- The system must be cross-platform and plugin-free, ensuring compatibility with web browsers, desktops, and mobile devices.
- Audit logs and security monitoring should track file access, encryption, decryption, and user activity to detect unauthorized access attempts.

3.7 Non-Functional Requirements :

- Security and Data Protection
- Scalability and Performance Optimization
- High Availability and Reliability
- Usability and User Experience
- Cross-Platform Compatibility
- Compliance and Regulatory Standards

3.8 System Requirements :

3.8.1 Software Requirements :

- Operating system - Windows XP
- Coding Language - Java/J2EE(JSP, Servlet)
- Front End - J2EE
- Back End - MySQL

3.8.2 Hardware Requirements :

- Processor - Pentium-IV
- RAM - 4GB(min)
- Hard Disk - 20GB

CHAPTER-4

SYSTEM DESIGN

4.1System Architecture :

System architecture is a conceptual model that defines a system's structure, behaviour, and interactions. It outlines the relationships between hardware, software, data flow, and communication mechanisms. A well-designed architecture ensures scalability, reliability, and efficiency. It serves as a blueprint for developers to build and maintain complex systems. System architecture can be represented using diagrams and models for better visualization.

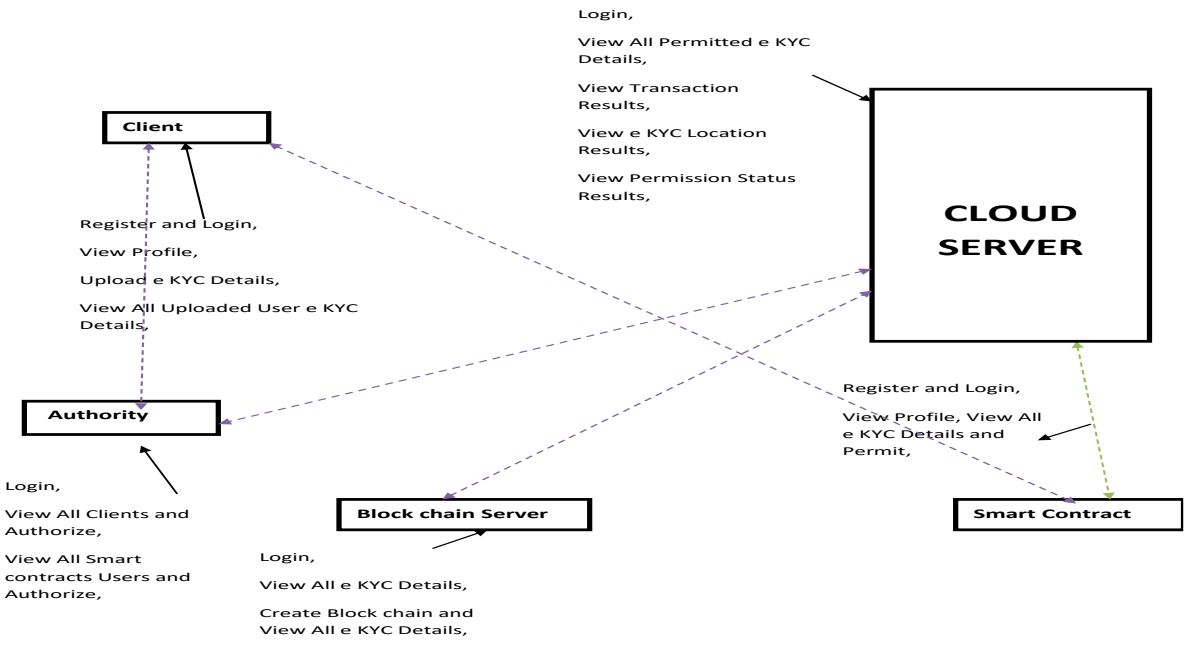


Figure 4.1.1 : System Architecture

4.2UML Diagrams :

UML (Unified Modeling Language) is a standardized method for visually representing system architecture and design. It serves as a blueprint for modeling complex and large-scale software systems effectively. By using graphical notations, UML simplifies the representation of object-oriented software structures and behaviours. It incorporates best engineering practices to enhance clarity and maintainability in software development. UML diagrams help developers, designers, and stakeholders understand system components and their interactions. Overall, UML plays a crucial role in streamlining the software development process.

4.2.1 Use Case Diagram:

A **Use Case Diagram** represents the functionality of a system by depicting the interactions between users (actors) and the system. It focuses on the system's behaviour from an external perspective, showing how users interact with different system functionalities. **Actors** are external entities, such as users or other systems, that communicate with the system to perform specific tasks. **Use cases** define the various operations or services the system provides. These diagrams help in understanding system requirements, improving communication between stakeholders, and ensuring a clear overview of system behaviour.

Use Case :

A **use case** represents a specific functionality or interaction within a system that provides value to an actor. It defines how the system responds to user inputs and describes the sequence of actions performed. Use cases help in identifying system requirements and ensuring a clear understanding of its functionalities.

Actor :

An **actor** is an external entity, such as a user, device, or another system, that interacts with the system to achieve a goal. Actors can be **primary** (initiating an interaction) or **secondary** (assisting in the process). They play a crucial role in defining system boundaries and functionalities.

Associations :

An **association** in a use case diagram represents the relationship between an actor and a use case. It indicates the interaction between the actor and the system's functionality. Associations are typically shown as solid lines connecting actors to their respective use cases in the diagram.

Relationships :

In a **Use Case Diagram**, relationships define interactions between actors and use cases. **Association** represents a direct link between an actor and a use case, while **include** indicates mandatory behaviour shared across multiple use cases. **Extend** represents optional functionality triggered under specific conditions, and **generalization** shows inheritance between actors or use cases to represent shared behaviours.

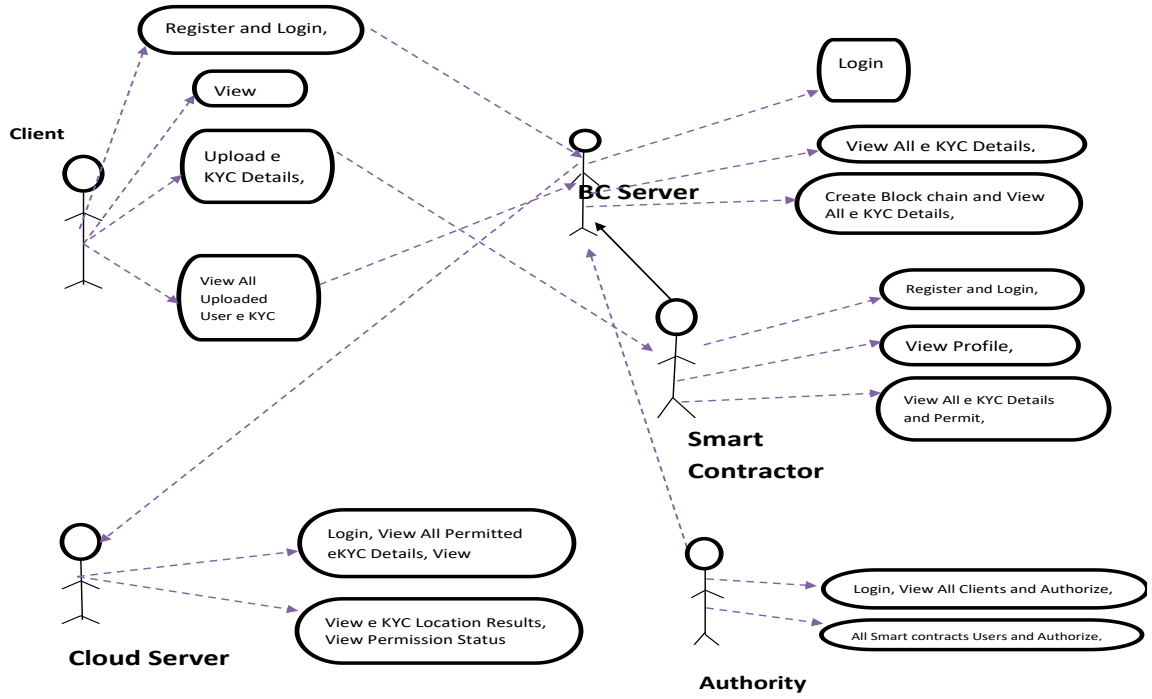


Figure 4.2.1.1 : Use Case Diagram

4.2.2 Class Diagram :

A **Class Diagram** is a UML diagram that represents the structure of a system by defining its classes, attributes, methods, and relationships. It helps in visualizing the object-oriented architecture and understanding how different components interact. The diagram includes **classes** with their properties and behaviours, along with relationships like **association, inheritance, aggregation, and composition**. Class diagrams are essential for designing, analysing, and documenting software systems. They serve as a blueprint for developers, ensuring a clear and organized system structure before implementation.

Classes :

A **class** is the fundamental building block of a class diagram, representing an entity in the system. It consists of a **class name, attributes (properties), and methods (functions)** that define its behaviour. Classes serve as templates for creating objects and help structure the system's functionality.

Attributes :

Attributes are the properties or characteristics of a class that define its state. They represent data stored within an object, such as a "User" class having attributes like "name" and "id".

Each attribute has a specific **data type** (e.g., String, int, boolean) that determines the kind of values it can hold. Attributes help in differentiating objects and play a

crucial role in data storage and manipulation. They are usually declared as **private** and accessed using getter and setter methods for encapsulation.

Methods :

Methods are the operations or behaviours that a class can perform. They define the actions that objects of the class can execute, such as a "User" class having methods like "Register()" and "Login()." Methods encapsulate functionality and allow interaction with the class.

Relationships :

Relationships in a class diagram show how different classes interact with each other. The main types include **association (simple connection), inheritance (generalization), aggregation (whole-part relationship), and composition (strong dependency).** These relationships define how data flows and how classes collaborate in the system.

