BLOCK CHAIN BASED SOLUTION FOR GENERATING AND VERIFYING DIGITAL CERTIFICATION

A project report submitted in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

by

MANDALA VAMSI	**21BF1A3738**
PASUPULETI HARI KRISHNA	**21BF1A3747**
ROJAKARI MASTAN RAKESH	**21BF1A3750**
ATLURU VAMSI KRISHNA	**21BF1A3704**

Under the guidance of

Dr. JASMINE SABEENA, Ph.D.

PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

SRI VENKATESWARA COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu
Accredited by NBA, New Delhi & NAAC with 'A' grade)

Karakambadi Road, TIRUPATI – 517507

2021 – 2025

SRI VENKATESWARA COLLEGE OF ENGINEERING

(AUTONOMOUS)

(Approved by AICTE, New Delhi & Permanently Affiliated to JNTUA, Ananthapuramu

Accredited by NBA, New Delhi & NAAC with 'A' Grade)

Karakambadi Road, TIRUPATI – 517507.

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY)

CERTIFICATE

This is to certify that the project work entitled,**" BLOCK CHAIN BASED SOLUTION FOR GENERATING AND VERIFYING DIGITAL CERTIFICATION "** is a bonafide record of the project work done and submitted by

MANDALA VAMSI	**21BF1A3738**	**R MASTAN RAKESH**	**21BF1A3750**
P HARI KRISHNA	**21BF1A3747**	**A VAMSI KRSIHNA**	**21BF1A3704**

under my guidance and supervision for the partial fulfillment of the requirements for the award of B.Tech degree in **COMPUTER SCIENCE AND ENGINEERING (CYBER SECURITY).** This project is the result of our own effort and that it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above

GUIDE **HEAD OF DEPARTMENT**

External Viva-Voce Exam held on _______________

INTERNAL EXAMINER **EXTERNAL EXAMINER**

DECLARATION

We hereby declare that the project entitled **“BLOCK CHAIN BASED SOLUTION FOR GENERATING AND VERIFYING DIGITAL CERTIFICATION”,** done by us under the esteemed guidance of **Dr. Jasmine Sabeena,** Professor and is submitted in partial fulfilment of the requirements for the award of the Degree of **Bachelor of Technology in Computer Science and Engineering (Cyber Security).**

This report is the result of our own effort and it has not been submitted to any other University or Institution for the award of any degree or diploma other than specified above.

MANDALA VAMSI	**21BF1A3738**
PASUPULETI HARI KRISHNA	**21BF1A3747**
ROJAKARI MASTAN RAKESH	**21BF1A3750**
ATLURI VAMSI KRISHNA	**21BF1A3704**

ACKNOWLEDGEMENT

We are thankful to our guide **Dr. Jasmine Sabeena, Ph.D, Professor, Department Of CSE (Cyber Security)** for her valuable guidance and encouragement. Her helping attitude and suggestions have helped in the successful completion of the project report.

We would like to express our gratefulness and sincere thanks to **Dr. Ch. Santhaiah , Professor & HOD, Department of CSE (Cyber Security) ,** for his kind help and encouragement during the course of study and in the successful completion of the project work.

We have great pleasure in expressing out hearty thanks to our beloved Principal **Dr. N. Sudhakar Reddy** for spending his valuable time with us to complete this project.

Successful completion of any project cannot be done without proper support and encouragement. We sincerely thank the Management for providing all the necessary facilities during course of study.

We would like to thank our parents and friends, who have the greatest contributions in all our achievements, for the great care and blessings in making us successful in all our endeavours.

MANDALA VAMSI	**21BF1A3738**
PASUPULETI HARI KRISHNA	**21BF1A3747**
ROJAKARI MASTAN RAKESH	**21BF1A3750**
ATLURU VAMSI KRISHNA	**21BF1A3704**

ABSTRACT

In the digital era, generating and verifying academic and professional certificates face challenges like data tampering, forgery, and slow manual verification. Existing methods lack security, efficiency, and accessibility. The above drawbacks, therefore, indicate the necessity of having a more secure and efficient solution for credential management. To address these issues, this project proposes a blockchain-based solution for secure and transparent credential management.

By leveraging blockchain's immutability and decentralization, certificates are issued and verified digitally, eliminating unauthorized modifications and intermediaries. Verification is streamlined using smart contracts and off-chain encrypted storage, ensuring data integrity while reducing costs. Cryptographic techniques secure documents and digital signatures. A user-friendly WebApp enhances accessibility, making the system scalable and efficient for institutions and certificate holders.

Key Words: Blockchain Technology, Off-Chain Database, Cryptographic Methods, WebApp Interface, Verification Process, Digital Signature.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

PKI	Public Key Infrastructure
CA	Certificate Authority
P2P	Peer-to-Peer
SHA	Secure Hash Algorithm
Dapp	Decentralized Application
NFT	Non-Fungible Token
UML	Unified Modeling Language

Chapter 1
INTRODUCTION

In an era defined by digital transformation and data proliferation, the management of educational records stands at the nexus of security, privacy, and efficiency. Traditional approaches to record storage and sharing are plagued by vulnerabilities, leaving sensitive data susceptible to breaches and manipulation. VeriSure emerges as a beacon of innovation in this landscape, harnessing the power of blockchain technology to revolutionize educational record management. By leveraging blockchain's inherent immutability and decentralization, VeriSure ensures the integrity and confidentiality of educational records, ushering in a new era of trust and reliability.

Educational institutions and individuals alike grapple with the challenges of securely storing and sharing academic credentials and achievements. VeriSure addresses these challenges head-on, offering a comprehensive solution that not only safeguards data but also streamlines the exchange process. Through a user-friendly interface and seamless integration with blockchain technology, VeriSure empowers users to take control of their educational records, facilitating secure and efficient sharing across diverse stakeholders.

The foundation of VeriSure lies in its commitment to security and privacy. In a digital landscape fraught with cyber threats and data breaches, protecting sensitive educational records is paramount. Verisure employs cutting-edge cryptographic techniques and robust security measures to fortify its platform against unauthorized access and manipulation. By decentralizing data storage and utilizing encryption protocols, VeriSure ensures that user data remains confidential and tamper-proof, instilling confidence in users and institutions alie.

Chapter 2

PROJECT DESCRIPTION

2.1 : PROBLEM DEFINITION

Educational institutions worldwide are confronted with the formidable challenge of securely managing and sharing vast amounts of sensitive data, including academic transcripts, certifications, and other educational records. Traditional methods of record-keeping, reliant on centralized databases and manual processes, are inherently vulnerable to security breaches, data tampering, and unauthorized access. Moreover, the proliferation of digital credentials and the increasing mobility of students and professionals demand a more agile and reliable solution for record management.
The existing landscape of educational record management is fraught with inefficiencies and risks. Centralized databases, while convenient, are susceptible to single points of failure and are prime targets for malicious actors seeking to exploit vulnerabilities. Furthermore, the manual verification processes inherent in traditional record management systems are time-consuming, error-prone, and lack transparency, leading to delays and discrepancies in credential verification.

Moreover, as educational records become increasingly digitized, concerns about data privacy and security loom large. The prevalence of data breaches and identity theft underscores the urgency of implementing robust security measures to protect sensitive educational information from unauthorized access and manipulation. Additionally, the lack of interoperability among disparate record management systems further exacerbates the challenges of data exchange and verification, hindering the seamless transfer of academic credentials across institutions and borders.

Furthermore, the emergence of new technologies and paradigms, such as blockchain, presents both opportunities and challenges for educational record management. While blockchain offers the promise of immutable and decentralized data storage, its adoption in the educational sector has been hindered by technical complexities, scalability concerns, and regulatory uncertainties. Integrating blockchain technology into existing record management systems requires careful planning and coordination to ensure compatibility, usability, and compliance with legal and regulatory frameworks.

In light of these challenges, there is a pressing need for innovative solutions that leverage cutting-edge technologies to address the complexities of educational record management. Such solutions must prioritize security, privacy, and interoperability while providing a seamless user experience and facilitating efficient data exchange. By harnessing the power of blockchain, cryptographic techniques, and decentralized protocols, these solutions can empower educational institutions and individuals to securely manage, share, and verify educational records, thereby ushering in a new era of trust and transparency in the digital age.

OBJECTIVES:

1. Secure Data Storage: Implement a secure and tamper-proof storage mechanism for educational records using blockchain technology. By leveraging the immutability and cryptographic properties of blockchain, ensure the integrity and confidentiality of stored data, safeguarding it against unauthorized access and manipulation.

2. Efficient Data Sharing: Develop streamlined processes and protocols for efficient sharing of educational records among stakeholders. Utilize blockchain-based mechanisms to facilitate peer-to-peer data exchange, eliminating the need for intermediaries and reducing delays and costs associated with traditional verification methods.

3. User-Friendly Interface: Design an intuitive and user-friendly interface for decentralized record management. Prioritize ease of use and accessibility to ensure that users, including students, educational institutions, and employers, can easily navigate the platform, upload and access records, and initiate data sharing transactions.

4. Interoperability: Ensure interoperability with existing record management systems and standards to facilitate seamless integration and data exchange. Adhere to industry-wide protocols and standards to enable compatibility with diverse educational institutions and credentialing bodies, enhancing the platform's utility and scalability

5. Privacy Protection: Implement robust privacy protection measures to safeguard sensitive user data and personal information. Employ encryption techniques and data anonymization protocols to prevent unauthorized access and ensure compliance with data protection regulations and best practices.

6. Browser Compatibility: Address browser compatibility issues to ensure a consistent user experience across different web browsers. Conduct thorough testing and optimization to mitigate compatibility issues related to browser extensions and plugins, such as MetaMask, ensuring smooth functionality across diverse user environments.

7. Performance Optimization: Optimize platform performance to enhance responsiveness and scalability, even under heavy loads and peak usage periods. Employ caching mechanisms, load balancing, and other performance optimization techniques to minimize latency and maximize throughput, ensuring a seamless user experience.

8. Security Measures: Implement robust security measures to protect user data, private keys, and transactional information from cyber threats and attacks. Utilize multi-factor authentication, encryption, and secure coding practices to fortify the platform against vulnerabilities and mitigate risks of data breaches and unauthorized access.

2.2 PROJECT DETAILS:

The interaction between the web interface and the blockchain is facilitated by Web3.js, a JavaScript library that ensures smooth communication. The web interface, designed using streamlit (python script) serves as the user's gateway to the system. Through this interface, users can register new documents or verify the authenticity of existing ones.

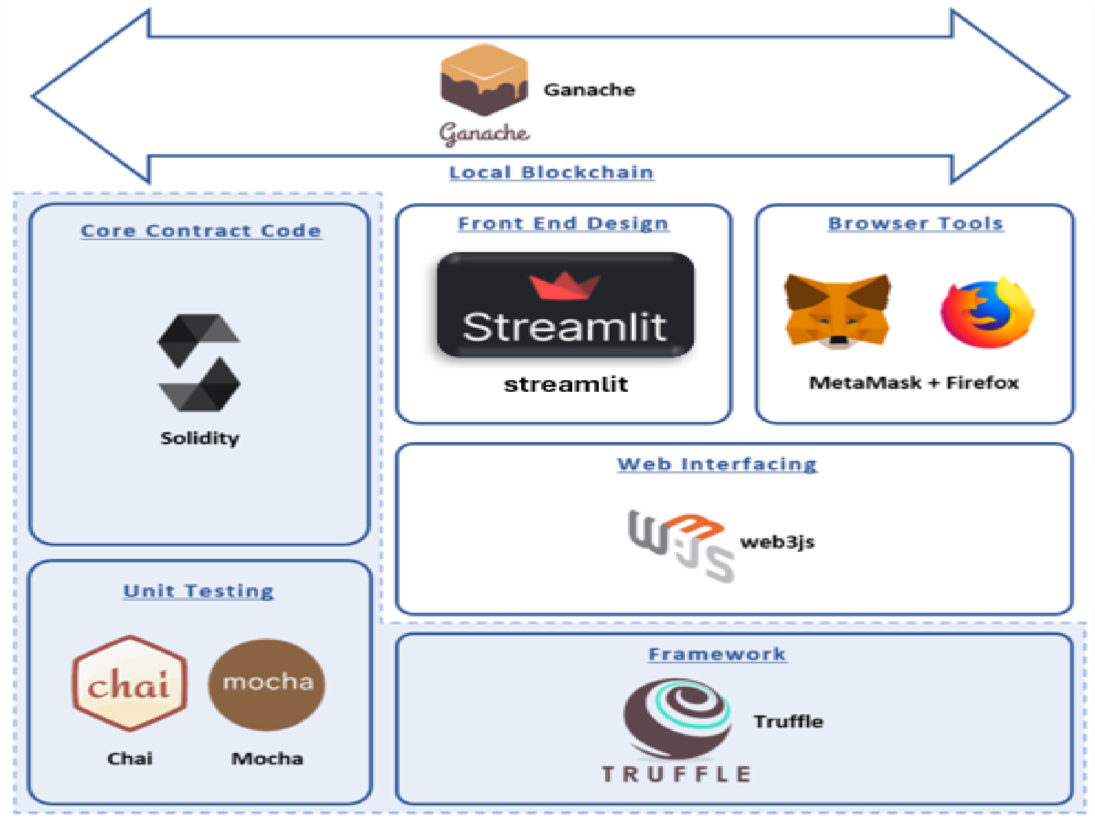


Fig 2.2 Project Details

The integration of Metamask, a browser extension, adds an extra layer of security and convenience, allowing users to manage their wallets and sign transactions securely. The entire system is architected to prioritize user experience, ensuring that document identification processes are not only secure but also intuitive and accessible. With the integration of these technologies, the system design fosters an ecosystem.

Chapter 3
COMPUTATIONAL ENVIRONMENT

3.1 : SOFTWARE REQUIREMENTS

Platform	: Windows 10
IDE	: VS Code
UML Design:	: Visual Paradigm
Coding Language	: Python
Frontend	: Streamlit (Python script)
Software	:Ganache, Solidity

3.2 : HARDWARE REQUIREMENTS

Processor	: Inter i3, AMD Ryzen3 or higher
Speed	: 1.6 ghz or higher
RAM	: 4 GB or higher
Storage Disk	: 500 GB or higher
Servers	: Blockchain & Web Hosting Server

3.3 : FUNCTIONAL REQUIREMENTS

VeriSure represents a paradigm shift in educational record management, centered around the innovative integration of blockchain technology. At its core, VeriSure provides a secure and decentralized platform for storing, sharing, and verifying educational records, leveraging the immutable and transparent nature of blockchain to ensure data integrity and trustworthiness. By anchoring encrypted records and hash information on the blockchain, VeriSure eliminates the need for centralized authorities or intermediaries, empowering users to take control of their own data and streamline the exchange process.

Central to the idea content of VeriSure is its commitment to user-centric design and accessibility. Recognizing the diverse needs and preferences of its users, VerifyLab prioritizes the development of an intuitive and user-friendly interface that caters to individuals, educational institutions, and employers alike. Through seamless integration with existing web browsers and compatibility with popular blockchain wallets such as MetaMask, VeriSure ensures a frictionless user experience, allowing users to effortlessly upload, access, and share their educational records with confidence.

