



**Project Title: DECENTRALIZED CERTIFICATE
AUTHENTICATION SYSTEM**

Submitted By

MD JAFRUL HASAN

ID: 192-35-461

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

Supervised By

Mr. Nuruzzaman Faruqui

Assistant Professor

Department of Software Engineering

DAFFODIL INTERNATIONAL UNIVERSITY

This Project report has been submitted to fulfil the requirements for the
Degree of Bachelor of Science in Software Engineering FALL- 2023



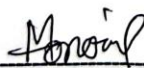

© All right Reserved by Daffodil International University

APPROVAL

APPROVAL

This project titled on “Decentralized Certificate Authentication System”, submitted by **Md Jafrul Hasan (ID: 192-35-461)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

BOARD OF EXAMINERS

 _____	Chairman
Dr. Imran Mahmud Associate Professor & Head Department of Software Engineering Faculty of Science and Information Technology Daffodil International University	
 _____	Internal Examiner 1
Nuruzzaman Faruqi Assistant Professor Department of Software Engineering Faculty of Science and Information Technology Daffodil International University	
 _____	Internal Examiner 2
Md. Monirul Islam Assistant Professor Department of Software Engineering Faculty of Science and Information Technology Daffodil International University	
 _____	External Examiner
Dr. Md. Sazzadur Rahman Associate Professor Institute of Information Technology, Jahangirnagar University	

DECLARATION

DECLARATION

I announce that I am rendering this study document under, Mr Nuruzzaman Faruqui, Assistant Professor, Department of Software Engineering, and Daffodil International University. I therefore, state that this work or any portion of it was not proposed here therefore for Bachelor's degree or any graduation.

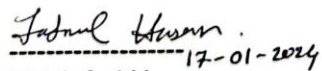
Supervised By



17-01-2024

Mr Nuruzzaman Faruqui
Assistant Professor
Department of Software Engineering
Daffodil International University

Submitted by



17-01-2024

Md Jafrul Hasan
ID: 192-35-461
Department of Software Engineering
Daffodil International University

ACKNOWLEDGEMENT

First and foremost, I wish to thank Mr. Nuruzzaman Faruqui, my noble mentor who has offered priceless direction and constant assistance in creating this work. His knowledge and experience have greatly influenced the course of this project whilst improving its overall standard.

Therefore, I extend my sincere gratitude to Daffodil International University for creating a conducive setting in which this project was completed. A conducive learning environment has also been enhanced by the academic environment and facilities of Daffodil International University.

I want to express, I would like to heartfelt thanks to my parents, students, and DIU members for their generous support. They were supporting me mentally to finish my work.

EXECUTIVE SUMMARY

“Decentralized Certificate Authentication System”, a new scheme for verifying the certificates based on blockchain technology. This project ensures higher security, reliability, and efficiency, as it addresses the limitations of traditional verification systems.

The system ensures the integrity of the certificates in a decentralized blockchain infrastructure. There are smart contracts for automated transactions and decentralization to avoid single points of failure, among others.

The project's objectives are clear: to protect the security of certificate data, improve reliability through decentralization, and ease the verification process for end users.

The system admits limitations such as scalability concerns and regulatory challenges as it focuses on producing, storing, and verifying certificates. The project is the foundation for future modifications, like scaling up and regulation.

“Decentralized Certificate Authentication System” promises to be a transformative solution, defining the security and usability paradigm of certificate authentication.

TABLE OF CONTENTS

APPROVAL	ii
ACKNOWLEDGEMENT	iv
EXECUTIVE SUMMARY	v
TABLE OF CONTENTS.....	vi
Chapter 1	1
INTRODUCTION	1
Project Overview	2
Background and Motivation	2
Objectives	3
Ensure Security	3
Decentralization	4
User Verification.....	4
Scope and limitations.....	5
Scope.....	5
Limitation.....	5
Stakeholders.....	6
Proposed System Model	8
Key Entities.....	8
Interaction Flow	9

Problem Statement	10
Chapter 2	13
Software Requirements & Specifications	13
2. Software Requirements Specification.....	14
2.1 Functional Requirements	14
2.1.1 Product Perspective.....	14
2.1.2 Product Features.....	14
2.1.3 User Classes and Features.....	14
2.1.4 Operating Environment.....	15
2.1.5 Design and Implementation Constraints.....	15
2.1.6 User Documentation	15
2.1.7 Assumptions and Dependencies	15
2.2 System properties.....	15
2.2.1 User Registration	15
2.2.2 Creation of Certificates	15
2.2.3 Certificate Verification	16
2.2.4 Authorization	16
2.2.5 Block chain Interaction	16
2.2.6 Event Logging.....	17
2.2.7 Smart Contracts.....	17
2.3 External Interface Requirements.....	17

2.3.1 User Interfaces	17
2.3.2 Hardware Interfaces	17
2.3.3 Software Interfaces	18
2.3.4 Communication Interfaces	18
2.4 Non-functional Requirements	18
2.4.1 Performance Requirements	18
2.4.2 Security Requirements	18
2.4.3 Reliability.....	18
2.4.4 Availability	18
2.4.5 Maintainability	18
2.4.6 Portability.....	19
2.5 Other Requirements	19
2.5.1 Legal and Regulatory Requirements.....	19
2.5.2 Ethical Considerations	19
2.5.3 Privacy Concerns	19
Chapter 3	20
DIAGRAM & REQUIREMENT ANALYSIS	20
3.1 Use Case Diagram.....	21
3.1.1 Use Case Diagram for USER, ISSUER & VERIFIER.....	21
3.1.2 Use Case: User Registration	22
3.1.3 Use Case: Certificate Creation.....	22

3.1.4 Use Case: Certificate Verification	23
3.1.5 Use Case: Block chain Interaction.....	23
3.2 Activity Diagram	24
3.3 Sequence Diagram	26
3.4 ER Diagram	27
3.5 Class Diagram.....	28
Chapter 4	31
System Design Specification	31
Development Tool & Technology	32
4.1 User Interface Technology	32
4.1.1 User Interface and User Experience Design Tool:	32
4.1.2 User Interface Development Tool:.....	32
Chapter 5	34
USER MANUAL.....	34
5.1 Authenticator & QR Scan	35
5.2 Upload.....	36
5.3 Verified or Not.....	36
Chapter 6	38
Smart Contract	38
6.1 Overview of the Smart Contract	39
6.1.1 Structs:	39

6.1.2 State Variables:	39
6.1.3 Events:.....	40
6.1.4 Modifiers:.....	40
6.1.5 Functions:.....	40
6.1.6 Function Modifiers:	40
6.1.7 Events:.....	40
6.2 Code of the Smart Contract.....	41
6.3 Receive MATIC & Deploy in Thirdweb	46
Chapter 7	50
Conclusion	50
7. Project Summary.....	51
7.1 Limitations	51
7.2 Obstacle & Achievements.....	51
7.3 Future Work	52
REFERENCE.....	53

Chapter 1

INTRODUCTION

Project Overview

A certificate authentication system using block-chain represents a state-of-the-art solution to enhance the security and trustworthiness of certificates.

This state-of-the-art innovative system capitalizes on the decentralized and immutable features of block-chain technology, establishing a robust and secure environment for certificate verification.

This chapter aims to provide a broad overview of the project, study its basic components and functions to clearly understand its significance and potential impact, its main aims and objectives.

Background and Motivation

The importance of safe and reliable testimonials in the fast-paced world of technology cannot be overstated. Traditional certificate authority systems have issues related to tampering and vulnerabilities arising from centralization.

This project is inspired by the desire to overcome these challenges. The employed blockchain technology can enable the creation of a stable and decentralized solution that is capable of minimizing threats while setting up new standards for certificate authentication in this digital age.

However, with the use of such technology, our certificates are centralized and decentralized so there will be more security. All of us know that a decentralized system cannot be hacked no matter what and one can therefore assume it to be safe.

Objectives

Ensure Security

Tamper proof Storage

To ensure the integrity of stored credentials, the system will use cryptographic methods. Each certificate entry will undergo cryptographic hashing and be linked to the previous entry, creating an immutable chain of blocks. This mechanism of tamper-proofing guarantees that any change to a certificate recorded in the block chain will require subsequent blocks to be altered, thereby protecting the integrity of the certificate.

Encryption Mechanisms

Encryption processes will secure sensitive information in certificates. This will protect against unauthorized access and preserve confidentiality. Secure communication between entities engaged in the issuance and verification of certificates may be enabled by making use of public-key cryptography.

Smart Contract Auditing

Smart contracts used in the certificate transactions will be audited. This refers to finding and fixing weaknesses present in the code that hackers, criminals, and other malicious actors can manipulate. In this case, auditing helps to verify if the smart contract operates safely and in line with accepted practices for block chain programming.