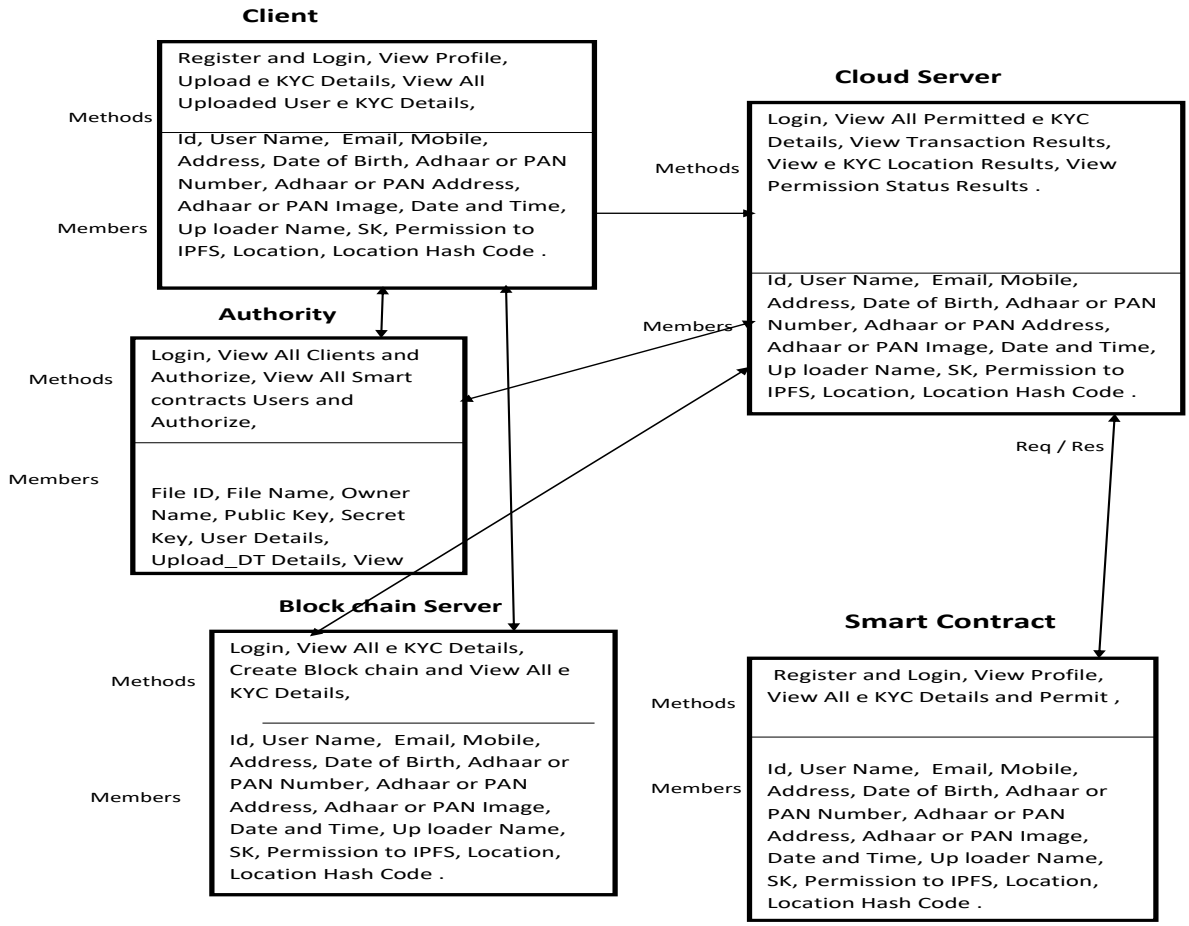


Figure 4.2.2.1 : Class Diagram

4.2.3 Communication Diagram :

A **Communication Diagram** in UML represents the interaction between objects in a system, focusing on the flow of messages. It shows how objects are linked and communicate to achieve specific functionality. The diagram consists of **objects (instances of classes), associations (connections between objects), and messages (interactions between objects)**. Messages are **sequentially numbered** to indicate the order of execution in a particular scenario. Unlike sequence diagrams, which emphasize the timeline, communication diagrams emphasize **object relationships**. They help in understanding **how components interact** within a system. These diagrams are useful for analyzing system behavior and improving design clarity.

Objects :

Objects represent instances of classes that participate in communication within the system. Each object is depicted as a box with its name, such as "Customer" or "Order System." These objects interact with each other by exchanging messages to perform specific tasks.

Messages :

Messages indicate the flow of information between objects, showing how they interact. Each message is **numbered sequentially** to depict the order of execution in a scenario. Messages are represented with arrows and often include method names or actions performed by objects.

Associations :

Associations define the links or connections between objects in a communication diagram. They represent relationships that enable objects to send and receive messages. These links help in visualizing the structure of communication and how objects collaborate in a system.

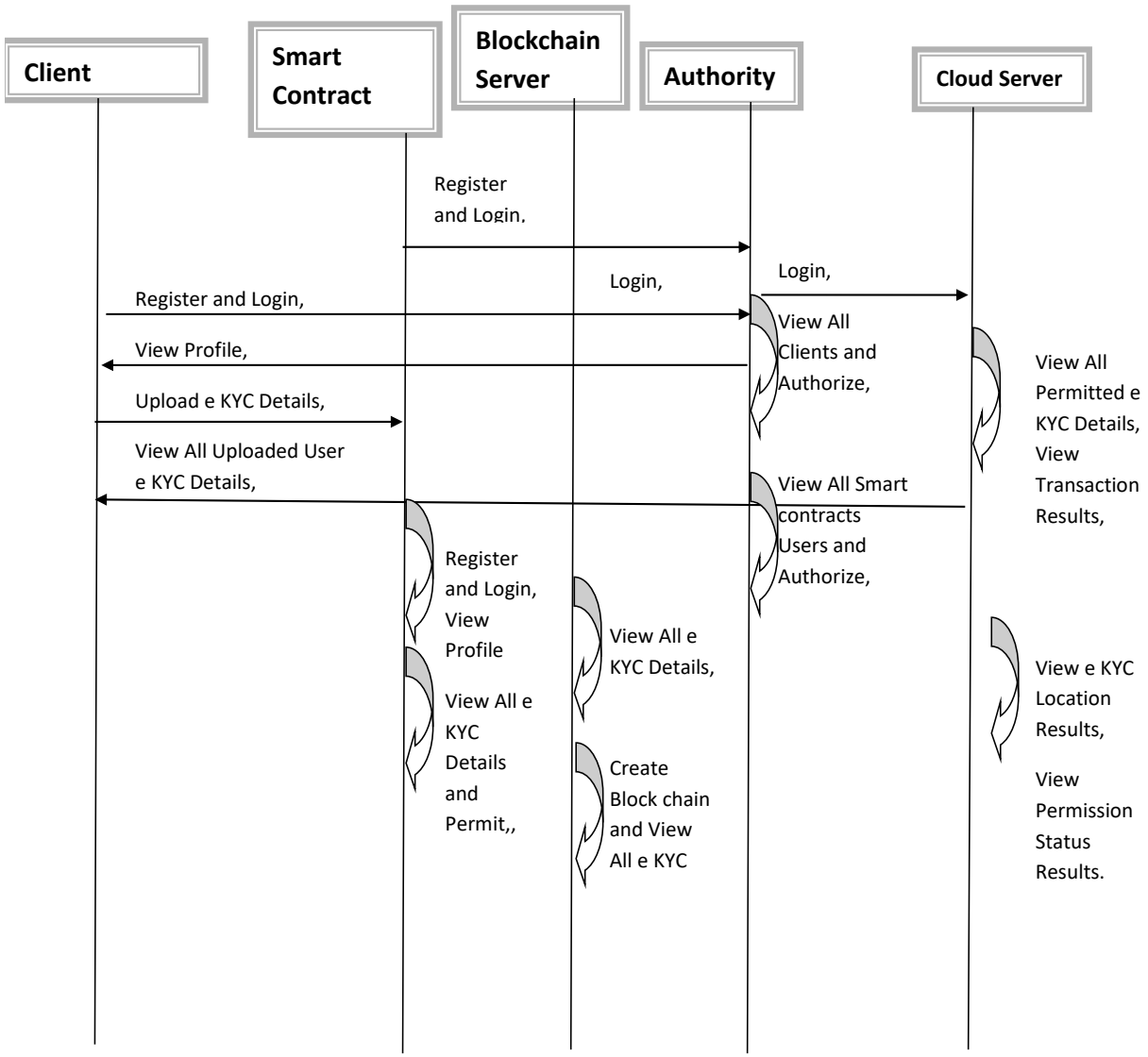


Figure 4.2.3.1 : Communication Diagram

4.2.4 Flow chart Diagram :

A **Flowchart Diagram** is a graphical representation of a process, algorithm, or workflow using different symbols and arrows. It helps in **visualizing the logical flow** of a system, making it easier to understand and analyse. Flowcharts use **standard symbols** like ovals for start/end points, rectangles for processes, diamonds for decisions, and arrows to show direction. They are widely used in **software development, business processes, and engineering** to document workflows. Flowcharts improve **problem-solving and debugging** by clearly outlining steps. They provide a **structured approach** to process execution, making complex logic easier to follow.

Process :

A **rectangle** represents a **process, operation, or action** in the flowchart. It defines tasks such as calculations, data processing, or function execution. This is the most commonly used symbol in a flowchart.

Decision :

A **diamond shape** represents a **decision point** where the process flow branches based on a condition. It usually has **two outgoing arrows** labeled **Yes/No** or **True/False**.

Arrows :

Arrows indicate the **direction of process flow** in a flowchart. They connect different symbols and guide the sequence of operations from one step to another.

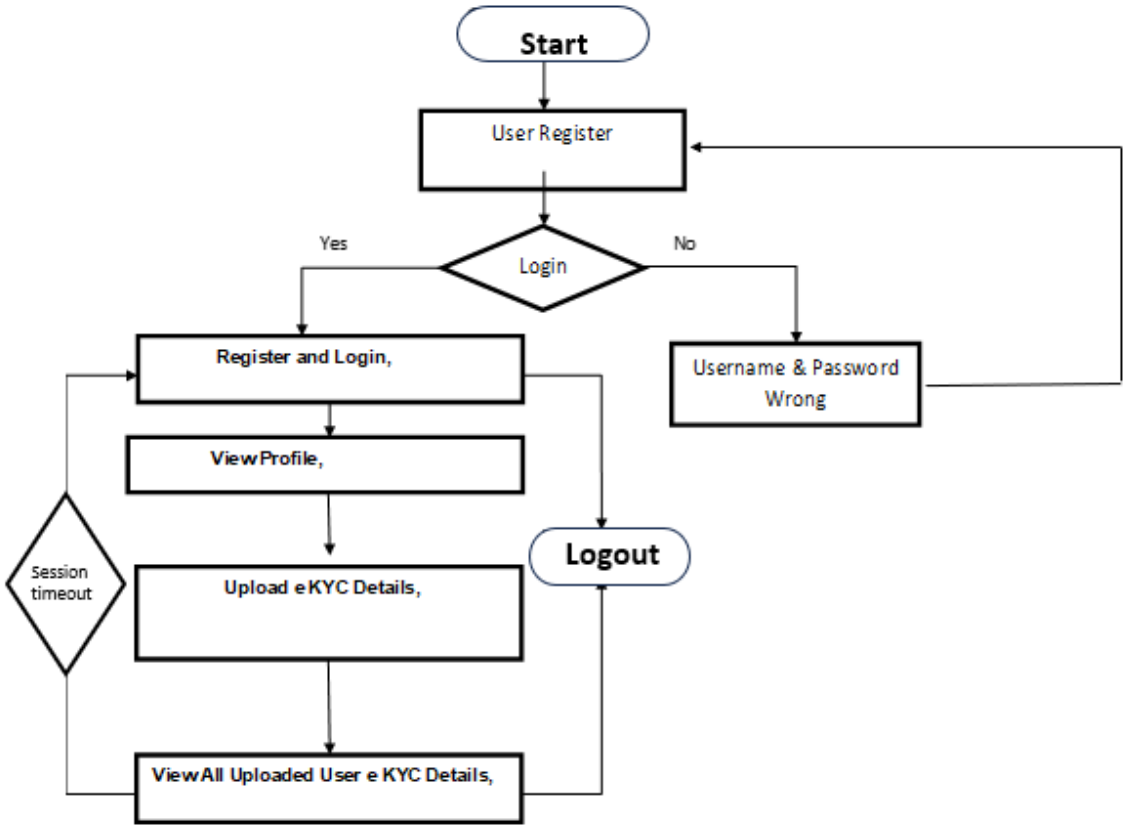


Figure 4.2.4.1 : Flowchart Diagram (Client)