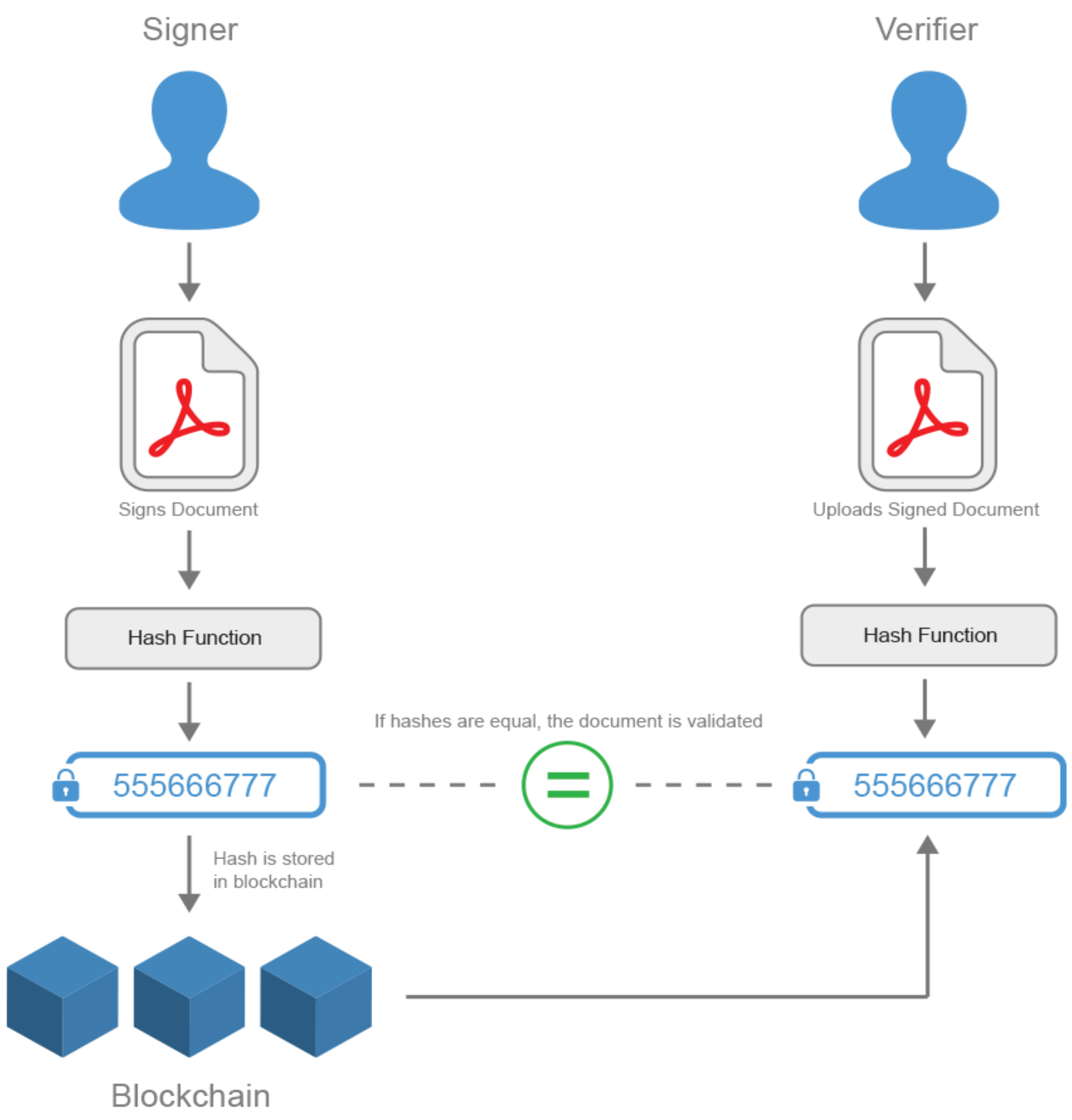
Moreover, VeriSure idea content extends beyond mere record management to encompass broader principles of privacy, security, and data sovereignty. By implementing robust encryption protocols and privacy protection measures, VeriSure safeguards sensitive user data and personal information from unauthorized access and manipulation. Furthermore, by decentralizing data storage and eliminating single points of failure, VeriSure enhances data sovereignty and resilience, empowering users to retain full control over their educational records while mitigating risks of data breaches and cyber attacks.

Overall, VeriSure content embodies the core principles of trust, transparency, and empowerment, ushering in a new era of secure and decentralized educational record management.

Drawbacks of Conventional System :

1. ***Vulnerability to Replication***
   - Traditional methods, such as holograms, are often easily replicated by sophisticated counterfeiters using advanced technologies, undermining the reliability of these identification measures.
2. ***Limited Traceability and Transparency***
   - Conventional systems may lack comprehensive traceability features, making it difficult to track the origin and movement of documents throughout the supply chain. This limitation reduces the ability to swiftly detect and address counterfeit documents.
3. ***Susceptibility to Tampering***
   - Physical identifiers, such as labels and tags, are prone to tampering. Counterfeiters can manipulate or remove these identifiers, compromising the integrity of the identification system and allowing counterfeit documents to go undetected**.**
4. ***Manual Verification Processes***
   - Many traditional systems rely on manual verification processes, introducing the risk of human error and making the detection of counterfeit documents a time-consuming and inefficient process. Automated verification processes are often needed for increased accuracy and efficiency.

3.4 : FUNCTIONING WORKFLOW



3.4 Functioning Workflow

3.5: SOFTWARE FEATURES

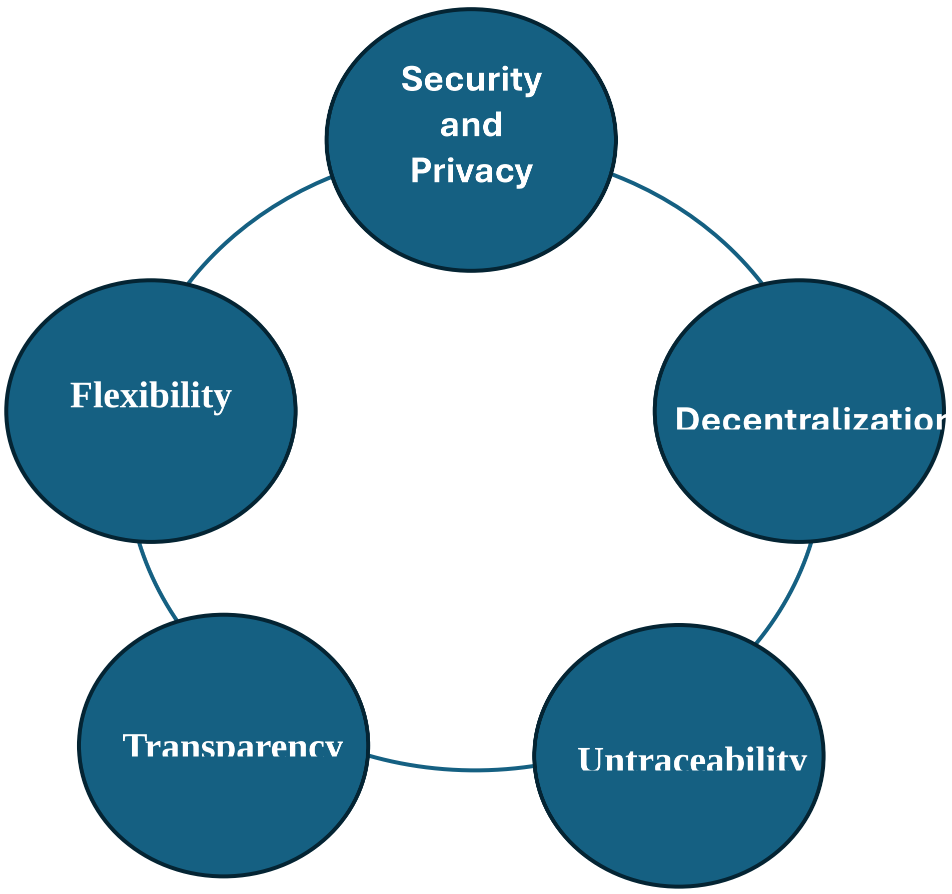


Fig 3.5 Features

1.Decentralization:

Decentralization is a foundational aspect of the system, distributing decision-making across a network of nodes to prevent a single point of failure and resist manipulation. The consensus mechanism ensures that no single authority has complete control over the records, enhancing the reliability and robustness of the system. This decentralized architecture eliminates the risks associated with centralized control, making the records more secure and resistant to malicious activity.

2.Flexibility:

The blockchain-based immutable records system boasts remarkable flexibility, accommodating a diverse array of data and transactions. Whether recording financial transactions, supply chain events, or healthcare information, the system's design allows it to seamlessly integrate and adapt to various types of data, making it highly versatile for different industries and applications. The blockchain-based immutable records system boasts remarkable flexibility, accommodating a diverse array of data and transactions. Whether recording financial transactions, supply chain events, or healthcare information, the system's design allows it to seamlessly integrate and adapt to various types of data, making it highly versatile for different industries and applications.

3.Security:

Security is at the forefront of the blockchain-based system, employing cryptographic hash functions and consensus mechanisms to ensure the integrity protection of each record. With a tamper-evident design and a decentralized
architecture, the system provides robust resistance against unauthorized alterations significantly reducing the risks of fraud, data breaches, and tampering. This inherent security instills trust in the recorded information, making it particularly well-suited for industries where data integrity is paramount.

4.Transparency:

Transparency is a key feature of the system, facilitated by the open and accessible nature of the blockchain. Participants within the network have clear visibility into the entire history of transactions or records. The cryptographic hashing adds an extra layer of security, creating a transparent and auditable trail. This transparency enhances trust among participants, allowing them to independently verify information and fostering increased accountability in the accuracy of records.

Chapter 4
FEASIBILITY STUDY

FEASIBILITY ANALYSIS:

1. **Technical Feasibility**

   - ***Assessment:*** The project's technical feasibility is high, given the availability of well-established technologies such as Truffle, Ganache, Web3.js, and blockchain protocols. Smart contracts facilitate secure data storage, and QR code generation is a widely supported technology.

   - ***Considerations:*** The technical expertise required for development, integration, and maintenance should be available. Compatibility with existing systems and the ability to adapt to emerging blockchain standards need consideration.

2. **Economic Feasibility**

   - ***Assessment:*** The economic feasibility of the project is justifiable due to the potential long-term cost savings in combating counterfeit goods and enhancing supply chain efficiency. Initial development costs may include software development, smart contract creation, and integration expenses.

   - ***Considerations:*** A cost-benefit analysis should factor in the potential reduction in losses due to counterfeit documents, the increased trust in the marketplace, and the long-term economic advantages for businesses adopting the system.

3. **Operational Feasibility**
   - ***Assessment:*** Operationally, the project is feasible as it integrates with existing technologies and involves processes (smart contract deployment, QR code generation) that are practical and user-friendly.

   - ***Considerations:*** Adequate training for system users, seamless integration with browsers and extensions, and adherence to industry.
     standards will be crucial for successful implementation and adoption.

4. **Legal and Regulatory Feasibility**
   - ***Assessment:*** The project aligns with legal and regulatory frameworks related to document identification, data privacy, and blockchain technology.

   - ***Considerations:*** Regular updates to comply with evolving regulations, ensuring data protection and user privacy, and obtaining necessary permissions for deploying the system are essential considerations.

5. **Schedule Feasibility**
   - ***Assessment:*** The project's schedule feasibility is contingent on the complexity of smart contract development, system integration, and user interface design. Clear milestones and timelines can ensure efficient project completion.

   - ***Considerations:*** Adequate planning, effective communication, and contingency measures for potential delays or unforeseen challenges are vital for adhering to the proposed schedule.

Chapter 5
SYSTEM ANALYSIS

5.1: EXISTING SYSTEM

- Document counterfeiting is a critical issue across industries, where forged documents lead to legal, financial, and reputational damages.
- Traditional methods such as paper-based documentation, physical seals, barcodes, and holograms are increasingly proving inadequate against sophisticated forgeries.
- The existing systems rely heavily on centralized databases and manual verification processes, making them vulnerable to data tampering, unauthorized access, and inefficiencies in record verification.
- The existing system used for document authentication generally follows these steps:
    1. Collection of document-related data from users or issuing authorities.
    2. Storage of records in centralized databases or on paper.
    3. Manual or semi-automated verification processes for document legitimacy.
    4. Validation based on visible security features or credentials.

5.1.1 : *CERTIFICATE GENERATION*

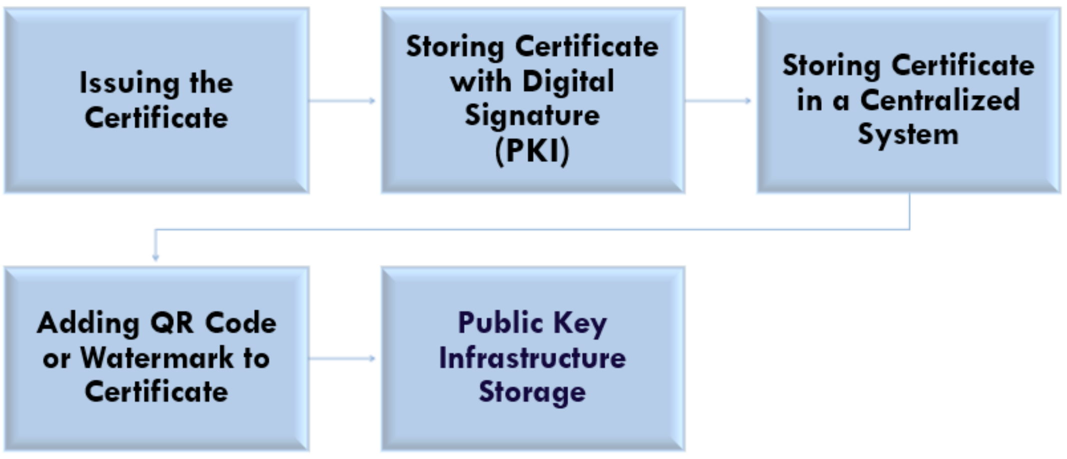

Issuing the Certificate
Storing Certificate with Digital Signature (PKI)
Storing Certificate in a Centralized System
Adding QR Code or Watermark to Certificate
Public Key Infrastructure Storage

Fig 5.1.1 Existing Certificate Generation System

5.2.2 : CERTIFICATE VERIFICATION

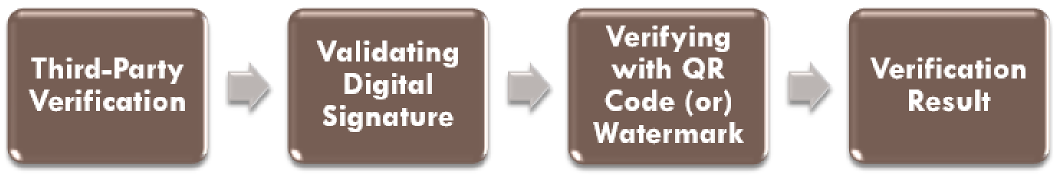

Third-Party Verification
Validating Digital Signature
Verifying with QR Code (or) Watermark
Verification Result

Fig 5.1.1 Existing Certificate Verification System

5.1.1 : CURRENT SYSTEM AND THEIR LIMITATIONS

1. **Centralized Databases**
   - **Current System:** Many industries rely on centralized databases to store and manage records, often maintained by a single authority or organization.
   - **Limitations:** Centralized systems are vulnerable to single points of failure and are more susceptible to security breaches. A breach in the central database can compromise the entire dataset. Additionally, these systems often lack transparency, as access and control are concentrated in the hands of a few entities, limiting visibility for end-users.