Decentralization

Elimination of Single Points of Failure

Traditional certification systems typically rely on centralized databases, making them vulnerable to single points of failure. In that case, the storage and verification methods will be divided into a network of nodes using the features of the block chain. This makes the system fault-tolerant and minimizes the risks that can come with central authority.

Consensus Mechanism

One aspect of decentralization is the consensus process used to come to an agreement and verify the state of the block chain. Factors such as security, efficiency, and environmental impact will be considered in choosing another consensus method.

User Verification

Simplified Verification Process

It will give users an easy and friendly way to check certificates. The users may use keys identifying the certificate, for instance, the unique ID of the certificate or the user's public key for querying the block chain. To ease this, the verification aspect will allow people, organizations, and institutions to verify the genuineness of a certificate readily.

Accessibility

Verification may be enabled through a website interface, mobile application, and/or any media used for that purpose by users in general. This is important because it makes it easier for people to authenticate credentials using common devices or platforms.

Scope and limitations

Scope

Development and Implementation

This project specifically focuses on building and deploying an end-to-end Certificate Authentication System based on the latest Block chain technology. This encompasses:

Certificate Creation: This is because it will help in the easy issuance of the certificates by the various issuers. This will enable issuers to enter their data like issuer details, certificate details, and user information.

Storage on Block chain: Smart contracts will store certificates safely on a block chain. Block chain is a decentralized secure mechanism that allows for the authenticity and unchangeability of certificate recordings.

Verification Process: The block chain enables users and third parties to validate certificate authenticity using queries. Such will involve communication with smart contracts for obtaining the required certificates' details.

Smart Contracts: Smart contracts will be implemented to automate and enforce rules on certificate generation, storage, and verifications. A block chain network will be utilized to execute these contracts.

Limitation

Scalability Concerns

- Challenge: As more and more transactions are processed in the block chain, especially, in Proof-of-Work chains, scaling becomes hard list.
- Mitigation: Best practice method will be adopted on how to manage issues related to scaling up. Some of these measures may involve looking for new ways of reaching a consensus and incorporating layer two scaling methods.

Regulatory Challenges

- Challenge: In different jurisdictions, there may be difficulties in adopting block chain technology into its use for such official documents as certificates.
- Mitigation: The process of the certificate issuance and verification will abide by existing certificate issuance regulation guidelines. A partnership could be established with the regulatory authorities to confirm that rules are followed.

Dependency on Block chain Infrastructure

- Challenge: As such, it largely depends upon the working environment of the underlying block chain support system.
- Mitigation: These robustness measures include choosing reputable block chain networks and integrating redundancy into the design. Secondly, I will examine various ways of minimizing possible consequences and consequences by itself.

Stakeholders



USERS	ISSUERS	DEVELOPERS	AUTHORITIES
--------------	----------------	-------------------	--------------------

It involves several layers of certificate validators who all play different roles in ensuring that the system works without any hitches. Here is an in-depth exploration of the key stakeholders:

- **Users:**
 - ✓ Definition: These include persons or organizations seeking certificate verification for varied functions, including among others academic credentials and professional qualifications.
 - ✓ Role: This system allows users to check on certificates being valid by depending on their decentralization and unforgeable properties to build greater trust and credibility towards the system.
- **Issuers:**
 - ✓ Definition: Those entities trusted with the task of generating, sealing, and verifying certificates.
 - ✓ Role: During the first stage issuers produce digital signatures that are then included in their certificates. They also verify the reliability of data stored on the block chain. They play a crucial part in confirming the genuineness of certificates.

- **Developers:**
 - ✓ Definition: Proficient personnel who are constantly improving and developing a certificate authentication system.
 - ✓ Role: Through bug fixing, updating, and addition of other features, developers enable system compatibility and usability. These are their works that determine how efficient, secure, and flexible a system can be.

- **Regulatory Authorities:**
 - ✓ Definition: Bodies or regulatory agencies charged with monitoring the compliance, legality, and ethics standards of the Certificate Authentication System by government organizations.
 - ✓ Role: However, regulatory authorities are important in ensuring that the system sticks to regulations about honesty and ethics that enhance user's confidence in the system. This makes their involvement enhance the system's credibility towards legal frameworks.

Proposed System Model

The following application is a block chain based voting system. Although Block chain technology is quite new but there are already a lot of decentralized voting projects are there in the industry. But in my system there are some unique offerings.

Key Entities

Users

Description: The users interact with the system to verify certificates or perform other relevant activities.

Issuers

Description: Certificate creator and issuing entity. Their responsibility is to enter certificate data and personal information into the system.

Smart Contracts

Description: They are encoded self-executing smart contracts with automatic certificate generation, storage, and verification rules. These contracts reside on the block chain.

Block chain

Description: A decentralized and distributed ledger wherein certificates are kept safely. Block chain makes the digital certificates' record immutable as it is transparent and can not be changed.

Certificate

Description: Description: Electronic copies of certificates issued by issuers. Users verify certifications that were encoded and are kept in the block chains.

Interaction Flow

1. Certificate Creation:

These certificates are generated by entering pertinent information into the issuer's application. Smart contracts serve as the means for validating and storing certificates on the block chain.

2. Certificate Verification:

Upon request, users can query the block chain using specific details. It also facilitates the verification process via smart contracts.

3. Smart Contract Execution:

smart contracts offer a mechanism for automating predetermined rules of transactions relating to issued certificates.

4. Block chain Security:

Data integrity, transparency, and decentralization are ensured via block chain.

Problem Statement



Figure 1.1

Data Tempering

Issue: However, centralized databases are vulnerable, and such vulnerabilities compromise the integrity of certificate data.

Centralized Vulnerabilities

Issue: Centralized systems are at risk of compromise due to single points of failure and their vulnerability to unauthorized access.

In-efficient Verification:

Issue: Some of the conventional methods of verification are complex and Exhausting.

- Traditional certificate authentication systems have **security and trust issues** due to centralized control.

- Certificates can be **easily tampered** with or forged, leading to fraudulent activities.
- Verification of certificates is a **time-consuming process**, and requires a centralized authority to perform the verification.

Proposed Solution

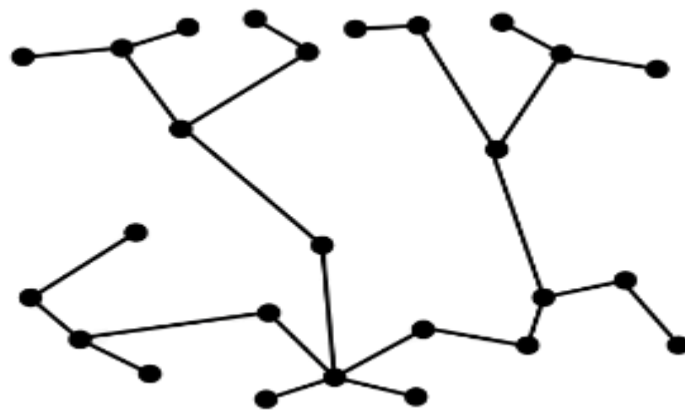


Figure 1.2

Decentralization

Solution: Using block chain technology to distribute certificate storage, eliminate single point of failure, and improve security.

Data Integrity

Solution: Block chain's immutability makes it impossible for anyone to change certificate information during its lifetime.

Simplified Verification

Solution: Automating the entire process through smart contracts based on smart contracts that will do automatic and efficient certificate verification.

- Implement a certificate authentication system using block chain technology to **provide a decentralized, secure** and transparent method of certificate verification.
- Certificates are **stored on the block chain**, which is an immutable and distributed ledger that eliminates the risk of tampering or forgery.
- The verification process is decentralized, and **any user can verify a certificate by accessing the block chain** network without relying on a central authority.
- **Smart contracts** are used to automate the verification process, reducing the time and cost of verification.
- The system increases transparency and trust by providing a transparent, secure and decentralized network for certificate verification.

Chapter 2

Software Requirements & Specifications

2. Software Requirements Specification

Software Requirements Specification (SRS) is a document that outlines the functional and non-functional requirements of a software system. It serves as a formal and structured representation of what the software should do and how it should behave. The SRS document acts as a contract between the stakeholders (clients, developers, testers, etc.) to ensure a shared understanding of the system's requirements.

The purpose of an SRS document is to provide a clear and unambiguous description of the software system's behavior, features, and constraints. It serves as a reference for the development team throughout the software development lifecycle and helps in effective project planning, design, implementation, testing, and maintenance.

2.1 Functional Requirements

Overall Description

2.1.1 Product Perspective

It is a DAPP that runs on a block chain system. It involves communication between users, issuers, and smart contracts for certificate integrity and security.

2.1.2 Product Features

These key features are user registration, certificate generation, verification, authorization, interaction with block chain, event logging, and smart contracts.