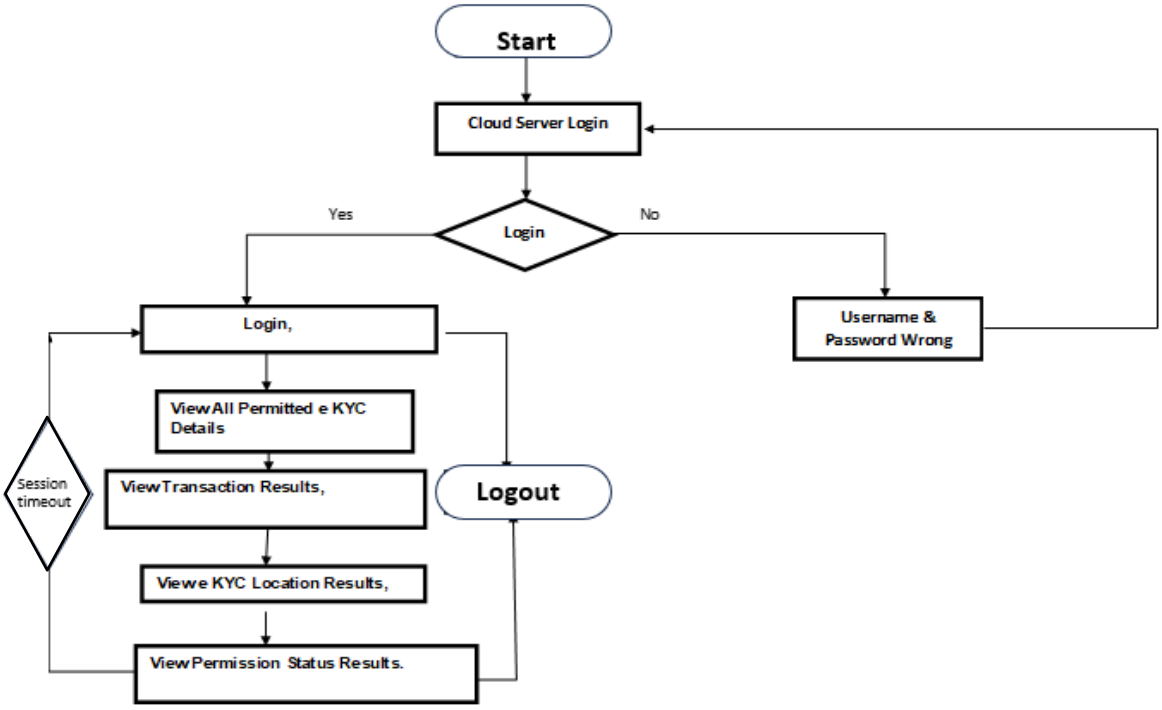


Figure 4.2.4.2 : Flowchart Diagram (Cloud Server)

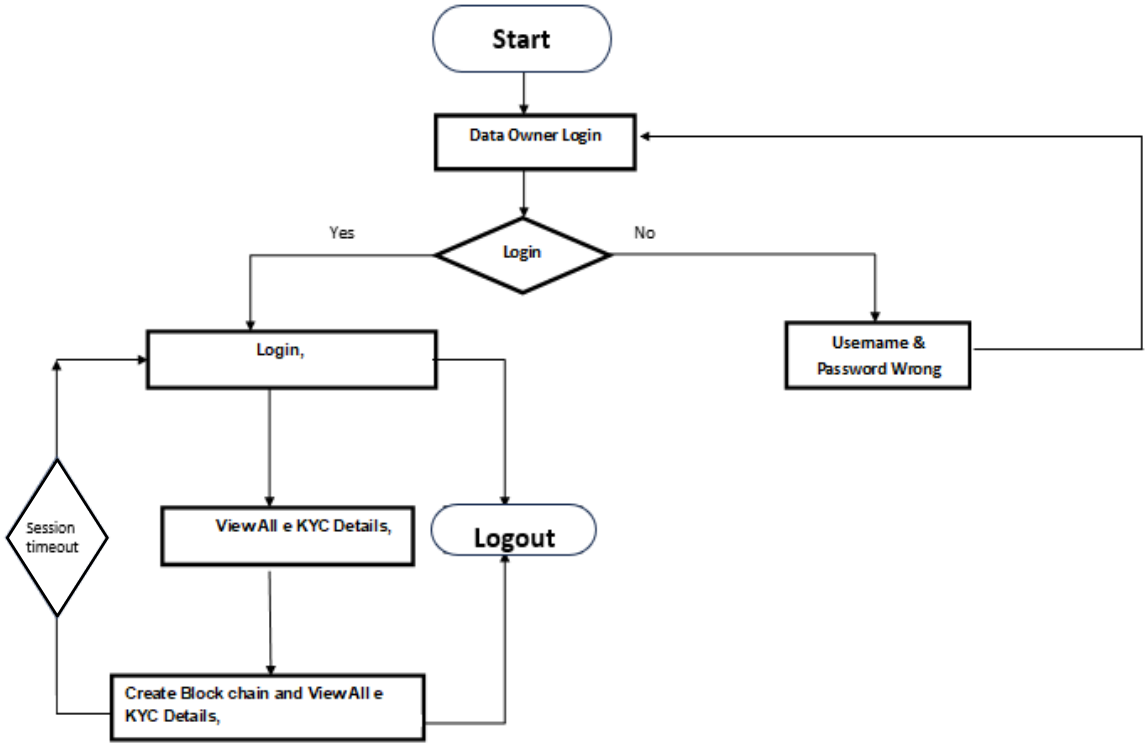


Figure 4.2.4.3 : Flowchart Diagram (Blockchain Server)

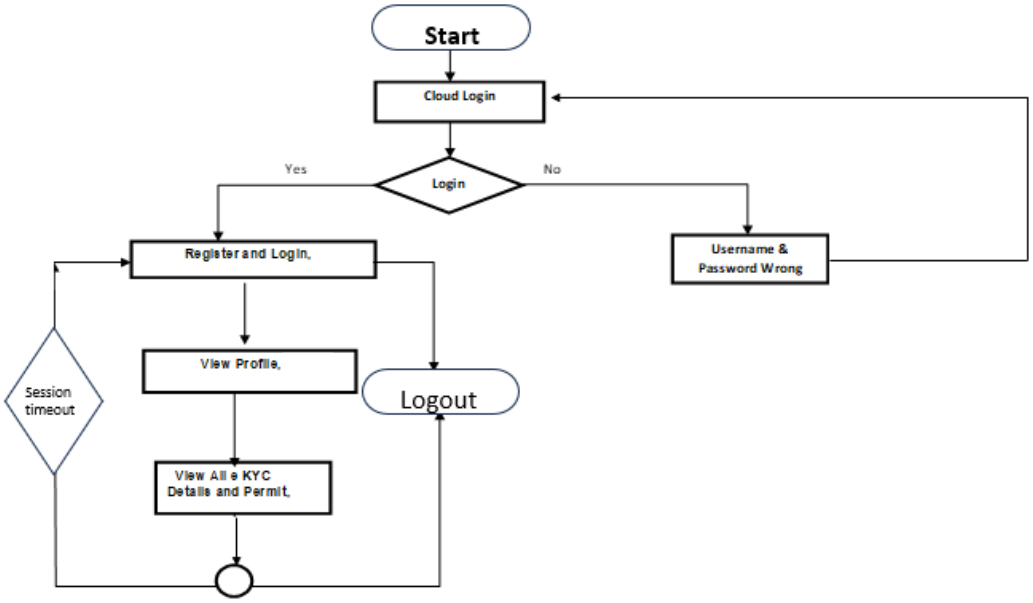


Figure 4.2.4.4 : Flowchart Diagram (Smart Contract)

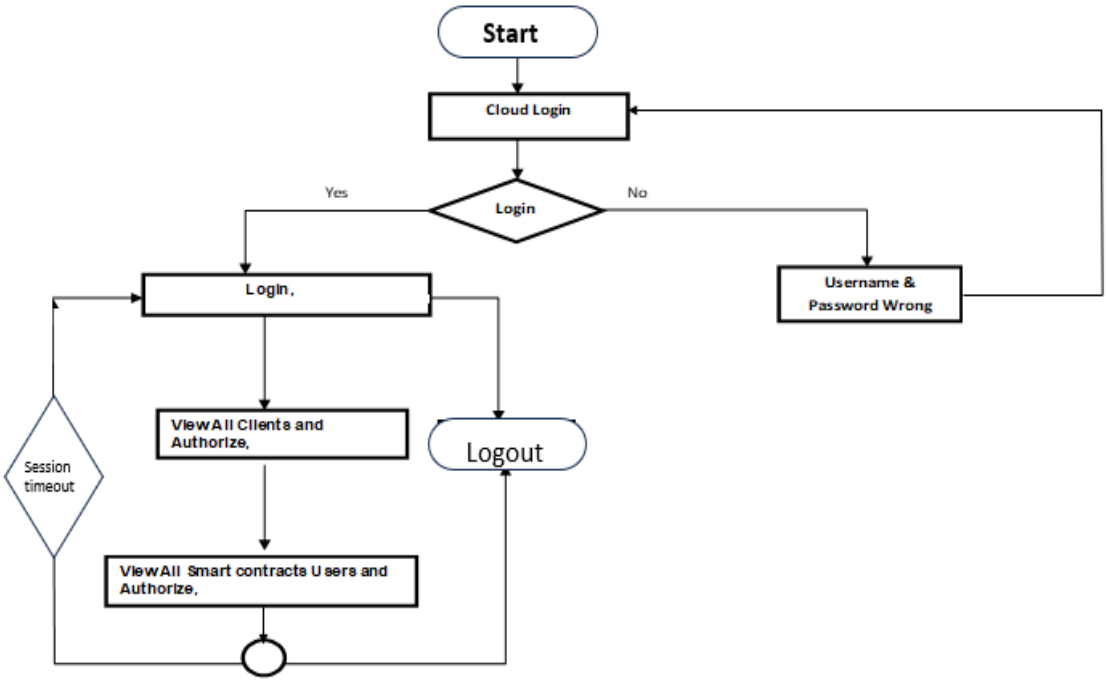


Figure 4.2.4.5 : Flowchart Diagram (Authority)

CHAPTER-5
MODULES

5.1 Clients :

Clients are the customers of financial institutes who join the blockchain-based KYC. Each customer has her own key pair used to encrypt and decrypt her credential data. To allow the credentials to be stored in any FIs database or in the cloud system, the FI must get the consent digitally signed by the client

5.2 IPFS:

IPFS is a cloud database that stores encrypted documents of KYC bound to each user account. It serves for users credentials to generate transaction for cryptocurrency. It houses distributed hash table (DHT)keeping the address of the hash value of the clients credential les which are encrypted in the IPFS storage.

5.3 **Blockchain**:

Blockchain is used to store the transactions of all KYC related activities. All sensitive transactions of the clients are encrypted. The data on the blockchain is tamper proof based on hash value and cryptography mechanism, which also prevents some illegal activities.