2. **Paper-Based Systems**
   - **Current System:** Some sectors still heavily rely on paper-based records for various processes, including documentation and transactions.
   - **Limitations:** Paper-based systems are prone to human errors, loss, and deterioration over time. Retrieval and verification processes can be time-consuming and inefficient. Moreover, these systems often lack the real-time accessibility and auditability required for modern, fast-paced environments.

3. **Traditional Authentication Methods**

    - **Current System:** Authentication methods such as passwords, PINs, and security questions are widely used to secure access to systems and data.
    - **Limitations:** Traditional authentication is susceptible to security breaches due to weak passwords, phishing attacks, and human negligence. Once compromised, these systems may not offer sufficient protection. Additionally, these methods often lack the robustness needed for ensuring secure access in an era of increasing cyber threats.

4. **Legacy Supply Chain Management Systems**

    - **Current System:** Some industries still operate with legacy supply chain management systems that rely on manual data entry and fragmented information storage.
    - **Limitations:** Legacy systems often lack real-time visibility and traceability in supply chains. Manual data entry is error-prone and can lead to inaccuracies, impacting the efficiency and transparency of the entire supply chain. Additionally, these systems may struggle to adapt to the dynamic and interconnected nature of modern supply chains.

5. **Traditional Financial Transaction Systems**

- **Current System:** Traditional financial transaction systems, such as credit card networks and wire transfers, remain prevalent in many sectors.
- **Limitations:** These systems can be susceptible to fraud, chargebacks, and delays. The reliance on intermediaries for transaction processing can lead to high fees and slow settlement times. The level of transparency and security demanded by modern financial ecosystems.

The limitations of these current systems underscore the need for innovative solutions that address issues of centralization, inefficiency, security vulnerabilities, and lack of transparency. Transitioning to advanced technologies, such as blockchain-based systems, can mitigate these limitations and usher in a new era of secure, transparent, and efficient record-keeping across various industries.

5.2: PROPOSED SYSTEM

- The proposed system leverages blockchain technology to ensure document authenticity, transparency, and traceability, reducing the risk of forgery and counterfeiting.
- Rather than replacing all traditional methods outright, this system is designed to complement and enhance existing document verification processes by introducing decentralized, immutable record-keeping and smart contract functionalities.

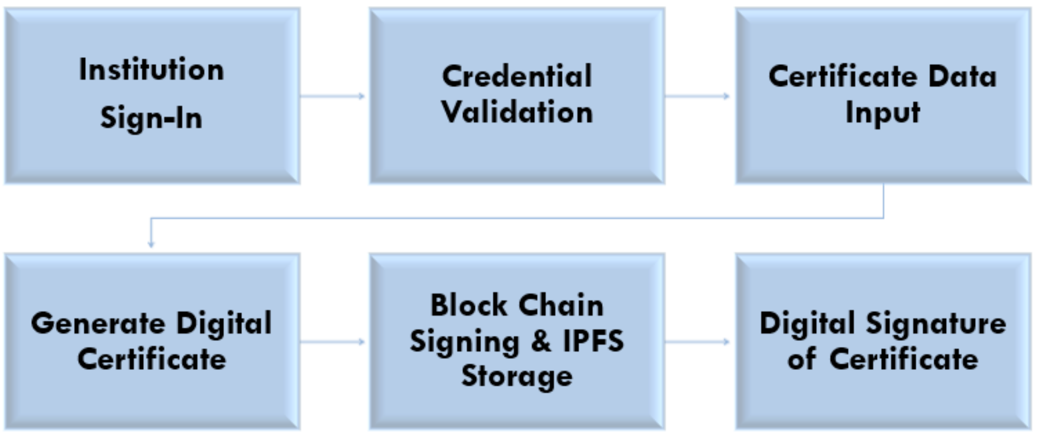

Institution Sign-In
Credential Validation
Certificate Data Input
Generate Digital Certificate
Block Chain Signing & IPFS Storage
Digital Signature of Certificate

Fig 5.2.1 Proposed Certificate Generation System

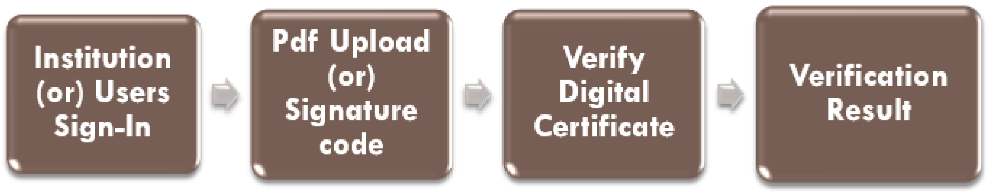

Institution (or) Users Sign-In
Pdf Upload (or) Signature code
Verify Digital Certificate
Verification Result

Fig 5.2.1 Proposed Certificate Verification System

- The working of the system begins with the collection of document data, followed by generation of a unique hash for each document. This hash is stored on the blockchain using smart contracts, and a QR code linked to the document's blockchain record is generated for verification purposes.
- The proposed system follows these steps to form an effective and secure document authentication framework:
    1. Data Collection and Processing
    2. Hash Generation and Blockchain Storage
    3. Smart Contract Deployment
    4. QR Code Generation for Document Linkage
    5. Document Verification via Blockchain Lookup
    6. User Interface for Uploading and Verifying Documents

Features of Proposed System

- Intuitive and user-friendly interface for document upload and verification
- Enhanced security through blockchain’s tamper-evident architecture
- Unique QR code generation for each document for easy verification
- Supports real-time tracking and auditability of documents

Advantages of Proposed System

- Eliminates reliance on centralized authorities
- Enhances trust through decentralized and immutable storage
- Reduces the risk of forgery and counterfeiting
- Saves time and resources by automating verification processes
- Scalable and adaptable to different industries (e.g., education, supply chain, legal, healthcare).

Chapter 6
SYSTEM DESIGN

System design is a transition from a user-oriented document to programmers or database personnel. The design is a solution, how to approach the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study.

Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of the implementation plan and prepare a logical design walkthrough. Systems design is the process of defining the architecture, components, modules, interfaces, and data for a system to satisfy specified requirements. One could see it as the application of systems theory to product development.

System design is the process of defining, specifying, and creating the architecture, components, modules, interfaces, and data for a software system to satisfy specified requirements. It involves breaking down a complex system into smaller, more manageable components and defining how these components will work together to achieve the desired functionality. The goal of system design is to create a system that is reliable, scalable, maintainable, and efficient. This involves identifying the functional and non-functional requirements of the system, determining the appropriate technologies and tools to use, and creating a plan for testing and deployment.

System design is an important aspect of software engineering, as it lays the foundation for the development and implementation of a successful software product or system.

6.1 System Architecture:

The below figure visualizes the architectural design of the project. This architecture provides the summarized view of the project. By this we can understand the rest of the project.

Blockchain based solution for generating and verifying digital certificates mainly consists following components :

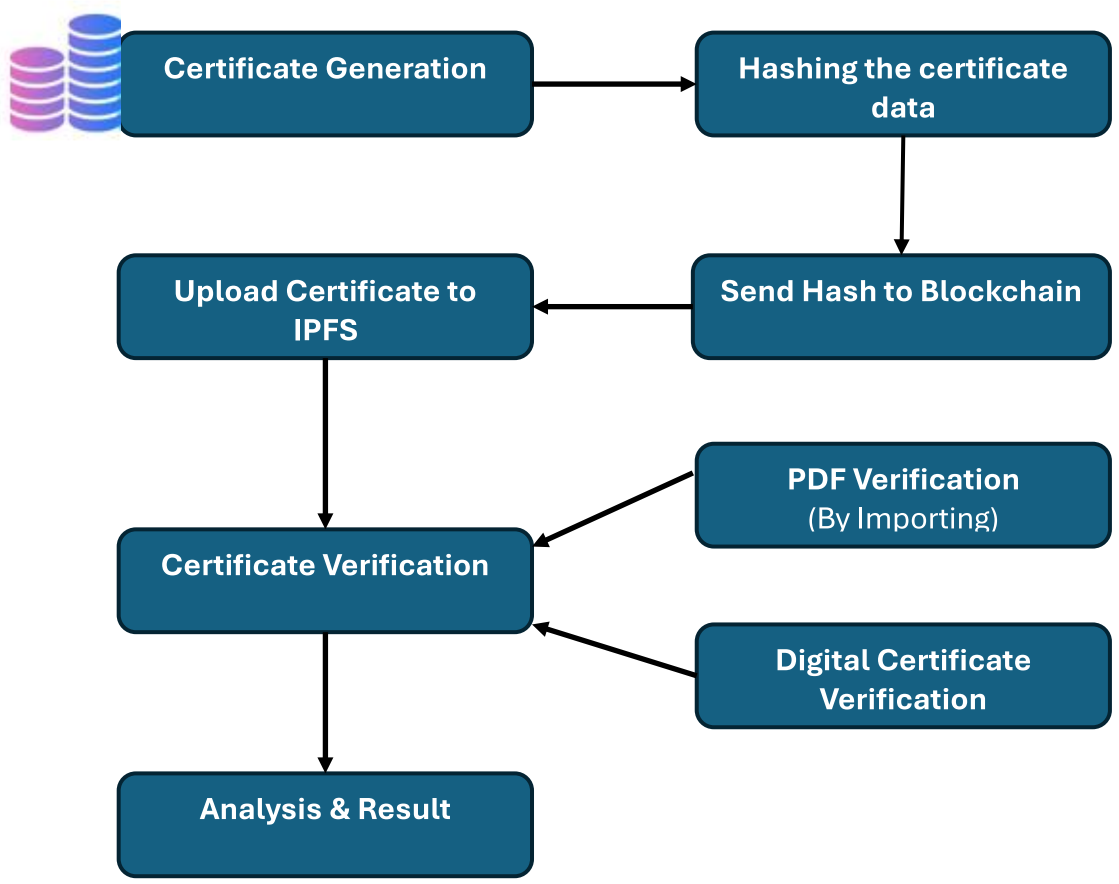

Certificate Generation
Hashing the certificate data
Send Hash to Blockchain
Upload Certificate to IPFS
PDF Verification (By Importing)
Certificate Verification
Digital Certificate Verification
Analysis & Result

Fig 6.1: System Architecture

6.2 UML DIAGRAMS

What is UML?

The Unified Modeling Language (UML) is a standard language for specifying, Visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects-oriented software and the software development process.

Advantages:

- Development time is reduced.
- The past faced issues by the developers are no longer exists.
- Has large visual elements to construct and easy to follow.
- To establish an explicit coupling between concepts and executable code.
- To creating a modeling language usable by both humans and machines UML defines several models for representing systems.

In this project six basic UML diagrams have been explained:

1. Use Case Diagram
2. Class Diagram
3. Sequence Diagram
4. Activity Diagram
5. Component Diagram
6. Deployment Diagram

6.2.1 USE CASE DIAGRAMS

A Use Case Diagram is a behavioral UML diagram that shows how users (actors) interact with a system to achieve goals. It gives a high-level view of system functionalities and includes use cases, actors, and relationships. Each use case represents a sequence of actions toward a specific goal. The diagram helps identify significant system activities from the user's perspective.

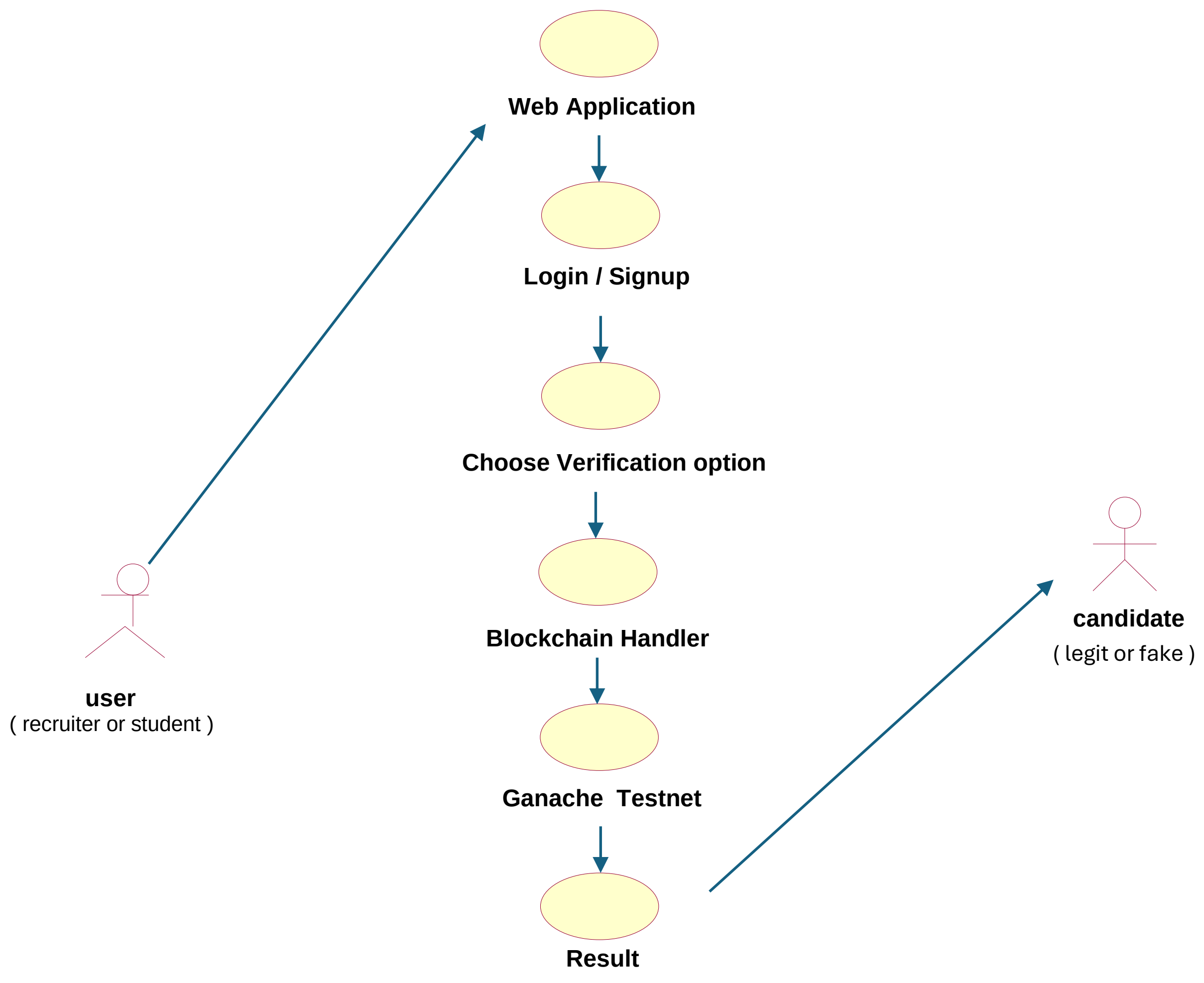


Fig 6.1.1: Use Case Diagram

6.2.2 CLASS DIAGRAMS

Class diagram is a static diagram. It represents the static view of an application. It is used for visualizing, describing, and documenting different aspects of a system and for constructing executable code. Class diagram describes the attributes and operations of a class and the constraints imposed on the system. It is widely used in modeling object-oriented systems as it can be directly mapped with object-oriented languages. Class diagram shows a collection of classes, interfaces, associations, and constraints. It is also known as a structural diagram and is useful for identifying relationships between classes and visualizing system structure. A Class Diagram consists of the following components: Classes, Attributes, Methods, Relationships.