2.1.3 User Classes and Features

Users: Subjects who interact with the system to obtain verified credentials.

2.1.4 Operating Environment

It uses a peer-to-peer (P2P) architecture running on a block chain network, which guarantees its decentralized and transparent character.

2.1.5 Design and Implementation Constraints

Block chain Reliance: It is a block chain-based system.

Scalability: Many transactions can raise concerns about scalability.

2.1.6 User Documentation

User documentation will come in the form of guides/tutorials.

2.1.7 Assumptions and Dependencies

Assumption: They usually know something about working with block chain protocols.

Dependability: Requires a reliable, robust, and advanced block chain network.

2.2 System properties

2.2.1 User Registration

2.2.1.1 Description

It requires users to register, and then provide details.

2.2.1.2 Dependencies

Existence of an operational block chain network.

2.2.2 Creation of Certificates

2.2.2.1 Description

The issuer generates a certificate using the correct information.

2.2.2.2 Dependencies

Availability of authorized issuers.

Smart contract for generating certificates.

2.2.3 Certificate Verification

2.2.3.1 Description

The system allows users to check the authenticity of certificates.

2.2.3.2 Dependencies

Details are provided for the availability of certificates on the block chain.

2.2.4 Authorization

2.2.4.1 Description

Authentication and authorization processes to manage user access.

2.2.4.2 Dependencies

Valid user credentials.

2.2.5 Block chain Interaction

2.2.5.1 Description

Secure, transparent transactions by interacting with the block chain.

2.2.5.2 Dependencies

Availability of block chain infrastructure.

2.2.6 Event Logging

2.2.6.1 Description

Recording events that are auditable and traceable.

2.2.6.2 Dependencies

Smart contracts for event triggers.

2.2.7 Smart Contracts

2.2.7.1 Description

Self-executing contracts based on predetermined rules.

2.2.7.2 Dependencies

Ensuring that smart contracts are effectively deployed and working properly.

2.3 External Interface Requirements

2.3.1 User Interfaces

2.3.1.1 Description

Wired communication interface that is easy to use in order registration and certificate production/verification.

2.3.1.2 Dependencies

Compatibility with standard web browsers.

2.3.2 Hardware Interfaces

2.3.2.1 Description

No specific hardware interfaces.

2.3.3 Software Interfaces

2.3.3.1 Description

Interfacing with the block chain network.

2.3.3.2 Dependencies

Compatible block chain client.

2.3.4 Communication Interfaces

2.3.4.1 Description

Authenticated communication channels to enable interactions between users.

2.3.4.2 Dependencies

Standard internet connectivity.

2.4 Non-functional Requirements

2.4.1 Performance Requirements

Response Time: Responding as quickly as possible to user actions is important.

Scalability: This system should be able to manage transaction traffic in varying intensities ranging from moderate to high volumes.

2.4.2 Security Requirements

Data Encryption: User's data and certificates need to be ciphered.

Access Control: Controls to prevent unauthorized activities by ensuring strict access constraints.

2.4.3 Reliability

System Uptime: There must be a high availability in a system.

2.4.4 Availability

Continuous Availability: It should give round-the-clock service.

2.4.5 Maintainability

System Updates: Upgrades should be easily effected on the system.

2.4.6 Portability

Cross-Browser Compatibility: Major browsers' interfaces must be supported by the user.

2.5 Other Requirements

2.5.1 Legal and Regulatory Requirements

Compliance: It is essential that the system respects data protection laws and protects the privacy of individuals.

2.5.2 Ethical Considerations

User Privacy: User data privacy must come first.

2.5.3 Privacy Concerns

Data Confidentiality: Protect the personal data of users.

Chapter 3

DIAGRAM & REQUIREMENT ANALYSIS

3. In this chapter we will describe about all diagrams which I used for this project such as Use case Diagram, Activity Diagram, Sequence Diagram, ER Diagram, Class Diagram.

3.1 Use Case Diagram

3.1.1 Use Case Diagram for USER, ISSUER & VERIFIER

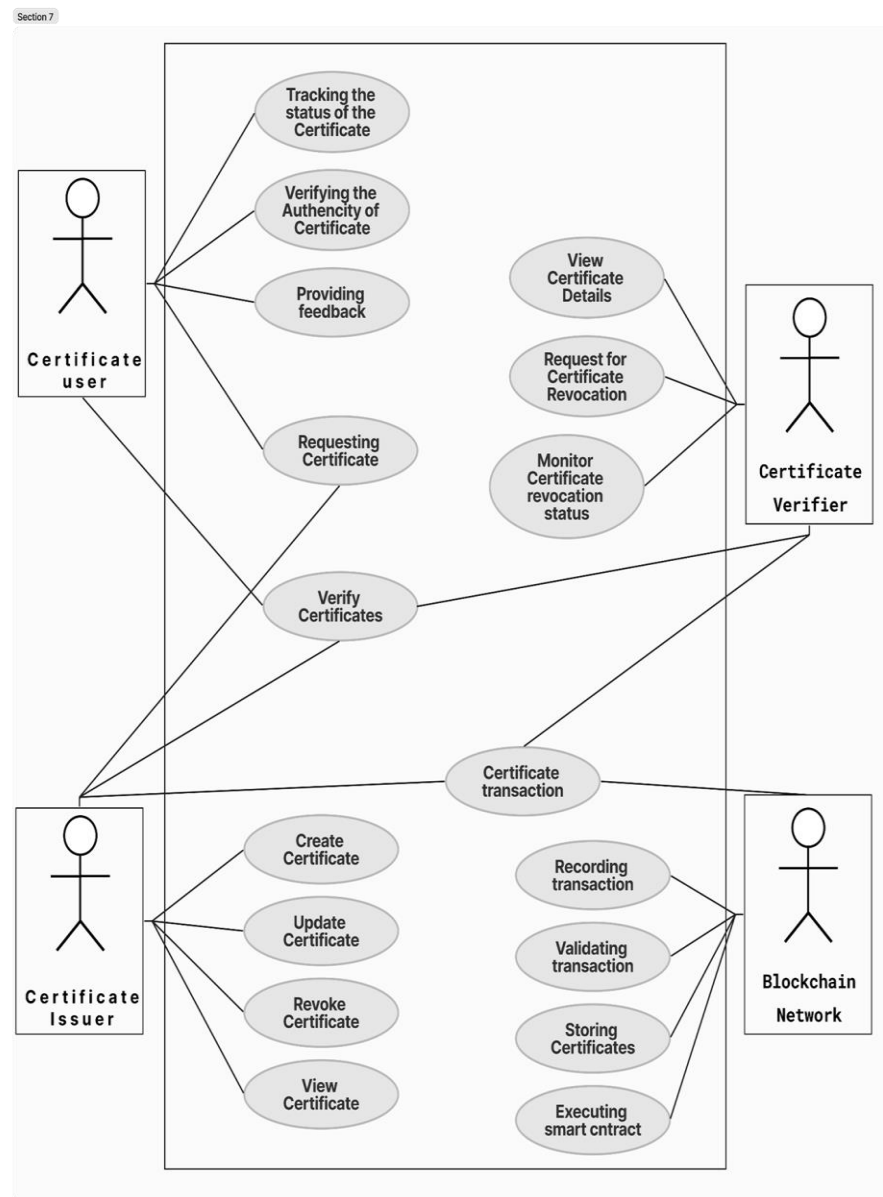


Figure: 3.1 – Use Case Diagram

3.1.2 Use Case: User Registration

Actors:

- User
- System

Description:

- i. It is the end-user who starts the registration procedure.
- ii. It will request from the user their first and last names, their public and private keys, e-mail addresses, and status updates.
- iii. The user keys in necessary data.
- iv. It verifies the data and gives a distinct login ID.
- v. This is added to the system to register the user in the list of registered users.
- vi. A confirmation message is sent to the user of a successful registration on the system.

3.1.3 Use Case: Certificate Creation

Actors:

- Issuer
- User
- System

Description:

- i. The issuer logs on to the system.
- ii. An individual is selected by the issuer, for which he wants to issue a certificate.
- iii. The system prompts the issuer to enter certificate details: Issuer, Block chain address, Issuer Date, Expiry date, Description, and title.
- iv. This means that the issuer does supply the required details.
- v. The system verifies the provided details by generating a specific serial number.
- vi. This makes it possible for the system to add a certificate to the user's record.

- vii. Hence, a generated certificate indicates that this is one successful event and sends out a confirmation note from the issuer.