5.4 Smart contracts:

These are used to control and automate all KYC processes. In our system, there are three smart contracts including Register contract is responsible for authenticating users, enrolling new users, and uploading the encrypted credentials to the IPFS, Master contract is responsible for controlling client pro les, keeping hash value of the citizen ID of all clients for interacting with IPFS, and e-consent generation, and Verify contract is responsible for KYC verification.

CHAPTER-6
TECHNOLOGIES

The system is built using various **technologies and frameworks** that ensure efficiency, security, and scalability. The core technologies used include **Java** for backend development, **SQL** for database management, **JSP** for dynamic content generation, and **HTML & CSS** for structuring and styling the frontend. This document provides a detailed overview of each technology and how they contribute to the system.

6.1 Java:

Introduction:

Java is the primary programming language used for developing the system's backend logic. It is an object-oriented, platform-independent language that provides security, portability, and high performance.

Features :

- **Platform Independence:** Java code runs on any OS using the Java Virtual Machine (JVM).
- **Security:** Java has built-in security features like the Security Manager and Bytecode Verification to prevent unauthorized access.
- **Multi-threading:** Java supports concurrent processing, allowing efficient execution of multiple tasks.
- **Memory Management:** Java's Garbage Collector (GC) automatically handles memory allocation and deallocation.
- **Robustness:** Java enforces strong typing and exception handling, reducing system crashes.
- **Object-Oriented Programming (OOP):** Java follows OOP principles like inheritance, polymorphism, encapsulation, and abstraction, making code reusable and modular.
- **High Performance:** Java achieves high performance through Just-In-Time (JIT) compilation and optimized runtime execution.

Usage :

Java plays a crucial role in developing robust and secure web applications by implementing business logic in Servlets and JavaBeans. It processes and validates user input, handles authentication, and manages secure transactions effectively. Java Database Connectivity (JDBC) allows seamless interaction with databases, enabling CRUD operations and data persistence. Security is enhanced through file encryption and decryption using PKG

(Public Key Generator), ensuring data protection. Multithreading improves system performance by enabling parallel processing of tasks. Additionally, Java Servlets manage HTTP requests and responses efficiently while facilitating the development of APIs that interact with frontend applications and external modules.

6.2 SQL :

Introduction :

SQL is a standardized query language used to store, manage, and retrieve data efficiently in relational databases. It allows the creation of tables, insertion of records, updating existing data, and deletion of unnecessary information. SQL ensures data integrity and consistency by enforcing constraints and relationships between tables. It supports complex queries to filter, sort, and aggregate data for analysis. Additionally, SQL provides transaction control to maintain data accuracy and handle concurrent operations effectively.

Features :

- **Data Integrity & Consistency:** SQL enforces constraints such as Primary Key, Foreign Key, and Unique constraints to maintain data accuracy and consistency.
- **Transaction Management:** It ensures ACID (Atomicity, Consistency, Isolation, Durability) properties, guaranteeing reliable and secure transaction handling.
- **Security:** SQL provides robust security features such as user authentication, role-based access control, and encryption to protect sensitive data.
- **Fast Query Processing:** SQL supports optimized query execution, enabling efficient data retrieval and manipulation to handle large datasets effectively.

Usage :

SQL is used to manage user authentication by securely storing and verifying login credentials. It facilitates file management by handling file storage, maintaining metadata, and tracking access logs. SQL also powers search and ranking mechanisms by dynamically fetching and ranking search results based on specified criteria. It supports data encryption to ensure secure storage and retrieval of sensitive information. Overall, SQL enhances system functionality by efficiently managing and securing data.

6.3 JSP(Java Server Pages) :

Introduction :

Java Server Pages (JSP) is a technology used to create dynamic web pages by embedding Java code directly into HTML, allowing seamless integration of backend logic with frontend design. It simplifies web application development by enabling developers to write server-side code using Java while maintaining an HTML-based structure. JSP supports tag libraries and custom tags, reducing code complexity and improving reusability. It processes user requests, interacts with databases, and dynamically generates web content. JSP is widely used in enterprise applications to build responsive and interactive web interfaces.

Features :

- **Separation of Concerns:** JSP enables separating business logic from UI**,** improving maintainability.
- **Dynamic Content Generation:** Pages can display real-time data retrieved from the database.
- **Reusable Components:** Supports JavaBeans and custom tags for modular development.
- **Session Management:** Handles user sessions efficiently using cookies and URL rewriting.
- **Automatic Form Handling:** Processes user inputs and interacts with the database.

Usage :

JSP dynamically displays search results and user dashboards by integrating backend logic with frontend design. It efficiently manages user interactions with backend services, ensuring seamless communication. JSP processes and validates form inputs, such as login, registration, and file uploads, enhancing security and accuracy. It also handles session tracking and error management to provide a smooth and consistent user experience. These features make JSP ideal for building responsive and interactive web applications.

6.4 HTML

Introduction :

HTML (HyperText Markup Language) is the standard language used for structuring and presenting content on web pages. It defines the layout and organization of various elements, such as headings, paragraphs, images, links, and forms, that are displayed

on the user's screen. HTML uses a system of nested tags to mark up content, ensuring that browsers interpret and render it correctly. It supports embedding multimedia elements and interactive content to enhance user experience. Additionally, HTML works seamlessly with CSS and JavaScript to create visually appealing and dynamic web applications.

Features :

- **Hyperlinking:** Enables navigation between different pages using <a> tags.

- **Structured Content:** Uses elements like headings, paragraphs, tables, and forms for organization.
- **Media Integration:** Supports embedding images, videos, and audio.
- **Cross-Platform Compatibility:** Works on all modern web browsers.
- **Semantic HTML:** Improves accessibility and SEO optimization.

Usage :

HTML is used to create static pages for login, registration, and file management by structuring content with appropriate elements. It organizes user forms and interactive components, ensuring seamless user interaction. HTML integrates with JavaScript to enable form validation and enhance UI functionality. It also supports embedding multimedia elements such as images, videos, and audio to improve the user experience. Together, these features make HTML essential for developing well-structured and interactive web applications. In addition to the above technologies used, we also have JDBC, Apache Tomcat, and a client-server architecture that play a vital role in the system's functionality.

JDBC (Java Database Connectivity) is a Java API that enables seamless interaction between Java applications and relational databases using SQL queries. It standardizes database connectivity, allowing the system to execute queries for data retrieval, manipulation, and transaction management. JDBC ensures secure and efficient communication between the

application and the database while maintaining portability across different database management systems. By leveraging features like prepared statements and connection pooling, JDBC enhances performance and prevents security vulnerabilities such as SQL injection. Within the system, JDBC is used for handling user authentication, file storage, and executing various SQL queries to ensure smooth database operations.

Apache Tomcat is an open-source web server and servlet container that provides an environment for running Java-based web applications. It acts as middleware that processes HTTP requests, executes Java Servlets and JSP pages, and manages application logic before delivering responses to the client. Tomcat supports essential features like session management,

The Client-Server Architecture forms the backbone of the system, where the client sends requests, and the server processes them before returning appropriate responses. This architecture enables centralized control over data management, authentication, and security while ensuring efficient communication between different system components. The client, typically a web browser, interacts with the server using HTTP requests, while the server processes these requests, retrieves data from the database, and sends responses. This model enhances security by restricting direct database access and improves performance by distributing workload between the client and the server. The system effectively utilizes this architecture to manage user authentication, data retrieval, and file transactions, ensuring a structured and scalable design.

The system effectively integrates multiple technologies to deliver a secure, scalable, and high-performing application. **Java Servlets and JavaBeans** handle business logic, user authentication, and data processing, while **JSP** dynamically generates web pages by embedding Java code into HTML. **JDBC** facilitates seamless interaction with relational databases, ensuring efficient data retrieval, manipulation, and transaction management. **Apache Tomcat** serves as the web server and servlet container, processing HTTP requests, executing JSP and Servlets, and managing application logic. **SQL** handles data storage, ensuring integrity, security, and optimized query performance. **HTML** structures the web pages and defines interactive elements, while **JavaScript** enhances UI functionality and validates user input. The **client-server architecture** centralizes control over data, authentication, and security, efficiently distributing workloads between the client and the server. Together, these technologies work cohesively to provide a robust, secure, and responsive system.

CHAPTER-7
IMPLEMENTATION

Index.html:-

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Home Page</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-aller.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/script.js"></script>
<script type="text/javascript" src="js/coin-slider.min.js"></script>
<style type="text/css">

.style1 {
font-size: 25px;
color: #33FF99;
}
.style2 {
color: #FF0000;
font-weight: bold;
}
.style3 {
font-size: 16px;
color: #FF0000;
font-weight: bold;
font-style: italic;
}
.style4 {color: #FF0000}
-->
</style>
</head>
<body>
<div class="main">
<div class="header">
<div class="header_resize">
<div class="logo">
<h1><a href="index.html" class="style1"> Blockchain for Enhancing Trust and Privacy in Electronic
Know Your Customer </a></h1>
</div>
<div class="menu_nav">
<ul>
<li class="active"><a href="index.html"><span>Home Page</span></a></li>
<li><a href="owner_login.jsp"><span>Clients</span></a></li>
<li><a href="user_login.jsp"><span>Smart contracts User</span></a></li>
<li><a href="cs_login.jsp"><span> Cloud Server</span>(IPFS)</a></li>
<li><a href="au_login.jsp"><span>Authority</span></a></li>
<li><a href="bs_login.jsp"><span>Blockchain Server</span></a></li>
</ul>
</div>
<div class="clr"></div>
<div class="slider">
<div id="coin-slider"> <a href="#"><img src="images/slide1.jpg" width="960"
height="320" alt="" /> </a> <a href="#"><img src="images/slide2.jpg" width="960"
height="320" alt="" /> </a> <a href="#"><img src="images/slide3.jpg" width="960"
height="320" alt="" /> </a> </div>
<div class="clr"></div>
</div>
```

```
<div class="clr"></div>


</div>
</div>
<div class="content">
<div class="content_resize">
<div class="mainbar">
<div class="article">
<div class="clr"></div>
<div class="img"><img src="images/img1.jpg" width="191" height="327" alt="" class="fl" /></div>
<div class="post_content">
<p align="justify" class="style2">The electronic know your customer (e-KYC) is a system for the
banking or identity
…..<p>
</div>
<div class="clr"></div>
</div>
</div>
<div class="sidebar">