.

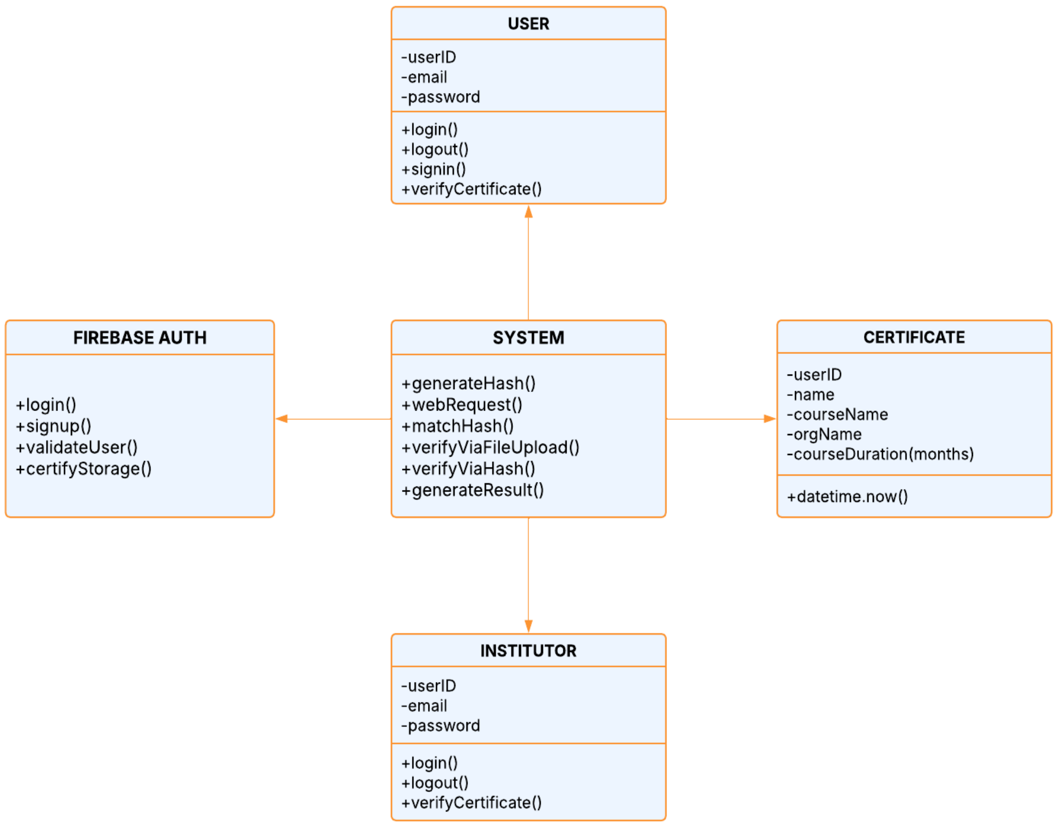
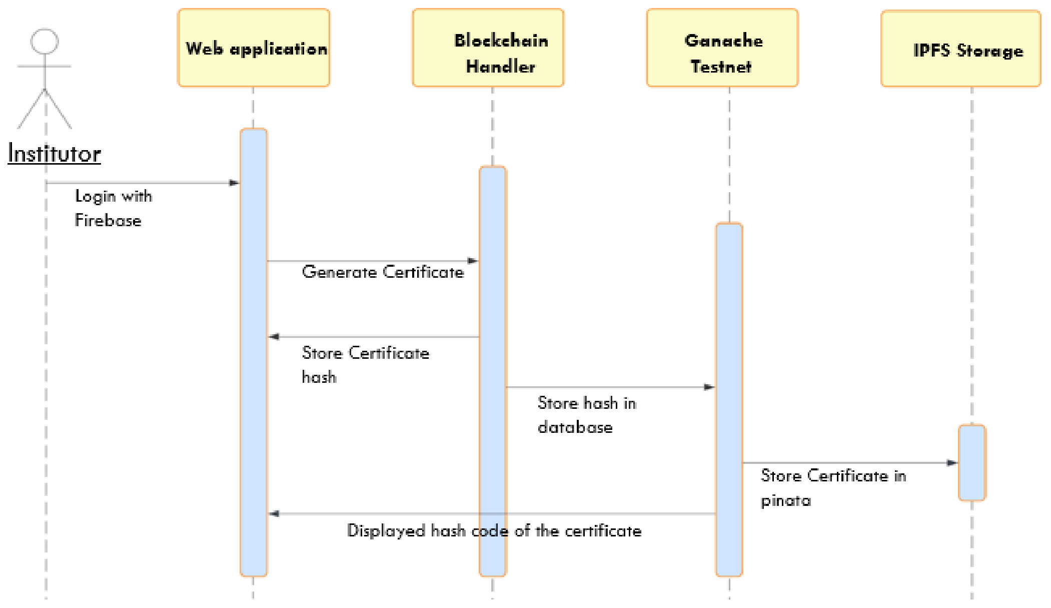


Fig 6.2.2 Class Diagram

6.2.3 SEQUENCE DIAGRAM

A Sequence diagram in Unified Modelling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A Sequence diagram depicts the sequence of actions that occur in a system. The invocation of methods in each object, and the order in which the invocation occurs is captured in a Sequence diagram. This makes the Sequence diagram a very useful tool to easily represent the dynamic behavior of a system. A sequence diagram is the most used interaction diagram. Since visualizing the interactions in a system can be a cumbersome task, we use different types of interaction diagrams to capture various features and aspects of interaction in a system.

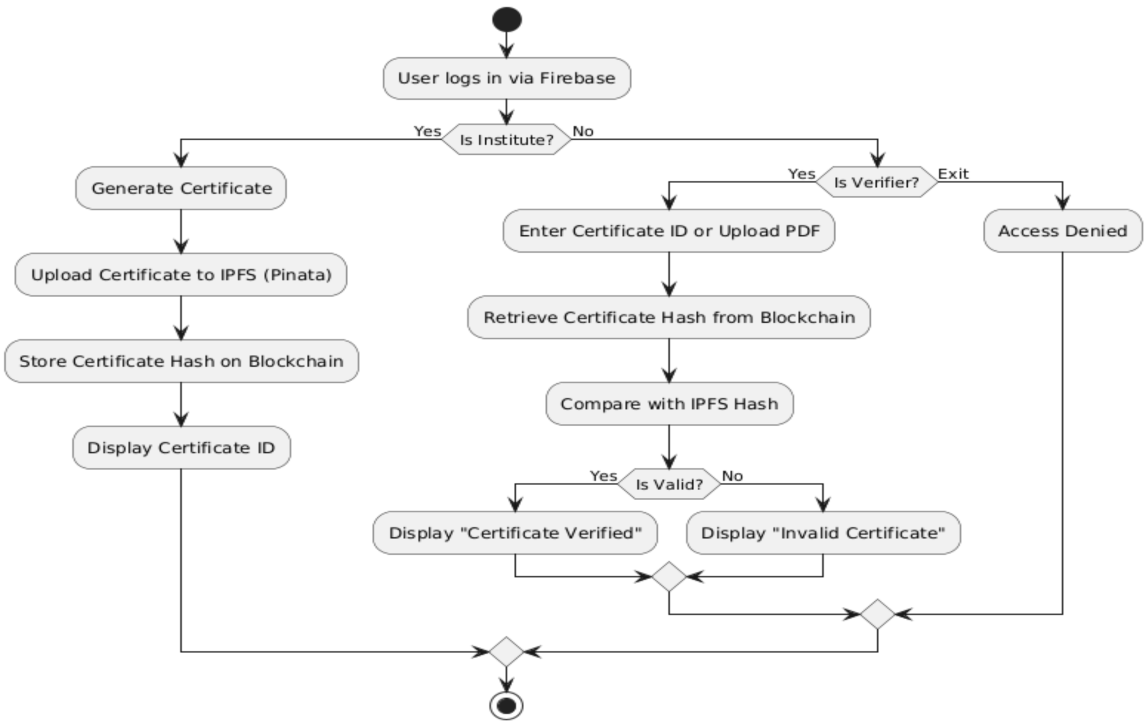
A sequence diagram simply depicts interaction between objects in a sequential order i.e., the order in which these interactions take place. Sequence diagrams describe how and in what order the objects in a system function.
A Sequence Diagram consists of the following components: Objects, Lifelines, Messages.



6.2.3 Certificate Generation Sequence Diagram.

6.2.4 ACTIVITY DIAGRAM

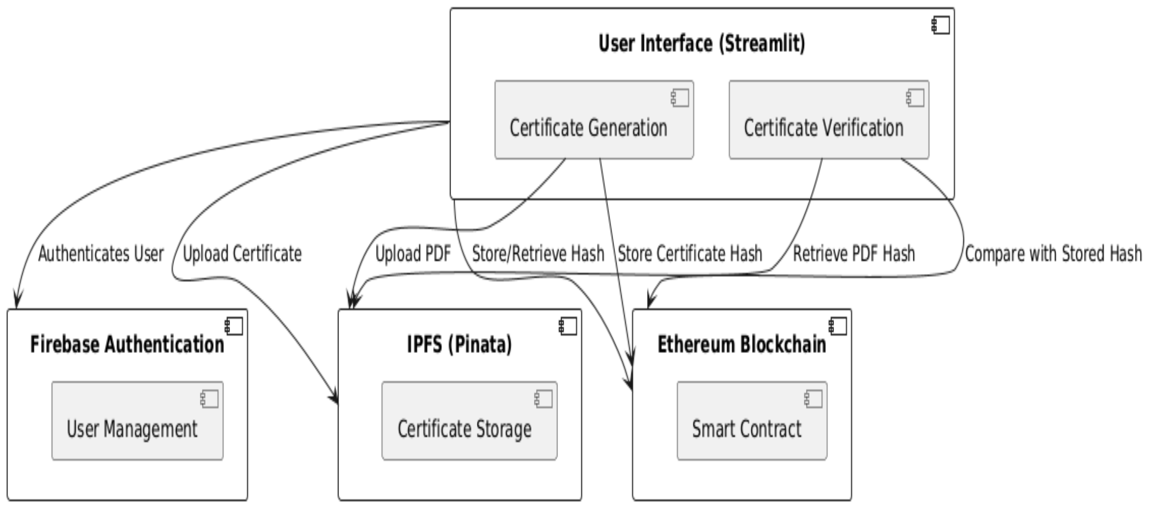
Activity diagram is another important diagram in UML to describe the dynamic aspects of the system. Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. An activity diagram is a type of UML (Unified Modeling Language) diagram that models the flow of activities or tasks in a system or process. It is a high-level view of the system that illustrates the steps that are required to accomplish a specific task or objective. Activity diagrams are useful for modeling the flow of activities or tasks in a system or process and for identifying potential errors or inefficiencies in the process. They can help to ensure that all the necessary activities are included in the process and that the process is designed to handle all possible scenarios. The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.


User logs in via Firebase
Yes
Is Institute?
No
Generate Certificate
Yes
Is Verifier?
Exit
Enter Certificate ID or Upload PDF
Access Denied
Upload Certificate to IPFS (Pinata)
Retrieve Certificate Hash from Blockchain
Store Certificate Hash on Blockchain
Compare with IPFS Hash
Display Certificate ID
Yes
Is Valid?
No
Display "Certificate Verified"
Display "Invalid Certificate"

6.2.4: Activity Diagram

6.2.5 COMPONENT DIAGRAM

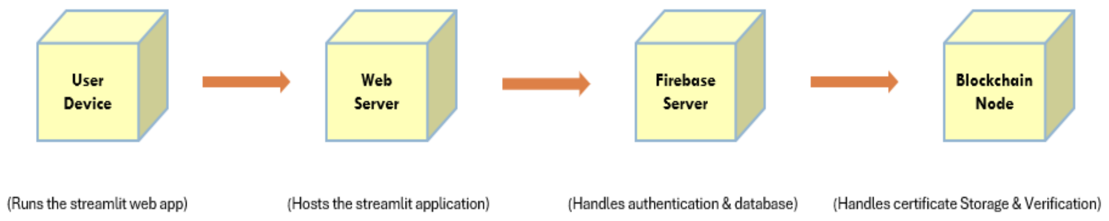
A component diagram is a type of UML (Unified Modeling Language) diagram that shows the structure of the components of a system and their interrelationships. It is a static view of the system that illustrates the physical and logical components of the system and how they work together. In a component diagram, components are represented as rectangles with the component name written inside. The interfaces of the component are shown as ports or small squares on the edges of the component. The relationships between the components are shown using connectors, such as associations or dependencies. Component diagrams are a powerful tool for system design, architecture, integration, and maintenance. They provide a clear and concise view of the system components and their relationships, which helps to ensure that the system is designed, built, and maintained correctly. Component diagrams are useful for visualizing the structure of a system and for identifying the relationships between the components.



6.2.5: Component Diagram

6.2.6 DEPLOYMENT DIAGRAM

The Deployment diagram represents the deployment view of a system. It is related to the component diagram. Because the components are deployed using the deployment diagrams. A deployment diagram consists of nodes. Nodes are nothing but physical hardware used to deploy the application. A deployment diagram is a type of UML (Unified Modeling Language) diagram that shows the physical deployment of software components and their relationships with hardware components in a system. It is a static view of the system that illustrates the physical architecture of the system and how the components are deployed on hardware devices or servers. In a deployment diagram, components are represented as nodes or boxes, and the relationships between the components are shown using connectors, such as associations or dependencies. Nodes represent physical devices, such as servers, and components represent software modules or applications. Deployment diagrams are useful for visualizing the physical deployment of software components in a system and for identifying potential issues or constraints in the deployment.


