**CHAPTER 1 INTRODUCTION**

## OVERVIEW

This project endeavors to revolutionize the conventional result management systems prevalent by implementing a decentralized framework based on blockchain technology. The primary objective is to address inherent challenges within current result management processes, including vulnerabilities to data tampering, lack of transparency, and inefficiencies in result distribution. Leveraging the immutable and transparent nature of blockchain, this initiative aims to establish a secure and trust-worthy system that ensures data integrity, facilitates transparent record-keeping, and streamlines result verification and distribution processes. By exploring the integration of blockchain technology within the result management infrastructure, this project seeks to provide a comprehensive blueprint for educational institutions to transition towards a decentralized system, fostering increased efficiency, trustworthiness, and accessibility of academic results.

## MOTIVATION

The project is motivated by the need to address the limitations of centralized result management systems in educational institutions. These systems are prone to data manipulation, lack transparency, and are vulnerable to cyber threats. The project's goal is to guarantee data integrity, improve transparency, and offer a safe platform for handling academic outcomes by utilizing public blockchain technology.

## PROBLEM DEFINATION OBJECTIVES

The objective of this project is to establish a robust and decentralized mechanism for document verification by leveraging Blockchain and PINATA technologies. The system operates by storing document hashes within the Blockchain network while housing the documents themselves on the PINATA network. This dual-layered approach ensures the integrity and immutability of the documents, as any attempts to tamper with or modify them would be readily detectable. Additionally, it facilitates easy retrieval and verification of documents by authorized entities, enhancing transparency and trust in the verification process.

## PROJECT SCOPE & LIMITATIONS

**1.4.1 Scope:**

Technical Implementation: The project will delve into the technical aspects of integrating blockchain technology into the existing result management system. This includes exploring suitable blockchain frameworks, designing smart contracts for result verification, and ensuring compatibility with the educational institution's infrastructure.

Data Security and Integrity: Scope includes ensuring the security and integrity of result data stored on the blockchain. Emphasis will be placed on cryptographic measures, access control, and the immutability of records to prevent unauthorized alterations or tampering.

Transparent Result Distribution: The project aims to streamline the result distribution process through blockchain, ensuring transparent and immediate access to academic results for stakeholders while maintaining data privacy.

Regulatory Compliance: Exploring the alignment of the decentralized system with relevant data protection laws and educational standards to ensure regulatory compliance within the project's framework.

**1.4.2 Limitations:**

Scalability Challenges: Due consideration will be given to potential scalability challenges associated with blockchain implementation in large educational institutions or high-volume result management systems.

Resource Constraints: The project may be limited by resource availability, including financial, technological, and human resources, impacting the depth of implementation and testing.

Adoption Hurdles: Limitations may arise concerning the adoption of a decentralized system within traditional educational settings. Resistance to change, institutional policies, and stakeholder acceptance could affect the project's implementation.

Comprehensive Integration: Complete integration with existing educational databases, systems, and legacy technologies may present challenges, potentially limiting the extent of the decentralized system's integration.

Evaluation Duration: The project may have limitations regarding the depth of evaluation or testing due to time constraints. Comprehensive long-term evaluation of the system's effectiveness might be limited within the project duration.

## METHODOLOGIES OF PROBLEM SOLVING

1. Problem Identification:

Define Challenges: Clearly articulate the specific issues or challenges faced in implementing the Decentralized Result Management System.

Understand Stakeholder Perspectives: Gather insights from stakeholders, including students, educators, administrators, and external verifiers, to identify potential pain points.

2. Requirement Analysis:

User Requirements: Understand the needs and expectations of end-users through interviews and surveys.

Regulatory Requirements: Analyze and document legal and regulatory requirements related to academic data and blockchain technology.

3. Technology Assessment:

Blockchain Platforms: Evaluate different public blockchain platforms (e.g., Ethereum, Binance Smart Chain) based on factors such as scalability, security, and smart contract capabilities.

Tools and Frameworks: Assess cybersecurity tools, encryption algorithms, and smart contract development frameworks suitable for the project.

4. Design and Architecture:

System Architecture: Design the overall architecture of the Decentralized Result Management System, specifying the roles of blockchain, smart contracts, and user interfaces.

Security Measures: Incorporate robust security measures, including encryption, decentralized identity, and access controls.

5. Development and Prototyping:

Smart Contract Development: Implement smart contracts for result recording, access control, and authentication.

User Interface Prototyping: Develop prototypes of user interfaces for students, educators, and administrators to gather feedback.

6. Testing:

Security Testing: Conduct thorough security testing, including penetration testing and vulnerability assessments.

User Acceptance Testing (UAT): Involve end-users in UAT to ensure the system meets their requirements and expectations.

7. Optimization and Performance Tuning:

Scalability Optimization: Identify and address potential scalability issues in the blockchain network.

Performance Tuning: Optimize the system for responsiveness and efficient transaction processing.

8. Deployment:

Gradual Rollout: Deploy the system in stages, starting with a controlled environment before full-scale implementation.

Monitoring: Implement monitoring tools to track system performance and security in real-time.

9. Documentation:

User Manuals: Prepare comprehensive user manuals for students, educators, and administrators.

System Documentation: Document the technical aspects of the system, including architecture, smart contract logic, and security measures.

10. Continuous Improvement:

Feedback Mechanism: Establish a feedback mechanism for continuous improvement, involving end-users and technical stakeholders.

Adaptation to Technological Advances: Stay informed about advancements in blockchain technology and cybersecurity to adapt the system accordingly.

**Chapter 2**

**Literature Survey**

## 2.1 STUDY OF RESEARCH PAPER

1. **Paper Name:** A Survey of IoT and Blockchain Integration: Security Perspective

**Author:** ELHAM A. SHAMMAR, AMMAR T. ZAHARY, ASMA A. AL-SHARGABI

**Abstract:**

Blockchain has recently attracted significant academic attention in research fields beyond the financial industry. In the Internet of Things (IoT), blockchain can be used to create a decentralized, reliable, and secure environment. The use of blockchain in IoT applications is still in its early stages, particularly at the low end of the computing spectrum. As a result, the future roadmap is hazy, and several challenges and questions must be addressed. Several articles combining blockchain technology with IoT have recently been released, but they are limited to shallow technological potential discussions, with very few providing an in-depth examination of the complexities of implementing blockchain technology for IoT. Therefore, this paper aims to provide current cutting-edge efforts coherently and comprehensively in this direction. It provides a literature review of IoT and blockchain integration by examining current research issues and trends in the applications of blockchain-related approaches and technologies within the IoT security context. We have surveyed published articles from 2017 to 2021 on blockchain-based solutions for IoT security, taking into consideration different security areas and then, we have organized the available articles according to these areas. The surveyed articles have been chronologically organized in tables for better clarity. In this paper, we try to investigate the vital issues and challenges to the integration of IoT and blockchain, and then investigate the research efforts that have been conducted so far to overcome these challenges.

1. **Paper Name:** A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare

**Author:** BAHAR HOUTAN, ABDELHAKIM SENHAJI HAFID, AND DIMITRIOS MAKRAKIS

## Abstract:

Convergence of physical and digital identity and integration of various individual records, such

as patient data, into a united repository remains a serious challenge. On one hand, collecting relevant data can help clinicians, specialists and healthcare service providers to facilitate care for patients. On the other hand, Self-Sovereign identity and the right to control personal data comes into question, because patients do not handle their data explicitly. Distributed Ledger Technology (DLT) is a novel method which would allow to securely record time-stamped data and enable patient-driven health and identity records. In this paper, we review the state-of-the-art in Blockchain (BC)-based self-sovereignty and patient data records in healthcare. Our motivation is to investigate the potential of BC technology for use in the patient data and identity management. As a distributed decentralized technology, BC can be very beneficial, giving patients control over their own data and self-sovereign identity. To the extent of our knowledge, there is no literature covering the same concerns. More specifically, the focus is on solutions that aim the realization of holistic BC-based Electronic Health Records (EHR) and Patient Health Records (PHR). EHR and PHR are used to record patient data, such as the doctor’s notes upon a visit and radiology images. Hence, they include critical information regarding patient’s privacy and identity. Therefore, development of pure decentralized Healthcare Information Systems (HIS) is a great challenge in terms of architectural and technical structure of the systems. Designing robust and reliable EHR and PHR, which represent the foundation of many other

healthcare services, relies on carefully finding the balance in a trade-off between many factors, such as level of decentralization, privacy, scalability, and data throughput. In this paper, we review the state-of-the-art and provide an analysis on the design trade-offs.

1. **Paper Name:** Analysis of Blockchain Solutions for IoT: A Systematic Literature Review

**Author:** SIN KUANG LO, YUE LIU, SU YEN CHIA, XIWEI XU1, QINGHUA LU, LIMING ZHU1, AND HUANSHENG NING

**Abstract:**

The Internet of Things (IoT) aims at connecting things to the Internet in a peer-to-peer paradigm for data collecting and data sharing in our daily life. A blockchain is an immutable append-only ledger maintained by a peer-to-peer network, where the whole network needs to reach a consensus on the transactional data stored on the ledger. With the decentralization nature, the design of IoT and blockchain aligns with each other well. Blockchain has been integrated with the IoT to solve the existing IoT problems. Our research focuses on analyzing the solutions proposed in academia and the methodologies used to integrate blockchain with the IoT. Through conducting a systematic literature review (SLR) on peer reviewed, published articles on blockchain-based solutions for IoT, we gather the knowledge on current technical approaches implemented to integrate blockchain into the IoT. Majority of the research in this space is either at a conceptual level or at a very early stage. However, we only found 35 published papers with the real implementation of blockchain in the IoT platforms. We elicit the challenges of the IoT that were being addressed, and the detailed design of the blockchain-based solutions from two perspectives, namely data management and thing management. The evaluation methods and metrics used by those works are also being recorded and analyzed. In addition to the analysis of the literature, we provide our insights on improving the existing solutions and research methodology based on our expertise and experience on the blockchain.

1. **Paper Name:** Applications of Blockchains in the Internet of Things: A Comprehensive Survey

**Author:**  Muhammad Salek Ali, Massimo Vecchio, Miguel Pincheira, Koustabh Dolui, Fabio Antonelli, and Mubashir Husain Rehmani

**Abstract:**

The blockchain technology has revolutionized the digital currency space with the pioneering cryptocurrency platform named Bitcoin. From an abstract perspective, a blockchain is a distributed ledger capable of maintaining an immutable log of transactions happening in a network. In recent years, this technology has attracted significant scientific interest in research areas beyond the financial sector, one of them being the Internet of Things (IoT). In this context, the blockchain is seen as the missing link toward building a truly decentralized, trustless, and secure environment for the IoT and, in this survey, we aim to shape a coherent and comprehensive picture of the current state-of-the-art efforts in this direction. We start with fundamental working principles of blockchains and how blockchain-based systems achieve the characteristics of decentralization, security, and auditability. From there, we build our narrative on the challenges posed by the current centralized IoT models, followed by recent advances made both in industry and research to solve these challenges and effectively use blockchains to provide a decentralized, secure medium for the IoT.

1. **Paper Name:** Blockchain for AI: Review and Open Research Challenges

**Author:** KHALED SALAH, M. HABIB UR REHMAN, NISHARA NIZAMUDDIN, AND ALA AL-FUQAHA

## Abstract:

Recently, artificial intelligence (AI) and blockchain have become two of the most trending and disruptive technologies. Blockchain technology has the ability to automate payment in cryptocurrency and to provide access to a shared ledger of data, transactions, and logs in a decentralized, secure, and trusted manner. Also with smart contracts, blockchain has the ability to govern interactions among participants

with no intermediary or a trusted third party. AI, on the other hand, offers intelligence and decision-making capabilities for machines similar to humans. In this paper, we present a detailed survey on blockchain applications for AI. We review the literature, tabulate, and summarize the emerging blockchain applications, platforms, and protocols specifically targeting AI area. We also identify and discuss open research challenges of utilizing blockchain technologies for AI.

1. **Paper Name:** Survey on Blockchain-Based Smart Contracts: Technical Aspects and Future Research

**Author:** THARAKA MAWANANE HEWA, YINING HU, MADHUSANKA LIYANAGE, SALIL S. KANHARE, AND MIKA YLIANTTILA

## Abstract:

The industrial and computing research context revolutionized in various directions during the last decades. The blockchain-based smart contract embraced as a significant research interest due to its distinguishing features such as decentralized storage of transactions, autonomous execution of contract codes, and decentralized establishment of the trust. Blockchain-based smart contracts can transform the working architecture of almost all industries towards elevated service standards. The use cases of blockchain based smart contracts range from industrial applications such as cryptocurrency systems towards logistics, agriculture, real estate, energy trading and so forth. The decentralization concept of blockchain is one of the biggest leaps in technology research since future computing got a super momentum towards the Internet of Things (IoT) and edge computing. A plethora of research is in progress to investigate the opportunities for the applicability of smart contracts and blockchain technologies to various industries. It is important to identify the technical aspects of blockchain-based smart contracts to further improve and sharpen the capabilities which they already owed. This survey is conducted to identify the significant technical aspects of blockchain-based smart contracts with the associated future research directions.

1. **Paper Name:** Survival Study on Blockchain Based 6G-Enabled Mobile Edge Computation for IoT Automation

**Author:** RAMESH SEKARAN, RIZWAN PATAN, ARUNPRASATH RAVEENDRAN, FADI AL-TURJMAN, MANIKANDAN RAMACHANDRAN, AND LEONARDO MOSTARDA

## Abstract:

Internet of Things (IoT) and Mobile Edge Computing (MEC) technology acts as a significant part of daily lives to facilitate control and monitoring of objects to revolutionize the ways that human interacts with physical world. IoT system includes large volume of data with network connectivity, power, and storage resources to transform data into meaningful information. Blockchain has decentralized nature to provide useful mechanism for addressing IoT challenges. Blockchain is distributed ledger with fundamental attributes, namely recorded, transparent, and decentralized. Blockchain formed participants in distributed ledger to record the transactions and communicate with other through trustless method. Security is considered as the most valuable features of Blockchain. IoT and Blockchain are emerging ideas for creating the applications to share the intrinsic features. Several existing works has been developed for the integration of blockchain with IoT. But, Blockchain protocols in the state-of-the-art works with IoT failed to consider the computational loads, delays, and bandwidth overhead which led to new set of problems. The review estimates main challenges in integration of Blockchain and IoT technologies to attain high-level solutions by addressing the shortcomings and limitations of IoT and Blockchain technologies.

1. **Paper Name:** Security, Performance, and Applications of Smart Contracts: A Systematic Survey

**Author:** SARA ROUHANI AND RALPH DETERS

**Abstract:**

Blockchain is the promising technology of recent years, which has attracted remarkable attention in both academic studies and practical industrial applications. The smart contract is a programmable transaction that can perform a sophisticated task, execute automatically, and store on the blockchain. The smart contract is the key component of the blockchain, which has made blockchain a technology beyond the scope of the cryptocurrencies and applicable for a variety of applications such as healthcare, IoT, supply chain, digital identity, business process management, and more. Although in recent years the progress toward improving blockchain technology with the focus on the smart contract has been impressive, there is a lack of reviewing the smart contract topic. This paper systematically views the key concepts and proposes the direction of recent studies and developments regarding the smart contract. The research studies are presented in three main categories:

1) security methods and tools;

2) performance improvement approaches;

3) decentralized applications based on smart contracts.

**Chapter 3**

**Software Requirement Specification**

## 3.0.1 Assumptions and Dependencies

1. User must require the Solidity
2. User must install the Remix on his/her pc.
3. User must have a MetaMask account.

## FUNCTIONAL REQUIREMENTS

* + 1. **System Feature1(Functional Requirement)**

Access Control: Test smart contracts governing access control for students, educators, and administrators. Confirm the correct implementation of Role-Based Access Control (RBAC) and Attribute Based Access Control (ABAC). Verify that access permissions align with user roles.

Result Recording: Verify that academic results are accurately recorded on the blockchain.

Ensure results are timestamped and linked in a chronological order for immutability. Validate the standardized format of academic records.

## System Feature2(Functional Requirement)

Smart Contracts: Verify the accuracy and security of smart contract logic. Test smart contracts for potential vulnerabilities and exploits. Confirm the proper execution of automated result management processes.

Authentication: Validate the cryptographic key-based authentication process. Ensure secure and seamless user authentication. Test the integration of decentralized identity management solutions.

## EXTERNAL INTERFACE REQUIREMENT

* + 1. **User Interface**

Application for collecting the Unique code and QR code and submit the QR code or Unique code to get the certificates and results.

## Hardware Interfaces:

RAM: 8GB

Hard Disk: 40 GB

Processor: Intel i5 Processor

IDE: Remix IDE/ VS Code

Coding Language: Solidity

Highly specified Programming Language for Smart Contracts because of avail- ability of High-Performance Libraries.

Operating System: Windows 10 or above

Latest Operating System that supports all type of installation and development Environment

## Software Interfaces

Operating System: Windows 10 or above

IDE: Remix IDE/ VS code

Programming Language: Solidity/ HTML/ CSS

## NON-FUNCTIONAL REQUIREMENT

* + 1. **Performance Requirements**

Test system scalability to handle increasing transaction volumes. Evaluate transaction processing speed and efficiency. Monitor and optimize system resource utilization. Conduct penetration testing to identify and mitigate security vulnerabilities. Ensure compliance with industry best practices for secure coding. Confirm the immutability of academic records through cryptographic measures.

## Safety Requirement

The application is designed in modules where errors can be detected and fixed easily. This makes it easier to install and update new functionality if required.

## Software Quality Attributes

Our software has many quality attribute that are given below:-

Usability: Conduct usability testing for user interfaces. Ensure a user-friendly experience for stakeholders. Provide clear and concise error messages for troubleshooting.

Interoperability: Test interoperability with various blockchain networks. Ensure compatibility with existing educational infrastructure. Confirm adherence to blockchain standards and protocols.

Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the blockchain developer.

Reliability: The performance of the software is better which will increase the reliability of the Software.

User Friendliness: Since, the software is a GUI application; the output generated is much user friendly in its behavior.

Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled

Security: Users are authenticated using many security phases so reliable security is provided.

Testability: The software will be tested considering all the aspects.

## SYSTEM REQUIREMENTS

* + 1. **Database Requirements**

No database used

## Software Requirements (Platform Choice)

* Operating system: Windows 7 or more.
* Coding Language: Solidity
* IDE: Remix IDE/ VS code

## Hardware Requirements

* System: Intel I3 Processor and above.
* Hard Disk : 20 GB
* Ram : 8GB

## ANALYSIS MODEL: SDLC MODEL TO BE APPLIED

The Agile model is a flexible approach to software development that focuses on iterative cycles, collaboration, and continuous improvement. It involves breaking projects into smaller sprints, adapting to changing requirements, collaborating closely with customers, delivering functional software in short increments, and fostering a culture of teamwork and constant enhancement.

1. Iterative Approach: Breaks projects into smaller cycles called sprints for iterative development.

2. Flexibility: Emphasizes adaptability to changing requirements and conditions throughout the project.

3. Customer Collaboration: Involves continuous customer feedback and collaboration for meeting user needs.

4. Cross-functional Teams: Small teams work collaboratively, covering various roles for efficient development.

5. Working Software Delivery: Focuses on delivering functional software in short, regular intervals.

6. Continuous Improvement: Encourages reflection and enhancement in processes for ongoing development.

The Agile model revolves around iterative progress, collaboration, adaptability, and continual refinement to achieve successful software development.

**Chapter 4**

**System Design**

## SYSTEM ARCHITECTURE

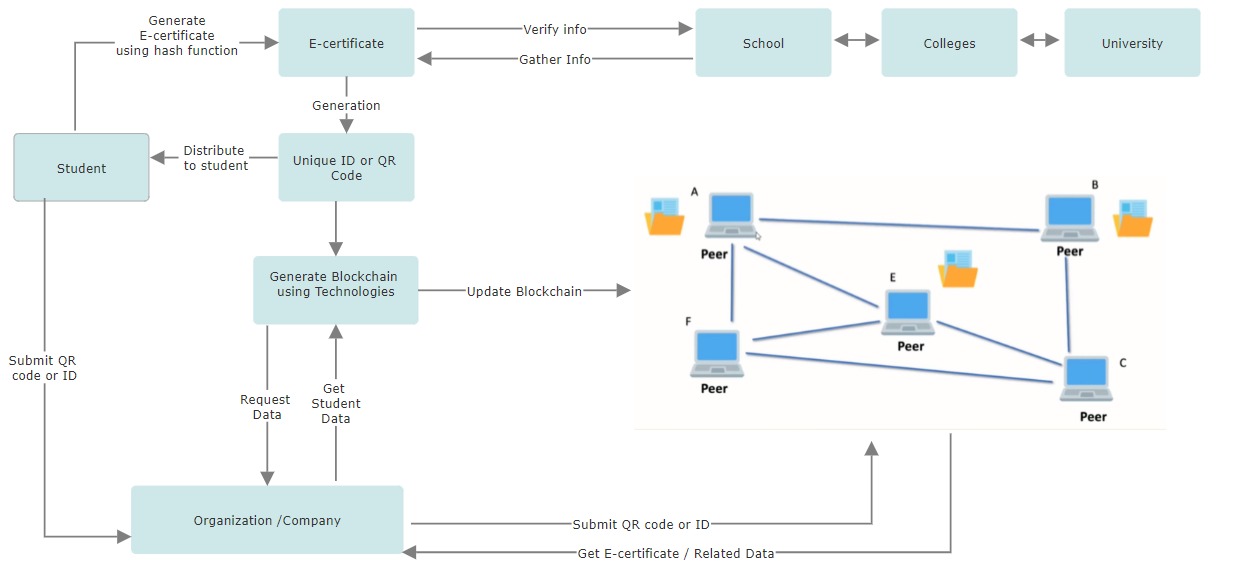
****

Figure 4.1: System Architecture

User Interface Layer: This layer consists of the interfaces for users to interact with the system, including web and mobile applications. It enables users to request, generate, and verify certificates.