<div class="clr"></div>
<div class="gadget">
<h2 class="star"><span>Sidebar</span> Menu</h2>
<div class="clr"></div>
<ul class="sb_menu">
<li><a href="index.html"><span>Home Page</span></a></li>
<li><a href="owner_login.jsp"><span>Client</span></a></li>
<li><a href="user_login.jsp"><span>SC User</span></a></li>
<li><a href="cs_login.jsp"><span>Cloud Server</span>(IPFS)</a></li>
<li><a href="au_login.jsp"><span>Authority</span></a></li>
</ul>
</div>
<div class="gadget">
<h2 class="star"><span>Concepts</span></h2>
<div class="clr"></div>
<ul class="ex_menu"><li class="style3">
</ul>
</div>
</div>
<div class="clr"></div>
</div>
</div>
<div class="fbg">
</div>
<div class="footer">
<div class="footer_resize">
<div style="clear:both;"></div>
<div align=center></div>
</body>
</html>
```

Clients :

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Owner LogIn</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-aller.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/script.js"></script>
<script type="text/javascript" src="js/coin-slider.min.js"></script>
<script language="javascript" type="text/javascript">
function valid()
{
var na3=document.s.userid.value;
if(na3=="")

{
alert("Please Enter Owner Name");
document.s.userid.focus();
return false;
}
else
{

}
var na4=document.s.pass.value;
if(na4=="")

{
alert("Please Enter Password");
document.s.pass.focus();
return false;
}
}
</script>
<style type="text/css">
<!--
.style1 {
  font-size: 25px;
  color: #33FF99;
}
.style2 {font-size: 25px}
.style3 {color: #FF0000}
```

```
	.style5 {color: #FF0000; font-weight: bold; }
-->
</style>
	</head>
	<body>
	<div class="main">
	  <div class="header">
	    <div class="header_resize">
	      <div class="logo">
	        <h1><a href="index.html" class="style1 style2">Blockchain for Enhancing Trust and
	        <td width="48%" height="25"><span class="style3 style2"><strong> <font size="5"
	face="Courier New, Courier, monospace">Pasword</font></strong></span></td>
	        <td width="52%" height="25"><input type="password"  name="pass" size="25" /></td>
	      </tr>
	      <tr>
	        <td height="78" colspan="2"><p align="center">
	            <input type="submit"  value="Login" name="B1" />
	            <input type="reset" value="Reset" name="B2" />
	        </p></td>
	      </tr>
	    </table>
	    <p><span class="style5">New User?</span>  <a href="ow_register.jsp">click here to
	Register</a><br />
	    </p>
	    </div>
	</form>
	          <p> </p>
	      <p> </p>
	        </div>
	      </div>
	      <div class="sidebar">
	        <div class="gadget">
	          <h2 class="star"><span>Sidebar</span> Menu</h2>
	          <div class="clr"></div>
	          <ul class="sb_menu">
	            <li><a href="index.html"><span>Home Page</span></a></li>
	            <li><a href="owner_login.jsp"><span>Client</span></a></li>
	            <li><a href="user_login.jsp"><span>SC User</span></a></li>
	            <li><a href="cs_login.jsp"><span>Cloud Server</span>(IPFS)</a></li>
	            <li><a href="au_login.jsp"><span>Authority</span></a></li>
	          </ul>
	        </div>
	      </div>
	      <div class="clr"></div>
	    </div>
	  </div>
	  <div class="fbg"></div>
	    </div>
	</div>
	<div align=center></div>
	</body>
	</html>
```

Smart contracts User:-

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>User LogIn</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-aller.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/script.js"></script>
<script type="text/javascript" src="js/coin-slider.min.js"></script>
<script language="javascript" type="text/javascript">
function valid()
{
var na3=document.s.userid.value;
if(na3=="")

{
alert("Please Enter User Name");
document.s.userid.focus();
return false;
}
else
{

}
var na4=document.s.pass.value;
if(na4=="")

{
alert("Please Enter Password");
document.s.pass.focus();
return false;
}


}
</script>
<style type="text/css">
<!--
.style1 {
  font-size: 25px;
  color: #33FF99;
}
.style2 {font-size: 25px}
.style3 {color: #FF0000}
```

```
-->
</style>
</head>
<body>
<div class="main">
  <div class="header">
    <div class="header_resize">
      <div class="logo">
    </div>
</form>
  <p> </p>
      <p> </p>
        </div>
      </div>
      <div class="sidebar">
        <div class="gadget">
          <h2 class="star"><span>Sidebar</span> Menu</h2>
          <div class="clr"></div>
          <ul class="sb_menu">
            <li><a href="index.html"><span>Home Page</span></a></li>
            <li><a href="owner_login.jsp"><span>Client</span></a></li>
            <li><a href="user_login.jsp"><span>SC User</span></a></li>
            <li><a href="cs_login.jsp"><span>Cloud Server</span>(IPFS)</a></li>
            <li><a href="au_login.jsp"><span>Authority</span></a></li>
          </ul>
        </div>
      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="fbg"></div>
  <div class="footer">
    <div class="footer_resize">
      <div style="clear:both;"></div>
    </div>
  </div>
</div>
<div align=center></div>
</body>
</html>
```

Cloud Server

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Server LogIn</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-aller.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/script.js"></script>
<script type="text/javascript" src="js/coin-slider.min.js"></script>
<script language="javascript" type="text/javascript">
function valid()
{
var na3=document.s.userid.value;
if(na3=="")

{
alert("Please Enter Name");
document.s.userid.focus();
return false;
}
else
{

}
var na4=document.s.pass.value;
if(na4=="")

{
alert("Please Enter Password");
document.s.pass.focus();
return false;
}


}
</script>
<style type="text/css">
<!--
.style1 {
  font-size: 25px;
  color: #33FF99;
}
```

```
.style2 {font-size: 25px}
.style3 {color: #FF0000}
-->
</style>
</head>
<body>
<div class="main">
  <div class="header">
    <div class="header_resize">
      <div class="logo">
        <h1><a href="index.html" class="style1 style2">Blockchain for Enhancing Trust and
Privacy in Electronic Know Your Customer
</a></h1>
      </div>
      <div class="menu_nav">
        <ul>
          <li><a href="index.html"><span>Home Page</span></a></li>
          <li><a href="owner_login.jsp"><span>PDS Owner</span></a></li>
          <li><a href="user_login.jsp"><span>User</span></a></li>
          <li class="active"><a href="cs_login.jsp"><span>Cloud Server</span></a></li>
          <li><a href="au_login.jsp"><span>Authority</span></a></li>
        </ul>
      </div>
      <div class="clr"></div>
      <div class="slider">
          <p> </p>
      </div>
      </div>
      <div class="sidebar">
        <div class="gadget">
          <h2 class="star"><span>Sidebar</span> Menu</h2>
          <div class="clr"></div>
          <ul class="sb_menu">
            <li><a href="index.html"><span>Home Page</span></a></li>
            <li><a href="owner_login.jsp"><span>PDS Owner</span></a></li>
            <li><a href="user_login.jsp"><span>User</span></a></li>
            <li><a href="cs_login.jsp"><span>Cloud Server</span></a></li>
            <li><a href="au_login.jsp"><span>Authority</span></a></li>
          </ul>
        </div>
      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="fbg"></div>
  <div class="footer">
    <div class="footer_resize">
      <div style="clear:both;"></div>
<div align=center></div>
</body>
</html>
```

Authority:-

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Authority LogIn</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-aller.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/script.js"></script>
<script type="text/javascript" src="js/coin-slider.min.js"></script>
<script language="javascript" type="text/javascript">
function valid()
{
var na3=document.s.userid.value;
if(na3=="")

{
alert("Please Enter Name");
document.s.userid.focus();
return false;
}
else
{

}
var na4=document.s.pass.value;
if(na4=="")

{
alert("Please Enter Password");
document.s.pass.focus();
return false;
}


}
</script>
<style type="text/css">
<!--
.style1 {
  font-size: 25px;
  color: #33FF99;
```

```
}
.style2 {font-size: 25px}
.style3 {font-family: "Times New Roman", Times, serif}
.style4 {color: #FFFFFF}
.style6 {font-size: 14px}
-->
</style>
</head>
<body>
<div class="main">
        <td width="48%" height="25" bgcolor="#FF0000"><span class="style4 style3
style6"><strong> Pasword</strong></span></td>
        <td width="52%" height="25"><input type="password"  name="pass" size="25" /></td>
      </tr>
      <tr>
        <td height="78" colspan="2"><p align="center">
            <input type="submit"  value="Login" name="B1" />
            <input type="reset" value="Reset" name="B2" />
        </p></td>
      </tr>
    </table>

    </div>
</form>
        </div>
      </div>
      <div class="sidebar">
        <div class="gadget">
          <h2 class="star"><span>Sidebar</span> Menu</h2>
          <div class="clr"></div>
          <ul class="sb_menu">
            <li><a href="index.html"><span>Home Page</span></a></li>
            <li><a href="owner_login.jsp"><span>Cleitns</span></a></li>
            <li><a href="user_login.jsp"><span>SC User</span></a></li>
            <li><a href="cs_login.jsp"><span>Cloud Server</span>(IPFS)</a></li>
            <li><a href="au_login.jsp"><span>Authority</span></a></li>
          </ul>
        </div>
      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="fbg"></div>
  <div class="footer">
    <div class="footer_resize">
      <div style="clear:both;"></div>
    </div>
  </div>
</div>
<div align=center></div>
</body>
</html>
```

Block Chain Server:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<title>Blockchain LogIn</title>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<link href="css/style.css" rel="stylesheet" type="text/css" />
<link rel="stylesheet" type="text/css" href="css/coin-slider.css" />
<script type="text/javascript" src="js/cufon-yui.js"></script>
<script type="text/javascript" src="js/cufon-aller.js"></script>
<script type="text/javascript" src="js/jquery-1.4.2.min.js"></script>
<script type="text/javascript" src="js/script.js"></script>
<script type="text/javascript" src="js/coin-slider.min.js"></script>
<script language="javascript" type="text/javascript">
function valid()
{
var na3=document.s.userid.value;
if(na3=="")

{
alert("Please Enter Name");
document.s.userid.focus();
return false;
}
else
{

}
var na4=document.s.pass.value;
if(na4=="")

{
alert("Please Enter Password");
document.s.pass.focus();
return false;
}


}
</script>
<style type="text/css">
<!--
.style1 {
  font-size: 25px;
  color: #33FF99;
}
```

```
.style2 {font-size: 25px}


.style3 {font-family: "Times New Roman", Times, serif}
.style4 {color: #FFFFFF}
.style6 {font-size: 14px}
-->
</style>
</head>
<body>
<div class="main">
  <div class="header">
    <div class="header_resize">
      <div class="logo">
        <h1><a href="index.html" class="style1 style2">Blockchain for Enhancing Trust and
Privacy in Electronic Know Your Customer<br />
        </a></h1>
      </div>

    </div>
  </div>
  <div class="content">
    <div class="content_resize">
      <div class="mainbar">
        </div>
      </div>
      <div class="sidebar">
        <div class="gadget">
          <h2 class="star"><span>Sidebar</span> Menu</h2>
          <div class="clr"></div>
          <ul class="sb_menu">
            <li><a href="index.html"><span>Home Page</span></a></li>
            <li><a href="owner_login.jsp"><span>Clients</span></a></li>
            <li><a href="user_login.jsp"><span>SC User</span></a></li>
            <li><a href="cs_login.jsp"><span>Cloud Server(IPFS)</span></a></li>
            <li><a href="au_login.jsp"><span>Authority</span></a></li>
          </ul>
        </div>
      </div>
      <div class="clr"></div>
    </div>
  </div>
  <div class="fbg"></div>
  <div class="footer">
    <div class="footer_resize">
      <div style="clear:both;"></div>
    </div>
  </div>
</div>
<div align=center></div>
</body>
</html>
```