User Device
Web Server
Firebase Server
Blockchain Node
(Runs the streamlit web app)
(Hosts the streamlit application)
(Handles authentication & database)
(Handles certificate Storage & Verification)

6.2.6: Deployment Diagram

Chapter 7

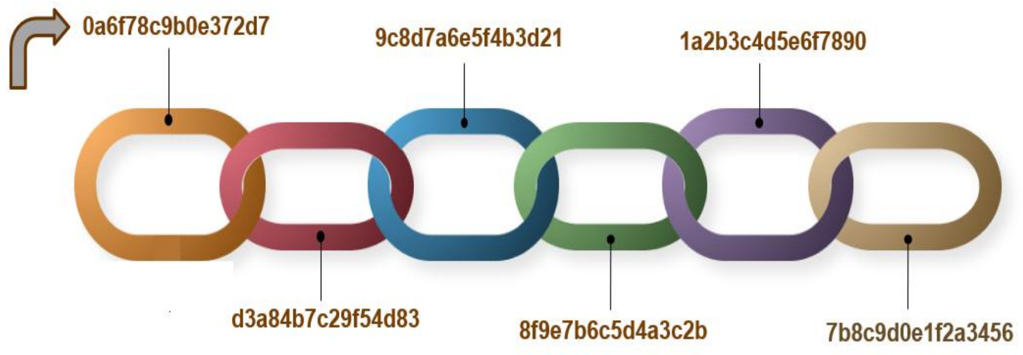
SYSTEM IMPLEMENTATION

7.1 BLOCKCHAIN HANDLING

Blockchain offers a decentralized, tamper-proof system for managing certifications, enhancing security, transparency, and reducing reliance on intermediaries. Ethereum supports smart contracts, allowing applications to run on a blockchain with data stored in blocks. To avoid real currency costs on the Ethereum mainnet, we use the Ganache framework for testing, which simulates real transactions. Our blockchain-based solution offers scalable and cost-effective credential management for both institutions and certificate holders.

EXAMPLE :

Hash Code



1.Hash: A Hash is a fixed-length string produced by hashing Algorithm**(SHA).**

2.Parent Hash: It points to the previous block's hash, ensuring continuity and linking the blocks in the chain .Parent hash links each block to its predecessor, creating a unbreakable chain.

3.Time Stamp: It represents exact time at which block is created and it follows **Unix time format .**

4.Nonce: Nonce is a “number used once” adjusted by miners to solve cryptographic puzzle, allowing them to create a valid block and add it to the blockchain.

7.2 SOLIDITY

7.2: Solidity Logo

Solidity is a programming language for creating smart contracts on blockchain platforms, mainly within the Ethereum ecosystem. Developed by Gavin Wood in 2014, it provides tools to build decentralized applications (DApps) using self-executing contracts. Solidity plays a key role in blockchain-based development.

Features of Solidity:

1. *Purposeful Design for Smart Contracts:*
Solidity's design centers on facilitating the creation of smart contracts, embodying predefined rules and conditions. This purpose-driven approach streamlines the development of decentralized applications by enabling the seamless integration of self-executing contracts into the blockchain.

2.Syntax Inspired by Familiar Languages:
Drawing inspiration from popular languages like JavaScript and C++, Solidity features a syntax that resonates with developers familiar with mainstream programming languages. This familiarity enhances accessibility for developers transitioning to blockchain development.

3.Versatile Data Types:

Solidity supports a diverse range of data types, from elementary types like integers and strings to more complex types like arrays and mappings. This versatility equips developers with the tools needed to handle a variety of data structures within their smart contracts.

Pros of Solidity:

1. Strong Ethereum Ecosystem Integration
2. Vibrant Community and Developer Support
3. Comprehensive Smart Contract Management
4. Wide Range of Development Tools and Libraries
5. High Adoption and Industry Relevance

Cons of Solidity:

1. Security Vulnerabilities and Risks
2. Steep Learning Curve for Beginners
3. Limited Debugging and Testing Tools
4. Gas Optimization Complexity
5. Rapid Language Evolution and Compatibility Issues

Chapter 8

TOOLS & TECHNOLOGIES

8.1 GANACHE

8.1: Ganache Logo

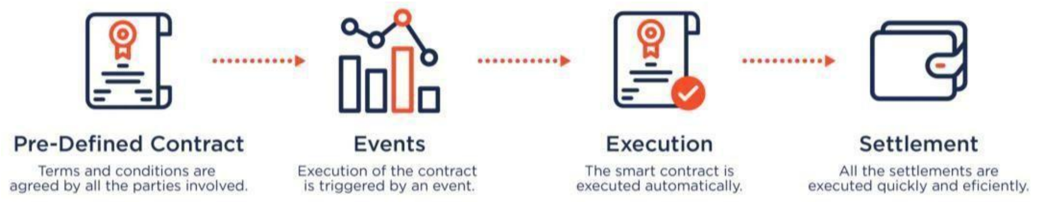
Ganache Blockchain Manager stands as a powerful tool within the blockchain development landscape, offering developers a user-friendly and efficient environment for testing and deploying smart contracts. Developed by Truffle, Ganache simplifies the complexities of blockchain interaction, making it an invaluable asset for both beginners and seasoned blockchain developers. By providing a local blockchain environment, Ganache enables developers to simulate and test their applications before deploying them on the live blockchain, fostering a streamlined and secure development process.

Ganache is particularly renowned for its ease of use and feature-rich environment, making it an essential component in the toolkit of blockchain developers working with the Ethereum network.

Features of Ganache Blockchain Manager:

1. Local Blockchain Simulation
2. User Friendly Interface
3. Quick Blockchain Deployment
4. Built-in Block Explorer

8.2 SMART CONTRACTS WORKING


Pre-Defined Contract
Terms and conditions are agreed by all the parties involved.
Events
Execution of the contract is triggered by an event.
Execution
The smart contract is executed automatically.
Settlement
All the settlements are executed quickly and eficiently.

8.2: Smart Contracts working

Smart contracts are self-executing agreements with predefined rules that automate and enforce transactions on the blockchain. Introduced on Ethereum, they ensure trust and transparency in decentralized applications and are now used across various platforms. Smart contracts revolutionize traditional contract execution by leveraging the security, immutability, and decentralized nature of blockchain technology. As an integral component of blockchain ecosystems, smart contracts facilitate the creation of trustless and tamper-resistant agreements, significantly impacting industries ranging from finance to supply chain management.

***Features of Smart Contracts*:**

1.Self-Executing Code
2.Decentralization
3.Immutable Record keeping
4.Trustless Transactions
5.Tamper-Resistant Security

Smart contracts run on decentralized networks, ensuring resilience and continuous execution without a central point of control. They automate processes, reduce manual intervention, and enhance efficiency. Their transparent and auditable nature fosters trust, allowing participants to verify terms and execution independently.

8.3 TRUFFLE WORKING

8.3: Truffle Logo

Truffle stands as a leading development framework in the blockchain ecosystem, designed to simplify and streamline the process of building decentralized applications (DApps) on the Ethereum blockchain.

Features of Truffle:

1.Smart Contract Compilation and Migration:

Truffle automates the compilation of smart contracts, simplifying the development workflow. Additionally, it facilitates the seamless migration of smart contracts to the Ethereum blockchain, ensuring a smooth transition from development to deployment.

2.Built-in Testing Framework:

Truffle incorporates a built-in testing framework that allows developers to create and execute comprehensive test suites for their smart contracts. This ensures the reliability and robustness of smart contract code before deployment, contributing to a more secure and trustworthy DApp.

4.Interactive Console (Truffle Console):

The interactive console provided by Truffle, known as the Truffle Console, allows developers to interact with deployed smart contracts in a real-time and iterative manner. This facilitates efficient debugging, testing, and exploration of contract functionalities directly from the command line.

8.4 WEB3.JS WORKING

8.4: Web3.js logo

Web3.js is a crucial JavaScript library in the realm of blockchain development, specifically tailored for interacting with Ethereum-based decentralized applications (DApps). Introduced to the community alongside Ethereum, web3.js serves as a bridge between applications and the Ethereum blockchain, enabling developers to build applications that seamlessly interact with smart contracts and decentralized networks. As an integral component of the Ethereum ecosystem, web3.js empowers developers to create dynamic and engaging user experiences while leveraging the capabilities of the blockchain.

Features of web3.js:

1.Etherum Blockchain Interaction

2.Smart Contract Integration

3.Event Handling and Listening

4.Account Management

5.Integration with Ethereum Wallets.

Chapter 9
TESTING

9. TESTING

Software testing ensures the quality and reliability of a system. In a blockchain-based document authentication system, it is vital for maintaining data integrity and security. This chapter outlines the testing levels used to ensure system robustness.

9.1 UNIT TESTING

Definition: Unit testing refers to the process of testing individual units or components of a software system in isolation. A unit is the smallest testable part of any software, typically a function, method, or object.

Importance in Blockchain System: In a blockchain-based document authentication system, unit testing is vital for testing smart contracts, individual methods involved in QR code generation, hashing functions, and user authentication modules.

Scope of Unit Testing:

- Smart Contract Functions (e.g., registerDocument, verifyDocument)
- Blockchain interactions using Web3.js
- QR Code generation and scanning functions
- Validation utilities (hash validation, input sanitization)
- User interface modules for input and output handling

Tools Used:

- **Truffle**: For testing smart contracts in the Solidity environment.
- **Ganache**: Local blockchain network for simulation.
- **Mocha & Chai**: JavaScript test framework and assertion library.

Sample Unit Test Cases:

Test Case	Input	Expected Output	Result
Register Document	Valid metadata	Document hash stored in blockchain	Pass
Verify Document	Tampered hash	Verification failure	Pass

9.1: Unit Testing

Benefits:

- Ensures correctness of individual modules
- Facilitates early detection of bugs
- Reduces the cost of defect fixing in later stages

Challenges:

- Mocking blockchain interactions for isolated testing
- Managing asynchronous behavior of blockchain confirmations

Conclusion: Unit testing provides a strong foundation for ensuring each component works as expected in isolation before integrating it into the broader system. It serves as a preventive mechanism against critical failures in production.

9.2 INTEGRATION TESTING

Definition: Integration testing is performed to test the interaction between modules and ensure that combined components function correctly as a group

Importance in Blockchain System: Integration testing is essential in a decentralized system where multiple components—smart contracts, UI, backend services, and blockchain nodes—must work together seamlessly.

Scope of Integration Testing:

- Interaction between frontend UI and blockchain through Web3.js
- Smart contract invocation from JavaScript modules
- QR code scanner communicating with hash verification module
- Document upload and retrieval from IPFS or decentralized storage.

Integration Test Scenarios:

1. **End-to-End Document Registration Flow:**
    - Upload a document
    - Generate hash and QR code
    - Store in blockchain
    - Verify success status

2. **Verification Workflow:**
    - Scan QR
    - Extract and match hash
    - Display verification result

3. **Unauthorized Access Prevention:**
    - Attempt to modify blockchain entry without permission
    - Expect transaction rejection

Tools Used:

- Postman (for API testing)
- Web3.js test scripts
- Truffle Suite (for end-to-end blockchain testing)

Sample Integration Issues Detected:

- Incompatible hash encoding between frontend and backend
- UI failure due to delayed blockchain confirmation
- Invalid QR code generation under specific edge cases

Benefits:

- Validates end-to-end functionality
- Catches interface mismatches early
- Improves system reliability under real-world usage

Conclusion: Integration testing ensures smooth communication among modules, revealing errors that unit tests cannot uncover. It is critical to confirm that components operate cohesively in a complex decentralized ecosystem.

9.3 SYSTEM TESTING

Definition: System testing is a high-level testing phase where the complete and fully integrated software system is tested in a simulated production environment. It focuses on validating the system's compliance with specified requirements.

Relevance to Blockchain System: In the context of a blockchain-based document authentication system, system testing ensures that the entire application—including user registration, document upload, blockchain interaction, QR code generation, and verification—functions as expected under realistic conditions.

Objectives of System Testing:

- To validate the end-to-end workflow
- To check for functional completeness
- To test performance under real use-cases
- To evaluate fault tolerance and error handling

Scope of System Testing:

- Smart contract deployment and execution
- Frontend interaction with blockchain and smart contracts
- Verification process for document authenticity
- UI response and feedback messages

Functional Test Cases:

Test Case	Expected Outcome
Register a new user	User stored and redirected to dashboard
Upload a document and register it	Document hash recorded on blockchain
Scan Cert details from uploaded document	Validation result shown to user
Attempt to modify a verified document	Change rejected and user alerted
System reacts to failed blockchain transaction	Error message displayed

9.2: Functional Testing

Non-Functional Testing Under System Testing:

- **Performance Testing**: Time taken to generate QR, write to blockchain.
- **Security Testing**: Tampering resistance, injection protection.
- **Compatibility Testing**: Different browsers, operating systems.
- **Usability Testing**: UI responsiveness, ease of use.

Test Environment Setup:

- Truffle and Ganache for smart contract testing
- React frontend environment
- Simulated IPFS node for decentralized storage
- Browser with extension-enabled QR reader

Challenges Encountered:

- Synchronization issues with blockchain confirmations
- QR codes not rendering properly on older browsers
- Occasional delays in hash verification due to blockchain congestion

Conclusion: System testing validates that the application as a whole behaves according to expectations. It uncovers defects related to integration, performance, and behavior in real-world usage, ensuring a robust and dependable blockchain solution for document authentication.

9.4 ACCEPTANCE TESTING

Definition: Acceptance testing is the final phase of software testing where the system is evaluated against business requirements to determine whether it is ready for delivery. It often involves feedback from end-users or stakeholders.

Purpose for Document Authentication System: For a blockchain-based document authentication system, acceptance testing ensures that the application solves the core problem—authenticating documents securely and transparently—and is ready for deployment in real scenarios like education certificates, legal documents, etc.

Types of Acceptance Testing:

- **User Acceptance Testing (UAT):** Conducted by end-users to validate system behavior.
- **Regulatory Acceptance Testing:** Ensures compliance with legal standards (e.g., data protection).
- **Alpha Testing:** Performed in-house by developers or QA teams.
- **Beta Testing:** Performed by external users in a controlled environment.

Key Acceptance Criteria:

- Users can register and log in without technical help.
- Users can successfully upload and authenticate documents.

- Verifiers can scan and verify documents without system errors.
- System maintains tamper-proof records.
- System complies with data protection and blockchain standards.

UAT Test Scenarios:

Scenario	Outcome
University admin uploads student certificate	Receives a tamper-proof digital Signature for attachment
Employer scans a candidate's certificate	Gets successful verification message
User scans a fake certificate	Verification failure is shown
Admin attempts to register duplicate document	Alert shown and action rejected

9.3: UAT Testing

Conclusion: Acceptance testing validates the business and functional readiness of the system. With successful UAT and stakeholder approval, the blockchain-based document authentication system is proven to meet its intended goals of security, efficiency, and transparency.