3.1.4 Use Case: Certificate Verification

Actors:

- Verifier
- User
- System

Description:

- i. The verifier gets logged in to the site.
- ii. Verifier chooses a particular user whose certificate requires verification.”
- iii. They show the user’s certificates in the system.
- iv. A particular certificate is chosen by the verifier.
- v. The system then retrieves a certificate’s details.
- vi. This is how a verifier compares the certificate details with the physical certificate or other trustworthy sources.
- vii. The comparison determines if the certificate is verified or not verified by the verifier and thus marked accordingly.
- viii. Verification is logged from the system

3.1.5 Use Case: Block chain Interaction

Actors:

- Smart Contract
- Block chain Network
- System

Description:

- i. An interaction with the block chain is initiated in this transaction.
- ii. Smart contracts mediate the transaction.
- iii. The block chain network validates it and puts it in one block.

- iv. The block chain's response is updated on the system and updated with the relevant records.

3.2 Activity Diagram

- A user registration process initiates the event diagram.
- First, a prompt instructs the user on information that he or she must enter like e-mail, first name, last name, public key, private, key, etc.
- The system ensures that all necessary input has been captured correctly.
- The system assigns a user identity after successful verification.
- On adding the specified ID's user into the database of registered users, the system operates accordingly.
- Once the user registers successfully, the system informs him about that fact.
- The activity diagram ends by showing that the user has finished registering.

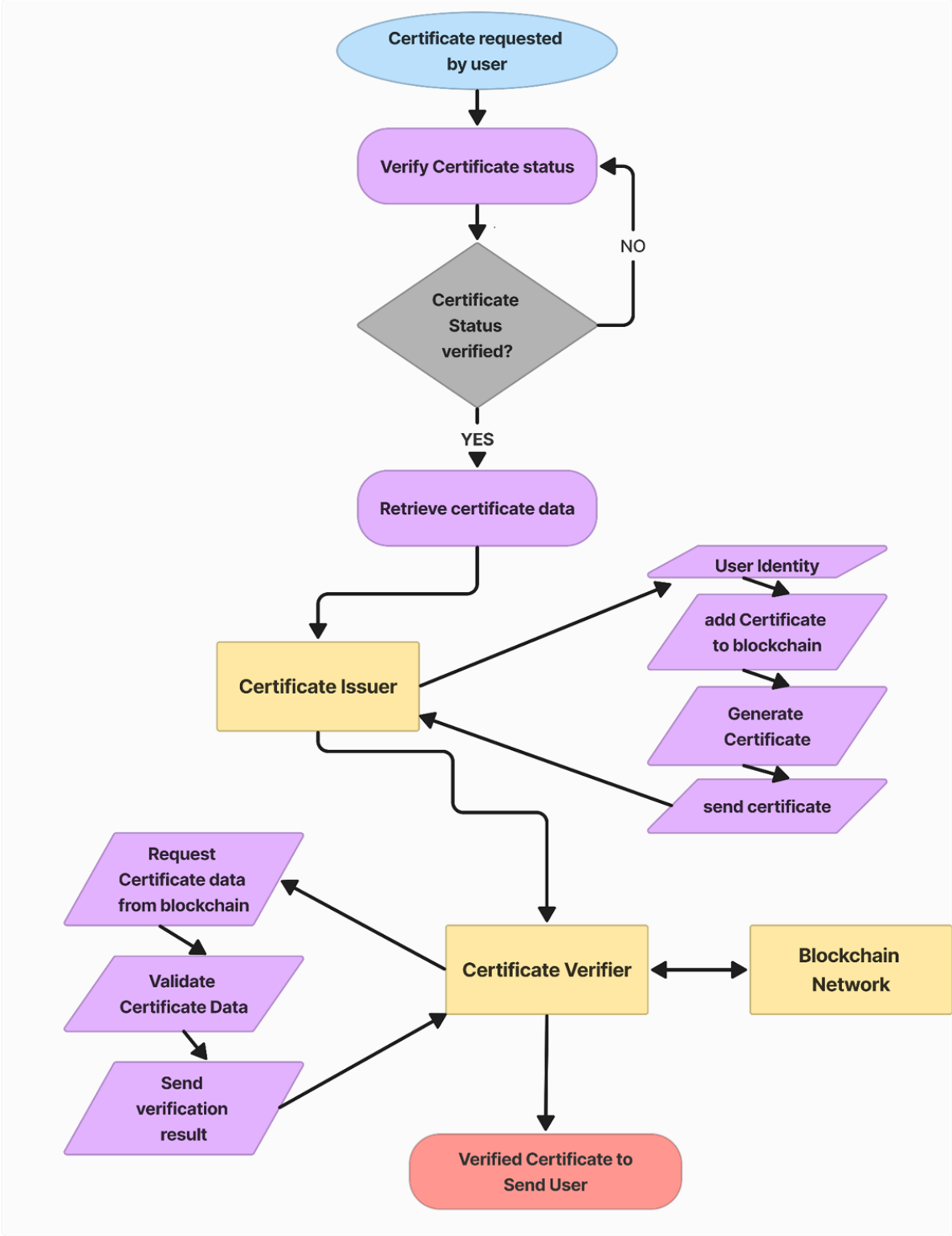


Figure: 3.2 – Activity Diagram

3.3 Sequence Diagram

On the user side of the sequence diagram, there is a request for a certificate from a certificate authority. Then, the certificate authority creates a transaction on the blockchain and sends the hash of the transaction to the user. The certificate authority forwards the certificate to the user. After that, the user sends a query to the blockchain network, which returns the corresponding checking output.