ENCRYPTION ALG

```
<title>Registration authentication</title>
<%@page import="java.io.BufferedInputStream"%>
<%@page import="java.security.DigestInputStream"%>
<%@page import="java.io.FileInputStream"%>
<%@page import="java.io.PrintStream"%>
<%@page import="java.io.FileOutputStream"%>
<%@page import="java.math.BigInteger"%>
<%@ page
import="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerator,javax.crypto.Cipher"%>


<%
        String user=(String)application.getAttribute("ow");
                    ArrayList list = new ArrayList();
                    ServletContext context = getServletContext();
                    String dirName =context.getRealPath("Gallery\\");
                    String paramname=null;
                    String file=null,aname=null,cont=null,mac=null,cld=null;
                    String a=null,b=null,d=null,image=null,aaddress=null,ano=null;
                    String ee[]=null;
                         String checkBok=" ";
                         int ff=0;
                         String bin = "";
                         String tname=null;
                         String desc=null;
                         int rank=0;                    FileInputStream fs=null;
                    File file1 = null;


try {
                        MultipartRequest multi = new MultipartRequest(request, dirName, 10 * 1024 * 1024);
// 10MB
                        Enumeration params = multi.getParameterNames();
                        while (params.hasMoreElements())

                        }

                        int f = 0;
                        Enumeration files = multi.getFileNames();
                        while (files.hasMoreElements())
                        {
                            paramname = (String) files.nextElement();


                            if(paramname != null)
```

```
                    {
                        f = 1;
                        image = multi.getFilesystemName(paramname);
                        String fPath = context.getRealPath("Gallery\\"+image);
                        file1 = new File(fPath);
                        fs = new FileInputStream(file1);
                        list.add(fs);

                        String ss=fPath;
                        FileInputStream fis = new FileInputStream(ss);
                        StringBuffer sb1=new StringBuffer();
                        int i = 0;
                        while ((i = fis.read()) != -1)
                         {
                            if (i != -1)
                            {
                                //System.out.println(i);
                                String hex = Integer.toHexString(i);
                                // session.put("hex",hex);
                                sb1.append(hex);
                                // sb1.append(",");

                                String binFragment = "";
                                int iHex;
                             int lyke=0;


KeyPairGenerator kg = KeyPairGenerator.getInstance("RSA");
Cipher encoder = Cipher.getInstance("RSA");
KeyPair kp = kg.generateKeyPair();
PublicKey pubKey = kp.getPublic();

// RSA produces 1024 bits Key

byte[] pub = pubKey.getEncoded();
String pk = pub.toString();

    String keys="q2e34rrfgfgfgg2a";
    byte[] keyValue = keys.getBytes();
    Key key = new SecretKeySpec(keyValue, "AES");
    Cipher c = Cipher.getInstance("AES");
    c.init(Cipher.ENCRYPT_MODE, key);

    String encaddr = new String(Base64.encode(addr.getBytes()));
    String encdob = new String(Base64.encode(dob.getBytes()));
    String encemail = new String(Base64.encode(email.getBytes()));
    String encmno = new String(Base64.encode(mno.getBytes()));
    String encano = new String(Base64.encode(ano.getBytes()));
    String encaaddress = new String(Base64.encode(aaddress.getBytes()));
        BigInteger bi1 = new BigInteger(md.digest());
        String h1 = bi1.toString(16);
```

```
                    String rk = "0", rk2 = "0", permit = "No";




          PreparedStatement ps=connection.prepareStatement("insert into
ekyc_details(uname,uaddress,dob,email,mobile,ano,aaddress,image,dt,owname,sk,permit,aloc,hcode)
values(?,?,?,?,?,?,?,?,?,?,?,?,?,?)");
                        ps.setString(1,uname);
                        ps.setString(2,encaddr);
                        ps.setString(3,encdob);
                        ps.setString(4,encemail);                    String task="Uploaded";
                    String strQuery222 = "insert into transaction(user,ano,task,dt)
values('"+user+"','"+ano+"','"+task+"','"+dt+"')";
                    connection.createStatement().executeUpdate(strQuery222);



                        }
            }
                connection.close();

    }
            catch (Exception e)
            {
                out.println(e.getMessage());
                e.printStackTrace();
            }
          %>
```

DECRYPTION

```
        <title>Registration authentication</title>
        <%@page import="java.io.BufferedInputStream"%>
        <%@page import="java.security.DigestInputStream"%>
        <%@page import="java.io.FileInputStream"%>
        <%@page import="java.io.PrintStream"%>
        <%@page import="java.io.FileOutputStream"%>
        <%@page import="java.math.BigInteger"%>
        <%@ page
        import="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerator,javax.crypto.Cipher"

        <%@page
        import="com.oreilly.servlet.*,java.sql.*,java.lang.*,java.text.SimpleDateFormat,java.util.*,java.io.
        *,javax.servlet.*,javax.servlet.http.*" %>


        <%@ page
            import="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerator,javax.crypto.Ciph
        er"%>
```

```
<%@page
    import="java.util.*,java.security.Key,java.util.Random,javax.crypto.Cipher,javax.crypto.spec.Sec
retKeySpec,org.bouncycastle.util.encoders.Base64"%>


<%
        String user=(String)application.getAttribute("ow");


                    ArrayList list = new ArrayList();
                    ServletContext context = getServletContext();



                    FileInputStream fs=null;
                    File file1 = null;

                        }

                        int f = 0;
                        Enumeration files = multi.getFileNames();
                        while (files.hasMoreElements())
                        {
                            paramname = (String)
files.nextElement()                                 {


            KeyPairGenerator kg = KeyPairGenerator.getInstance("RSA");
            Cipher encoder = Cipher.getInstance("RSA");
            KeyPair kp = kg.generateKeyPair();
            PublicKey pubKey = kp.getPublic();

            // RSA produces 1024 bits Key

            byte[] pub = pubKey.getEncoded();
            String pk = pub.toString();

                String keys="q2e34rrfgfgfgg2a";
                byte[] keyValue = keys.getBytes();
                Key key = new SecretKeySpec(keyValue, "AES");
                Cipher c = Cipher.getInstance("AES");
                c.init(Cipher.ENCRYPT_MODE, key);

                String encaddr = new String(Base64.encode(addr.getBytes()));
                String encdob = new String(Base64.encode(dob.getBytes()));
                String encemail = new String(Base64.encode(email.getBytes()));
                String encmno = new String(Base64.encode(mno.getBytes()));
                String encano = new String(Base64.encode(ano.getBytes()));
                String encaaddress = new String(Base64.encode(aaddress.getBytes()));
```

```
            Statement st = connection.createStatement();


            String query="select * from ekyc_details where uname='"+uname+"'";
            ResultSet rs=st.executeQuery(query);

            if(rs.next()==true)

            }
             else if(rs.next()!=true)
             {

                 String filename = context.getRealPath("filename.txt");
                    try (PrintStream p = new PrintStream(new FileOutputStream(filename))) {
                        p.print(new String(aloc));
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    MessageDigest md = MessageDigest.getInstance("SHA1");
                    try (FileInputStream fis11 = new FileInputStream(filename);
                         DigestInputStream dis1 = new DigestInputStream(fis11, md);
                         BufferedInputStream bis1 = new BufferedInputStream(dis1)) {
                        while (true) {
                            int b1 = bis1.read();
                            if (b1 == -1)
                                break;
                        }
                    }


                        }
%>

          </p><br />
        <p>  <a href="ow_main.jsp">BACK</a></p>

<%
            }
                connection.close();

    }
            catch (Exception e)
            {
                out.println(e.getMessage());
                e.printStackTrace();
            }
          %>
```

GENERATING KEY

```
        <title>Registration authentication</title>

<%@page import="java.io.BufferedInputStream"%>
<%@page import="java.security.DigestInputStream"%>
<%@page import="java.io.FileInputStream"%>
<%@page import="java.io.PrintStream"%>
<%@page import="java.io.FileOutputStream"%>
<%@page import="java.math.BigInteger"%>
<%@ page
import="java.security.Key,java.security.KeyPair,java.security.KeyPairGenerator,javax.crypto.Cipher"%>
<%@page
import="java.util.*,java.security.Key,java.util.Random,javax.crypto.Cipher,javax.crypto.spec.SecretKeySpec,o
rg.bouncycastle.util.encoders.Base64"%>

                                        while(binFragment.length() < 4)
                                        {
                                            binFragment = "0" + binFragment;
                                        }
                                        bin += binFragment;
                                        //System.out.print(bin);
                                    }
                                }
                            }
                        }
                    }

                /*  FileInputStream fs1 = null;
                    //name=dirName+"\\Gallery\\"+image;
                    int lyke=0; ;

                    Date now = new Date();

                    String strDate = sdfDate.format(now);
                    String strTime = sdfTime.format(now);
                    String dt = strDate + "   " + strTime;



            KeyPairGenerator kg = KeyPairGenerator.getInstance("RSA");
            Cipher encoder = Cipher.getInstance("RSA");
            KeyPair kp = kg.generateKeyPair();
            PublicKey pubKey = kp.getPublic();

            // RSA produces 1024 bits Key

            byte[] pub = pubKey.getEncoded();
            String pk = pub.toString();
```

```
                String keys="q2e34rrfgfgfgg2a";
                byte[] keyValue = keys.getBytes();
                Key key = new SecretKeySpec(keyValue, "AES");
                Cipher c = Cipher.getInstance("AES");
                c.init(Cipher.ENCRYPT_MODE, key);

                String encaddr = new String(Base64.encode(addr.getBytes()));
                String encdob = new String(Base64.encode(dob.getBytes()));
                String encemail = new String(Base64.encode(email.getBytes()));
                ……



            Statement st = connection.createStatement();


            String query="select * from ekyc_details where uname='"+uname+"'";
            ResultSet rs=st.executeQuery(query);

            if(rs.next()==true)
            {
              out.println("Username Already Exist");

              %>

          </p>
            }
             else if(rs.next()!=true)
             {

                 String filename = context.getRealPath("filename.txt");
                    try (PrintStream p = new PrintStream(new FileOutputStream(filename))) {
                        p.print(new String(aloc));
                    } catch (IOException e) {
                        e.printStackTrace();
                    }

                    MessageDigest md = MessageDigest.getInstance("SHA1");

…….
<%
            }
                connection.close();

    }
            catch (Exception e)
            {
                out.println(e.getMessage());
                e.printStackTrace();
            }
          %>
```

CHAPTER-8

TESTING

8.1 Introduction to Testing :

The Purpose of Testing is to discover Errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies or a finished product. It is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of testing. Each test type addresses a specific testing requirement. Testing is a process, which reveals errors in the program. It is the major quality measure employed during software development. During testing, the program is executed with a set of test cases and the output of the program for test cases is evaluated to determine if the program is performing as it is expected to perform.

8.2 Testing Strategies :

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results. Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

8.3 Unit Testing :

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements. Each module can be tested using the following two Strategies:

Black Box Testing :

In this strategy some test cases are generated as input conditions that fully execute all functional requirements for the program. This testing has been uses to find errors in the following categories:

- Incorrect/missing functions

- Interface errors
- Data structure or database access issues
- Performance errors
- Initialization/termination errors

In this testing only the output is checked for correctness. The logical flow of the data is not checked.