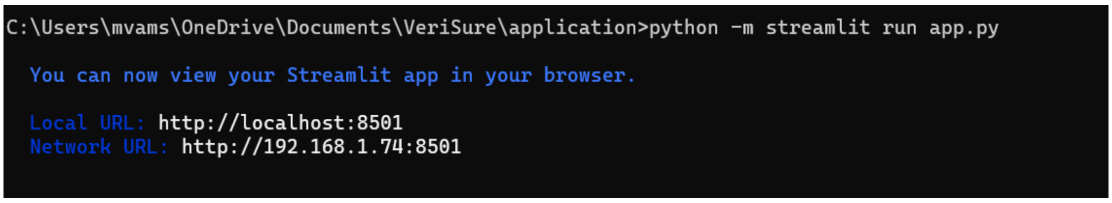
Chapter 10
SOURCE CODE

10.1 SERVER IMPLEMENTATION

Ganache serves as a crucial tool in the development of the Fake document Identification System by providing a local and controlled blockchain environment for testing. Integrated with Truffle, Ganache facilitates a seamless development workflow, allowing for the deployment of smart contracts to a local blockchain.
In this setup:

1. Local Blockchain Environment
2. Controlled Workspace
3. Smart Contract Deployment
4. Quick Feedback Loop
5. Blockchain Explorer
6. No Real Ether Transactions

In essence, Ganache's role in test blockchain handling is to enhance development efficiency by providing a simulated yet realistic testing environment for smart contracts in the Fake document Identification System.



```
C:\Users\mvams\OneDrive\Documents\VeriSure\application>python -m streamlit run app.py

  You can now view your Streamlit app in your browser.

  Local URL: http://localhost:8501
  Network URL: http://192.168.1.74:8501
```

Fig 10.1: Server Implementation

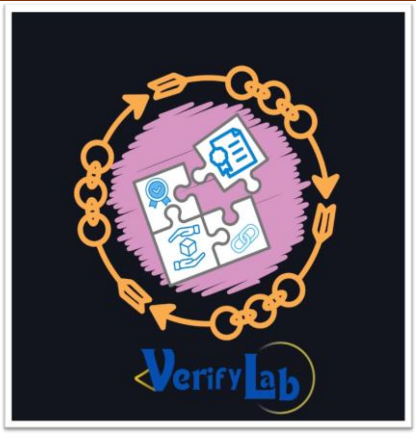


Fig 10.1.1: Our Project Logo

// Our project ensures the trust worthiness between organizations and users by integrating tamper-proof technology.

10.2 **CODE :** migrations. sol

```
pragma solidity ^0.8.13;
contract Migrations {
address public owner;
uint public last_completed_migration;
modifier restricted() {
if (msg.sender == owner) _;
}
constructor() {
owner = msg.sender;
}
function setCompleted(uint completed) public restricted {
last_completed_migration = completed;
}
function upgrade(address new_address) public restricted {
Migrations upgraded = Migrations(new_address);
upgraded.setCompleted(last_completed_migration);
}
}
```

10.3 CODE : initial_migration.js

```
let Migrations = artifacts.require("./Migrations.sol");

module.exports = function(deployer) {
  deployer.deploy(Migrations);
};
```

10.4 CODE : deploy-contract.js

```
let Certification = artifacts.require("./Certification.sol");
const fs = require('fs');
module.exports = async function (deployer) {
 await deployer.deploy(Certification);
 const deployedCertification = await Certification.deployed();
 // Always start with an empty object for configData
  let configData = {};
 // Update or add the contract address
  configData.Certification = deployedCertification.address;
  // Save the updated configuration back to the file
  fs.writeFileSync('./deployment_config.json', JSON.stringify(configData,
 null, 2));
 console.log(`Certification contract deployed at address:
${deployedCertification.address}`);
};
```

10.5 CODE : cert-utils.py

```
import pdfplumber
from reportlab.lib.pagesizes import letter, landscape
from reportlab.pdfgen import canvas
from reportlab.lib.colors import HexColor
from datetime import datetime
import re


def generate_certificate(pdf_file_path, uid, candidate_name, course_name, org_name,
logo_path):
    """
    Generates a PDF certificate with professional formatting and design.
    """

    # Asset paths (should ideally be passed as parameters or configured)
    signature_path = r"C:\Users\mvams\Downloads\MySignNorm.png"
    slide_design_path = r"C:\Users\mvams\Downloads\slideDesign.jpg"
    award_logo_path = r"C:\Users\mvams\Downloads\AwardLogo.png"

    # Create canvas
    c = canvas.Canvas(pdf_file_path, pagesize=landscape(letter))
    width, height = landscape(letter)

    # Set styles

 border_color = HexColor("#001f3f")

  text_color = HexColor("#2E2E2E")
```

```
    # Draw border
    border_margin = 20
    c.setStrokeColor(border_color)
    c.setLineWidth(4)
    c.rect(border_margin, border_margin, width - 2 * border_margin, height - 2 *
border_margin)

    # Add logos and design
    c.drawImage(award_logo_path, width - 320, height - 300, width=300, height=280)
    c.drawImage(slide_design_path, border_margin, height - 520, width=500,
height=500)
    c.drawImage(logo_path, (width - 200) / 2, height - 540, width=200, height=100)

    # Certificate headings
    c.setFont("Helvetica-Bold", 50)
    c.setFillColor(text_color)
    c.drawCentredString(width / 2, height - 140, "CERTIFICATE")

    c.setFont("Helvetica", 25)
    c.drawCentredString(width / 2, height - 180, "OF ACHIEVEMENT")

    # Candidate name
    c.setFont("Helvetica-Bold", 20)

c.drawCentredString(width / 2, height - 260, "THIS IS TO CERTIFY THAT")

    c.setFont("Helvetica-Bold", 36)
    c.drawCentredString(width / 2, height - 330, candidate_name)
    # Course description
    c.setFont("Helvetica", 14)
    c.drawString(border_margin + 60, height - 370, f"has successfully completed the
course {course_name} with a duration of 6 months.")
    c.drawString(border_margin + 60, height - 390, "In recognition of outstanding
performance and commitment, this certificate is awarded.")

    # Signature block
    c.drawImage(signature_path, 55, 120, width=170, height=50)
    c.setFont("Helvetica-Bold", 12)
    c.drawString(95, 120, "(Vamsi Mandala)")
    c.line(border_margin + 60, 115, 195, 115)
    c.setFont("Helvetica", 14)
    c.drawString(95, 95, "Manager")

    # Issue date
    current_date = datetime.now().strftime("%d %B %Y")

  c.setFont("Helvetica-Bold", 12)
    c.drawString(width - 185, 120, current_date)
    c.line(width - 195, 115, width - 80, 115)
    c.setFont("Helvetica", 14)

 c.drawString(width - 165, 95, "Issue Date")

    # Footer metadata
    c.setFont("Helvetica-Bold", 10)
    c.drawString(width / 2 - 180, height - 560, "Certificate ID:")
    c.setFont("Helvetica", 10)
    c.drawString(width / 2 - 100, height - 560, uid)
```

```
c.setFont("Helvetica-Bold", 10)
    c.drawString(width / 2 + 30, height - 560, "Issued by:")
    c.setFont("Helvetica", 10)
    c.drawString(width / 2 + 90, height - 560, org_name)

    # Save certificate
    c.save()



def extract_certificate(pdf_path):
    """
    Extracts key certificate details from the generated PDF using pdfplumber.

   """


    extracted_text = ""
    with pdfplumber.open(pdf_path) as pdf:
        for page in pdf.pages:
            extracted_text += page.extract_text() + "\n"

    lines = extracted_text.splitlines()


    # Extract Certificate ID
    uid_match = re.search(r"Certificate ID:\s*([\w\d]+)", extracted_text)
    uid = uid_match.group(1).strip() if uid_match else "Not Found"

    # Extract Organization Name
    org_match = re.search(r"Issued by:\s*(.+)", extracted_text)
    org_name = org_match.group(1).strip() if org_match else "Not Found"
    # Extract Candidate Name
    try:
        name_index = lines.index("THIS IS TO CERTIFY THAT") + 1
        candidate_name = lines[name_index].strip()
    except (ValueError, IndexError):
        candidate_name = "Not Found"

    # Extract Course Name

course_match = re.search(r"has successfully completed the course\s+(.+?)\s+with a duration", extracted_text)

course_name = course_match.group(1).strip() if course_match else "Not Found"

    return uid, candidate_name, course_name, org_name
```

10.6 CODE : institute.py

```
import streamlit as st
import requests
import json
import os
import hashlib
from dotenv import load_dotenv

from utils.cert_utils import generate_certificate
from utils.streamlit_utils import view_certificate, hide_icons, hide_sidebar, remove_whitespaces
from streamlit_extras.switch_page_button import switch_page
from connection import contract, w3
```

```
# Set up page layout
st.set_page_config(layout="wide", initial_sidebar_state="collapsed")
hide_icons()

hide_sidebar()
remove_whitespaces()

# Sidebar logout option
sideb = st.sidebar
if sideb.button("Logout"):
    switch_page('app')

# Load environment variables for Pinata

load_dotenv()
api_key = os.getenv("PINATA_API_KEY")
api_secret = os.getenv("PINATA_API_SECRET")

# Upload generated certificate to IPFS using Pinata
def upload_to_pinata(file_path, api_key, api_secret):
    pinata_api_url = "https://api.pinata.cloud/pinning/pinFileToIPFS"
    headers = {
        "pinata_api_key": api_key,
        "pinata_secret_api_key": api_secret,
    }

    with open(file_path, "rb") as file:
        files = {"file": (file.name, file)}
        response = requests.post(pinata_api_url, headers=headers, files=files)

 result = json.loads(response.text)
```

```
        if "IpfsHash" in result:
            return result["IpfsHash"]
        else:
            print("Error uploading to Pinata:", result.get("error", "Unknown error"))
            return None

# Certificate generation or view selection
options = ("Generate Certificate", "View Certificates")

selected = st.selectbox("", options, label_visibility="hidden")

# Certificate Generation Section
if selected == "Generate Certificate":
    form = st.form("Generate-Certificate")
    uid = form.text_input("UID")
    candidate_name = form.text_input("Name")
    course_name = form.text_input("Course Name")
    org_name = form.text_input("Organization Name")
    submit = form.form_submit_button("Submit")

    if submit:
        pdf_file_path = "certificate.pdf"
        logo_path = "C:/Users/mvams/Downloads/ProjectOrg.jpg"

        generate_certificate(pdf_file_path, uid, candidate_name, course_name,
org_name, logo_path)

        ipfs_hash = upload_to_pinata(pdf_file_path, api_key, api_secret)
        os.remove(pdf_file_path)
```

```
 data_to_hash  =  f"{uid}{candidate_name}{course_name}{org_name}".encode("utf-
8")
        certificate_id = hashlib.sha256(data_to_hash).hexdigest()

        contract.functions.generateCertificate(

 certificate_id, uid, candidate_name, course_name, org_name, ipfs_hash
        ).transact({'from': w3.eth.accounts[1]})

        st.success(f"Certificate generated successfully with ID: {certificate_id}")

# Certificate Viewing Section
else:
    form = st.form("View-Certificate")
    certificate_id = form.text_input("Enter the Certificate ID")
    submit = form.form_submit_button("Submit")

    if submit:
        try:

view_certificate(certificate_id)
        except Exception:
            st.error("Invalid Certificate ID!")
```

10.7 CODE : verifiers.py

```
import streamlit as st
import os
import hashlib

from utils.cert_utils import extract_certificate
from utils.streamlit_utils import view_certificate, displayPDF, hide_icons,
hide_sidebar, remove_whitespaces
from connection import contract
from streamlit_extras.switch_page_button import switch_page

# Configure page layout
st.set_page_config(layout="wide", initial_sidebar_state="collapsed")

# Hide UI elements as per custom theme
hide_icons()
hide_sidebar()
remove_whitespaces()

# Logout button in sidebar
if st.sidebar.button("Logout"):
    switch_page("app")

# Certificate verification options

options = ("Verify Certificate using PDF", "View/Verify Certificate using Certificate
ID")
selected = st.selectbox("", options, label_visibility="hidden")

# Option 1: Verify by uploading PDF
if selected == options[0]:
  uploaded_file = st.file_uploader("Upload the PDF version of the certificate")

   if uploaded_file is not None:
       # Save uploaded file temporarily
       with open("certificate.pdf", "wb") as file:
           file.write(uploaded_file.getvalue())

       try:
           # Extract certificate details
           uid, candidate_name, course_name, org_name =
extract_certificate("certificate.pdf")
           displayPDF("certificate.pdf")
           os.remove("certificate.pdf")

           # Generate hash for verification
           data_to_hash =
f"{uid}{candidate_name}{course_name}{org_name}".encode("utf-8")
           certificate_id = hashlib.sha256(data_to_hash).hexdigest()

           # Verify certificate ID using smart contract
           result = contract.functions.isVerified(certificate_id).call()

           if result:
               st.success("Certificate validated successfully!")
           else:

  st.error("Invalid Certificate! It may have been tampered.")

       except Exception as e:
           st.error(f"Error: {str(e)}. The certificate might be tampered or unreadable.")
```

```
# Option 2: Verify using Certificate ID

elif selected == options[1]:
    form = st.form("Validate-Certificate")
    certificate_id = form.text_input("Enter the Certificate ID")
    submit = form.form_submit_button("Validate")

    if submit:
        try:
            # View certificate details
            view_certificate(certificate_id)

            # Validate Certificate ID on blockchain
            result = contract.functions.isVerified(certificate_id).call()

            if result:
                st.success("Certificate validated successfully!")
            else:
                st.error("Invalid Certificate ID!")

        except Exception:
            st.error("Invalid Certificate ID or unable to retrieve data.")
```

10.8 CODE : app.py

```
import streamlit as st
from PIL import Image
from utils.streamlit_utils import hide_icons, hide_sidebar, remove_whitespaces
from streamlit_extras.switch_page_button import switch_page
from streamlit.components.v1 import html
```

```
# JavaScript code to connect MetaMask
js_code = """

<script>
    async function connectMetaMask() {
        if (window.ethereum) {
            try {
                await window.ethereum.request({ method: 'eth_requestAccounts' });
                const account = window.ethereum.selectedAddress;
                document.getElementById('wallet').textContent = account;
            } catch (error) {
                console.error(error);
                alert('Connection failed.');
            }
        } else {
            alert('MetaMask not installed!');
        }
    }
</script>
<button onclick="connectMetaMask()">Connect MetaMask</button>
<p id="wallet">Not connected</p>
"""
html(js_code)



# Set up page layout

st.set_page_config(layout="wide", initial_sidebar_state="collapsed")
hide_icons()
hide_sidebar()
remove_whitespaces()
```

```
# Centered logo display
col1, col2, col3 = st.columns([1, 2, 1])
with col2:
    st.image(".assets/MyLogo3.gif", width=300)

# Title and role selection
st.title("VeriFicate")
st.subheader("Select Your Role")

col1, col2 = st.columns(2)

# Institute role selection
institite_logo = Image.open("../assets/institute_logo.png")
with col1:
    st.image(institite_logo, width=230)
    clicked_institute = st.button("Institute")

# Verifier role selection
company_logo = Image.open("../assets/company_logo.jpg")
with col2:
    st.image(company_logo, width=230)
    clicked_verifier = st.button("Verifier")

# Page switching based on role

if clicked_institute:
    st.session_state.profile = "Institute"

   switch_page("login")

elif clicked_verifier:
    st.session_state.profile = "Verifier"
    switch_page("login")
```