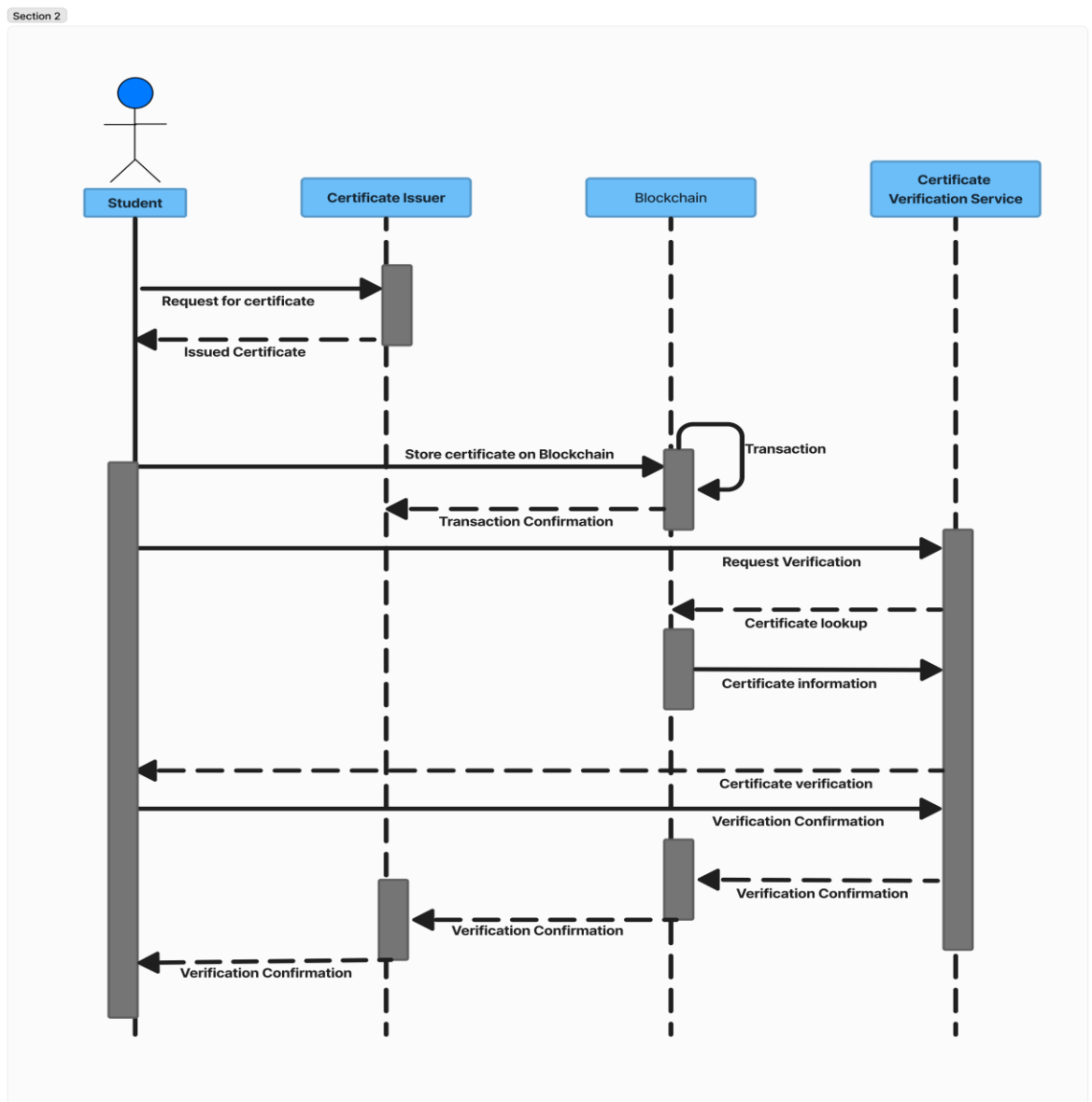


Figure: 3.3 – Sequence Diagram

3.4 ER Diagram

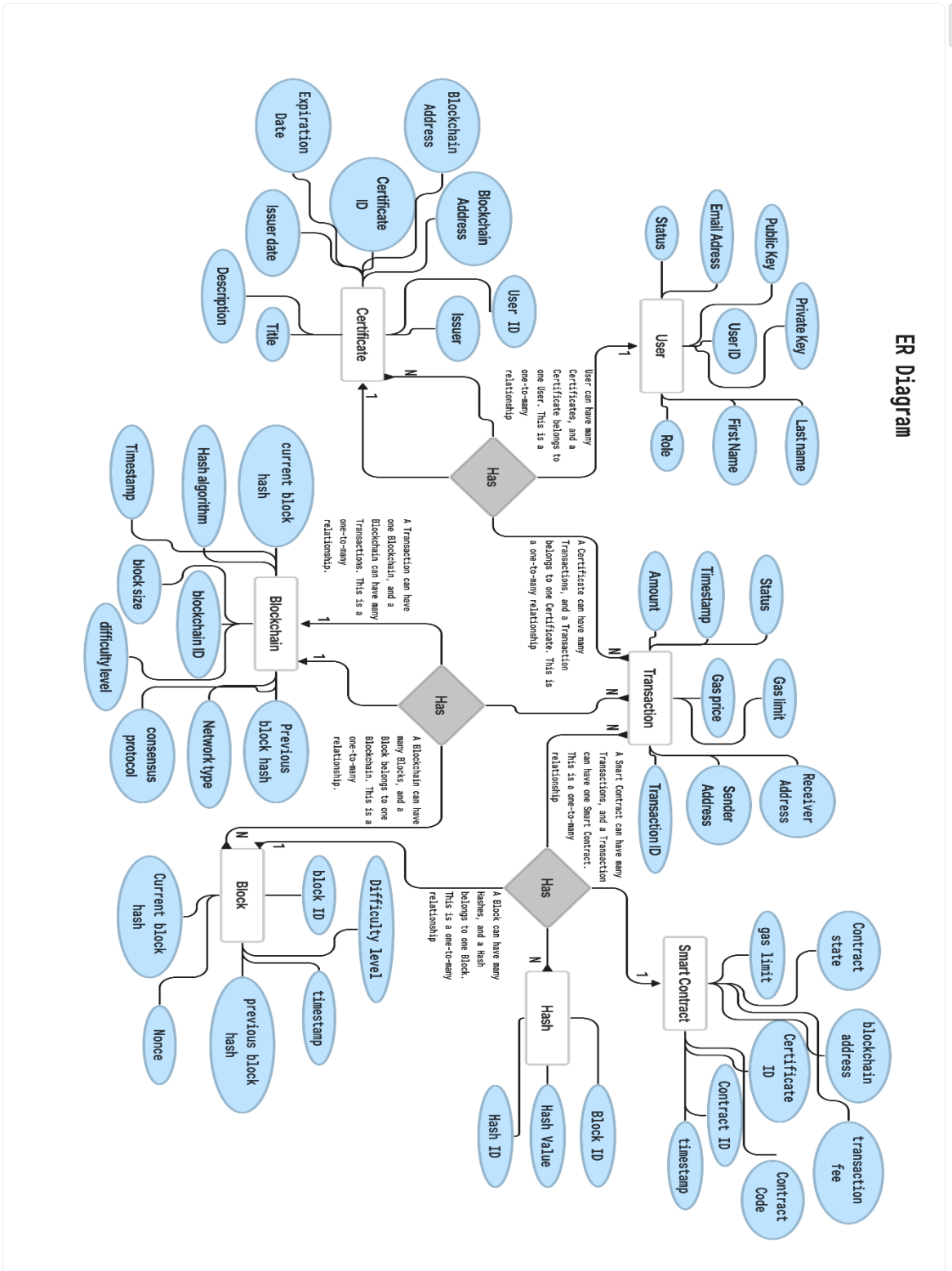


Figure 3.4: ER Diagram

- One User may have several Certificates while each Certificate belongs to just one User. This kind of relationship is a one-to-many relationship.
- One Certificate may contain multiple Transactions, and each Transaction is related only to one Certificate. Such relations are known as one-to-many.
- A transaction is capable of having only a blockchain, whereas a blockchain may comprise several transactions. One-on-many relationship.
- A particular Blockchain consists of several Blocks and they all belong to that one Blockchain. This is a case of one to many relationships.⁹
- One Block may involve several Hashes while a single Hash belongs in one block. A one-to-many relationship.
- There are multiple Transactions in a Smart Contract while there is only one Smart Contract in a Transaction. That's a many to one relation.

3.5 Class Diagram

- The certificate class has a many-to-one relationship with the User class, means that many certificates can be issued by the same user.
- The certificate class has a many-to-one relationship with the Transaction class, means that many certificates can be associated with the same transaction.
- Transaction class has a many-to-one relationship with Block class, means that many transactions can be included in the same block.
- Block class has a one-to-many relationship with Transaction class, means that one block can contain many transactions.
- Block class has a one-to-one relationship with Hash class, means that each block has a unique hash value.

- The smart Contract class has a many-to-one relationship with the Certificate class, means that many certificates can be associated with the same smart contract.
- The smart Contract class has a many-to-one relationship with the Blockchain class, means that many smart contracts can exist in the same blockchain.
- Blockchain class has a one-to-many relationship with Block class, means that one blockchain can contain many blocks.

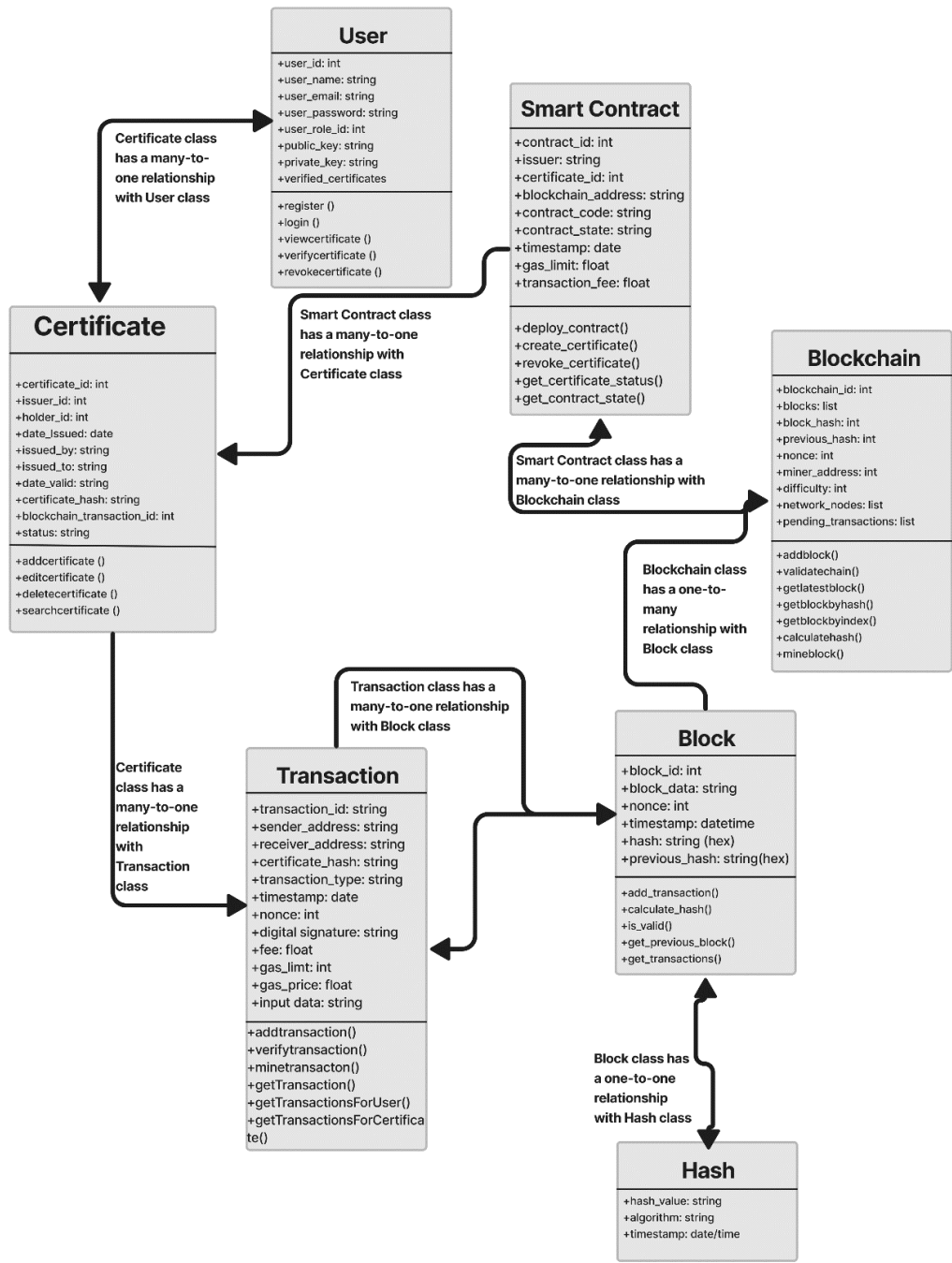


Figure 3.5: Class Diagram

Chapter 4

System Design Specification

Development Tool & Technology

4.1 User Interface Technology

4.1.1 User Interface and User Experience Design Tool:

For UI/UX design I have used Figma Pro to develop and research the proper interface and

user experience for the application.