White Box Testing :
In this the test cases are generated on the logic of each module by drawing flow graphs of that module and logical decisions are tested on all the cases. It has been uses to generate the test cases in the following cases and ensures :

- All independent paths are executed.
- Logical decisions are tested.
- Loops run within boundaries.
- Internal data structures are valid.

8.4 Integration Testing :

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design. There are two types of integration testing. They are :

Top Down Integration :

This method follows an incremental approach to constructing the program structure, integrating modules by moving downward through the control hierarchy, starting from the main program module. Subordinate modules are incorporated into the structure using either a depth-first or breadth-first approach. The software is initially tested using the main module, with individual stubs serving as placeholders for lower-level modules. As testing progresses, these stubs are systematically replaced with actual modules, ensuring a controlled and efficient testing process

Bottom Up Integration :

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

- The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
- A driver (i.e.) the control program for testing is written to coordinate test case input and output.
- The cluster is tested.
- Drivers are removed and clusters are combined moving upward in the program structure.

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

8.5 System Testing :

Software, once validated, must be integrated with other system elements such as hardware, people, and databases to ensure seamless functionality. System testing plays a crucial role in verifying that all these components interact correctly and that the overall system meets its intended performance. This phase of testing ensures that the system functions as expected and aligns with its original objectives, current specifications, and system documentation. Additionally, system testing helps identify discrepancies that may exist between the designed system and its actual implementation, ensuring that any deviations are addressed before deployment. By thoroughly evaluating the system's functionality, integration, and performance, system testing guarantees a reliable and efficient final product.

8.6 Acceptance Testing :

User acceptance is a crucial factor in determining the success of any system. To ensure this, the system is continuously tested for user acceptance by actively involving prospective users during the development process and incorporating necessary changes based on their feedback. A well-designed system prioritizes a user-friendly interface, making it intuitive and easy to navigate, even for individuals unfamiliar with the system. By maintaining close communication with users and addressing their requirements, the system enhances usability, ensuring a seamless experience and increasing overall user satisfaction.

8.7 Functional Testing :

Functional testing systematically verifies that the system meets business and technical requirements as outlined in system documentation and user manuals. This type of testing ensures that all specified functions operate correctly by evaluating various aspects, including valid and invalid input handling, function execution, expected output generation, and interactions with interfacing systems or procedures. The organization and preparation of functional tests focus on key requirements, critical functions, and special test cases. Additionally, systematic coverage includes business process flows, data fields, predefined processes, and successive processes to ensure comprehensive validation of system functionality.

8.8 Test Approach : Testing can be done in two ways. They are :

1. Bottom-Up Approach
2. Top-Down Approach

Bottom-Up Approach :

Testing can be performed starting from smallest and lowest level modules and proceeding one at a time. For each module in bottom up testing a short program executes the module and provides the needed data so that the module is asked to perform the way it will when embedded within the larger system. When bottom level modules are tested attention turns to those on the next level that use the lower-level ones they are tested individually and then linked with the previously examined lower-level modules.

Top-Down Approach :

This type of testing starts from upper-level modules. Since the detailed activities usually performed in the lower-level routines are not provided stubs are written. A stub is a module shell called by upper-level module and that when reached properly will return a message to the calling module indicating that proper interaction occurred. No attempt is made to verify the correctness of the lower-level module.

CHAPTER-9
SCREENSHOTS

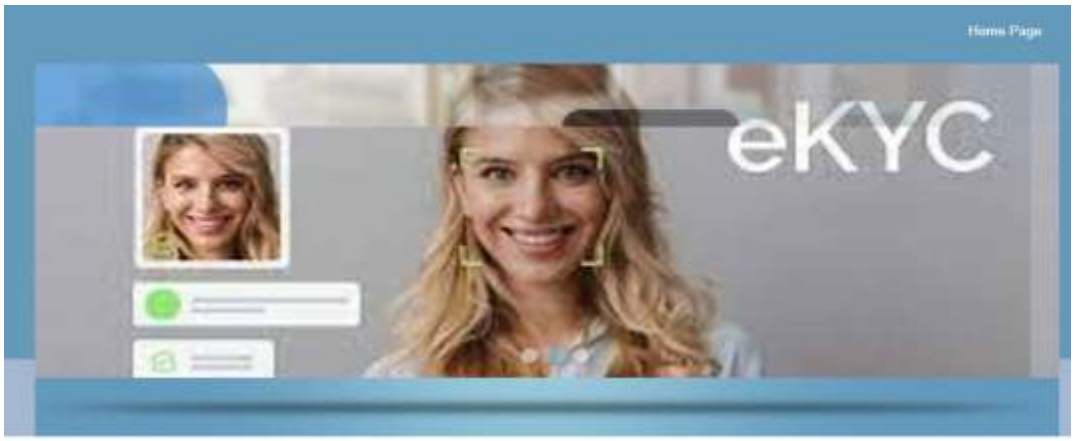
Sidebar Menu

Concepts

e-KYC, authentication, CP-ABE, key management, access control, blockchain

Blockchain for Enhancing Trust and Privacy in Electronic Know Your Customer

The electronic know your customer (e-KYC) is a system for the banking or identity proxies to establish a customer identity data verification process between relying parties. Due to the efficient resource consumption and the high degree of accessibility and availability of cloud computing, most banks implement their e-KYC system on the cloud. Essentially, the security and privacy of e-KYC related documents stored in the cloud becomes the crucial issue. Existing e-KYC platforms generally rely on strong authentication and apply traditional encryption to support their security and privacy requirement. In this model, the KYC system owner encrypts the file with their trust's key and uploads it to the cloud. This method induces encryption dependency and communication and key management overheads. In this paper, we introduce a novel blockchain-based e-KYC scheme called e-KYC Trustblock based on the ciphertext policy attribute based encryption (CP-ABE) method binding with the client consent enforcement to deliver trust, security and privacy compliance. In addition, we introduce attribute-based encryption to enable the privacy preserving and fine-grained access of sensitive transactions stored in the blockchain. Finally, we conduct experiments to show that our system is efficient and scalable in Practica...



Sidebar Menu

Home Page

Client

SC User

Cloud Server(IPFS)

Authority

Client Log In

Name

Pasword

Login Reset

New User? Click here to Register

Client Registration :

Sidebar Menu

Home Page
Client
SC User
Cloud Server(PFS)
Authority

SC Log In!!!

Name
Pasword

Login Reset

New User? click here to Register

Sidebar Menu

Home Page
Client
SC User
Cloud Server(PFS)
Authority

User Registration

Register

User Name (required)

Password (required)

Financial Institute Name (required)

--Select--

Specialist (required)

--Select--

Email Address (required)

Mobile Number (required)

Your Address

Date of Birth (required)

Select Gender (required)

-Select-

Enter Pincode (required)

Select Profile Picture (required)

Choose file No file chosen

REGISTER

Back

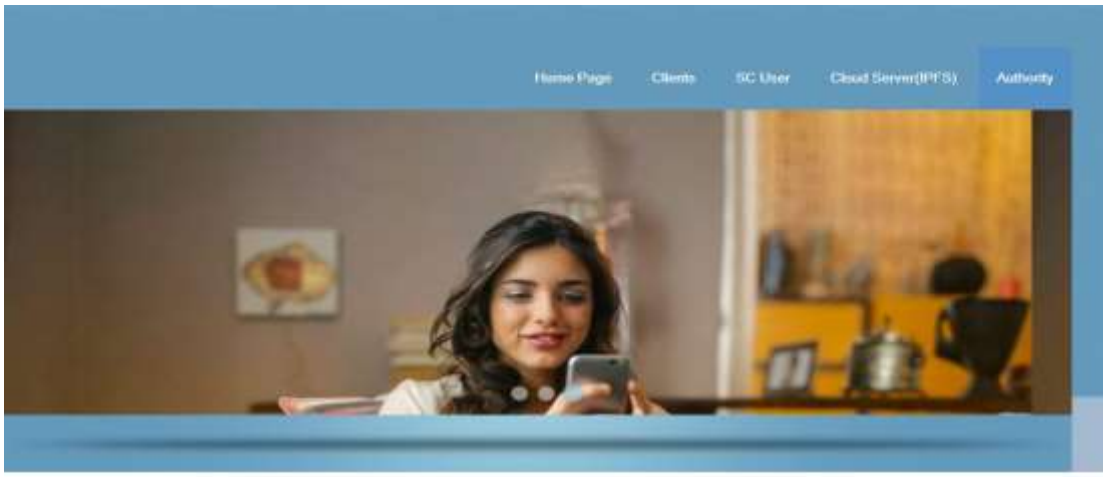
SC User Menu

View Profile

View All eKYC Details and Permit

Log Out

Welcome Smart Contract :: lalith



Sidebar Menu

Home Page

Clients

SC User

Cloud Server(IPFS)

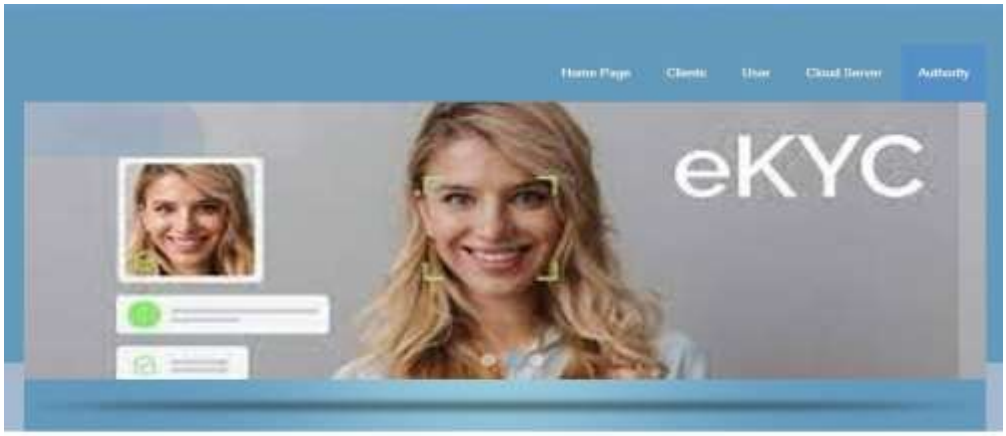
Authority

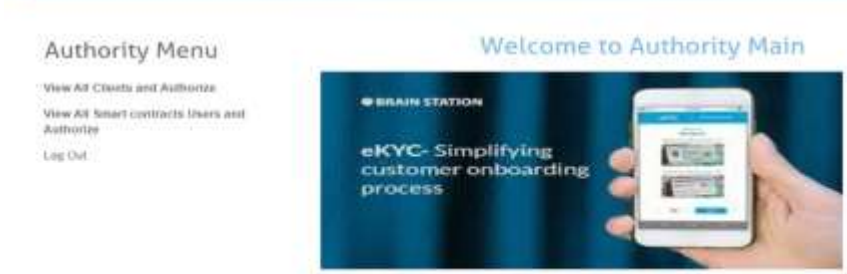
Authority Log In

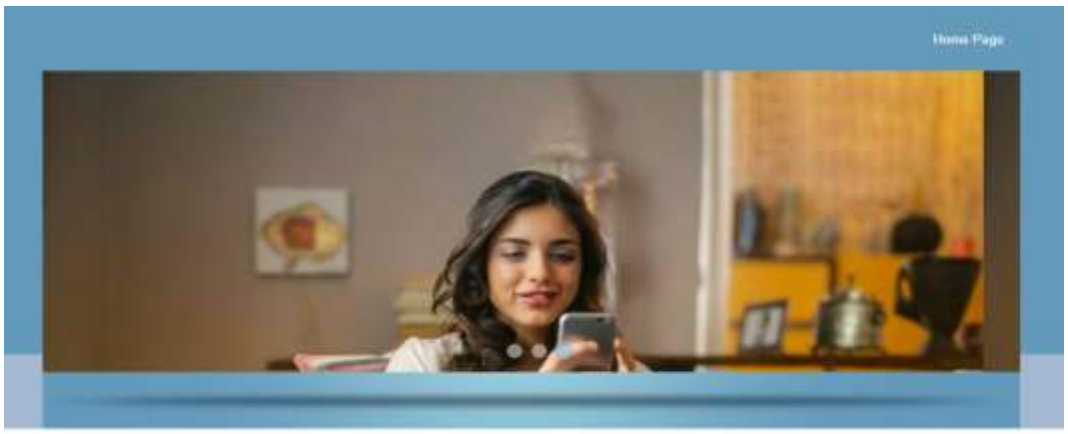
Name

Password

Login Reset


Home Page
Clients
User
Cloud Server
Authority
eKYC


Authority Menu
View All Clients and Authorize
View All Smart contracts Users and Authorize
Log Out
Welcome to Authority Main
BRAIN STATION
eKYC- Simplifying customer onboarding process