10.7 CODE : institute.py

```
import streamlit as st
import requests
import json
import os
from dotenv import load_dotenv
import hashlib
from utils.cert_utils import generate_certificate
from utils.streamlit_utils import view_certificate
from connection import contract, w3
from utils.streamlit_utils import hide_icons, hide_sidebar, remove_whitespaces
from streamlit_extras.switch_page_button import switch_page

st.set_page_config(layout="wide", initial_sidebar_state="collapsed")
sideb = st.sidebar
check1 = sideb.button("Logout")
if check1:
    switch_page('app')
hide_icons()
hide_sidebar()
remove_whitespaces()

load_dotenv()

api_key = os.getenv("PINATA_API_KEY")
api_secret = os.getenv("PINATA_API_SECRET")
```

```
def upload_to_pinata(file_path, api_key, api_secret):
    # Set up the Pinata API endpoint and headers
    pinata_api_url = "https://api.pinata.cloud/pinning/pinFileToIPFS"
    headers = {
        "pinata_api_key": api_key,
        "pinata_secret_api_key": api_secret,
    }

    # Prepare the file for upload
    with open(file_path, "rb") as file:
        files = {"file": (file.name, file)}

        # Make the request to Pinata
        response = requests.post(pinata_api_url, headers=headers, files=files)

        # Parse the response
        result = json.loads(response.text)

        if "IpfsHash" in result:
            ipfs_hash = result["IpfsHash"]
            print(f"File uploaded to Pinata. IPFS Hash: {ipfs_hash}")
            return ipfs_hash
        else:
            print(f"Error uploading to Pinata: {result.get('error', 'Unknown error')}")
            return None
options = ("Generate Certificate", "View Certificates")
selected = st.selectbox("", options, label_visibility="hidden")

if selected == options[0]:
    form = st.form("Generate-Certificate")
    uid = form.text_input(label="UID")
```

```
candidate_name = form.text_input(label="Name")
    course_name = form.text_input(label="Course Name")
    org_name = form.text_input(label="Org Name")



    submit = form.form_submit_button("Submit")
    if submit:
        pdf_file_path = "certificate.pdf"
        logo_path = "C:/Users/mvams/Downloads/ProjectOrg.jpg"

                generate_certificate(pdf_file_path, uid, candidate_name, course_name,
org_name, logo_path)

        # Upload the PDF to Pinata
        ipfs_hash = upload_to_pinata(pdf_file_path, api_key, api_secret)
        os.remove(pdf_file_path)
        data_to_hash = f"{uid}{candidate_name}{course_name}{org_name}"
        .encode('utf-8')
        certificate_id = hashlib.sha256(data_to_hash).hexdigest()

        # Smart Contract Call
                contract.functions.generateCertificate(certificate_id, uid, candidate_name,
course_name, org_name, ipfs_hash).transact({'from': w3.eth.accounts[1]})
                st.success(f"Certificate successfully generated with Certificate ID:
{certificate_id}")

else:
    form = st.form("View-Certificate")
    certificate_id = form.text_input("Enter the Certificate ID")
    submit = form.form_submit_button("Submit")
    if submit:
```

```
    try:
        view_certificate(certificate_id)
    except Exception as e:
        st.error("Invalid Certificate ID!")
```

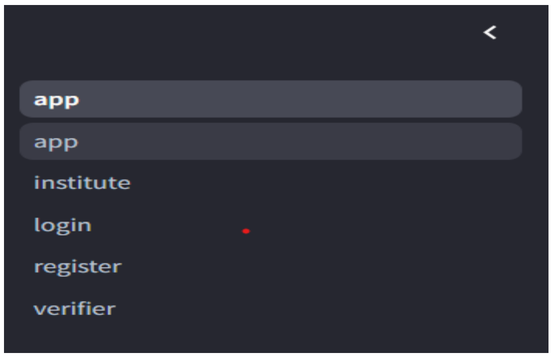
10.7 CODE : connection.py

```
import json
from pathlib import Path
from web3 import Web3
w3 = Web3(Web3.HTTPProvider('http://127.0.0.1:7545'))

def get_contract_abi():
    certification_json_path = Path('../build/contracts/Certification.json')

    try:
        with open(certification_json_path, 'r') as json_file:
            certification_data = json.load(json_file)
            return certification_data.get('abi', [])
    except FileNotFoundError:
        print(f"Error: {certification_json_path} not found.")
        return []

contract_abi = get_contract_abi()
deployment_config_fpath = Path("../deployment_config.json")
with open(deployment_config_fpath, 'r') as json_file:
    address_data = json.load(json_file)
contract_address = address_data.get('Certification')

contract = w3.eth.contract(address=contract_address, abi=contract_abi)
```

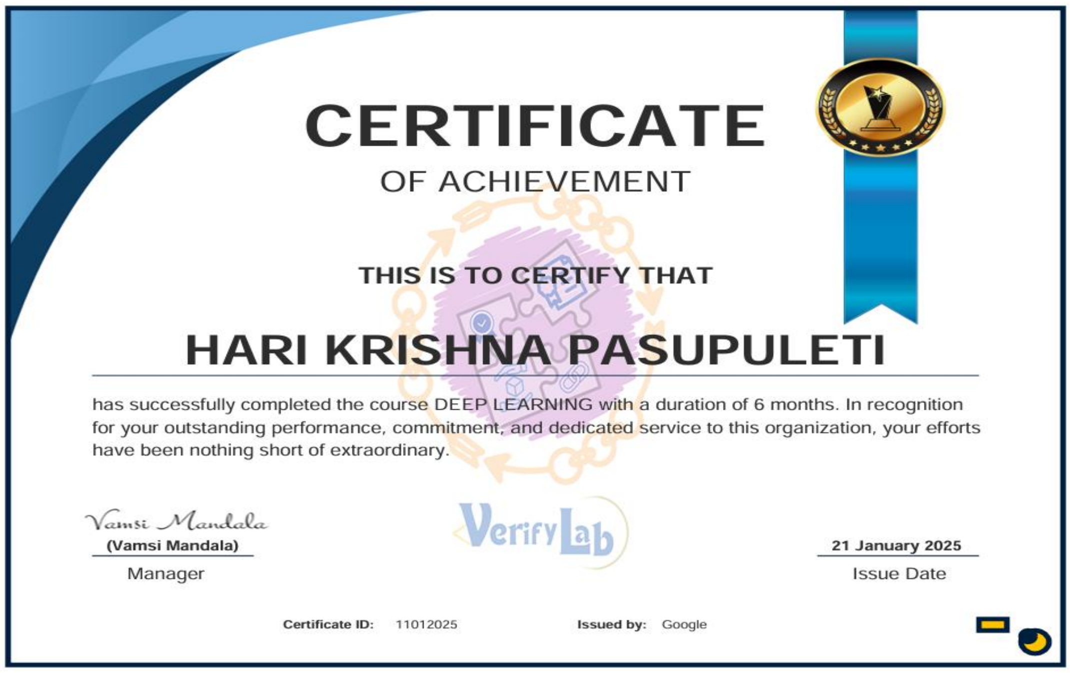
Chapter 11

SCREEN LAYOUTS

// After running streamlit script elements (**python -m streamlit run app.py)** in cmd Interface, our website contains following elements.



11.1 CERTIFICATE LAYOUT : certification (demo)



CERTIFICATE
OF ACHIEVEMENT

THIS IS TO CERTIFY THAT

HARI KRISHNA PASUPULETI

has successfully completed the course DEEP LEARNING with a duration of 6 months. In recognition for your outstanding performance, commitment, and dedicated service to this organization, your efforts have been nothing short of extraordinary.

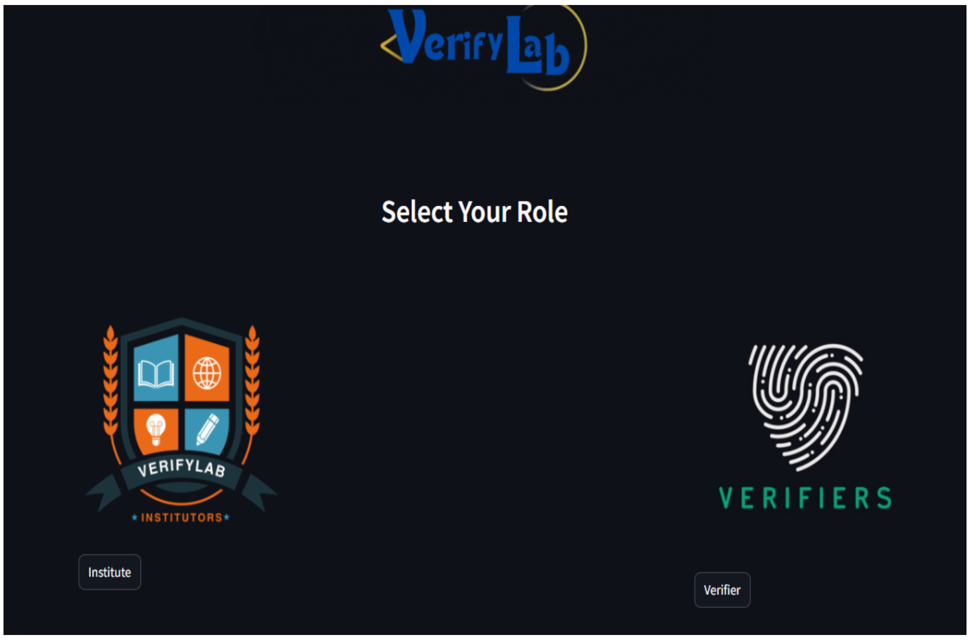
Vamsi Mandala
(Vamsi Mandala)
Manager

VerifyLab

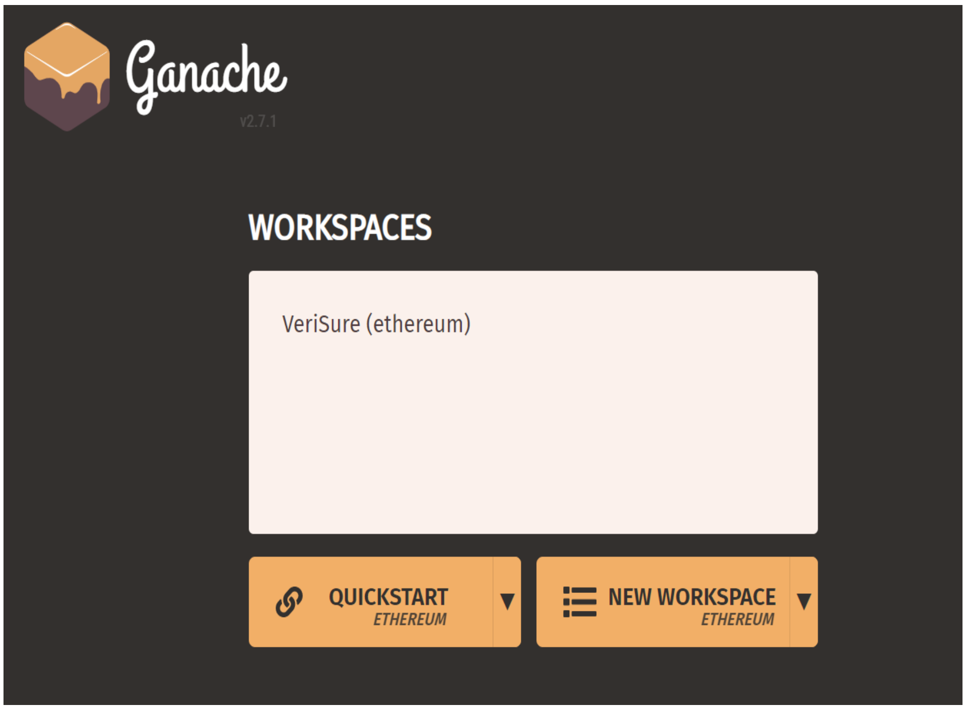
21 January 2025
Issue Date

Certificate ID: 11012025 Issued by: Google

11.2 WEBSITE LAYOUT : web interface (streamlit)

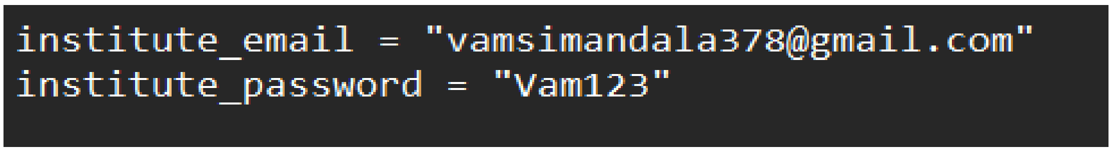


11.3 : GANACHE LAYOUT : blockchain (server)



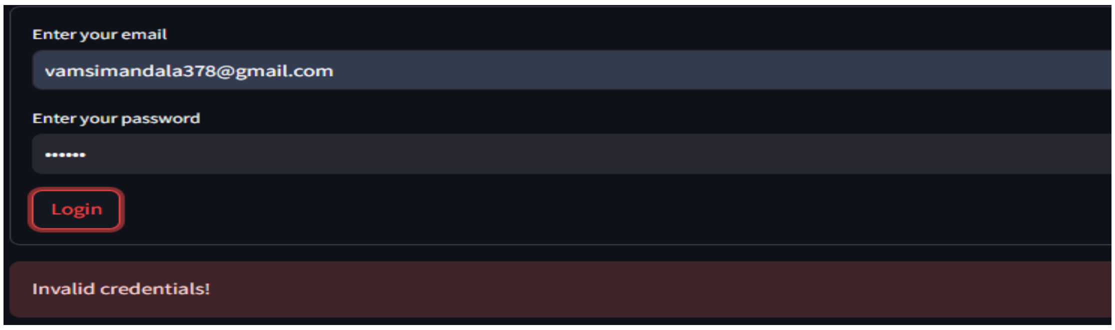
11.4 : CERTIFICATE GEN LAYOUT :

1. To access this first we want sign in with authorized credentials this certificate generation only accessible for authorized persons only. It is initialized in **.env** file



```
institute_email = "vamsimandala378@gmail.com"
institute_password = "Vam123"
```

2. If entered details are incorrect it shows like this "Invalid Credentials!!"



Enter your email

vamsimandala378@gmail.com

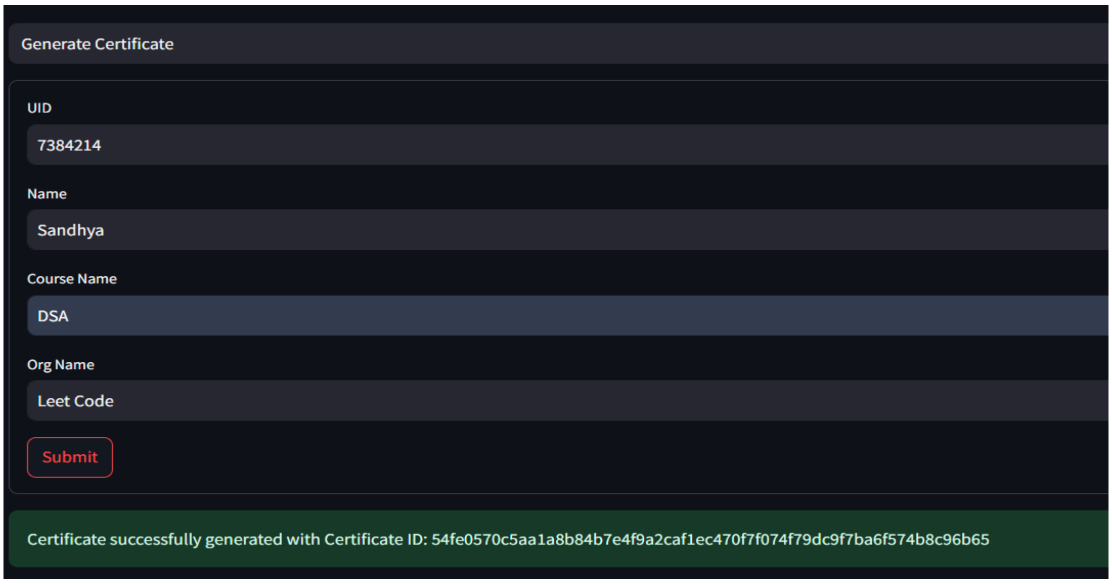
Enter your password

••••••

Login

Invalid credentials!