Figma is a collaborative web-based design tool that allows teams to design, prototype, and

collaborate on projects in real time| It is a vector graphics editor that offers a variety of features for UI and UX design, including:

1. Real-time collaboration
2. Prototyping tools
3. Design systems
4. FigJam (a collaborative whiteboarding tool)
5. Code generation
6. A free plan for individual users

4.1.2 User Interface Development Tool:

For User Interface development I have used React and React native to develop web and mobile application for this application.

React: React is a JavaScript library for building user interfaces. It is used to create interactive web applications that can be rendered on the client-side or the server-side|

React is based on a component-based architecture, which makes it easy to create reusable and maintainable code.

Here are some of the benefits of using React:

- **Performance:** React is one of the most performant library where you can build with React is a very performant library that can be used to create. fast and responsive user interfaces.
- **Reusability:** React components are very easily reusable, saving a considerable amount of time and energy when. building large applications.
- **Scalability:** React can easily scale for big applications with high traffic.
- **Community:** Therefore, it should be noted that react has a big and busy community so there is a lot documentation, and support resources for developers of libraries to aid in learning and using the library.

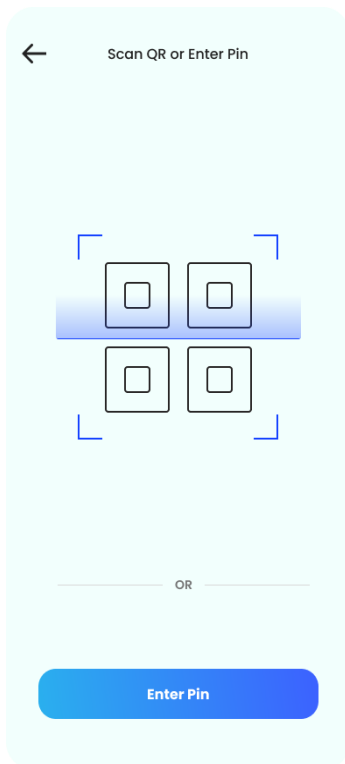
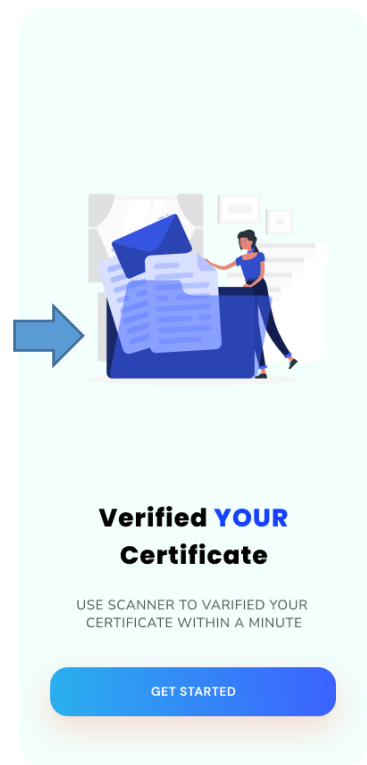
Visual Studio Code (VSCode):

- **Description:** The most extensible code-source editor known as vscode supports several coding languages. The software provides a good environment for running web applications.
- **Use Case:** A general-purpose code editor for use in front-end programming.