Application Layer: This layer handles the business logic and interacts with the blockchain layer. It includes modules for certificate request handling, verification, and generation.

Smart Contract Layer: Smart contracts are deployed on the blockchain to manage certificate issuance and verification. They define the rules for generating and validating certificates, ensuring transparency and security.

Blockchain Layer: This layer is composed of the decentralized network where the data is stored. A suitable blockchain platform like Ethereum or a custom-built blockchain can be used, depending on the specific requirements of the system.

Data Layer: Data storage components, including decentralized storage protocols or IPFS (Inter Planetary File System), can be utilized to securely store certificate data

and ensure its availability across the network.

Identity Management Layer: To ensure authenticity and prevent fraud, an identity management layer can be integrated, utilizing technologies such as digital signatures and cryptographic techniques.

Security Layer: This layer includes encryption protocols, secure communication channels, and measures to protect against unauthorized access and data breaches.

API Layer: For interoperability, an API layer can be established to facilitate communication between the system and external applications or services.

## Data Flow Diagram

In Data Flow Diagram, we Show that flow of data in our system in DFD0 we show that base DFD in which rectangle present input as well as output and circle show our system, In DFD1 we show actual input and actual output of system input of our system is text or image and output is rumor detected likewise in DFD 2 we present operation of user as well as admin.

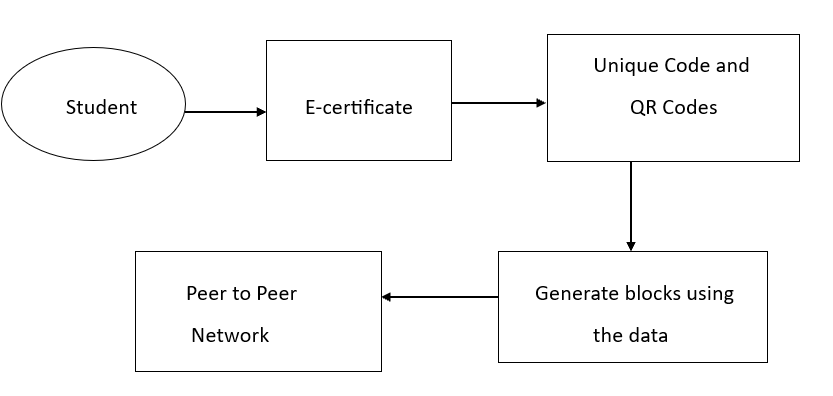


Figure 4.2: Dataflow (1) diagram

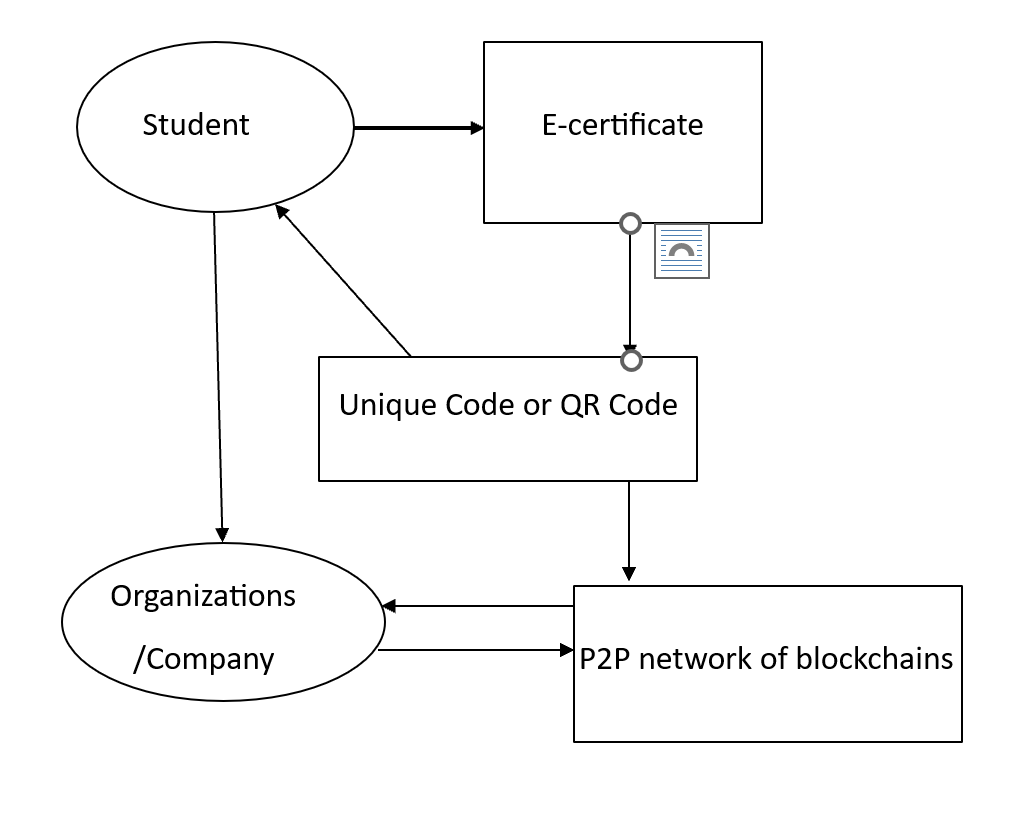


Figure 4.3 Dataflow (2) diagram

## ENTITY RELATIONSHIP DIAGRAM

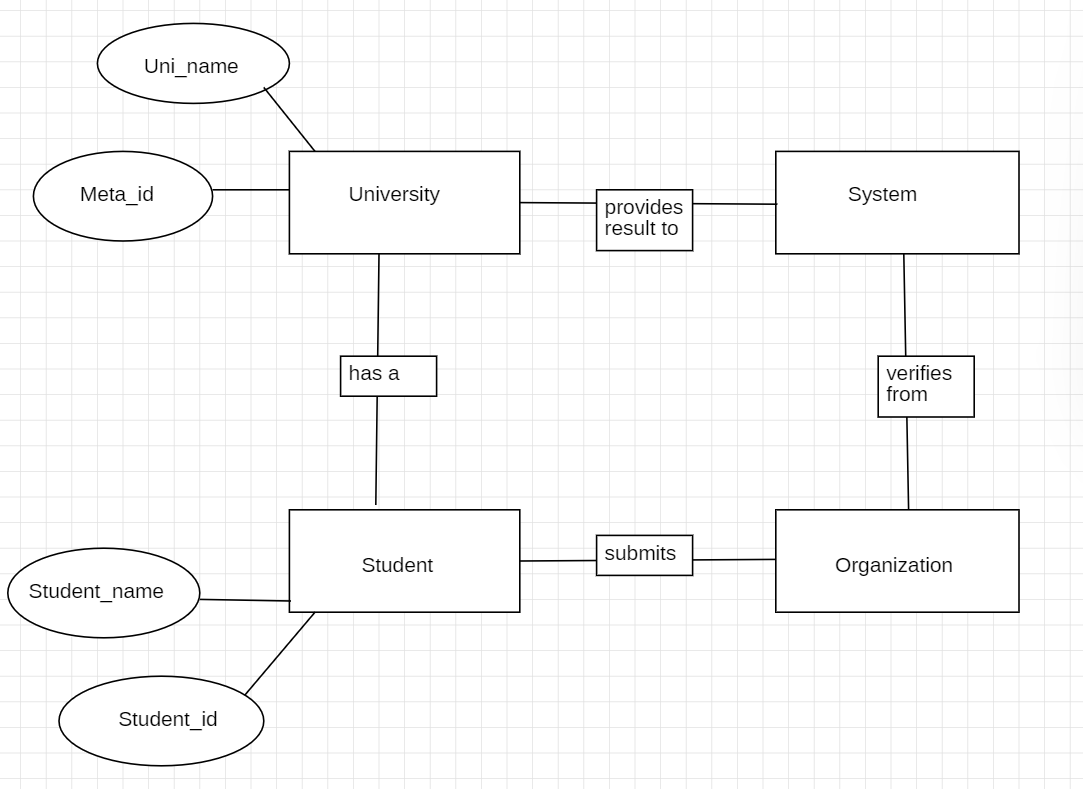
****

Figure 4.5: Entity Relationship Diagram

## UML DIAGRAMS

Unified Modeling Language is a standard language for writing software blueprints. The UML may be used to visualize, specify, construct, and document the artifacts of a software intensive system. UML is process independent, although optimally it should be used in process that is use case driven, architecture-centric, iterative, and incremental. The Number of UML Diagram is available.

Class Diagram.

Use case Diagram.

Activity Diagram.

Sequence Diagram.

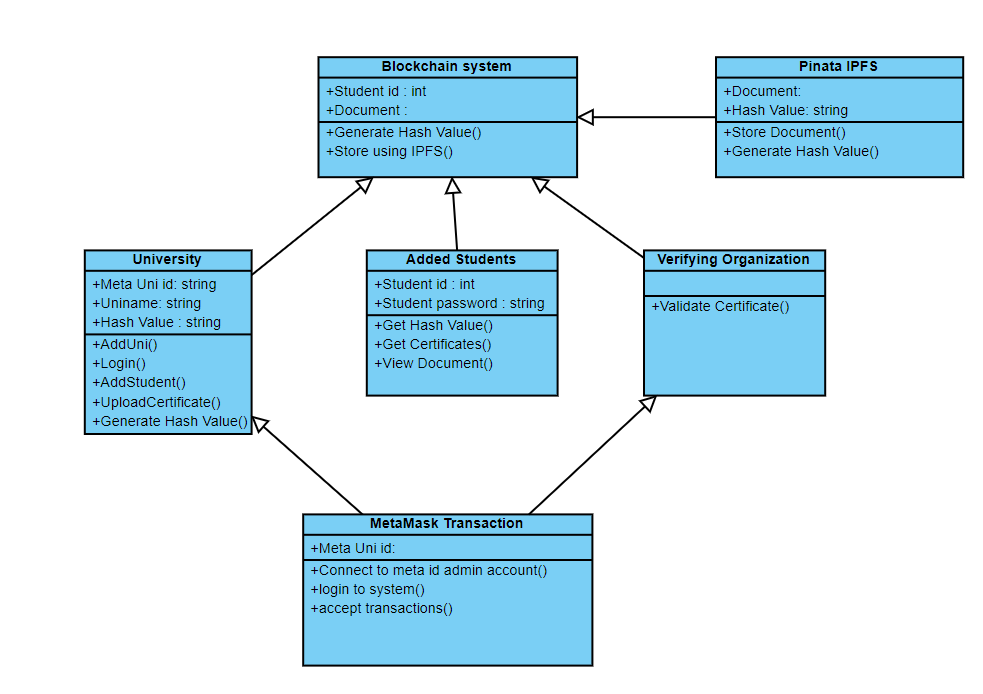


Figure 4.6: Class Diagram

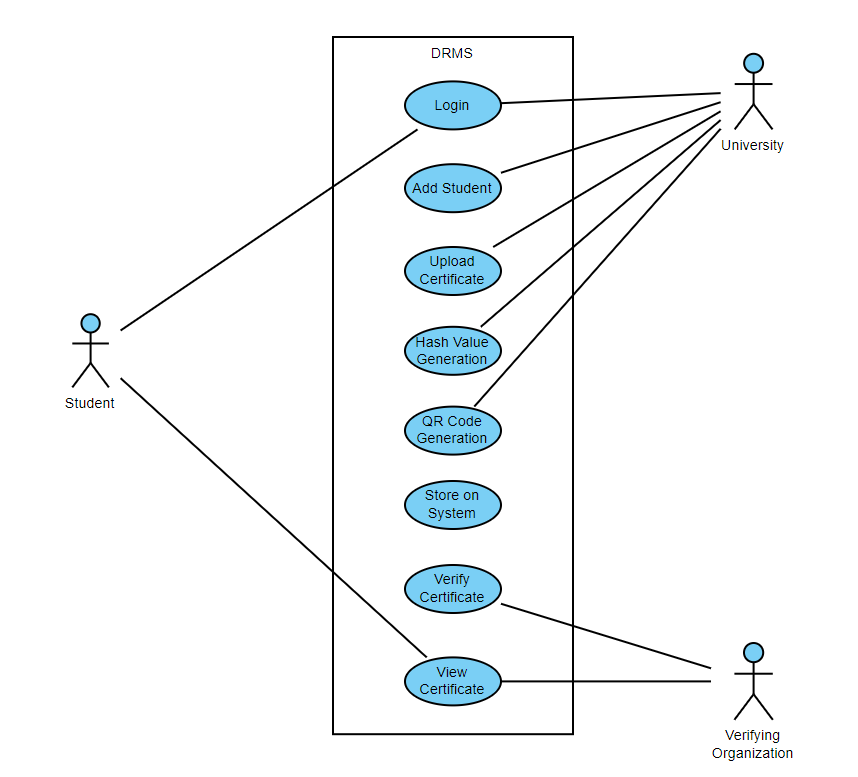


Figure 4.7: Use case Diagram

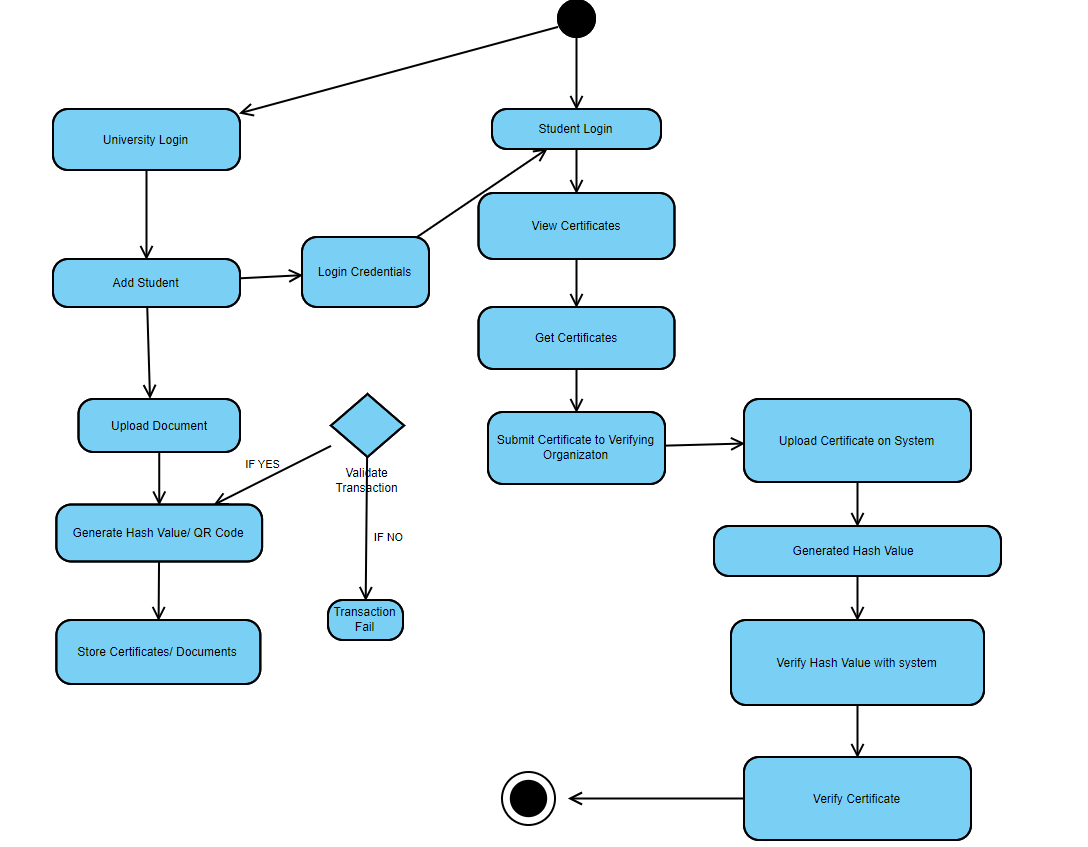


Figure 4.8: Activity Diagram

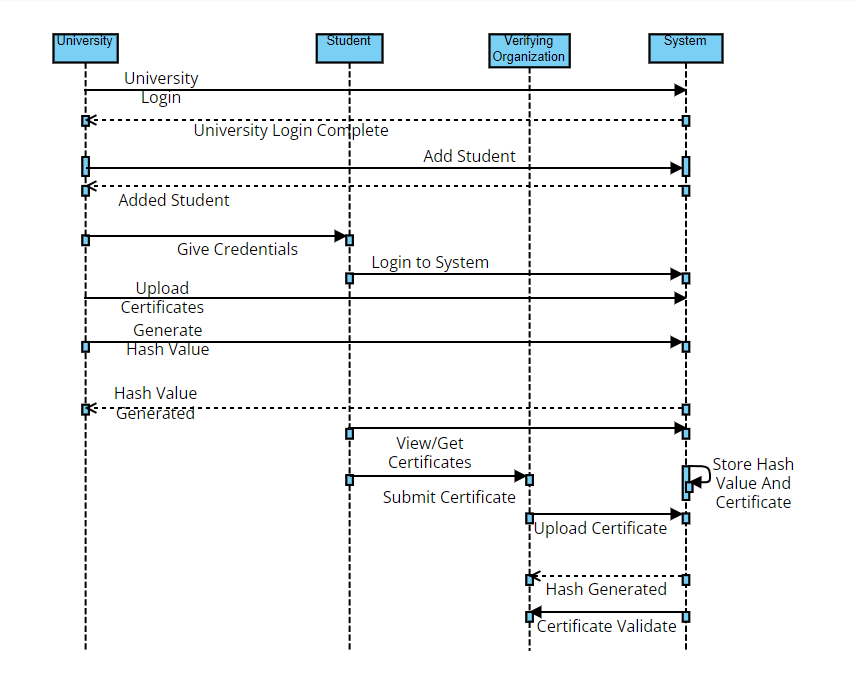


Figure 4.9: Sequence Diagram

Chapter 5

Project Plan

## PROJECT ESTIMATES

We are using Agile model for our project estimation.

1. Iterative Development:

- Sprints: Break down the development process into short iterations or sprints, typically 2-4 weeks long.

- Incremental Progress: Each sprint focuses on delivering a functional increment of the result management system, starting with basic features and progressively adding functionality.

1. Continuous Stakeholder Collaboration:

- Regular Feedback: Encourage regular feedback from stakeholders, including educational institutions, students, and administrators, at the end of each sprint.

- Adaptation to Changes: Agile allows flexibility to accommodate changing requirements or emerging needs, ensuring that the system aligns with stakeholders' expectations.

1. Prioritized Product Backlog:

- Product Backlog: Maintain a prioritized list of features, functionalities, and user stories required for the decentralized result management system.

- Sprint Planning: Select items from the product backlog for each sprint based on priority and feasibility.

1. Cross-Functional Teams:

- Collaborative Teams: Form cross-functional teams comprising developers, blockchain specialists, testers, and domain experts (educational professionals).

- Collaboration and Transparency: Encourage collaboration among team members to share knowledge and address challenges collectively.

1. Continuous Integration and Testing:

- Continuous Integration (CI): Implement continuous integration practices to ensure that

code changes are integrated regularly and tested.

- Automated Testing: Utilize automated testing for smart contracts and system functionalities to maintain quality standards.