3. Certificate digital signature is generated after clicking the submit button by the institutor.



Generate Certificate

UID

7384214

Name

Sandhya

Course Name

DSA

Org Name

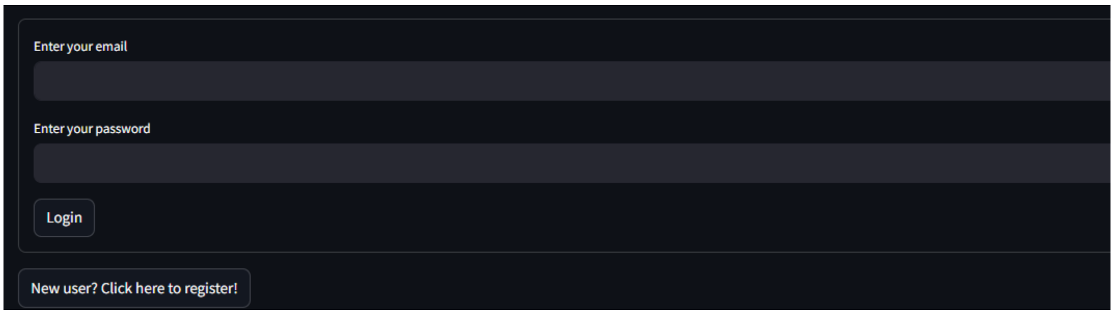
Leet Code

Submit

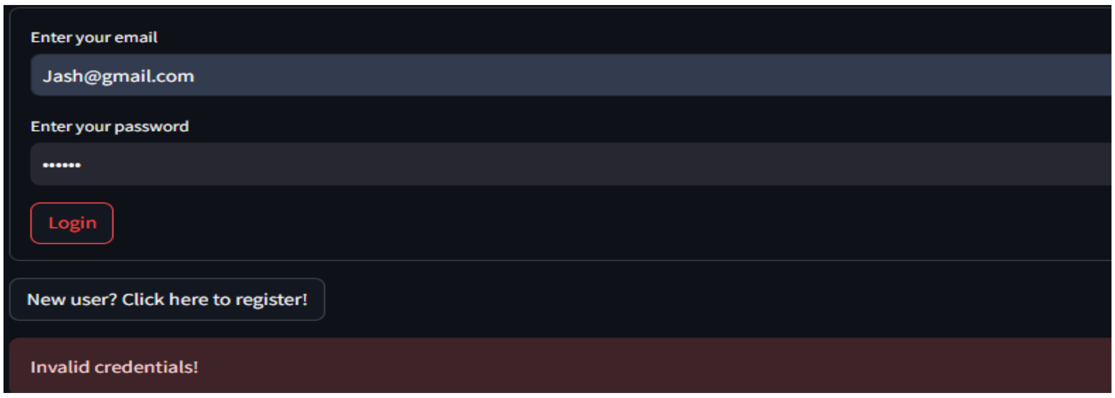
Certificate successfully generated with Certificate ID: 54fe0570c5aa1a8b84b7e4f9a2caf1ec470f7f074f79dc9f7ba6f574b8c96b65

11.5 : CERTIFICATE VERIFICATION LAYOUT :

1. In this verification layout we have two options one is sign in (if you've already registered) and another is sign up (new user).



2. If you choose sign in, but if you entered wrong credentials it shows like this "Invalid credentials!!"

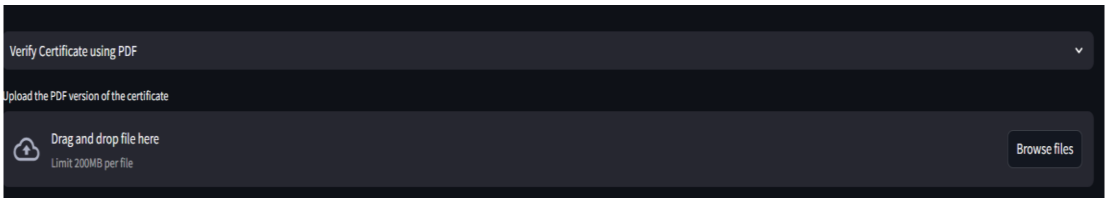


Jash@gmail.com

••••••

Invalid credentials!

11.5.1 : VERIFICATION MECHANISM - I: (by importing)


Verify Certificate using PDF
Upload the PDF version of the certificate
Drag and drop file here
Limit 200MB per file
Browse files

Steps :

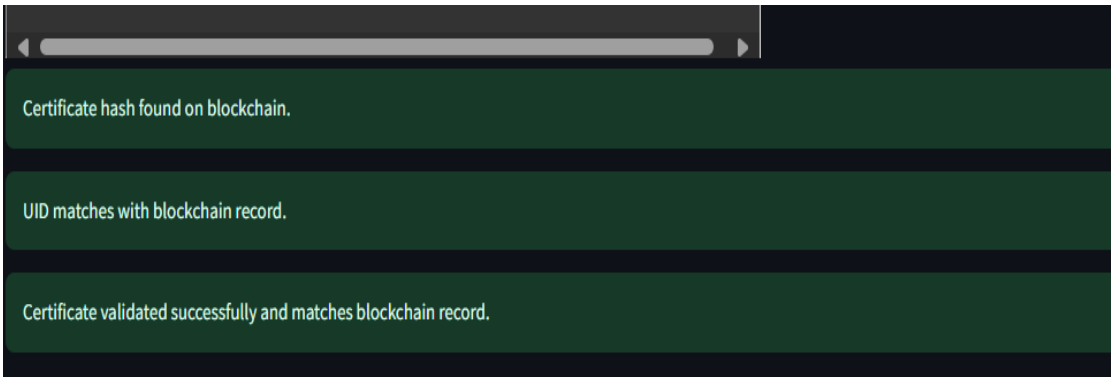
1. By first, the mechanism checks the uploaded certificate hash is found on the block chain.
2. It Checks UID (acts as Primary Key i.e, it is unique and not null) matches with block chain record.

3. At last, it checks all the records (i.e, uploaded certificate data) with existed blockchain certificate data,

If it matches, 11.5.1.1 output

Else 11.5.1.2 output

11.5.1.1 : OUTPUT

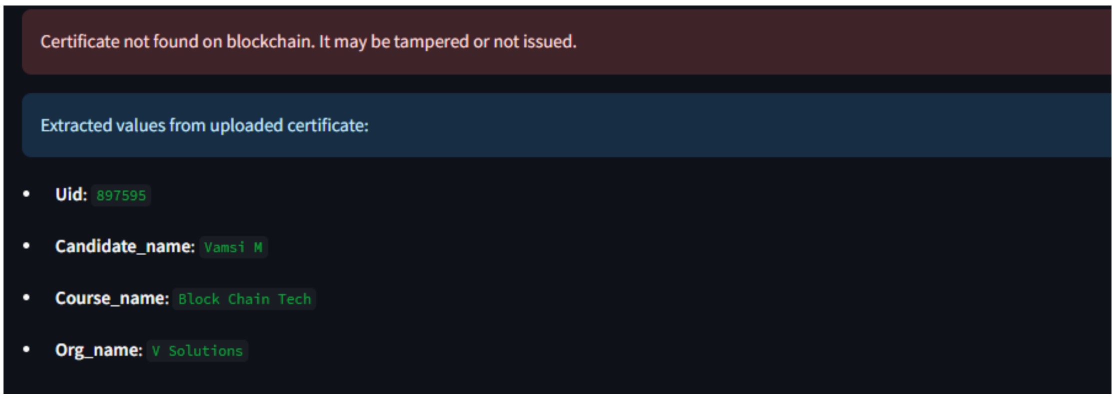


Certificate hash found on blockchain.

UID matches with blockchain record.

Certificate validated successfully and matches blockchain record.

11.5.1.2 : OUTPUT

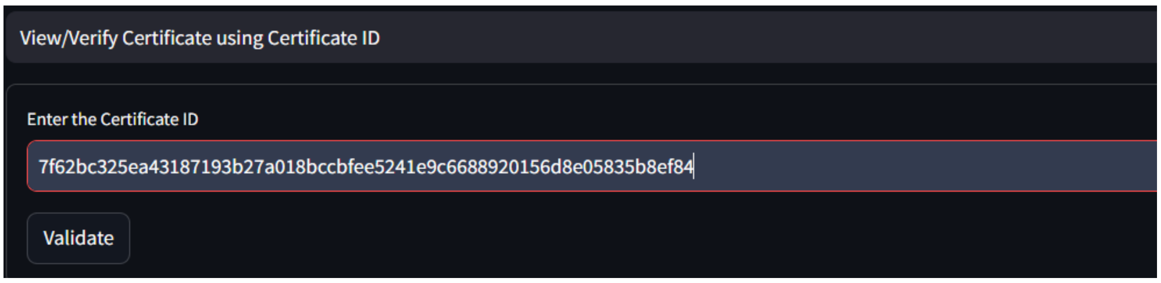


Certificate not found on blockchain. It may be tampered or not issued.

Extracted values from uploaded certificate:

- **Uid:** `897595`
- **Candidate_name:** `Vamsi M`
- **Course_name:** `Block Chain Tech`
- **Org_name:** `V Solutions`

1.5.2 : VERIFICATION MECHANISM - II: (by digital signature)



Enter the Certificate ID

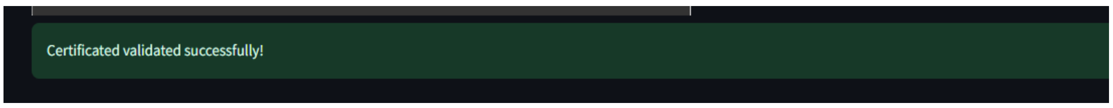
7f62bc325ea43187193b27a018bccbfee5241e9c6688920156d8e05835b8ef84

// By first, the mechanism checks the uploaded certificate hash with block chain existing certification hash,

If it matches, 11.5.2.1 output

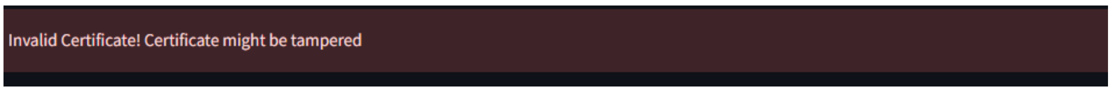
Else 11.5.2.2 output

11.5.2.1 : OUTPUT



Certificated validated successfully!

11.5.2..2 : OUTPUT



Invalid Certificate! Certificate might be tampered

Chapter 12

CONCLUSION

VerifyLab revolutionizes educational record management by leveraging blockchain to ensure secure, transparent, and decentralized data handling. It eliminates intermediaries, guaranteeing data integrity and user trust. The platform offers a seamless, user-friendly interface with wallet compatibility like MetaMask for effortless access and sharing. With strong encryption and privacy measures, VerifyLab ensures users retain full control of their records. Its browser-based accessibility enhances convenience and adoption. By removing single points of failure, it boosts data resilience. Overall, VerifyLab empowers users and institutions with a modern, efficient, and secure approach to record-keeping in the digital age.

Chapter 13
FUTURE ENHANCEMENTS

1. Dedicated Mobile Application:

Enhances accessibility by allowing users to upload, verify, and scan documents anytime, anywhere through a user-friendly mobile interface. The app can support push notifications for status updates, QR code scanning for instant verification, and offline access for remote areas with limited connectivity.

2. Biometric Authentication and Identity Linking:

Strengthens user verification by integrating fingerprint or facial recognition technologies, ensuring that only authorized users can access or share their records. Linking with national identity databases like Aadhaar or government-issued IDs adds an extra layer of trust and compliance, especially for high-stakes applications such as job recruitment or higher education admissions.

3. Upgradeable Smart Contracts:

Ensures system flexibility by allowing future feature enhancements and protocol updates without disrupting or affecting existing records. This allows the platform to remain future-proof, accommodating changes in compliance requirements, technology updates, or user needs without redeploying the entire system.

4. Cross-Blockchain Interoperability:

Broadens platform compatibility by supporting verification across multiple blockchain networks like Ethereum, Polygon, and Binance Smart Chain. This enhances flexibility for institutions already using different blockchain ecosystems and encourages broader collaboration and trust between diverse platform.

Chapter 14
BIBLIOGRAPHY

[1] Yasmeen Dabbagh, Reem Khoja, Leena AlZahrani, Ghada AlShowaier, Nidal Nasser, "A Blockchain-Based Fake Document Identification System."

[2] W. Viriyasitavat and D. Hoonsopon, "Blockchain characteristics and consensus in modern business processes," Journal of Industrial Information Integration,2019.

[3] Asem Ghaleb, Karthik Pattabiraman, Julia Rubin, "AChecker: Statically Detecting Smart Contract Access Control Vulnerabilities."

[4] Mohammad Jalalzai, Chen Feng, Jianyu Niu, Fangyu Gai, "Fast-HotStuff: A Fast and Robust BFT Protocol for Blockchains."

[5] IBM Training. IBM, IBM Corporation, 2018–2019.

[6] "Trade in fake goods is now 3.3% of world trade and rising," OECD, Mar. 2019. [Online]. Available: https://www.oecd.org/newsroom/trade-in-fake-goods-is-now-33-of-world-trade-and-rising.html

[7] E. Segran, "The volume of the problem is astonishing: Amazon's battle against fakes may be too little too late," Fast Company, May 2021. [Online]. Available: https://www.fastcompany.com/90636859/the-volume-of-the-problem-is-astonishing-amazons-battle-against-fakes-may-be-too-little-too-late

[8] "Nearly 1 in 10 EU Consumers Have Mistakenly Purchased a Counterfeit Document Over the Past Year Per Report," The Fashion Law, Jun. 2021. [Online]. Available:https://www.thefashionlaw.com/nearly-1-in-10-eu-consumers-have-mistakenly-bought-a-counterfeit-document-over-the-past-year-per-report/

[9] Avni Rustemi, Fisnik Dalipi, Atanasovski & Aleksandar (2023), " A Review on Blockchain-Based System for Academic Certificate Verification ," By IEEE Senior Association.

[10] Singhal, A., and Pavithra. R.S. (2018), "Degree Certificate Authentication using QR Code and Smartphone ". International Journal of Computer Applications, 120(16), 0975–8887.