Home Page


Sidebar Menu
Home Page
Clients
SC User
Cloud Server(IPFS)
Authority
Blockchain Log In
Name
Pasward
Login
Reset


Home Page
eKYC


Blockchain Menu
View All eKYC Details
Create Blockchain and View All eKYC Details
Log Out
Welcome :: Blockchain Server
SignDesk

CHAPTER-10
CONCLUSION AND FUTURE SCOPE

10.1 Conclusion :

We have presented the privacy-preserving e-KYC approach based on the blockchain. Our proposed scheme delivers secure and decentralized authentication and verification of the e-KYC process with the users consent enforcement feature. In our scheme, the privacy of both customers identity documents stored in the cloud is guaranteed by the symmetric key and public key encryption while the sensitive transaction data stored in the blockchain is encrypted by symmetric key encryption and CP-ABE. Our scheme also allows the KYC data to be updated by the data owner or the customer. In addition, we devised an access policy update algorithm to enable dynamic access authorization. For the evaluation, we performed comparative analysis between our scheme and related works in terms of the computation cost, the communication cost, and performance. The experimental results showed that our scheme outperforms existing schemes in terms of performance, comprehensive KYC compliance features, and the scalable access control mechanism. For future works, we will test a larger sample of data in the real cloud environment and measure the throughput of the system in accommodating high number of e-KYC registration and verification requests. In addition, we will investigate the technique to enable batch veri cation of e-KYC transactions stored in the blockchain with the searchable encryption feature.

10.2 Future Scope :

The future scope of blockchain-based e-KYC systems focuses on strengthening privacy, scalability, and compliance in digital identity verification. By integrating advanced cryptographic methods like quantum-resistant encryption and Ciphertext-Policy Attribute-Based Encryption (CP-ABE), the system can offer fine-grained access control while safeguarding against emerging security threats. Smart contracts will evolve to support dynamic consent management, enabling users to control data sharing in real-time while maintaining transparency and auditability across financial institutions.

Incorporating technologies like Progressive Web Applications (PWAs), biometric authentication, and edge computing with Web Assembly will enhance user experience and reduce processing latency. AI-driven analytics can be used for automated fraud detection and anomaly tracking, ensuring proactive security. Blockchain integration with decentralized identifiers (DIDs), multi-cloud environments, and IoT devices will expand the system’s practical applications, making it a robust, future-ready solution for global, privacy-preserving e-KYC processes.

CHAPTER-11
REFERENCES

[1] Y. Zhong, M. Zhou, J. Li, J. Chen, Y. Liu, Y. Zhao, and M. Hu, ``Distributed blockchain-based authentication and authorization protocol for smart grid," *Wireless Commun. Mobile Comput.*, vol. 2021, pp. 1_15, Apr. 2021, doi: 10.1155/2021/5560621.

[2] S. Y. Lim, P. T. Fotsing, A. Almasri, O. Musa, M. L. M. Kiah, T. F. Ang, and R. Ismail, ``Blockchain technology the identity management and authentication service disruptor: A survey," *Int. J. Adv. Sci. Eng. Inf. Tech.*, vol. 8, pp. 1735_1745, Sep. 2018.

[3] A. A. Mamun, A. Al Mamun, S. R. Hasan, S. R. Hasan, M. S. Bhuiyan, M. S. Bhuiyan, M. S. Kaiser, M. S. Kaiser, M. A. Yousuf, and M. A. Yousuf, ``Secure and transparent KYC for banking system using IPFS and blockchain technology," in *Proc. IEEE Region Symp. (TENSYMP)*, Jun. 2020, pp. 348_351.

[4] M. Pic, G. Mahfoudi, and A. Trabelsi, ``RemoteKYC: Attacks and countermeasures," in *Proc. Eur. Intell. Secur. Informat. Conf. (EISIC)*, Nov. 2019, pp. 126_129.

[5] W. Shbair, M. Steichen, and J. François, ``Blockchain orchestration and experimentation framework: A case study of KYC," in *Proc. 1st IEEE/IFIP Int. Workshop Manag. Managed Blockchain (Man Block)*, Jeju Island, South Korea, Aug. 2018, pp. 23_25.

[6] R. Norvill, M. Steichen, W. M. Shbair, and R. State, ``Demo: Blockchain for the simpli_cation and automation of KYC result sharing," in *Proc. IEEE Int. Conf. Blockchain Cryptocurrency (ICBC)*, May 2019, pp. 9_10, doi: 10.1109/BLOC.2019.8751480.

[7] T. Mikula and R. H. Jacobsen, ``Identity and access management with blockchain in electronic healthcare records," in *Proc. 21st Euromicro Conf. Digit. Syst. Design (DSD)*, Prague, Czech Republic, Aug. 2018, pp. 699_706.

[8] S. Wang, R. Pei, and Y. Zhang, ``EIDM: A ethereum-based cloud user identity management protocol," *IEEE Access*, vol. 7, pp. 115281_115291, 2019, doi: 10.1109/ACCESS.2019.2933989. [9] N. Ullah, K. A. Al-Dhlan, and W. M. Al-Rahmi, ``KYC optimization by blockchain based hyperledger fabric network," in *Proc. 4th Int. Conf. Adv. Electron. Mater., Comput. Softw. Eng. (AEMCSE)*, Mar. 2021, pp. 1294_1299.

[10] N. Kapsoulis, A. Psychas, G. Palaiokrassas, A. Marinakis, A. Litke, and T. Varvarigou, ``Know your customer (KYC) implementation with smart contracts on a privacy-oriented decentralized architecture," *Future Internet*, vol. 12, no. 41, pp. 1_13, 2020.

[11] J. Bethencourt, A. Sahai, and B.Waters, ``Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy*, Oakland, CA, USA, May 2007, pp. 321_334.

[12] I. Gutierrez-Aguero, S. Anguita, X. Larrucea, A. Gomez-Goiri, and B. Urquizu, ``Burnable pseudo-identity: A non-binding anonymous identity method for ethereum," *IEEE Access*, vol. 9, pp. 108912_108923, 2021.

[13] S. Nakamoto. (2008). *Bitcoin: A Peer-to-Peer Electronic Cash System*. Accessed: Jan. 8, 2022. [Online]. Available: https://bitcoin.org/bitcoin.pdf

[14] J. P. Moyano and O. Ross, ``KYC optimization using distributed ledger technology," *Bus. Inf. Syst. Eng.*, vol. 59, no. 6, pp. 411_423, Dec. 2017.

[15] A. Chowdhary, S. Agrawal, and B. Rudra, ``Blockchain based framework for Student identity and educational certi_cate veri_cation," in *Proc. 2nd Int. Conf. Electron. Sustain. Commun. Syst. (ICESC)*, Aug. 2021, pp. 916_921.

[16] *GDPREuropeanUnionGuidelines*. Accessed: Aug. 12, 2021. [Online]. Available: https://gdprinfo.eu/

[17] G. Bramm, M. Gall, and J. Schütte, ``BDABE-blockchain-based distributed

attribute based encryption," in *Proc. 15th Int. Conf. e-Bus. Telecommun.*, 2018, pp. 99_110.

[18] Y. Fan, X. Lin, W. Liang, J. Wang, G. Tan, X. Lei, and L. Jing, ``TraceChain: A blockchain-based scheme to protect data con_dentiality and traceability," *Softw., Pract. Exper.*, vol. 52, no. 1, pp. 115_129, Jan. 2022, doi: 10.1002/spe.2753.

[19] C. Yuan, M. Xu, X. Si, and B. Li, ``Blockchain with accountable CP-ABE: How to effectively protect the electronic documents," in *Proc. IEEE 23rd Int. Conf. Parallel Distrib. Syst. (ICPADS)*, Dec. 2017, pp. 800_803.

[20] A.Wu, Y. Zhang, X. Zheng, R. Guo, Q. Zhao, and D. Zheng, ``Ef_cient and privacy-preserving traceable attribute-based encryption in blockchain," *Ann. Telecommun.*, vol. 74, nos. 7_8, pp. 401_411, Aug. 2019.

[21] L. Guo, X. Yang, and W.-C. Yau, ``TABE-DAC: Ef_cient traceable attribute-based encryption scheme with dynamic access control based on blockchain," *IEEE Access*, vol. 9, pp. 8479_8490, 2021, doi: 10.1109/ACCESS.2021.3049549.

[22] M. Barati, G. S. Aujla, J. T. Llanos, K. A. Duodu, O. F. Rana, M. Carr, and R. Ranjan, ``Privacy-aware cloud auditing for GDPR compliance veri_cation in online healthcare," *IEEE Trans. Ind. Informat.*, vol. 18, no. 7, pp. 4808_4819, Jul. 2022, doi: 10.1109/TII.2021.3100152.

[23] *PBC (Pairing-Based Cryptography) Library*. Accessed: Jan. 5, 2022. [Online]. Available: https://crypto.stanford.edu/pbc/

[24] S. Gao, G. Piao, J. Zhu, X. Ma, and J. Ma, ``TrustAccess: A trustworthy secure ciphertext-policy and attribute hiding access control scheme based on blockchain," *IEEE Trans. Veh. Technol.*, vol. 69, no. 6, pp. 5784_5798, Jun. 2020.

[25] A. D. Dwivedi, R. Singh, U. Ghosh, R. R. Mukkamala, A. Tolba, and O. Said, ``Privacy preserving authentication system based on noninteractive zero knowledge proof suitable for Internet of Things," *J. Ambient Intell. Humanized Comput.*, pp. 1_11, Sep. 2021, doi: 10.1007/s12652-021-03459-4.