1. Regular Reviews and Retrospectives:

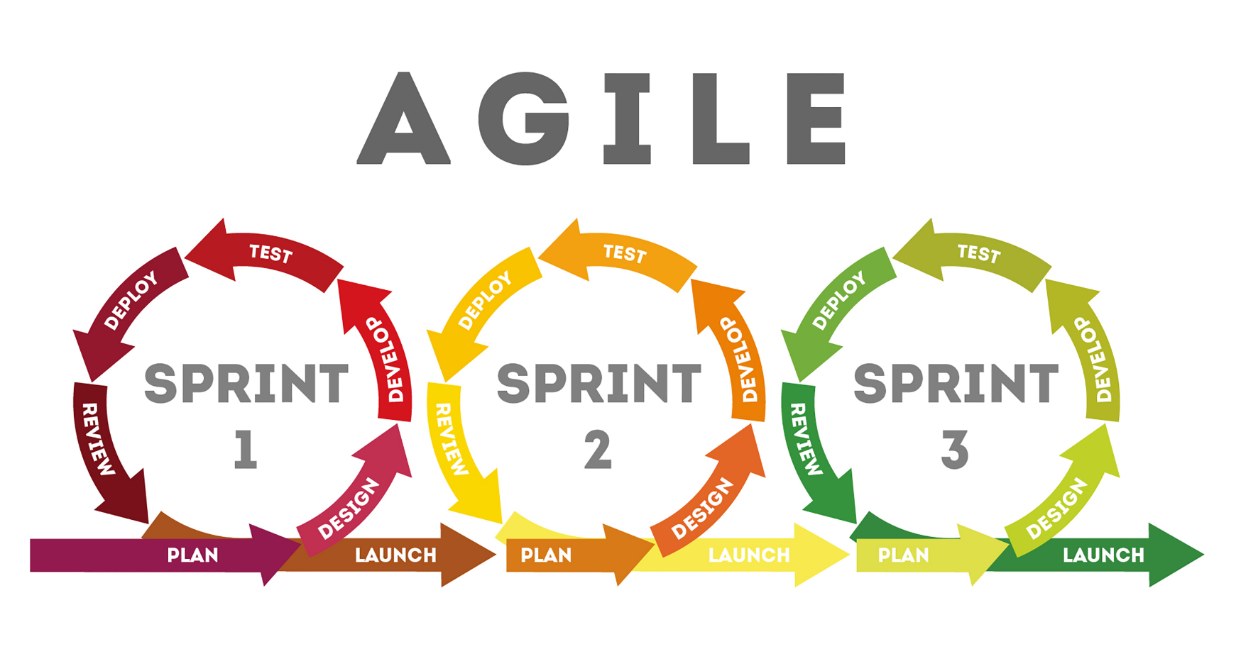
- Sprint Reviews: Conduct sprint reviews at the end of each iteration to demonstrate completed functionalities to stakeholders and gather feedback.

- Sprint Retrospectives: Reflect on each sprint's successes and challenges to identify areas for improvement in subsequent iterations.

1. Adaptation to Blockchain Development:

- Smart Contract Development: Plan sprints specifically for developing, testing, and deploying smart contracts on the chosen blockchain network.

- Blockchain Integration: Iteratively integrate blockchain functionalities into the system, ensuring compatibility and addressing technical challenges along the way.



## Reconciled Estimates

Project reconciliation management is a component of Project management which ar- ranges every one of the parts of a project. Project reconciliation guarantees smooth execution of all procedures.

## Project Resources

Well configured Laptop, Remix IDE, 2 GHZ CPU speed, 8 GB RAM, Internet connection

## RISK MANAGEMENT W.R.T. NP HARD ANALYSIS

Complexity Issues: Identify specific NP-hard problems within the context of blockchain technology, such as cryptographic hashing, consensus mechanisms (e.g., proof-of-work), or complex smart contract execution.

## Risk Identification

Security Risks: Evaluate the potential security risks stemming from NP-hard problems. For instance, cryptographic vulnerabilities due to the difficulty of factorization or discrete logarithm problems.

Scalability Challenges: NP-hard problems may hinder scalability, leading to increased computational resources required for consensus, longer transaction confirmation times, or network congestion.

## Risk Analysis

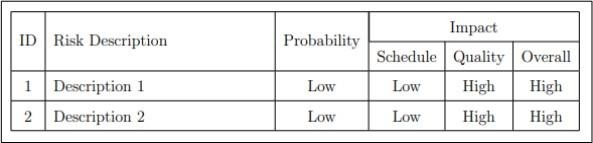
The risks for the Project can be analyzed within the constraints of time and quality

Figure 6.1: Risk Table

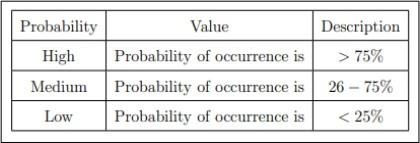


Figure 6.2: Risk Probability definitions

Figure 6.3: Risk Impact definitions

## Overview of Risk Mitigation, Monitoring, Management

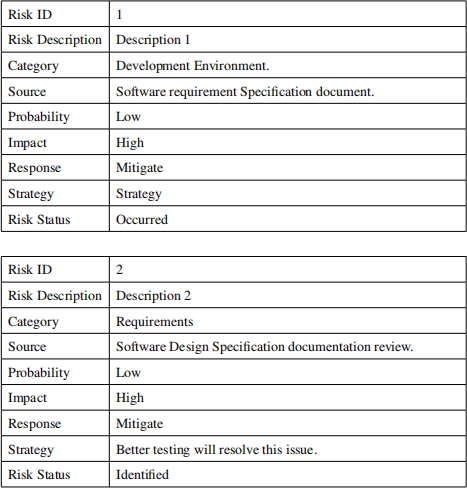


Figure 6.4

## PROJECT SCHEDULE

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.No. | Task Description | Date | Duration |
| 1 | Project Kickoff | 15/07/2023 | 1 day |
| 2 | Research and Data Collection | 16/07/2023 - 30/07/2023 | 2 weeks |
| 3 | Literature Review | 31/07/2023 - 07/08/2023 | 1 week |
| 4 | Study of Technologies | 08/08/2023 - 24/10/2023 | 10 weeks |
| 5 | Framework Design | 25/10/2023 - 01/11/2023 | 1 week |
| 6 | Development of Blockchain Module | 02/11/2023 - 15/11/2023 | 2 weeks |
| 7 | Integration of Storage networks | 16/11/2023 - 30/11/2023 | 2 weeks |
| 8 | Model Verification | 01/12/2023 -15/12/2023 | 2weeks |
| 9 | Documentation and Report | 16/12/2023 - 23/12/2023 | 1 week |
| 10 | Review and Finalization | 24/12/2023 - 08/01/2024 | 2 weeks |
| 11 | Presentation and Submission | 08/01/2024 - 15/01/2024 | 1 week |

## TASK NETWORK

Major Tasks in the Project stages are:

* Task 1: correctness
* Task 2: availability
* Task 3: integrity

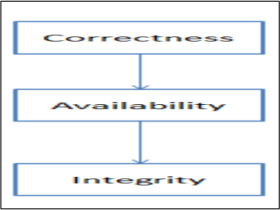


Figure 6.6: Task Network

## Timeline Chart

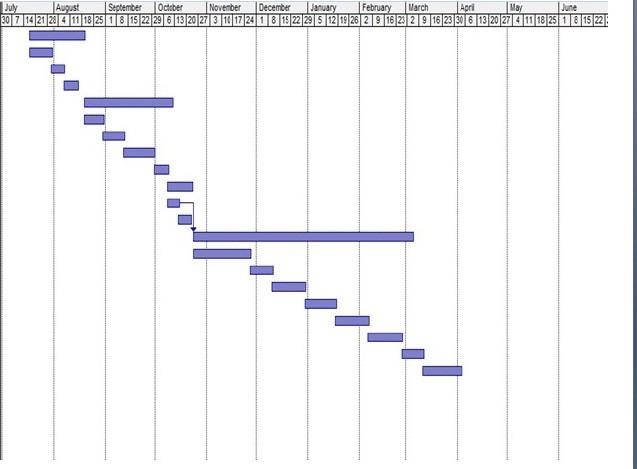


Figure 6.7: Timeline Chart

## TEAM ORGANIZATION

Team consists of 3 members and proper planning mechanism are used and roles of each member are defined.

## Team structure

The team structure for the project is identified. There is total 3 members in our team and roles are defined. All members are contributing in all the phases of project.

## Management reporting and communication

Well planning mechanisms are used for progress reporting and inter/intra team communication are identified as per requirements of the project and Augmented Reality (AR) Vision Tests.

Chapter 6

Project Implementation

**6.1 Overview of Project Modules:**

**Blockchain Integration Module:**

Responsible for connecting the system to a public blockchain network (e.g., Ethereum, Biance Smart Chain). Manages the interaction with the blockchain, including transactions and data storage.

**Smart Contract Module:**

Implements smart contracts for key functionalities, such as result recording, access control, and authentication. Defines the rules and logic governing interactions on the blockchain.

**Result Recording Module:**

Handles the recording of academic results on the blockchain. Ensures the data is standardized, timestamped, and linked to maintain an immutable and transparent record.

**Access Control Module:**

Implements access control mechanisms using smart contracts. Defines roles and permissions for different user types (students, educators, administrators) to access and modify academic records.

**Authentication Module:**

Manages user authentication through cryptographic keys. Validates the identity of users and ensures secure interactions with the blockchain.

6.2 TOOLS AND TECHNOLOGIES USED:

**SOLIDITY:** Solidity is a high-level, statically typed programming language primarily utilized for creating smart contracts on blockchain platforms like Ethereum. It serves as the core language for writing the logic and functionality of smart contracts that operate on the Ethereum Virtual Machine (EVM) and other compatible blockchains.

This language's syntax resembles that of JavaScript, making it relatively approachable for developers familiar with C-like languages or JavaScript. Solidity emphasizes security and safety in smart contract development, but due to the immutable nature of blockchain, any bugs or vulnerabilities in deployed contracts can be critical and challenging to rectify.

Developers use Solidity to define smart contract logic, which is then compiled into bytecode executable on the Ethereum Virtual Machine. This process is vital for integrating decentralized applications (dApps) and blockchain-based systems.

Solidity comes with a variety of development tools and libraries that aid in writing efficient, secure, and functional smart contracts. Tools such as Truffle and Remix, along with libraries like OpenZeppelin, offer support and resources to Solidity developers.

An essential aspect of Solidity is its role in creating various decentralized applications, ranging from simple token contracts to complex decentralized finance (DeFi) protocols and non-fungible tokens (NFTs). Continuous learning and staying abreast of best practices in Solidity are crucial for developers to produce robust and secure smart contracts in the ever-evolving blockchain landscape.

**REMIX:** Remix stands as a robust and user-friendly web-based Integrated Development Environment (IDE), designed specifically for creating, debugging, and deploying smart contracts on the Ethereum blockchain. Developed by the Ethereum Foundation, Remix offers a comprehensive suite of features within a browser-based interface, making it accessible to both novice and experienced developers interested in Solidity smart contract development.

At its core, Remix provides an intuitive code editor enabling developers to write Solidity smart contracts effortlessly. This editor is equipped with syntax highlighting, autocomplete, and error detection features, facilitating efficient coding practices. Furthermore, Remix integrates a compiler that transforms Solidity code into bytecode, ensuring compatibility with the Ethereum Virtual Machine (EVM).

One of Remix's standout features is its robust analysis tools. These tools enable developers to perform static analysis on their smart contracts, identifying potential security vulnerabilities and bugs before deployment. Additionally, Remix offers a debugging environment that allows step-by-step code execution, variable inspection, and breakpoint setting, streamlining the debugging process, and ensuring contract reliability.

For deployment and testing purposes, Remix seamlessly integrates with Ethereum's test networks (e.g., Rinkeby, Ropsten) and the Ethereum mainnet. This integration enables developers to deploy contracts to these networks directly from the Remix interface and interact with deployed contracts through transactions, facilitating comprehensive testing and validation of contract functionalities.

Moreover, Remix's extensibility through plugins enhances its capabilities. Developers can leverage various plugins for additional functionalities such as security analysis, gas estimation, and integration with external tools, expanding Remix's utility and adaptability to specific development needs.

Lastly, Remix supports project management features, allowing developers to save, manage, and share their Solidity projects. The platform enables version control and easy sharing of contracts via links or GitHub integration, promoting collaboration among developers and facilitating knowledge sharing within the community.

Overall, Remix serves as an indispensable tool in the Solidity development ecosystem, providing a seamless and comprehensive environment for writing, testing, debugging, and deploying smart contracts on the Ethereum blockchain. Its user-friendly interface, robust features, and integration capabilities make it an invaluable resource for both learning Solidity and building production-ready decentralized applications.

PINATA: Using the Inter Planetary File System (IPFS), PINATA is a web service that offers decentralized file hosting and content delivery. An outline of PINATA and its main characteristics   
  
1. Decentralized Storage: IPFS, a distributed file system that distributes file and content storage over a network of nodes, is utilized by PINATA. As opposed to conventional centralized storage systems, which store data on a single server or a group of servers under the ownership of a single company, IPFS distributes files among several nodes, providing redundancy, fault tolerance, and resistance to censorship.

2. Immutable Content Addressing: IPFS uses content-based addressing, where each piece of content is identified by its unique cryptographic hash. This hash is derived from the content itself, meaning that if the content changes even slightly, its hash will change, resulting in a different identifier. This ensures the integrity and immutability of content stored on IPFS, as any modification to the content would result in a different hash, making it easily detectable.

3. Content Pinning: PINATA allows users to "pin" their content to the IPFS network. Pinning ensures that the content remains available and accessible even when the original uploader is offline. By pinning content to IPFS through PINATA, users can ensure that their files are always available and retrievable by anyone with the corresponding hash.

4. User-Friendly Interface: PINATA provides a user-friendly web interface that allows users to upload, manage, and distribute their files easily. Users can drag and drop files directly onto the web interface or use the provided APIs for programmatic access. The interface also provides features such as file organization, search, and analytics to help users manage their content effectively.

5. APIs and Integrations: PINATA offers APIs and integrations with other platforms and services, allowing developers to incorporate decentralized file storage into their applications seamlessly. These APIs provide functionality for uploading, pinning, retrieving, and managing files programmatically, enabling developers to build decentralized applications (DApps) that leverage the power of IPFS and PINATA.

6. Security and Privacy: PINATA emphasizes security and privacy, ensuring that users' files and data are encrypted and protected. Content uploaded to PINATA is encrypted before being stored on IPFS, and access to content is controlled using cryptographic keys and access control mechanisms. This ensures that only authorized users can access and retrieve the content, enhancing security and privacy.

METAMASK:

MetaMask is a cryptocurrency wallet and browser extension that allows users to interact with the Ethereum blockchain and decentralized applications (DApps) directly from their web browsers. Here's a detailed overview of how MetaMask works:

1. Installation and Setup:

MetaMask is available as a browser extension for popular web browsers like Chrome, Firefox, Brave, and Edge. Users can install MetaMask from the respective browser's extension store.

After installation, users are guided through the setup process, where they create a new wallet or import an existing one using a seed phrase (mnemonic phrase). This wallet serves as their gateway to interact with Ethereum and store their Ethereum based assets.

2. Wallet Functionality:

Once set up, MetaMask provides users with a wallet interface within their browser. This wallet displays their Ethereum address, balance, and transaction history.

Users can send and receive Ethereum (ETH) and Ethereum based tokens directly from their MetaMask wallet. They can also view and manage their token balances and add custom tokens if needed.

3. Interaction with DApps:

MetaMask acts as a bridge between users and Ethereum based decentralized applications (DApps). When users visit a website or web application that integrates with MetaMask, they can connect their wallet to the DApp with a few clicks.

After connecting their wallet, users can perform various actions within the DApp, such as making transactions, interacting with smart contracts, participating in token sales (ICOs/IEOs), and accessing decentralized finance (DeFi) protocols.

4. Transaction Signing:

When users initiate a transaction within a DApp (e.g., sending tokens, interacting with a smart contract), MetaMask prompts them to review and confirm the transaction details.

Users can customize transaction parameters such as gas price and gas limit before confirming the transaction. Once confirmed, MetaMask securely signs the transaction using the user's private key and broadcasts it to the Ethereum network.

5. Security and Privacy:

MetaMask prioritizes security and privacy to protect users' funds and data. It encrypts and stores users' private keys locally on their device, ensuring that only they have access to their funds.

MetaMask also implements various security features such as password protection, seed phrase backup, and support for hardware wallets (e.g., Ledger, Trezor) to enhance security and mitigate the risk of unauthorized access or loss of funds.

6. Network Selection:

MetaMask allows users to choose which Ethereum network they want to interact with, including the Ethereum mainnet, testnets (Ropsten, Rinkeby, Kovan, Sepolia), and custom networks. This flexibility enables developers to test their DApps on different networks before deploying them to the mainnet.

CONNECTIVITY:

1. Solidity Smart Contract:

Solidity is a programming language used for writing smart contracts on the Ethereum blockchain.

Smart contracts are self-executing contracts with the terms of the agreement directly written into code. They run on the Ethereum Virtual Machine (EVM) and can automate processes and enforce rules in a decentralized manner.

In the context of connectivity between Solidity and HTML/CSS/JavaScript, you would write Solidity smart contracts to define the business logic and data storage for your decentralized application (DApp).

2. Web3.js:

Web3.js is a JavaScript library that provides an interface for interacting with the Ethereum blockchain.

It allows you to connect your frontend application (HTML/CSS/JavaScript) to the Ethereum blockchain, enabling communication with smart contracts deployed on the blockchain.

Web3.js provides functions for deploying smart contracts, sending transactions, and calling smart contract methods from JavaScript code.

3. HTML/CSS/JavaScript Frontend:

HTML (HyperText Markup Language), CSS (Cascading Style Sheets), and JavaScript are the standard technologies used for creating web-based user interfaces.

HTML defines the structure of the webpage, CSS styles the elements, and JavaScript provides interactivity and dynamic behavior.

In the context of connectivity with Solidity, you would create HTML/CSS/JavaScript files to build the frontend interface for your DApp. This interface would include forms for user input, buttons for triggering actions, and areas to display information fetched from the blockchain.

4. Communication Flow:

To connect the frontend with the Solidity smart contract, you would use Web3.js to:

Connect to an Ethereum node or provider (e.g., MetaMask, Infura).

Load the compiled ABI (Application Binary Interface) of the smart contract, which defines its methods and events.

Instantiate a JavaScript object representing the deployed instance of the smart contract.

Call methods of the smart contract to read data from the blockchain or trigger statechanging transactions.

Users interact with the frontend interface, which triggers JavaScript functions to interact with the smart contract via Web3.js.

When a user submits a form or clicks a button, JavaScript code sends transactions to the Ethereum network, invoking the corresponding smart contract functions.

The smart contract executes the requested actions and updates its state on the blockchain.

Events emitted by the smart contract can be captured by the frontend and used to update the user interface in real-time.

5. Testing and Deployment:

During development, you can test your DApp locally using a local blockchain (e.g., Ganache) and tools like Remix IDE for Solidity compilation and debugging.

Once tested, you deploy your smart contract to the Ethereum mainnet or a testnet using tools like Truffle or Remix.

Users can then access your DApp through a web browser, interacting with it via the frontend interface.

6.3 Algorithms Details:

6.3.1 System Algorithm:

1) User logs in.

2) System provides the address of the University wallet and the University Name.

3) System adds the exporter to the platform.

4) System processes transactions requested by the users.

5) If the Exporter address exists, the system allows the addition of a new address for a new university

6) System prompts for student information to add them to the platform.

7) System requests the student ID for identification purposes.

8) System prompts for the name of the student.

9) System prompts for a password for student authentication.

10) System adds the student to the platform with provided details.

11) System initiates the file upload process.

12) System prompts for the student ID associated with the file to be uploaded

13) System allows the selection of the file to be uploaded.

14) System ensures that the transaction is executed from the university wallet.

15) Upon transaction completion, the system generates a QR code for the certificate, displaying GEI required, CID hash, normal hash, and timestamp.

16) System provides options to download the QR code and verify the document.

17) The university uploads the document to be verified.

18) System verifies the document; if valid, it shows verification status, student information, timestamp of verification, and hash value.

19) System allows the university to delete a student by providing their student ID.

20) Students log in using their student ID and password.

21) System displays the number of verified certificates and generates links for individual document access.

22) System shows the document requested by the student.

23) System presents the QR code of the document and the student's profile.

24) Student’s certificate is verified.

6.3.2 Algorithms Used:

1. Algorithm 1:

SHA 1: To secure hash algorithm it takes input and produces 160-bit hash value.

SHA1 (Secure Hash Algorithm 1) is a cryptographic hash function that takes an input message and produces a fixed size output hash value, typically 160 bits (20 bytes) in length.

1. Input Message:

SHA-1 operates on an input message of arbitrary length. This message can be any data, such as a text string, binary file, or even another hash value.

2. Padding:

Before processing the input message, SHA-1 pads it to ensure that its length is a multiple of 512 bits. This padding includes appending a single '1' bit followed by a sequence of '0' bits, and then appending the length of the original message (in bits) as a 64bit representation.

3. Message Processing:

The padded message is divided into blocks of 512 bits each. Each block is further divided into 16 words of 32 bits each.

4. Initialization Vector (IV):

SHA1 uses a set of five 32bit initial hash values, denoted as `H0`, `H1`, `H2`, `H3`, and `H4`. These values are constants defined by the SHA1 specification.

5. Hash Computation:

SHA1 operates in rounds, with each round consisting of multiple steps that modify the initial hash values based on the input message. Each message block is processed sequentially through multiple rounds of operations, including bitwise operations (such as AND, OR, XOR), modular addition, and cyclic rotations.

The processing of each block produces a temporary hash value, which is combined with the initial hash values using bitwise addition (modular addition).

After processing all blocks, the final hash value is obtained by concatenating the five 32bit words representing the updated `H0`, `H1`, `H2`, `H3`, and `H4` values.