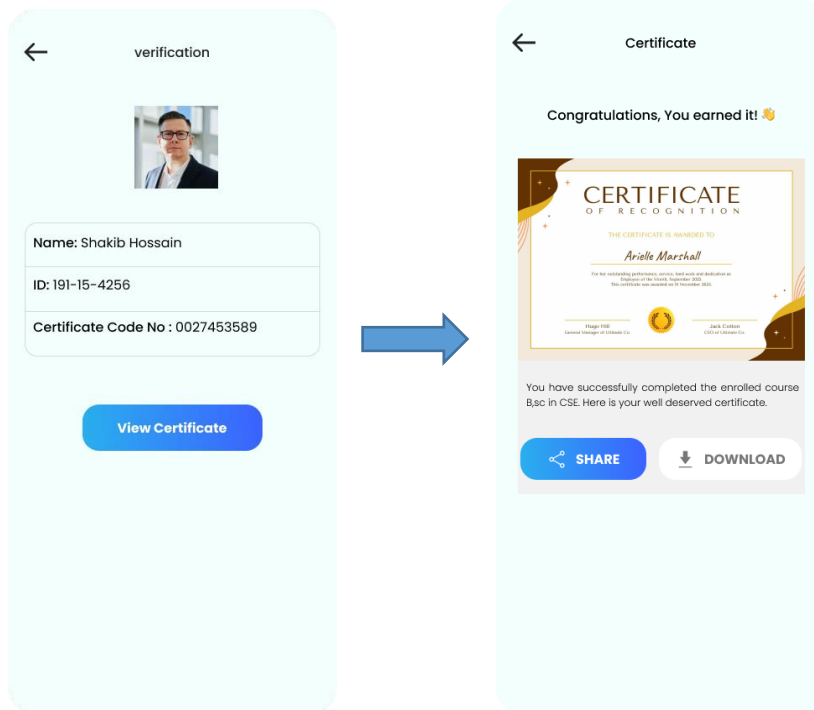
Chapter 5

USER MANUAL

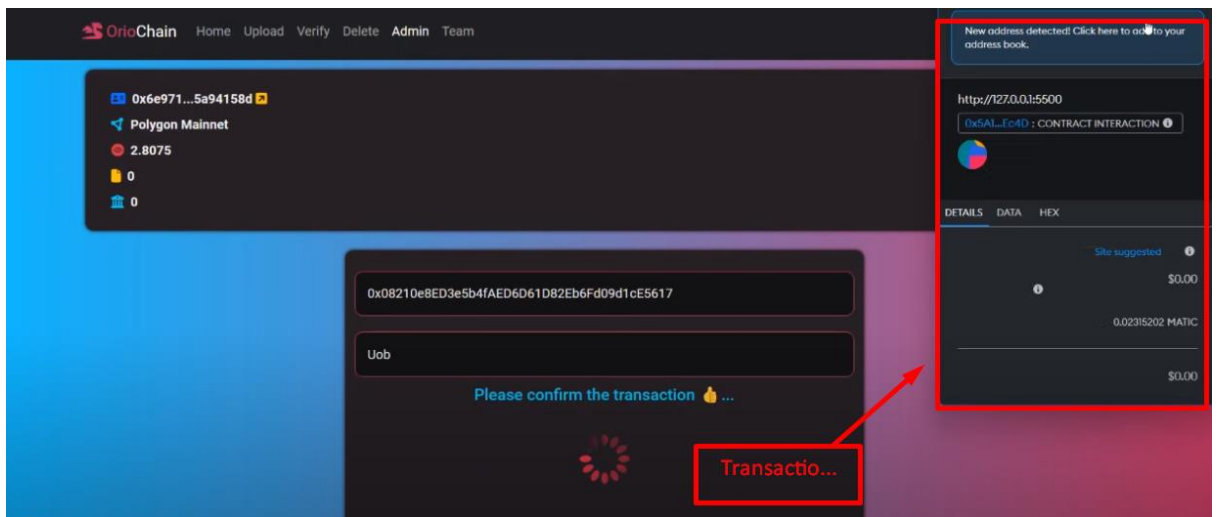
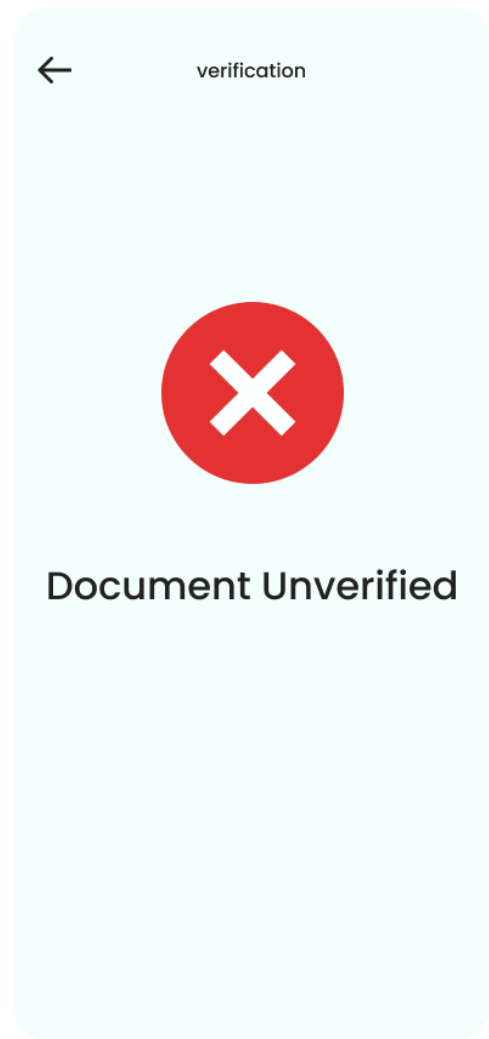
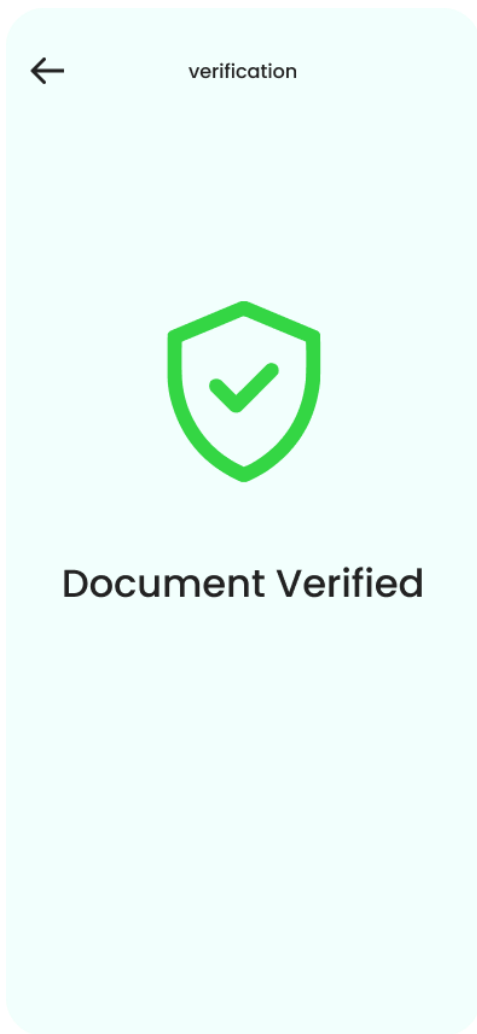
5.1 Authenticator & QR Scan



5.2 Upload



5.3 Verified or Not



Verified with Metamask Transaction

Chapter 6

Smart Contract

6.1 Overview of the Smart Contract

Smart contracts are autonomous contracts that consist of terms written in code. They are built on blockchain technology and perform operations after conditions are met. The use of smart contracts provides several benefits:

1. Clarity
2. Safety
3. Efficiency
4. Cost savings
5. Faithless execution.

6.1.1 Structs:

User: It holds data on a particular user which includes but is not limited to ID number, name, public key, private key, email address, and status.

Certificate: Gives the information regarding a certificate including its ID, address of the issuer, user ID, block chain address, issue date, description, and title.

Transaction: It is composed of data regarding a sale that includes a unique code, value, result, gas limit, gas price, recipients' address, and senders' address.

6.1.2 State Variables:

users: A data structure for storing user data.

certificates: Certificate data storage system which consists of an array.

transactions: A buffer to hold transaction data.

authorizedUsers: An authorization map to indicate who is allowed.

6.1.3 Events:

UserAdded: Fire upon adding a new user.

CertificateAdded: When you add another certificate, fire.

TransactionAdded: Firing on adding a new transaction.

6.1.4 Modifiers:

onlyAuthorizedUser: An add-on to authenticate some operations for just a few users.

Constructor:

Sets up the contract with the deployer's address as an authenticated person.

6.1.5 Functions:

addUser: Creates an additional account for a user on the system.

addCertificate: Certificate additions to the system.

addTransaction: A new transaction is added to the system.

authorizeUser: Authorizes a user.

revokeAuthorization: Revokes a user's authorization.

getAllCertificates: Retrieves all certificates.

getAllTransactions: Retrieves all transactions.

getAllUsers: Retrieves all users.

6.1.6 Function Modifiers:

onlyAuthorizedUser: Verifies that only particular functions are accessible to some selected users.

6.1.7 Events:

UserAdded: Generated when a fresh user is created.

CertificateAdded: This happens when a new certificate is emitted.

TransactionAdded: Generated when a new transaction is appended.

A standard basis for a decentralized certificate verifying scheme using smart contracts.

The Ethereum blockchain is capable of adding users, issuing certificates, recording transactions, and managing authorization. Events are used for transparency and it is possible to control through Ethereum transactions.

6.2 Code of the Smart Contract

```
// SPDX-License-Identifier: GPL-3.0
```

```
pragma solidity ^0.8.0;
```

```
// the version of the Solidity
```

```
contract CertificateAuthentication {
```

```
    // Definition
```

```
    struct User {
```

```
        // representing a user.
```

```
        uint256 userId; // Unique identifier for user.
```

```
        string firstName; // First name of the user.
```

```
        string lastName; // Last name of the user.
```

```
        string publicKey; // Public key of the user.
```

```
        string privateKey; // Private key of the user.
```

```
        string emailAddress; // Email address of the user.
```

```
        bool status; // User is active or not.
```

```
    }
```

```
    struct Certificate {
```

```

// representing a certificate.
uint256 certificateId; // Unique identifier for the certificate.
address issuer; // Address of the entity issuing the certificate.
uint256 userId; // ID of the user to whom the certificate belongs.
address blockchainAddress; // Address on the blockchain where the certificate is
stored.

string issuerDate; // Date when the certificate was issued.
string expirationDate; // Date when the certificate expires.
string description; // Description of the certificate.
string title; // Title or name associated with the certificate.
}

struct Transaction {
// Structure representing a transaction.
uint256 transactionId; // Unique identifier for the transaction.
uint256 amount; // Amount involved in the transaction.
string status; // Status of the transaction.
uint256 gasLimit; // Gas limit for the transaction.
uint256 gasPrice; // Gas price for the transaction.
address receiverAddress; // Address of the transaction receiver.
address senderAddress; // Address of the transaction sender.
}

User[] public users; // Array to store user data.
Certificate[] public certificates; // Array to store certificate data.
Transaction[] public transactions; // Array to store transaction data.
mapping(address => bool) public authorizedUsers; // Mapping to track authorized
users.

event UserAdded(uint256 indexed userId, string firstName, string lastName);
// Event emitted when a new user is added.
event CertificateAdded(uint256 indexed certificateId, address issuer, string title);
// Event emitted when a new certificate is added.