6. Output Hash Value:

The final output hash value produced by SHA1 is a fixed size 160bit (20byte) binary string. This hash value is typically represented as a hexadecimal string of 40 characters (160 bits / 4 bits per hexadecimal digit).

7. Security Considerations:

While SHA1 was once widely used for cryptographic purposes, it is now considered to be vulnerable to collision attacks, where different input messages produce the same hash value. Due to these vulnerabilities, SHA1 is no longer recommended for cryptographic applications, and more secure hash functions like SHA256 or SHA3 are preferred.

SHA1 is a cryptographic hash function that produces a 160bit hash value from an input message, providing a fixed size representation of the input data that is useful for integrity verification, digital signatures, and other security applications. However, its vulnerability to collision attacks has led to its deprecation in favor of more secure alternatives.

2. Algorithm 2:

SHA 256 was used to create new hash value Secure password hashing.

SHA-256 (Secure Hash Algorithm 256) is a cryptographic hash function that generates a fixed-size output hash value of 256 bits (32 bytes). It is one of the members of the SHA-2 (Secure Hash Algorithm 2) family, which also includes SHA-224, SHA-384, SHA-512, SHA-512/224, and SHA-512/256. How SHA-256 works and its applications in secure password hashing:

1. Input Message: SHA-256 operates on an input message of arbitrary length, similar to other cryptographic hash functions. This message can be any data, such as a password, text string, file contents, or binary data.

2. Padding: Before processing the input message, SHA-256 pads it to ensure that its length is a multiple of 512 bits. The padding scheme includes appending a single '1' bit followed by a sequence of '0' bits, and then appending the length of the original message (in bits) as a 64-bit representation.

3. Message Processing: The padded message is divided into blocks of 512 bits each. Each block is further divided into 16 words of 32 bits each.

4. Initialization Vector (IV): SHA-256 uses a set of eight 32-bit initial hash values, denoted as `H0`, `H1`, `H2`, `H3`, `H4`, `H5`, `H6`, and `H7`. These values are constants defined by the SHA-256 specification.

5. Hash Computation: SHA-256 operates in rounds, with each round consisting of multiple steps that modify the initial hash values based on the input message. Each message block is processed sequentially through multiple rounds of operations, including bitwise operations (such as AND, OR, XOR), modular addition, and cyclic rotations. The processing of each block produces a temporary hash value, which is combined with the initial hash values using bitwise addition (modular addition). After processing all blocks, the final hash value is obtained by concatenating the eight 32-bit words representing the updated `H0` to `H7` values.

6. Output Hash Value: The final output hash value produced by SHA-256 is a fixed-size 256-bit (32-byte) binary string. This hash value is typically represented as a hexadecimal string of 64 characters (256 bits / 4 bits per hexadecimal digit).

7. Secure Password Hashing: SHA-256 is commonly used for secure password hashing in various applications, including authentication systems, user account management, and cryptographic protocols. When a user creates or updates their password, the password is hashed using SHA-256 before being stored in a database or directory service. During authentication, the user-provided password is hashed using SHA-256, and the resulting hash value is compared to the stored hash value to verify the authenticity of the password. By using SHA-256 for password hashing, plaintext passwords are not stored directly, providing an additional layer of security against unauthorized access and data breaches.

SHA-256 is a widely used cryptographic hash function that generates a secure and fixed-size hash value of 256 bits, making it suitable for secure password hashing and other cryptographic applications where data integrity and authenticity are essential.

3.SHA 3:

SHA 3 provides a random mapping from a string from solidity.

SHA-3 (Secure Hash Algorithm 3) is a cryptographic hash function that generates a fixed-size hash value from an input message. While SHA-3 itself does not directly provide a "random mapping" from a string, it can be used in conjunction with other techniques to achieve a pseudo-random mapping.

How SHA-3 can be utilized in Solidity to create a pseudo-random mapping from a string:

1. Hashing in Solidity: In Solidity, the `keccak256` function is used to compute the SHA-3 hash of a given input. This function takes an arbitrary-length input (such as a string) and returns a fixed-size 256-bit hash value. The `keccak256` function is commonly used in Solidity smart contracts for cryptographic purposes, such as generating unique identifiers, verifying data integrity, or creating pseudo-random values.

2. Seed for Randomness: To create a pseudo-random mapping from a string in Solidity, you can use the `keccak256` hash of the input string as a seed for generating random values. The input string can represent any data that you want to use as a basis for randomness, such as user-provided data, timestamps, block hashes, or other variables. By hashing the input string using `keccak256`, you obtain a fixed-size hash value that serves as a deterministic representation of the input. This hash value can then be used as a seed for generating pseudo-random numbers or other random outputs.

3. Random Number Generation: Once you have obtained the `keccak256` hash of the input string, you can use it as a seed for generating random numbers or other random outputs in Solidity. There are various techniques for generating pseudo-random numbers in Solidity, such as using the block timestamp, block hash, or cryptographic techniques like the Mersenne Twister algorithm. By combining the `keccak256` hash of the input string with other sources of randomness or entropy, you can create a more robust and unpredictable pseudo-random mapping in your Solidity smart contracts.

SHA-3 itself does not directly provide a random mapping from a string, it can be combined with other techniques in Solidity to create pseudo-random mappings that are deterministic based on the input string. This approach allows for the generation of pseudo-random values in Solidity smart contracts based on user-provided data or other sources of input.

**Chapter 7**

**Software Testing**

* 1. TYPE OF TESTING:

There were two types of testing conducted for this particular model 1) Unit testing 2) Integration Testing.

1. Unit Testing: This testing was mainly conducted on the smart contracts included in the system Remix IDE is was use for the verification and working of the smart contracts. All the contracts have been tested if the contracts work in properly or not.
2. Integration Testing: Integration testing was done when the frontend and backend was connected. The smart contracts and frontend were overlapped and checked id the functions in smart contract and the frontend performing the function for the user work together or not.

These were the two major testing carried out during the testing phase.

* 1. Test Cases and Test Results:

1. Test Case Title: Verify Certificate Functionality

Test Case ID: TC001

Priority: High

1. Preconditions:

1) The blockchain certificate verification website is accessible.

2) The user has a valid certificate hash and the associated document.

3. Test Steps:

1) Launch the blockchain certificate verification website.

2) Navigate to the certificate verification section.

3) Enter the certificate hash into the provided input field.

4) Click on the "Verify" button.

5) Wait for the verification process to complete.

4. Expected Results:

1) The system should display a message confirming the verification status of the certificate.

2) If the certificate is valid:

The system should display details such as the certificate holder's name, issuing organization, and date of issuance.

The document should be accessible for download or viewing.

3)If the certificate is invalid or not found:

The system should display an error message indicating the certificate could not be verified.

The reason for the failure should be clearly stated, e.g., "Invalid hash" or "Certificate not found."

5.Post Conditions:

1) The certificate verification process should be completed successfully.

2) The system should return to the main interface or remain on the verification page for further actions.

6. Test Data:

1) Valid certificate hash: [Provide a valid certificate hash]

2) Invalid certificate hash: [Provide an invalid certificate hash]

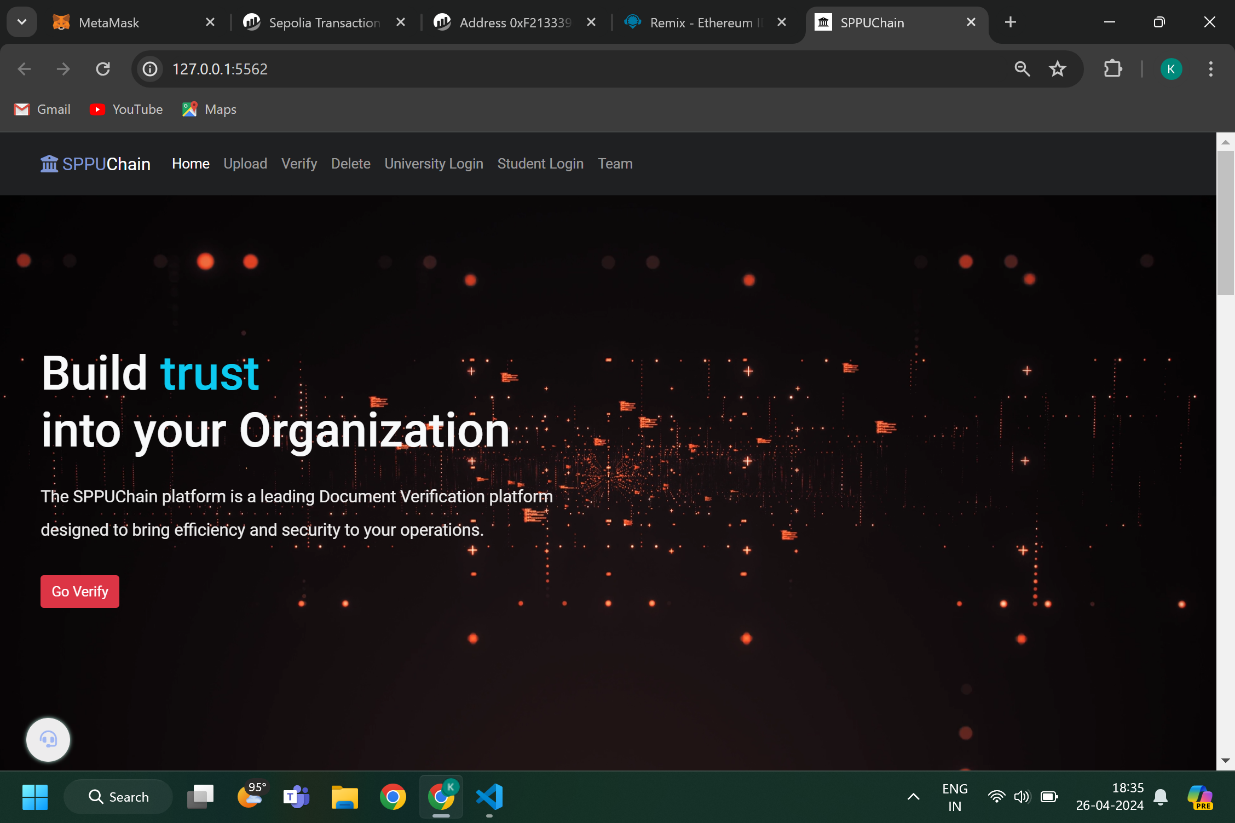
3) Certificate document: [Provide a sample certificate document for testing]

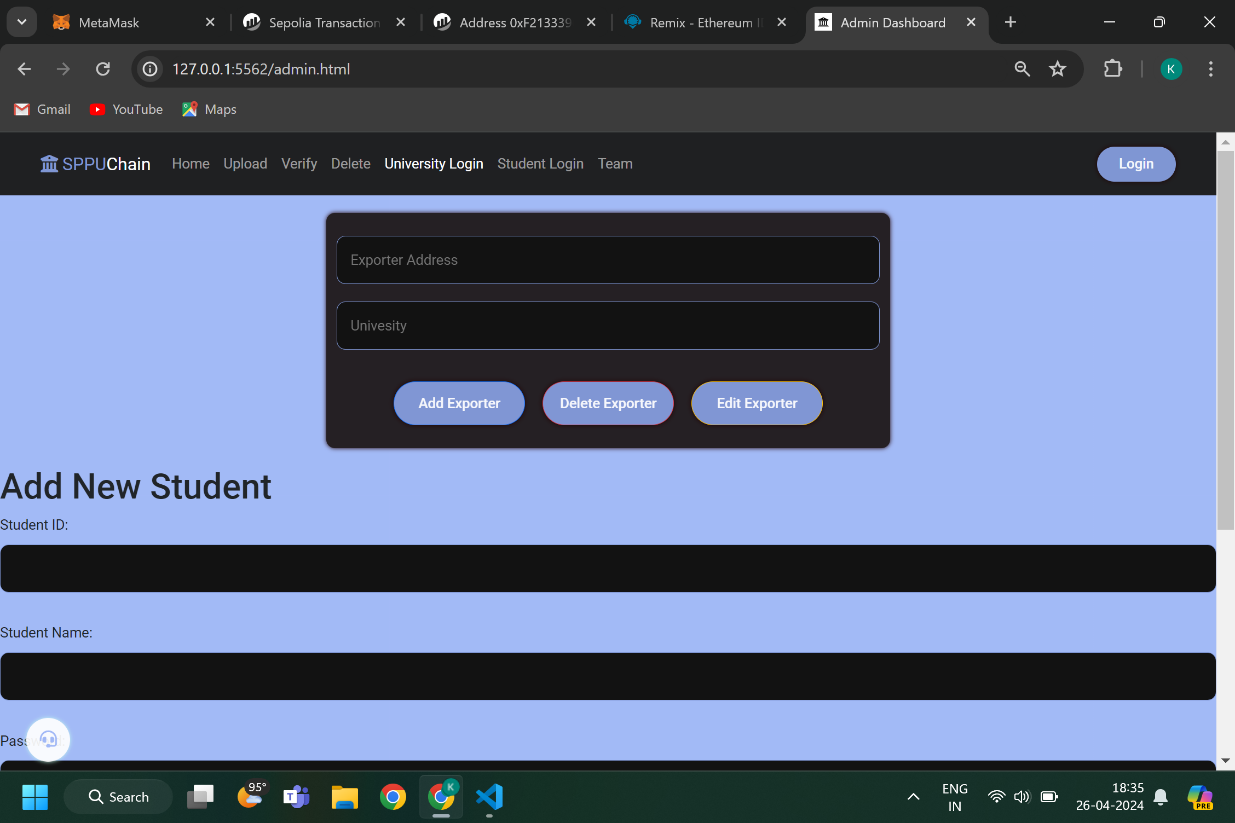
**Chapter 8**

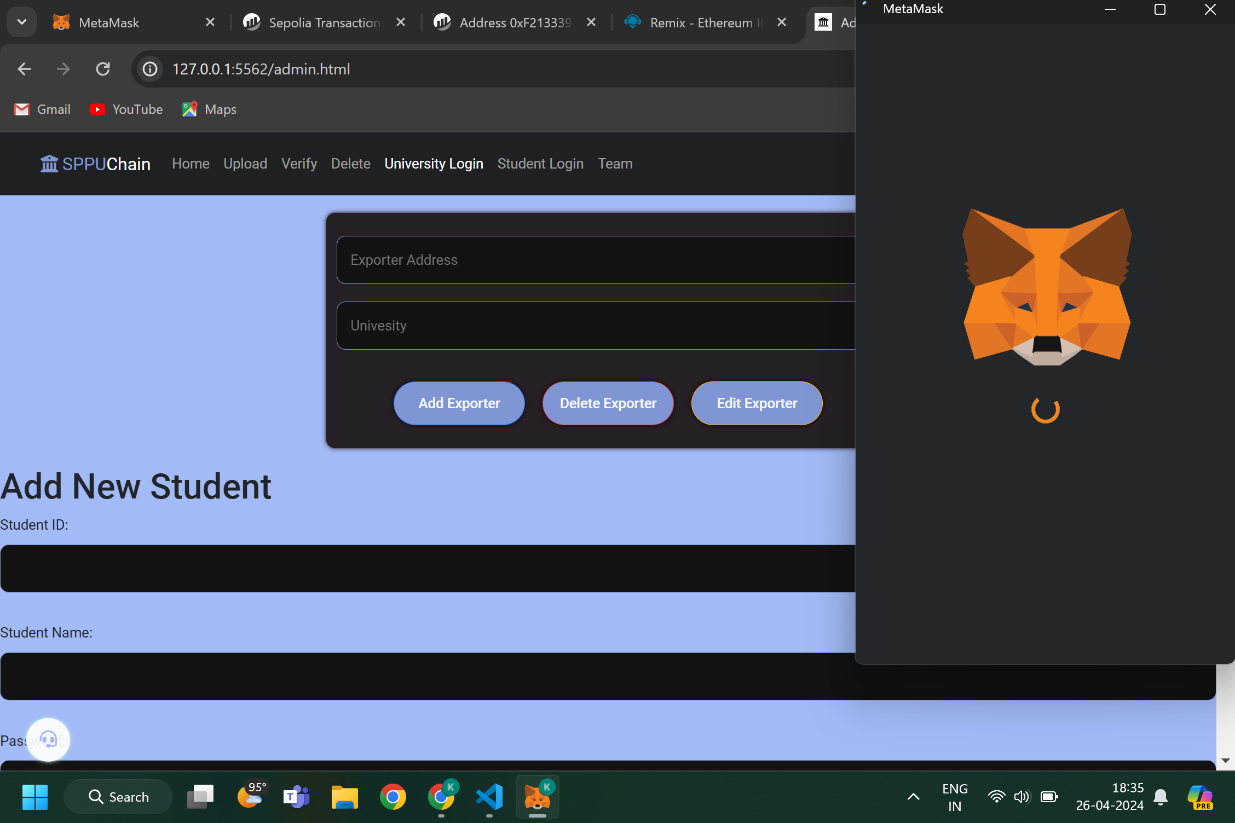
**Results**

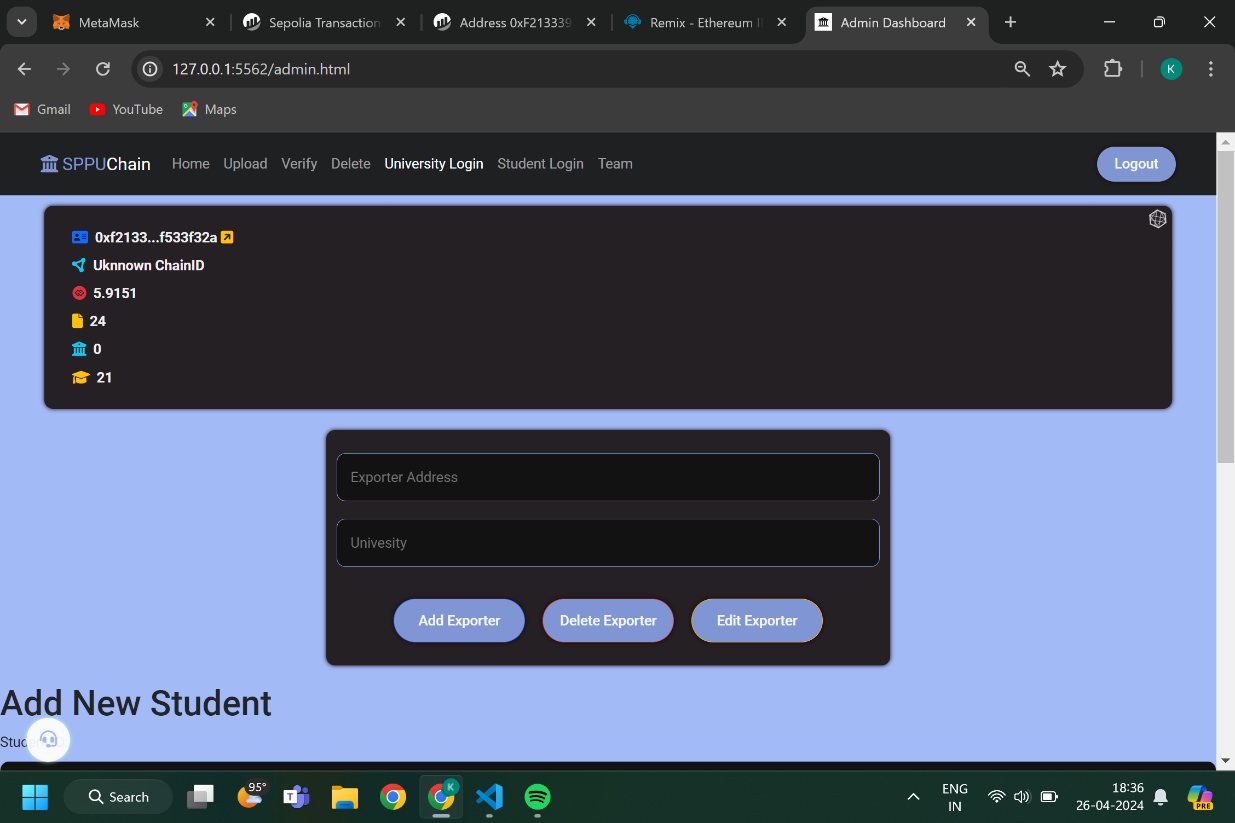
**8.1 OUTCOMES:**

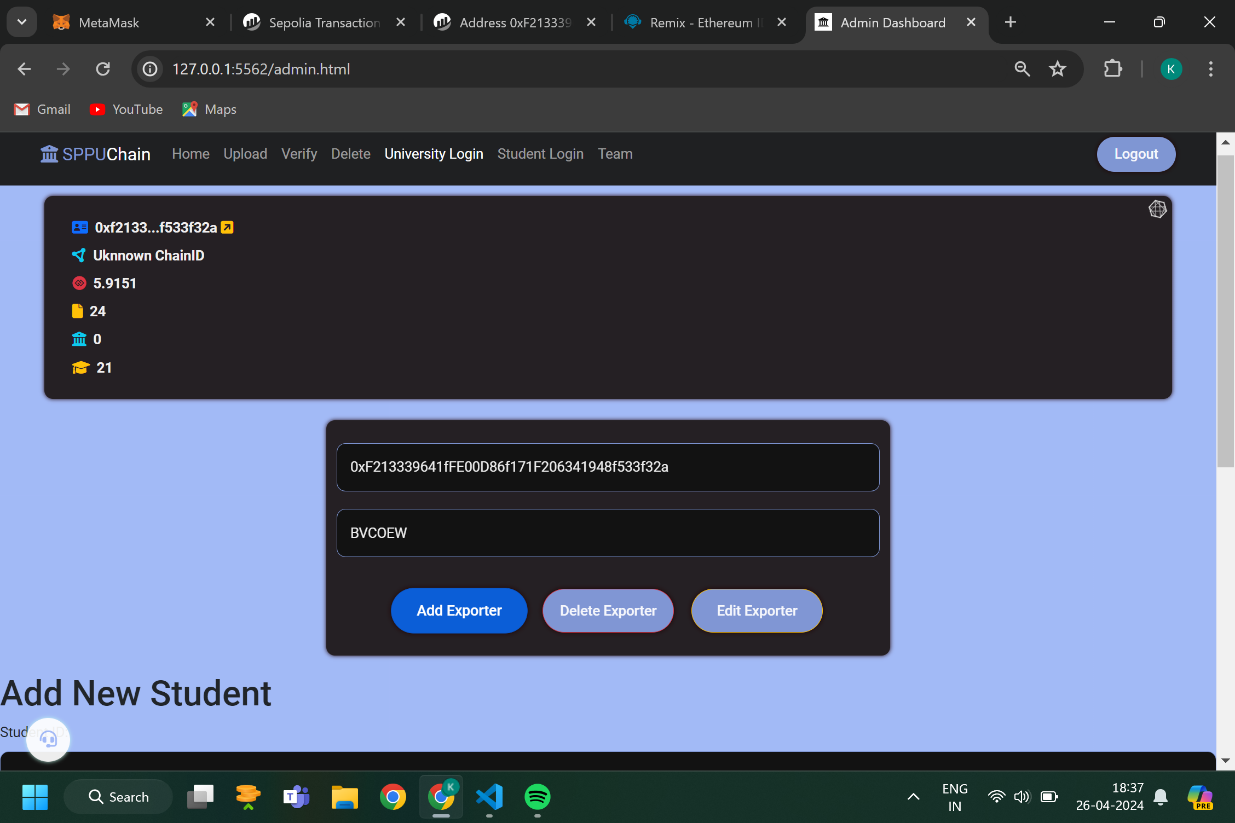
**8.2 SCREENSHOTS:**

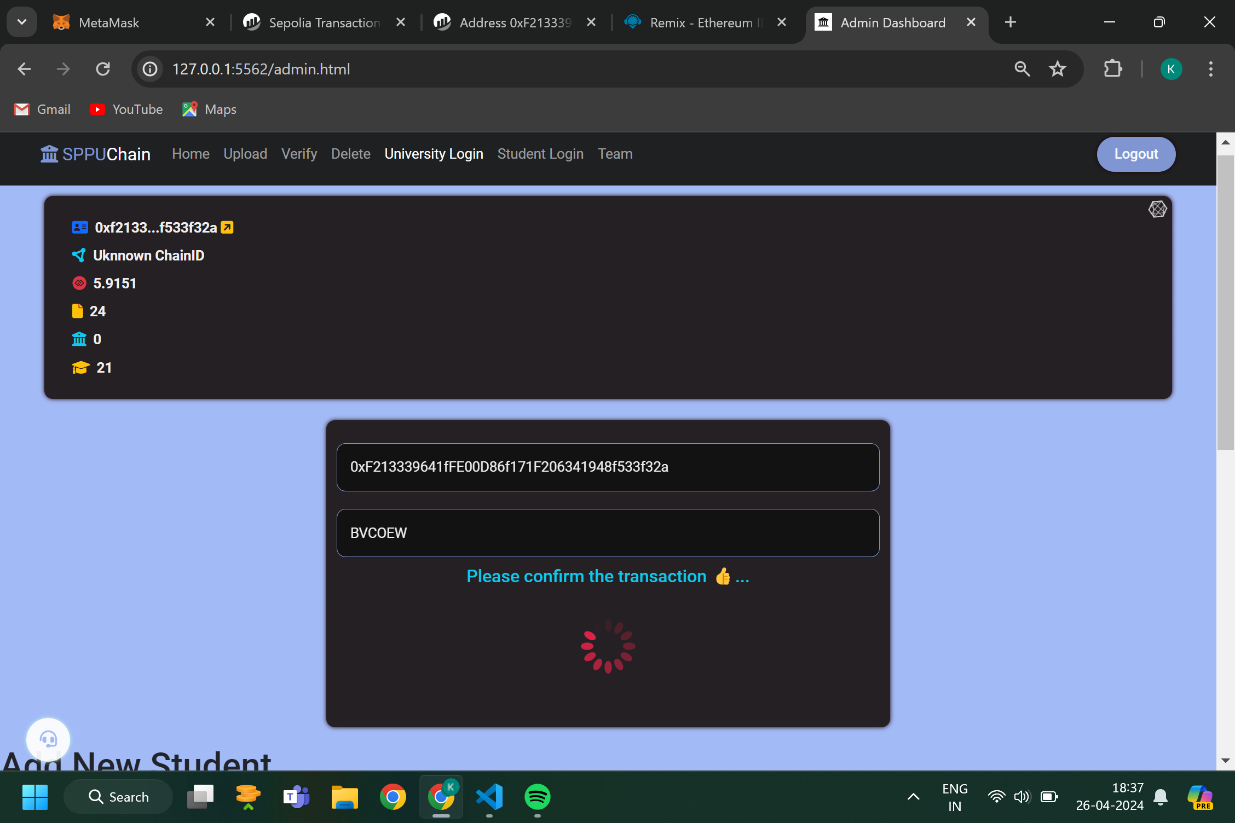
****

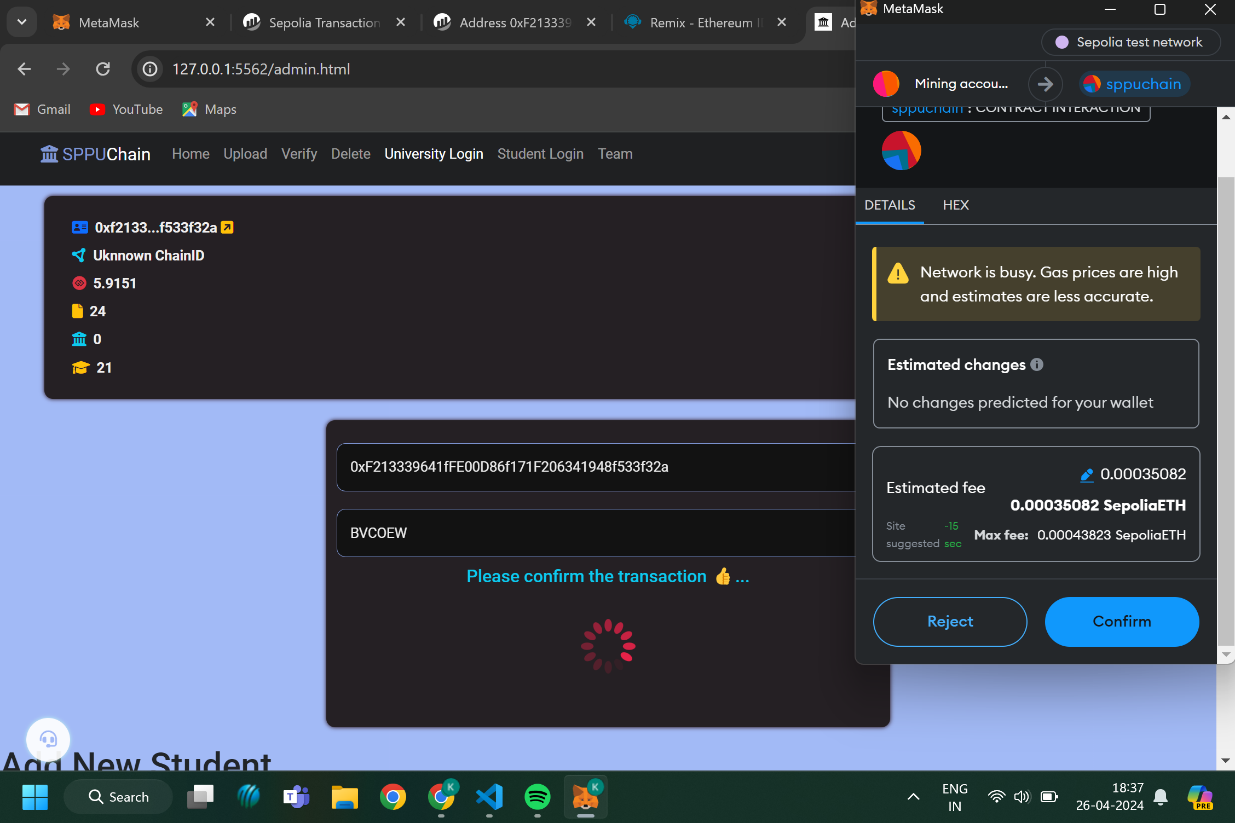


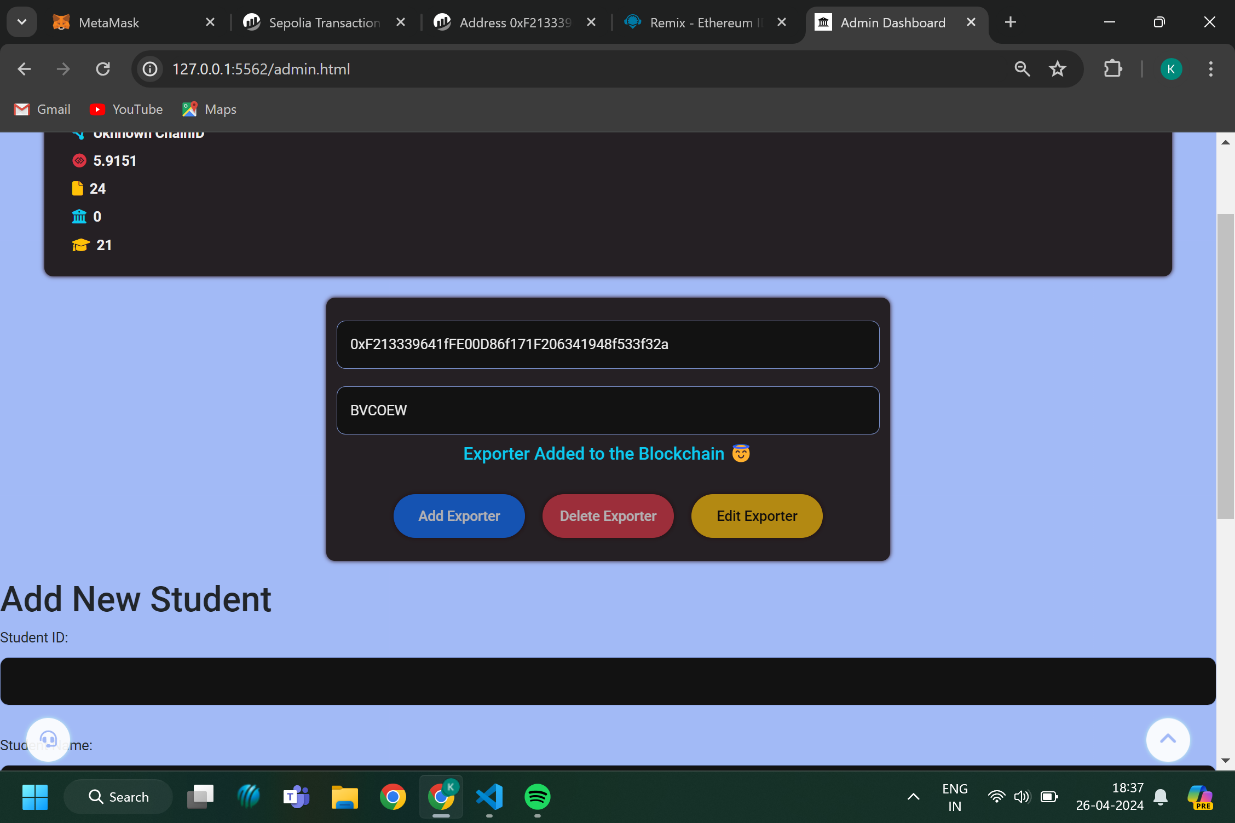


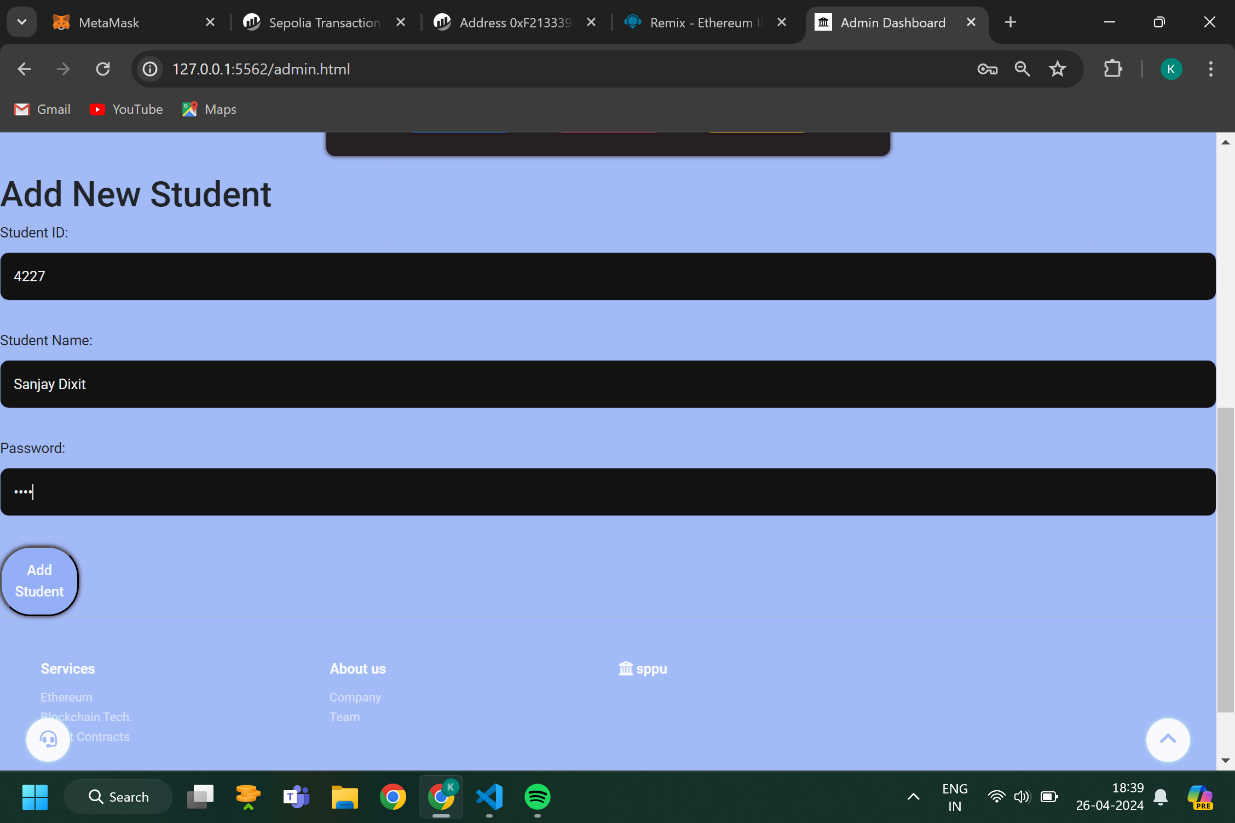


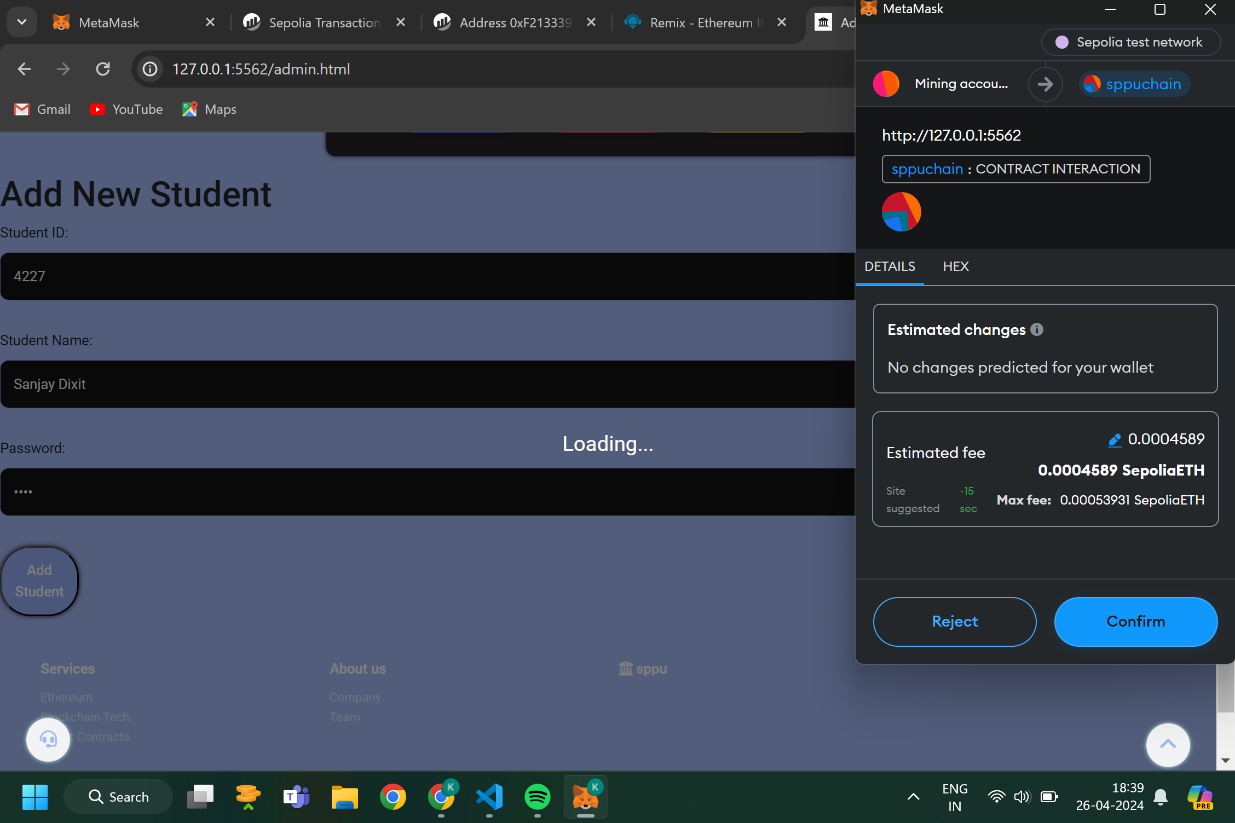


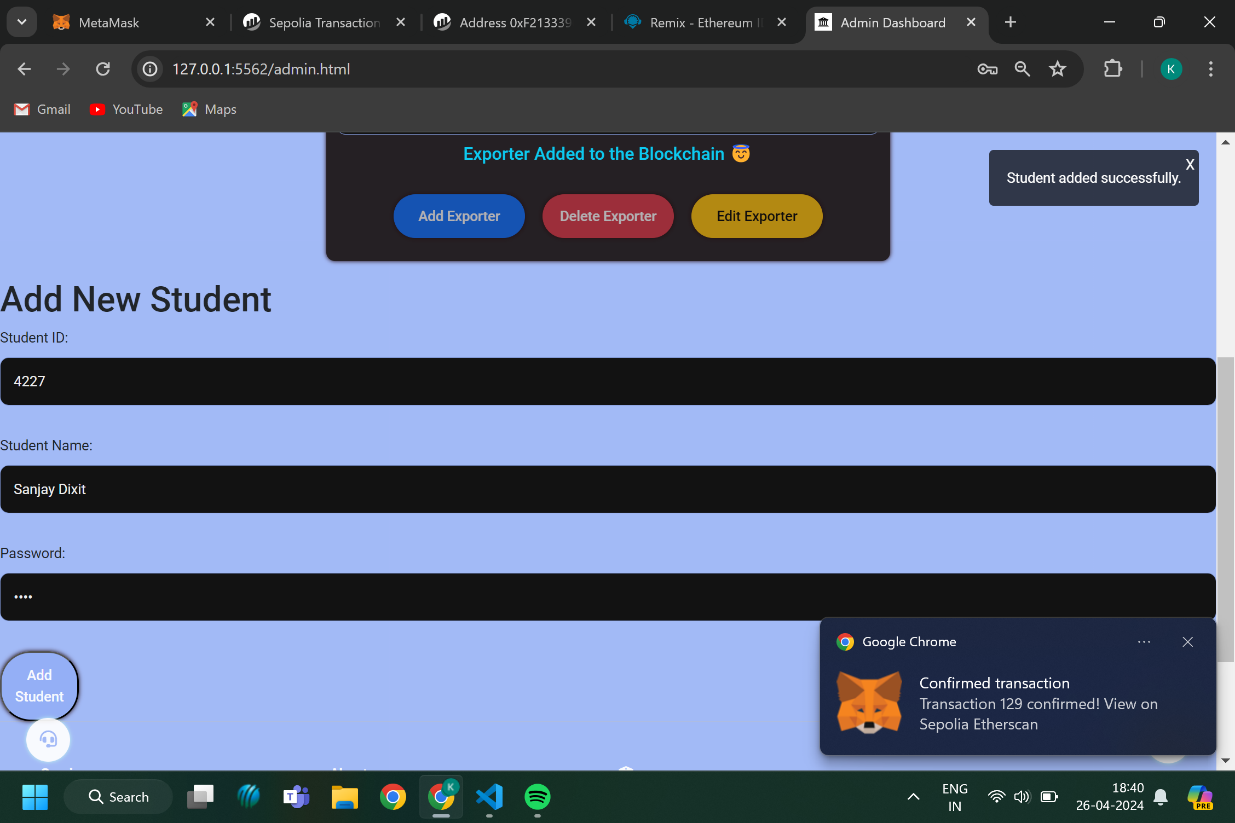


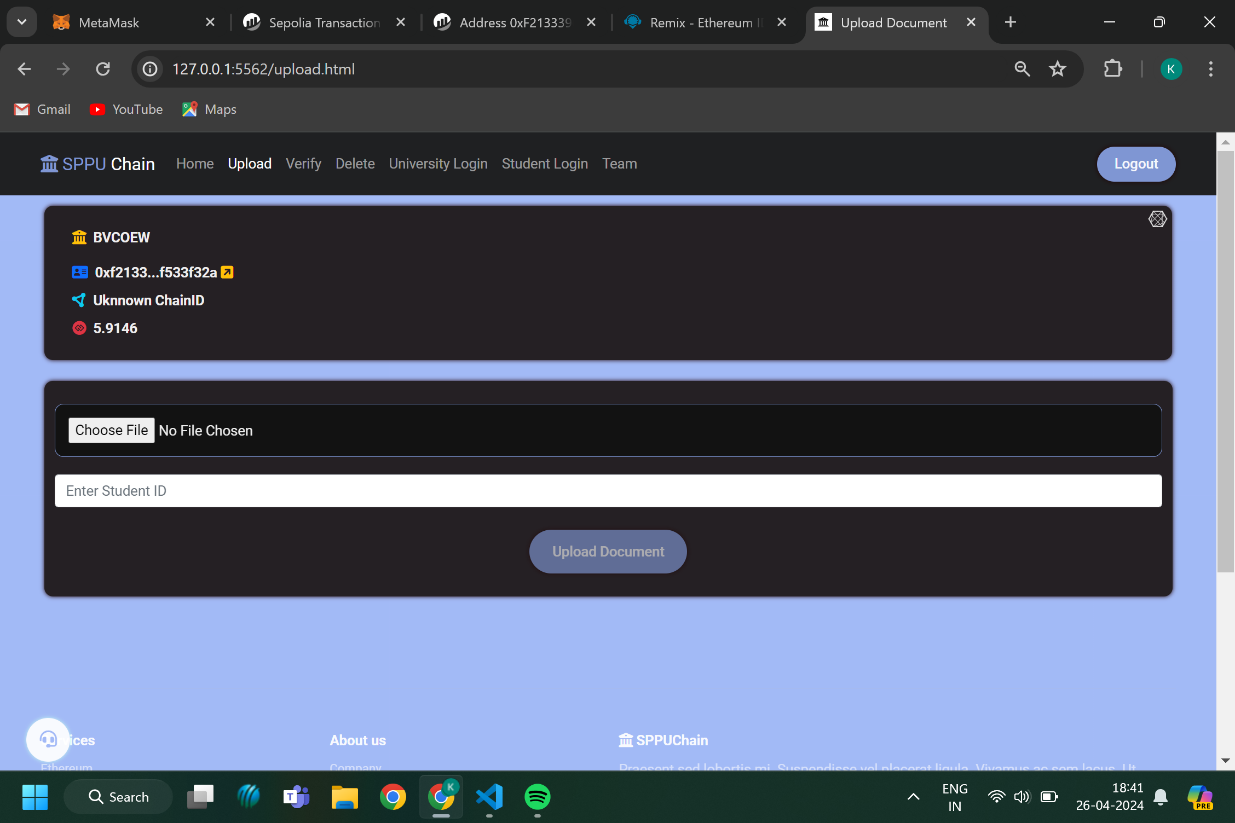


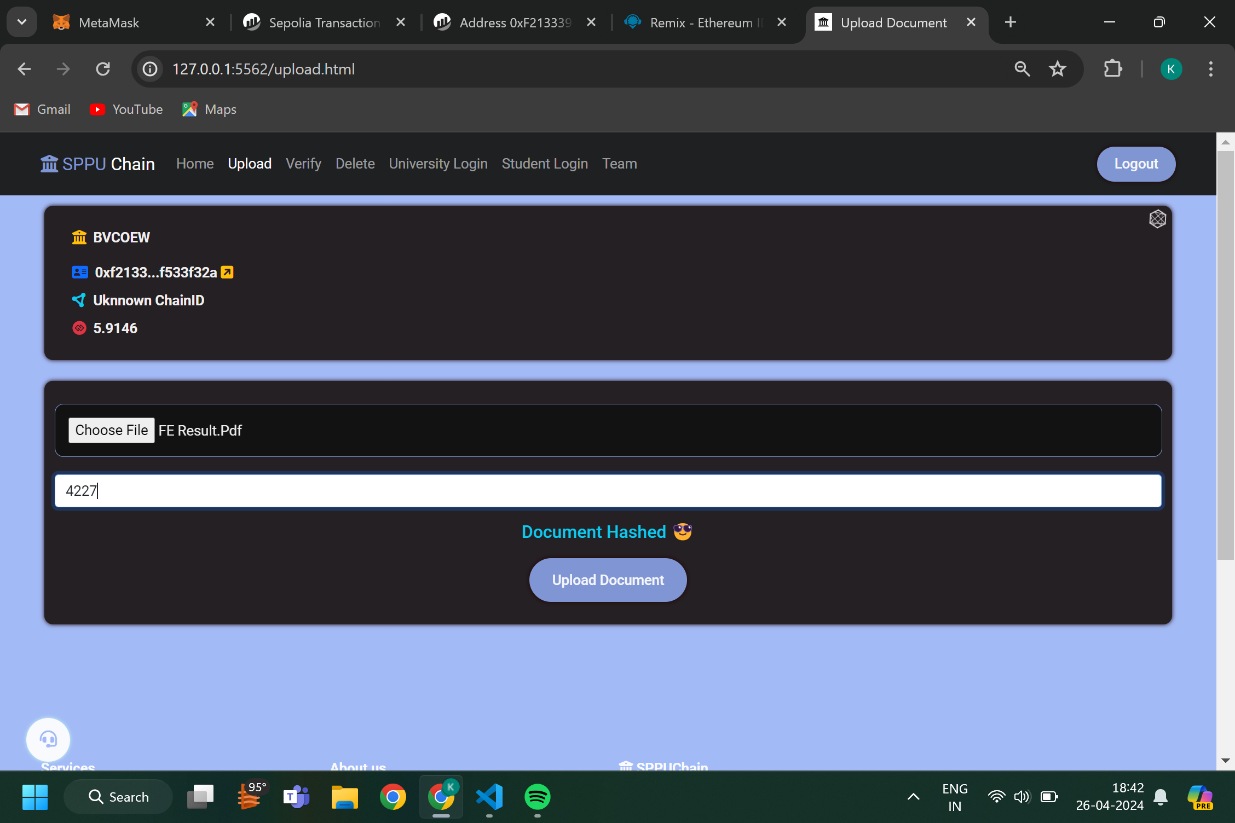


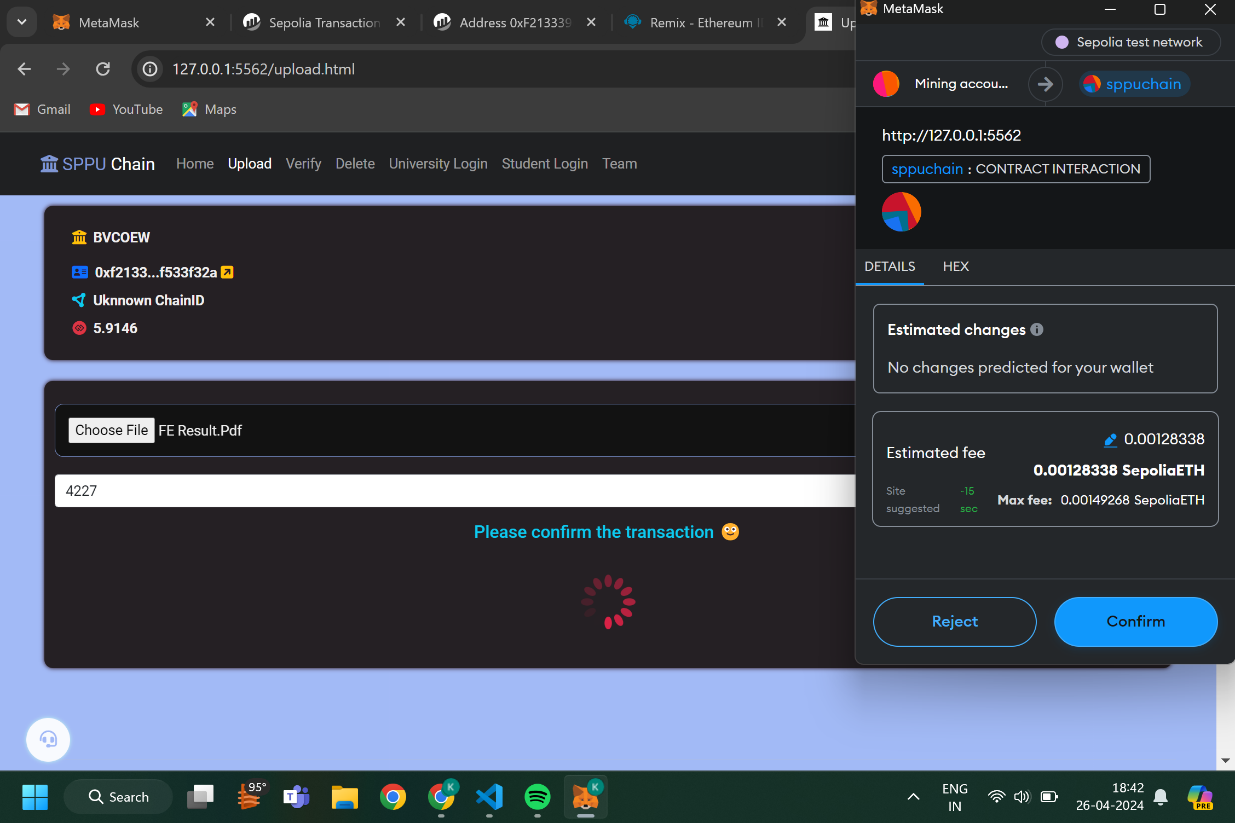


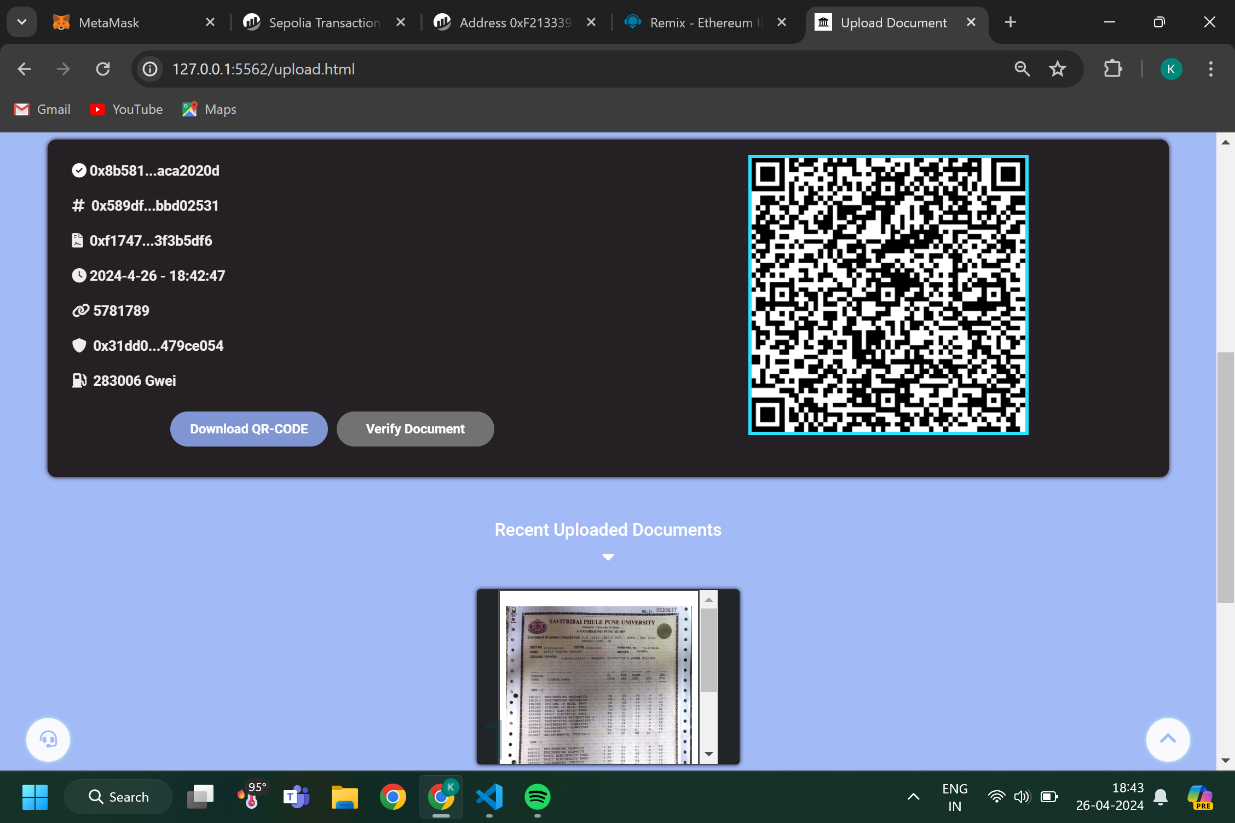


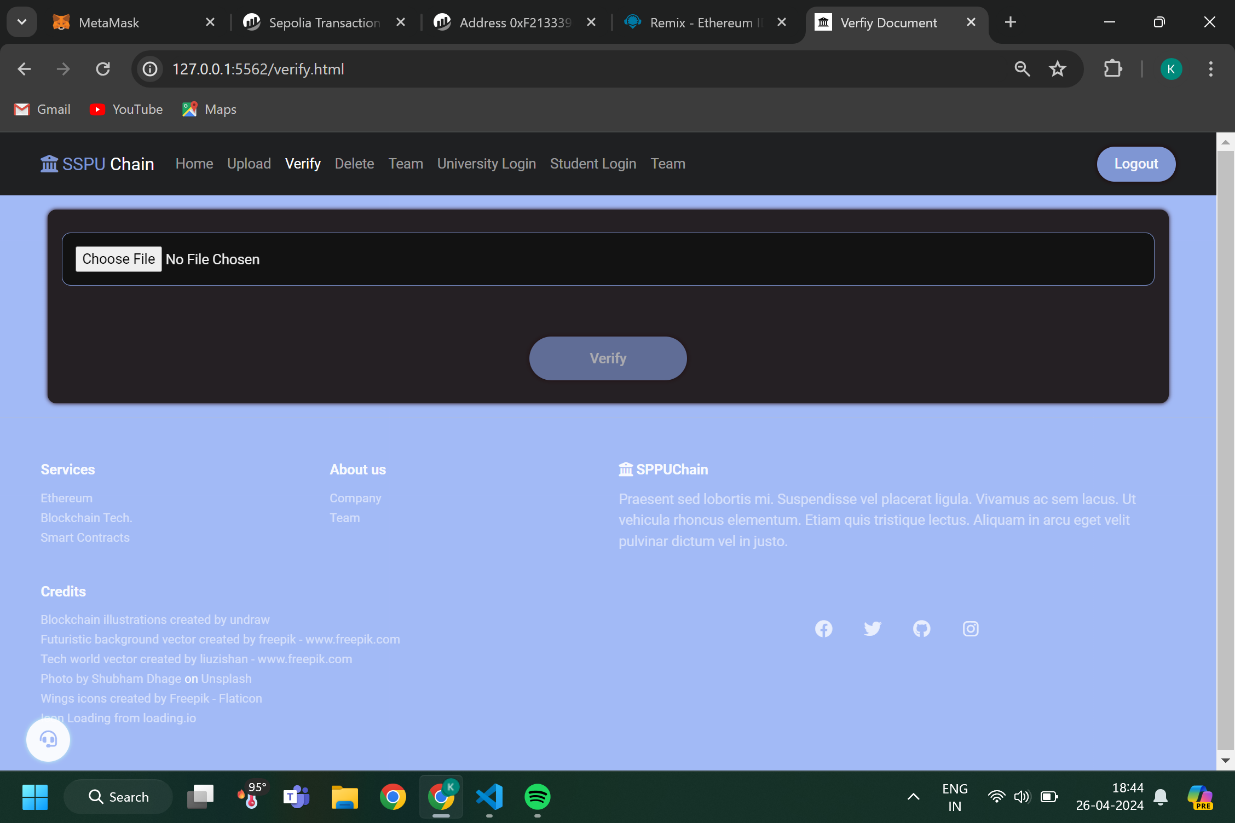


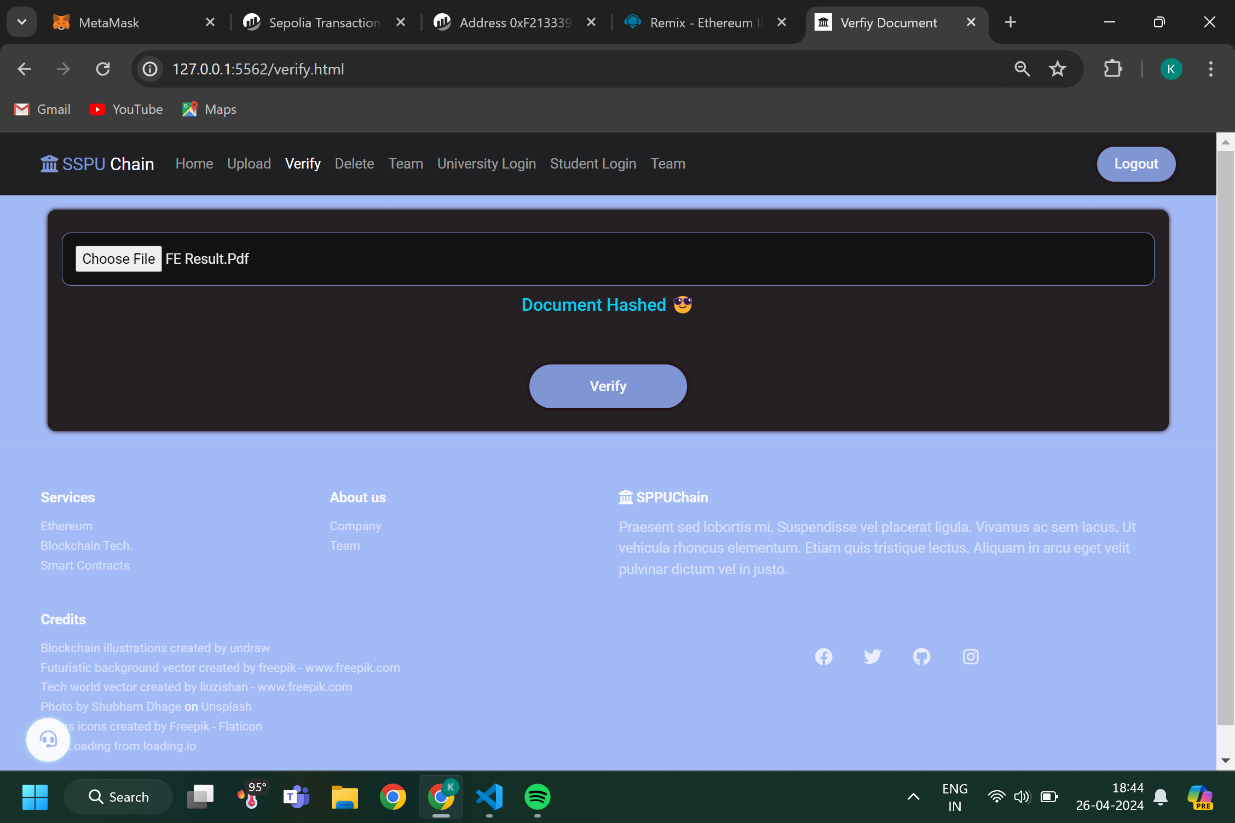


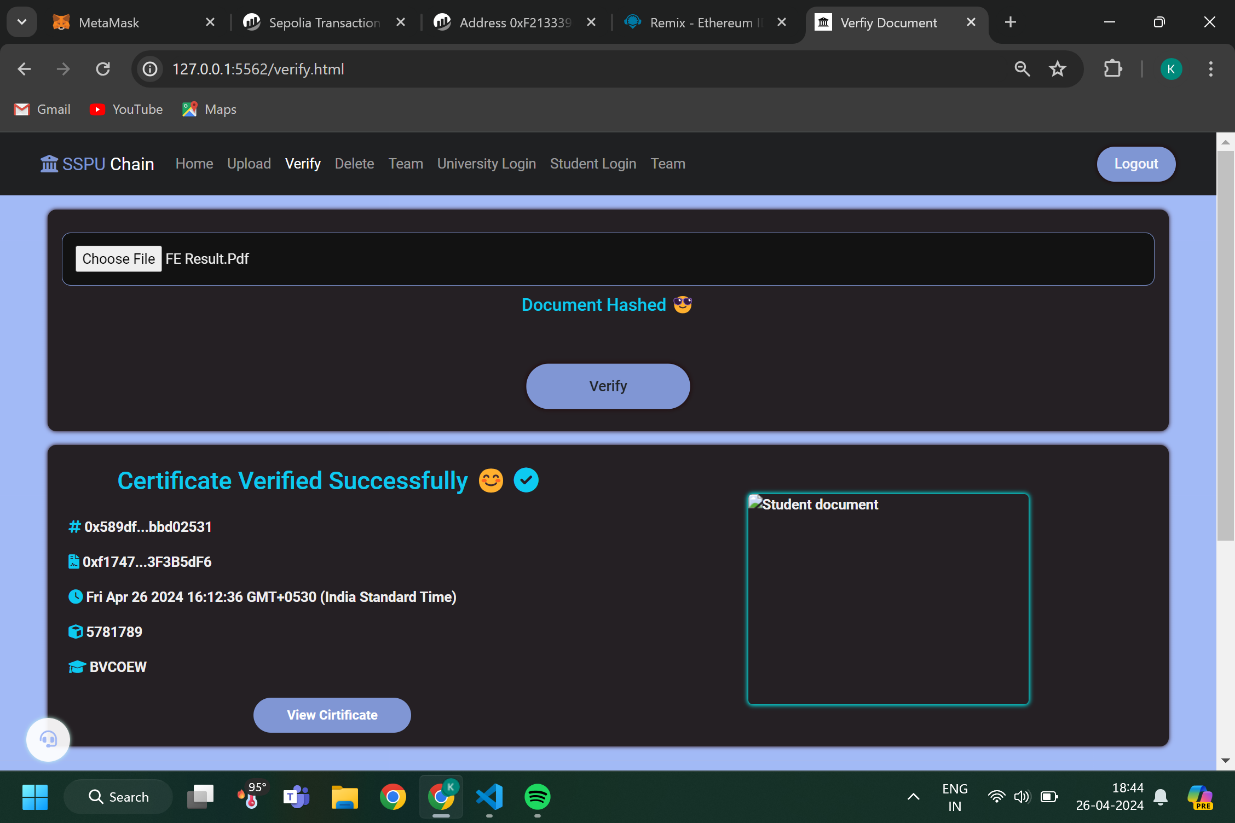




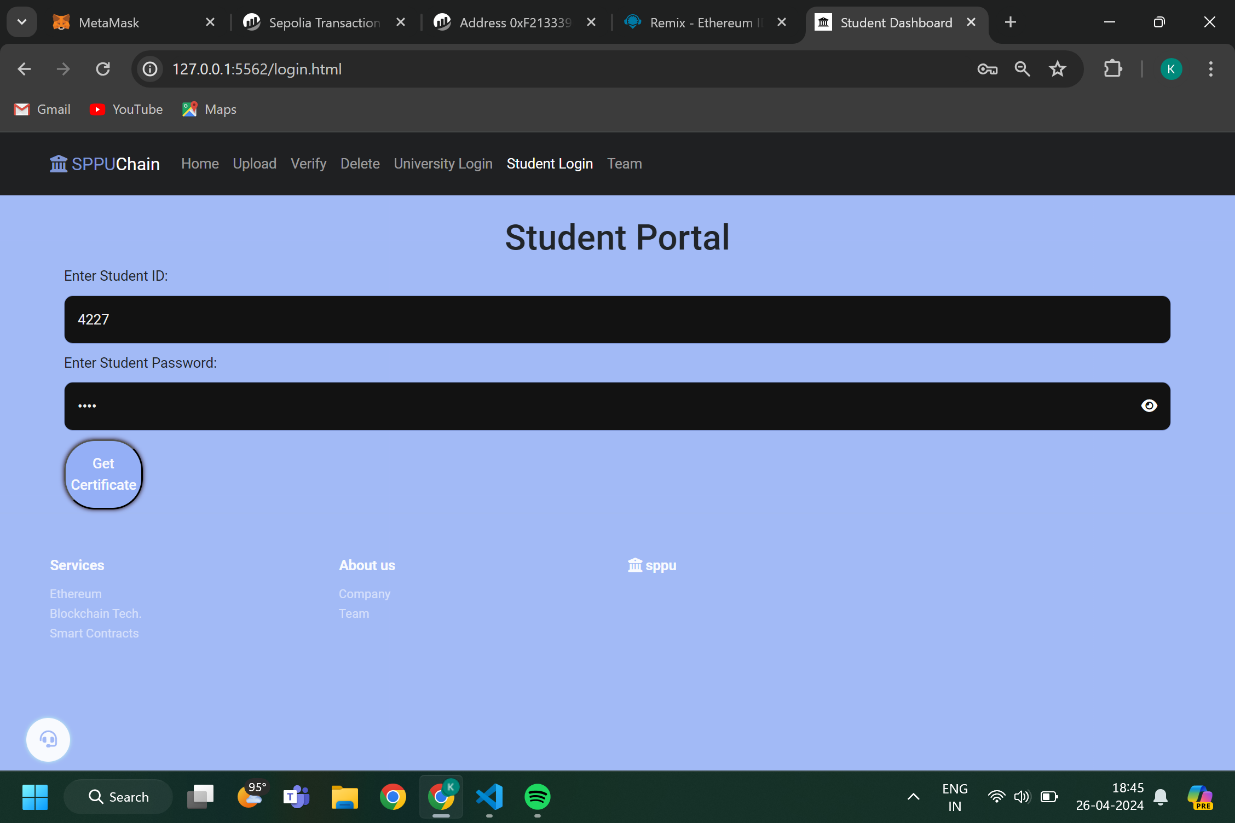


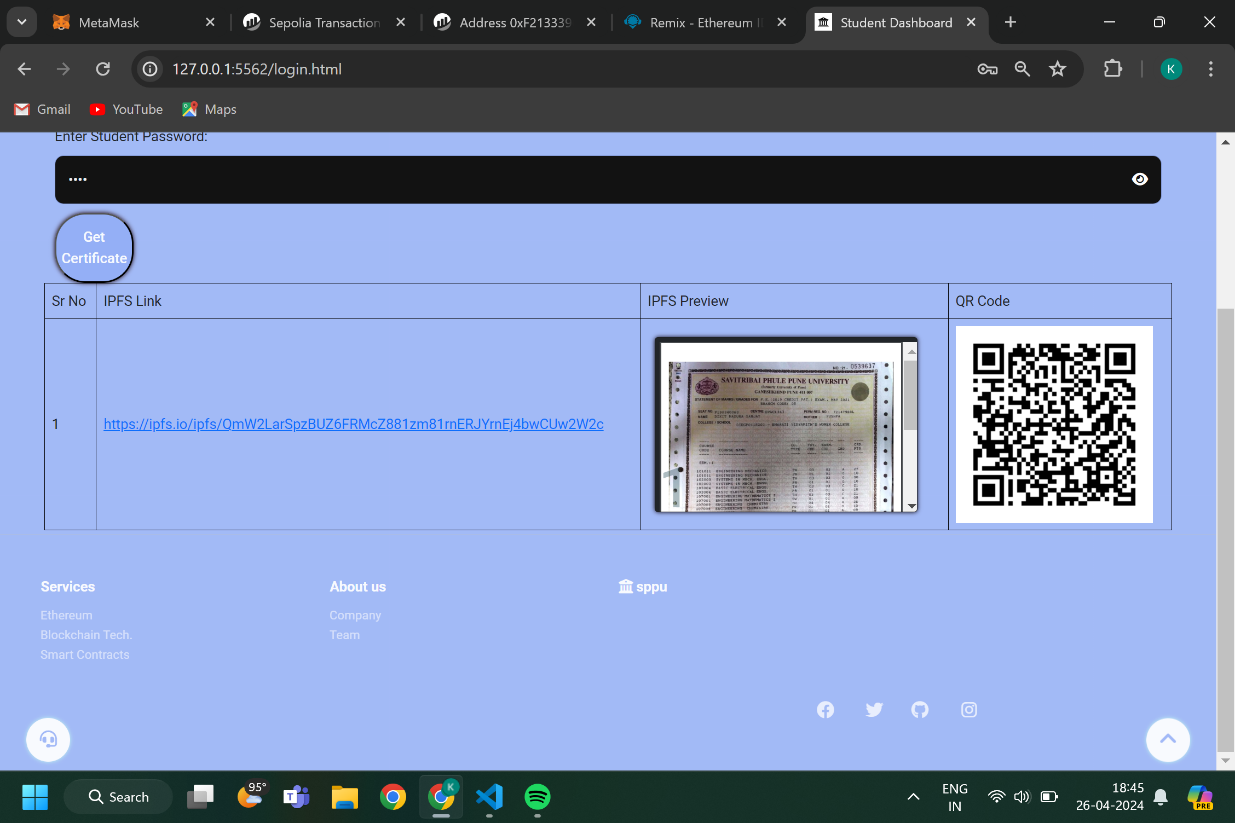


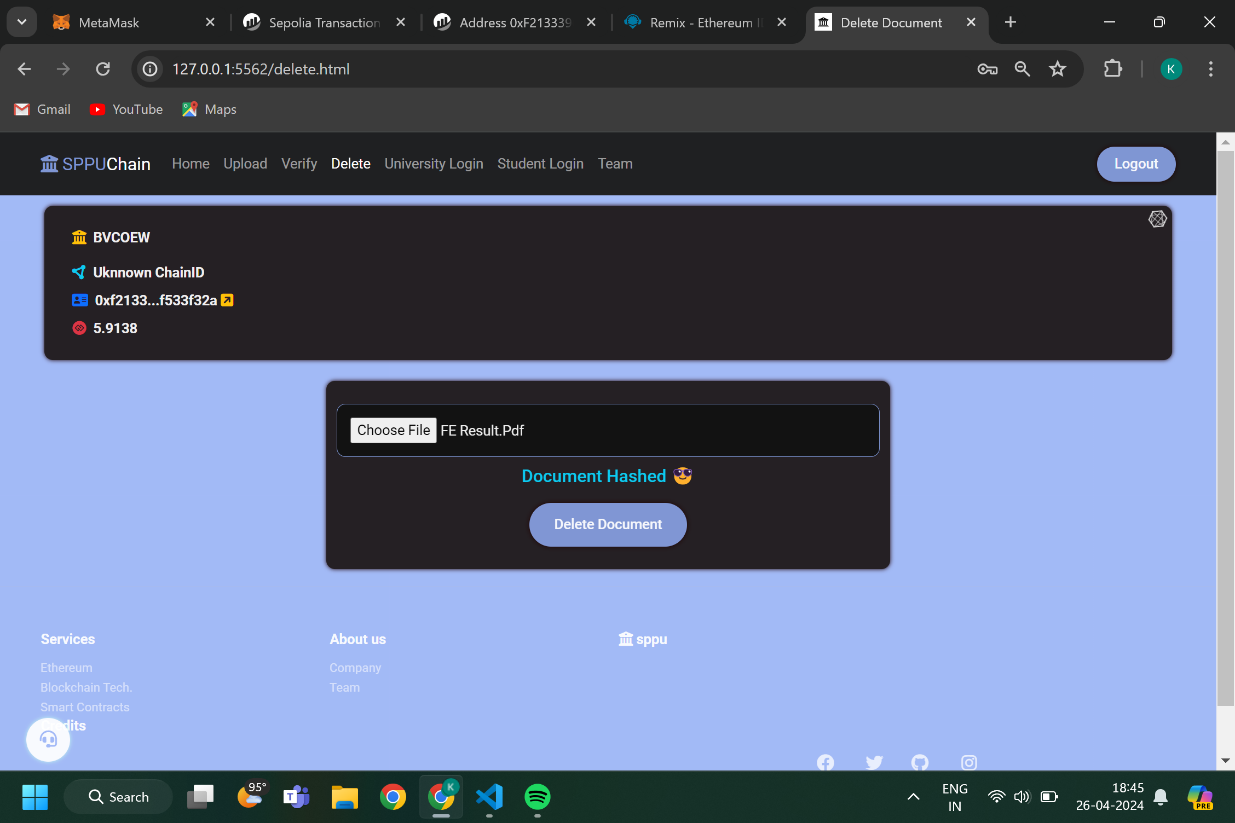


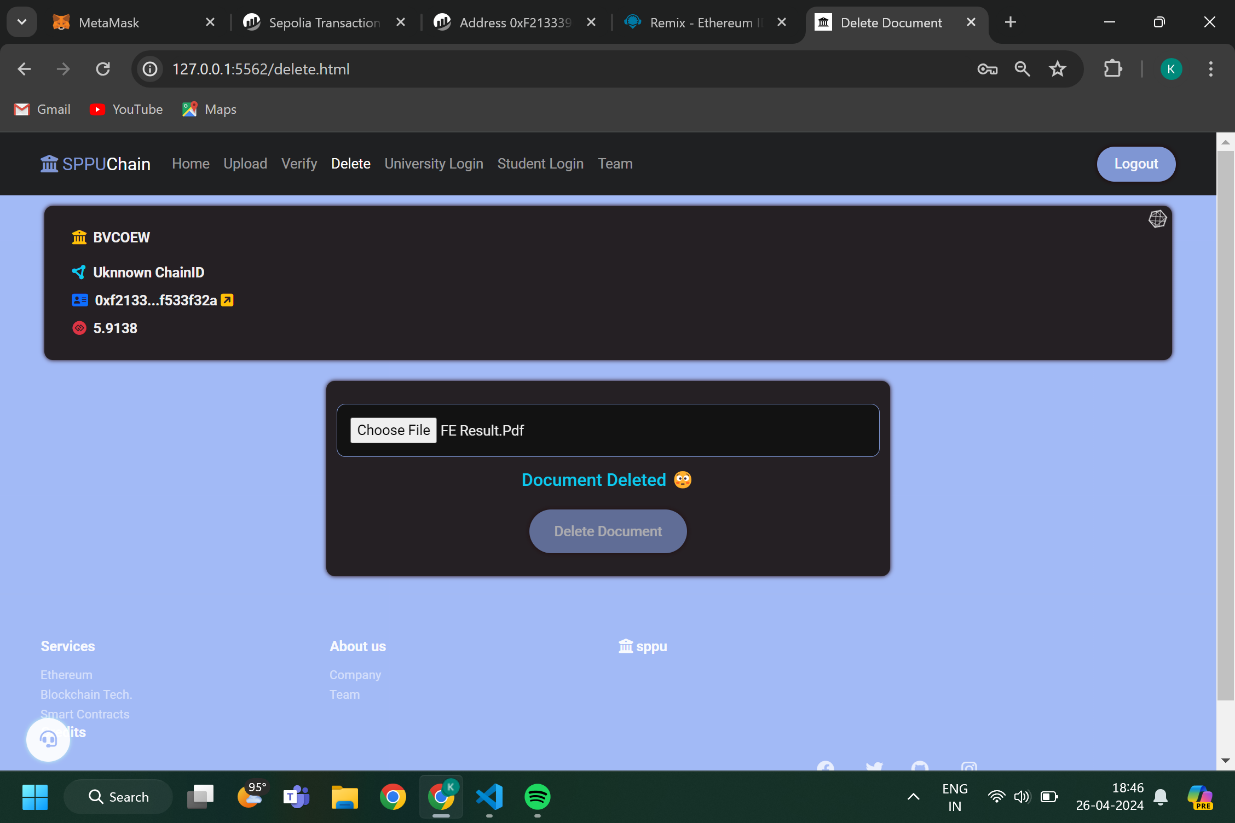


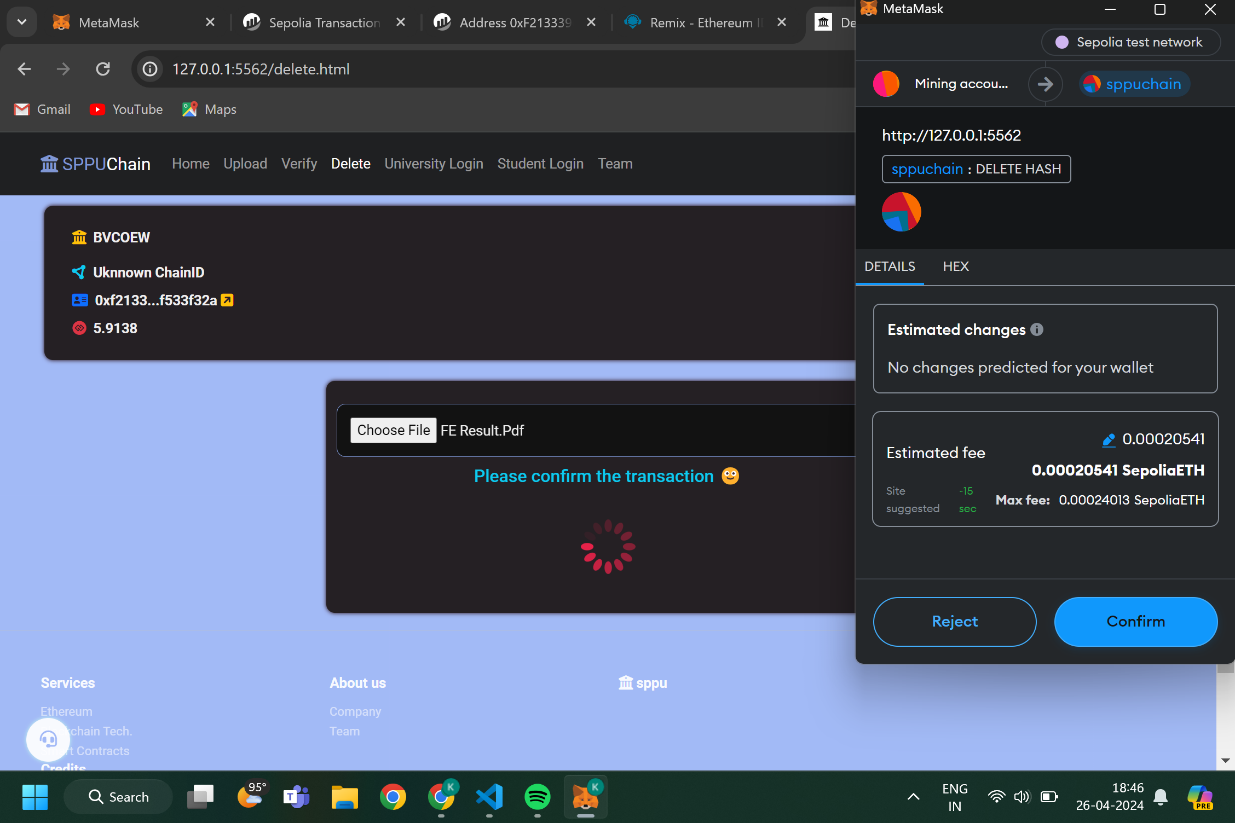


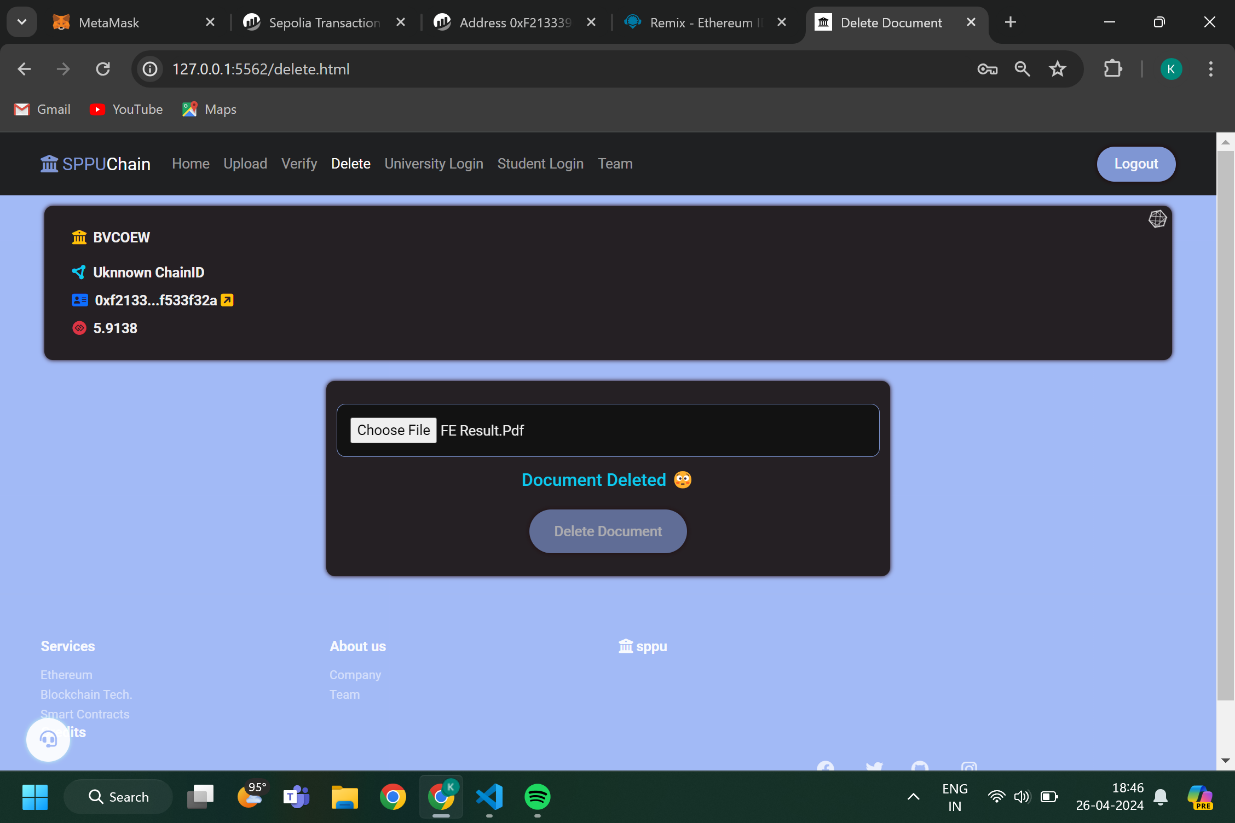


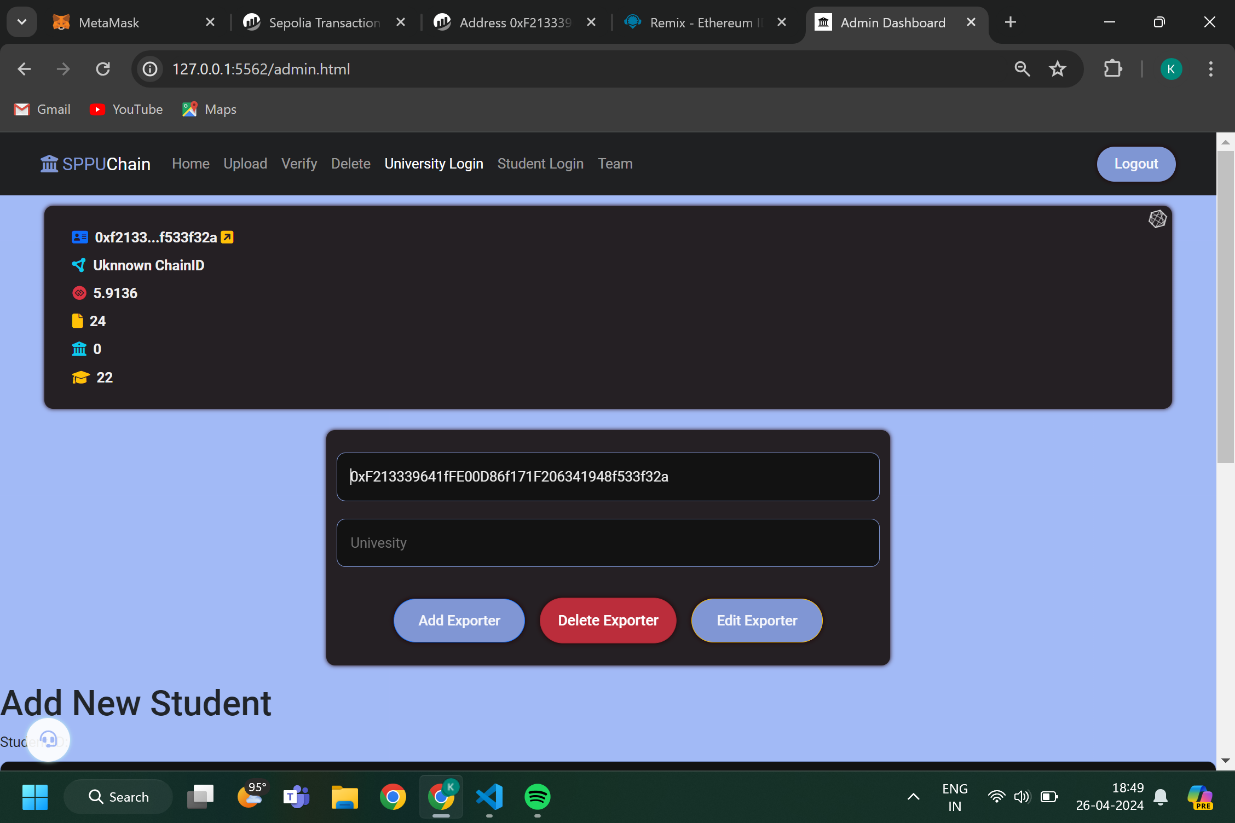


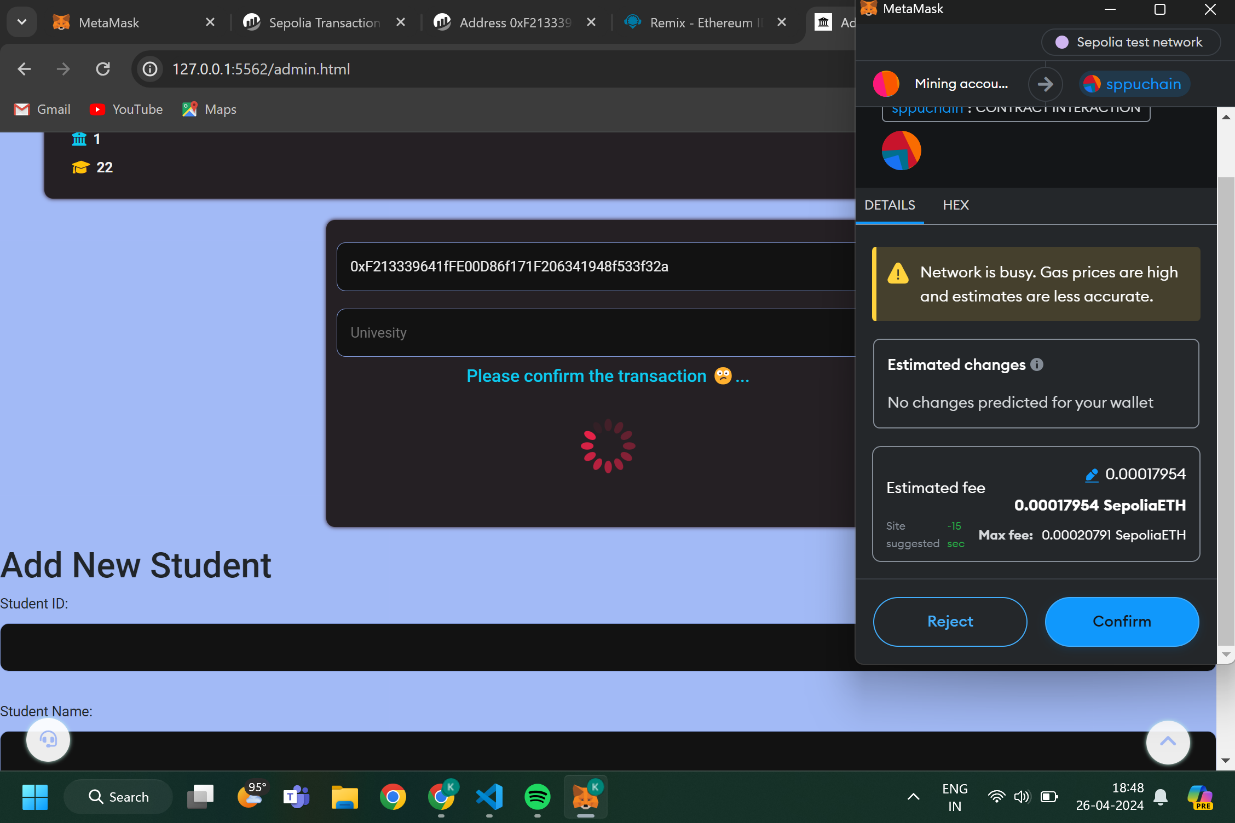




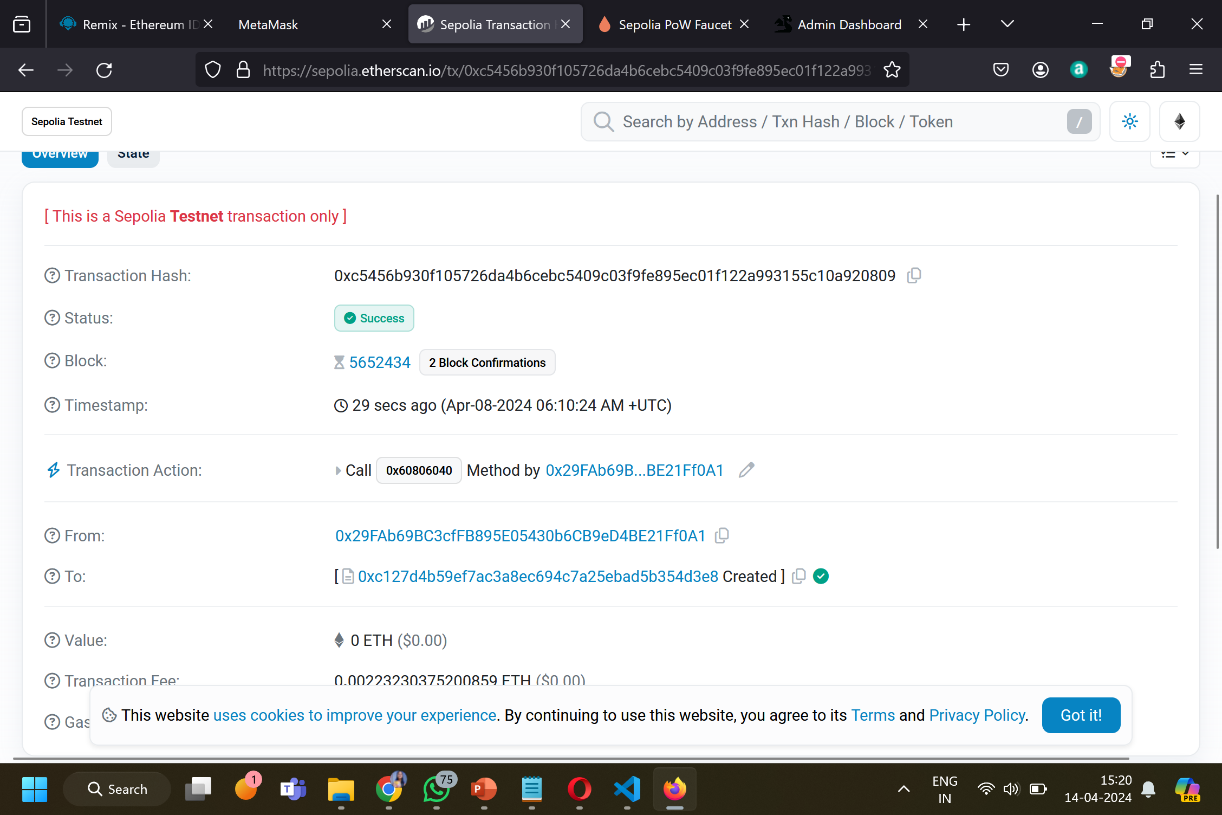


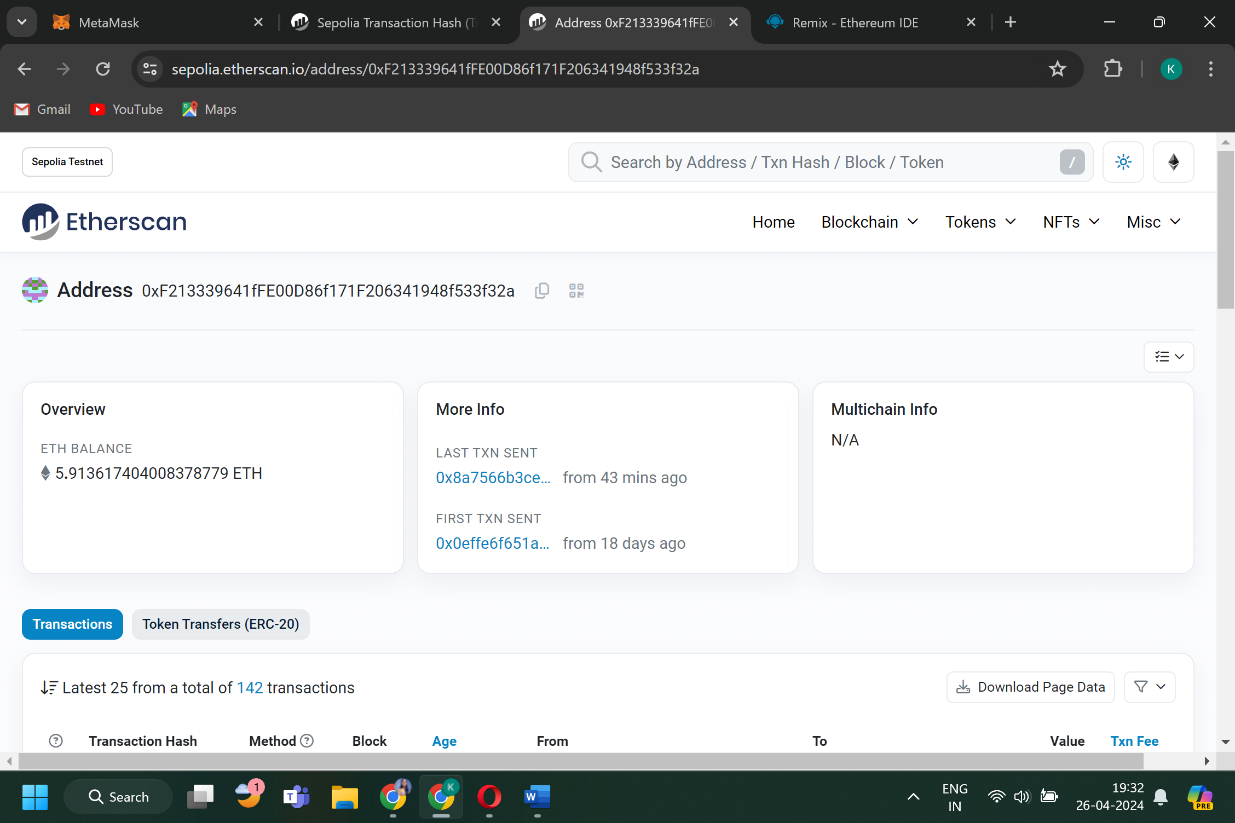


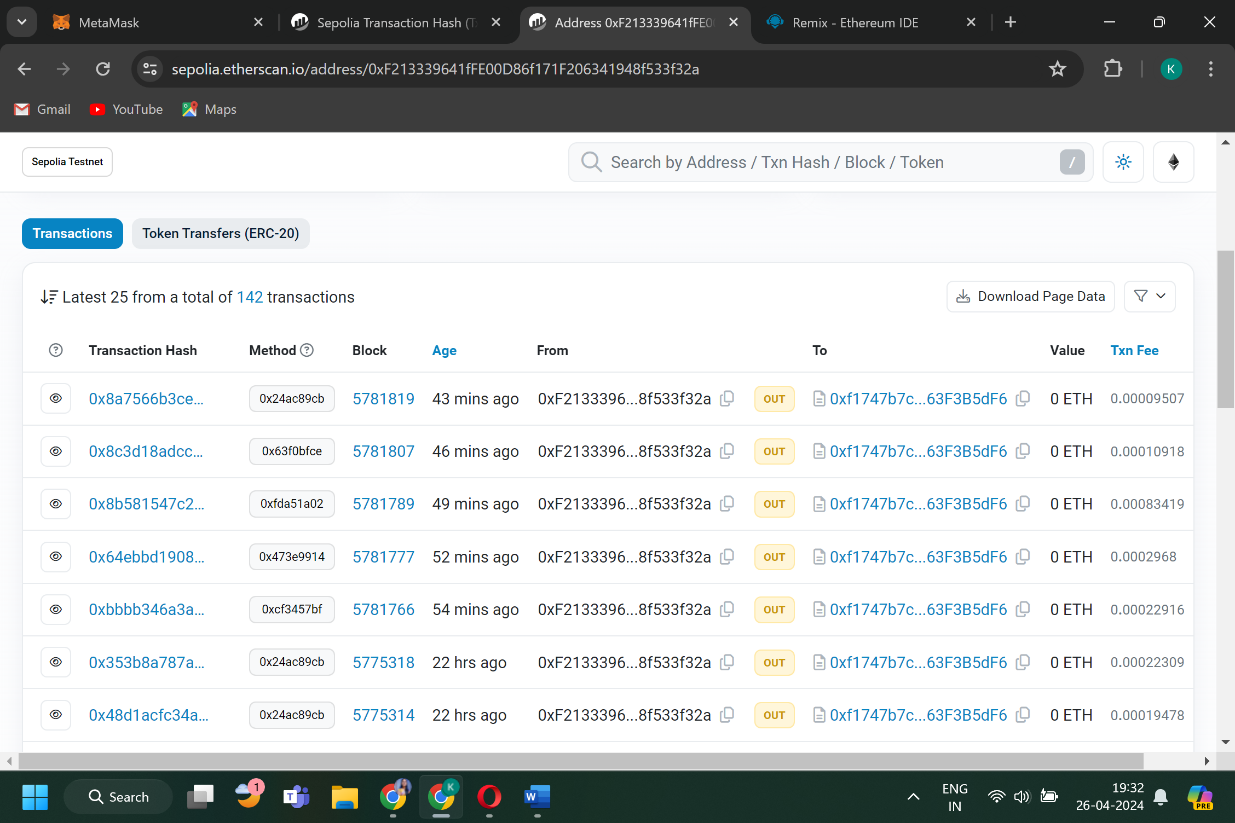












**Chapter 7**

**Conclusion**

**Conclusion:**

In conclusion, here we implement the application of blockchain technology in result management that holds significant potential for enhancing data security, transparency, and efficiency. By leveraging blockchain's decentralized framework, educational institutions can ensure tamper-proof record-keeping, streamlined verification processes, and improved data integrity. Embracing this technology can foster a more reliable and trustworthy result management system, contributing to the overall integrity of educational evaluation processes.

**Chapter 8**

**References**

**References:**

1. E. A. Shammar, A. T. Zahary and A. A. Al-Shargabi, "A Survey of IoT and Blockchain Integration: Security Perspective," in IEEE Access, vol. 9, pp. 156114-156150, 2021, doi: 10.1109/ACCESS.2021.3129697.
2. B. Houtan, A. S. Hafid and D. Makrakis, "A Survey on Blockchain-Based Self-Sovereign Patient Identity in Healthcare," in IEEE Access, vol. 8, pp. 90478-90494, 2020, doi: 10.1109/ACCESS.2020.2994090.
3. S. K. Lo et al., "Analysis of Blockchain Solutions for IoT: A Systematic Literature Review," in IEEE Access, vol. 7, pp. 58822-58835, 2019, doi: 10.1109/ACCESS.2019.2914675.
4. M. S. Ali, M. Vecchio, M. Pincheira, K. Dolui, F. Antonelli and M. H. Rehmani, "Applications of Blockchains in the Internet of Things: A Comprehensive Survey," in IEEE Communications Surveys & Tutorials, vol. 21, no. 2, pp. 1676-1717, Secondquarter 2019, doi: 10.1109/COMST.2018.2886932.
5. K. Salah, M. H. U. Rehman, N. Nizamuddin and A. Al-Fuqaha, "Blockchain for AI: Review and Open Research Challenges," in IEEE Access, vol. 7, pp. 10127-10149, 2019, doi: 10.1109/ACCESS.2018.2890507.
6. T. M. Hewa, Y. Hu, M. Liyanage, S. S. Kanhare and M. Ylianttila, "Survey on Blockchain-Based Smart Contracts: Technical Aspects and Future Research," in IEEE Access, vol. 9, pp. 87643-87662, 2021, doi: 10.1109/ACCESS.2021.3068178.
7. R. Sekaran, R. Patan, A. Raveendran, F. Al-Turjman, M. Ramachandran and L. Mostarda, "Survival Study on Blockchain Based 6G-Enabled Mobile Edge Computation for IoT Automation," in IEEE Access, vol. 8, pp. 143453-143463, 2020, doi: 10.1109/ACCESS.2020.3013946.
8. S. Rouhani and R. Deters, "Security, Performance, and Applications of Smart Contracts: A Systematic Survey," in IEEE Access, vol. 7, pp. 50759-50779, 2019, doi: 10.1109/ACCESS.2019.2911031