```

```
event TransactionAdded(uint256 indexed transactionId, address senderAddress,
address receiverAddress);
```

```
// Event emitted when a new transaction is added.
```

```
modifier onlyAuthorizedUser() {
```

```
    // Modifier to restrict certain functions to authorized users.
```

```
    require(authorizedUsers[msg.sender], "Not authorized");
```

```
    _;
```

```
}
```

```
constructor() {
```

```
    // Constructor to initialize the contract.
```

```
    authorizedUsers[msg.sender] = true;
```

```
    // The deployer's address is set as an authorized user during contract deployment.
```

```
}
```

```
function addUser(
```

```
    string memory _firstName,
```

```
    string memory _lastName,
```

```
    string memory _publicKey,
```

```
    string memory _privateKey,
```

```
    string memory _emailAddress,
```

```
    bool _status
```

```
) public onlyAuthorizedUser {
```

```
    // Function to add a new user to the system.
```

```
    uint256 userId = users.length; // Assign a unique ID to the new user.
```

```
    User memory newUser = User(
```

```
        userId,
```

```
        _firstName,
```

```
        _lastName,
```

```
        _publicKey,
```

```
        _privateKey,
```

```
        _emailAddress,
```

```
        _status
```

```

);
users.push(newUser); // Add the new user to the array.
emit UserAdded(userId, _firstName, _lastName); // Emit an event for the added
user.
}

```

```

function addCertificate(
    uint256 _certificateId,
    address _issuer,
    uint256 _userId,
    address _blockchainAddress,
    string memory _issuerDate,
    string memory _expirationDate,
    string memory _description,
    string memory _title
) public onlyAuthorizedUser {
    // Function to add a new certificate to the system.
    Certificate memory newCertificate = Certificate(
        _certificateId,
        _issuer,
        _userId,
        _blockchainAddress,
        _issuerDate,
        _expirationDate,
        _description,
        _title
    );
    certificates.push(newCertificate); // Add the new certificate to the array.
    emit CertificateAdded(_certificateId, _issuer, _title); // Emit an event for the added
certificate.
}

```

```

function addTransaction(
    uint256 _transactionId,

```

```

uint256 _amount,
string memory _status,
uint256 _gasLimit,
uint256 _gasPrice,
address _receiverAddress,
address _senderAddress
) public onlyAuthorizedUser {
    // Function to add a new transaction to the system.
    Transaction memory newTransaction = Transaction(
        _transactionId,
        _amount,
        _status,
        _gasLimit,
        _gasPrice,
        _receiverAddress,
        _senderAddress
    );
    transactions.push(newTransaction); // Add the new transaction to the array.
    emit TransactionAdded(_transactionId, _senderAddress, _receiverAddress);
    // Emit an event for the added transaction.
}

function authorizeUser(address _user) public onlyAuthorizedUser {
    // Function to authorize a user.
    authorizedUsers

```

6.3 Receive MATIC & Deploy in Thirdweb

[This is a POLY Testnet transaction only]

Transaction Hash:	0xb2ea72b0cd75a4fb2ab3d37eea5f2696989b2ec261b9ec314454f38864a187da
Status:	Success
Block:	42031290 10 Block Confirmations
Timestamp:	30 secs ago (Nov-05-2023 07:58:35 AM +UTC)
From:	0x5Ff40197C83C3A2705ba912333Cf1a37BA249eB7
To:	0x0CcD5089b9fF9164Edb22c4FEF78EA2791ad6562
Value:	0.2 MATIC (\$0.00)
Transaction Fee:	0.00006605875227 MATIC (\$0.00)
Gas Price:	3.14565487 Gwei (0.00000000314565487 MATIC)

0x0Cc...6562

1.0938 MATIC
\$0.67 USD

Buy Send Swap Bridge Portfolio

Tokens NFTs Activity

Oct 24, 2023

Receive Confirmed 0.2 MATIC \$0.12 USD

Oct 13, 2023

Receive Confirmed 0.5 MATIC \$0.31 USD

I got 0.2 MATIC from faucet and receive in metamask wallet for blockchain transaction.

Figure: 6.1 – MATIC receive

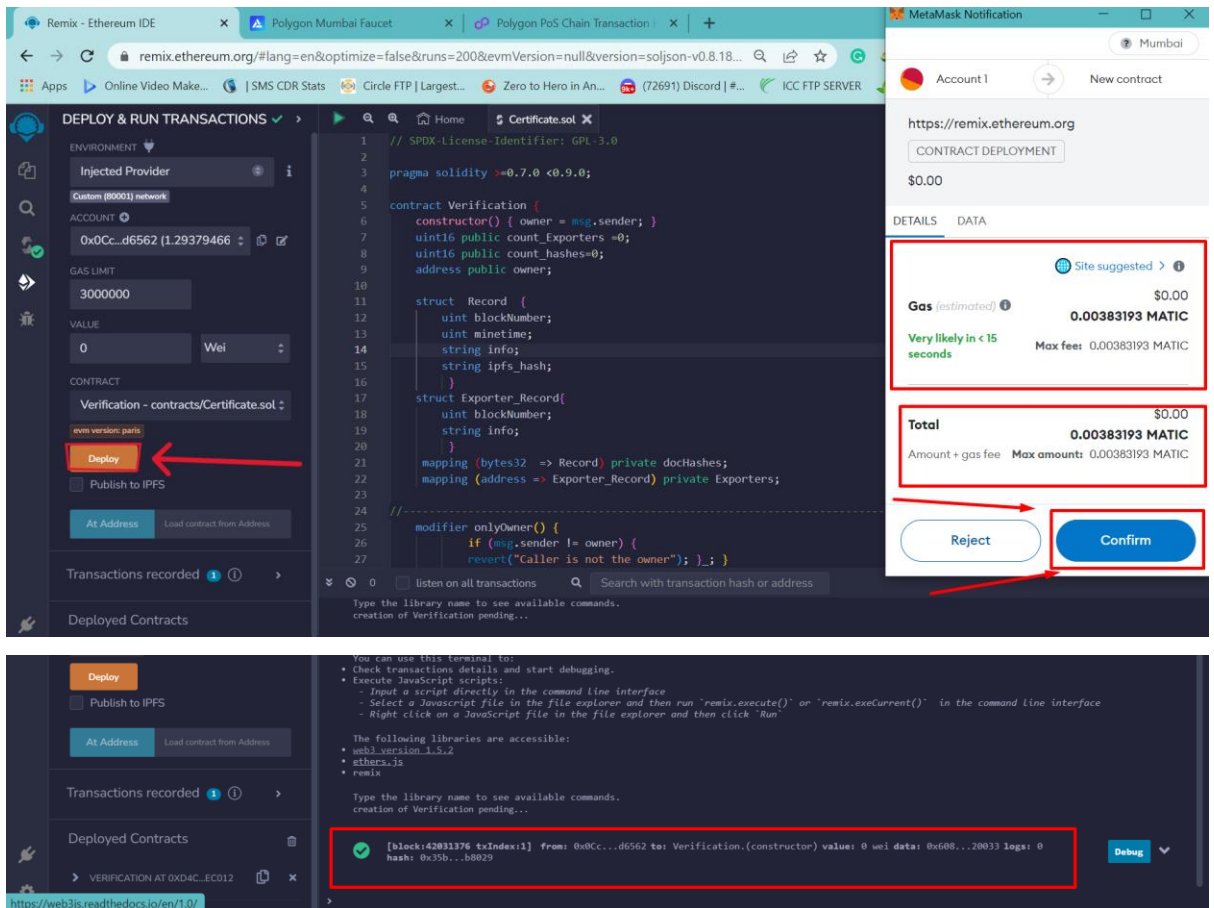
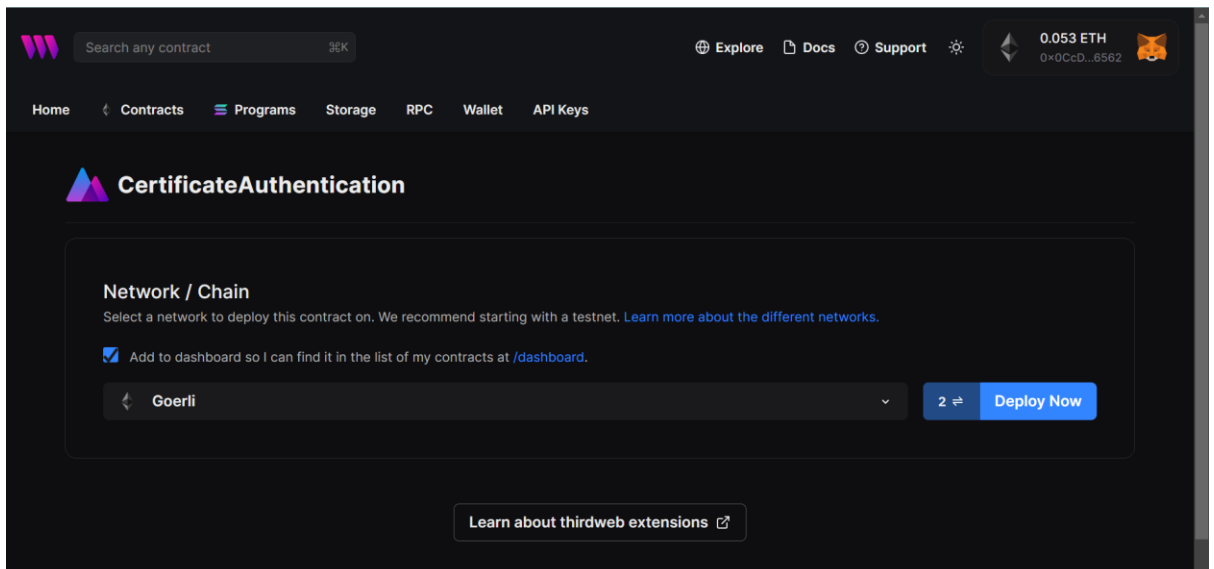


Figure: 6.2 – Deploy successful & Transaction



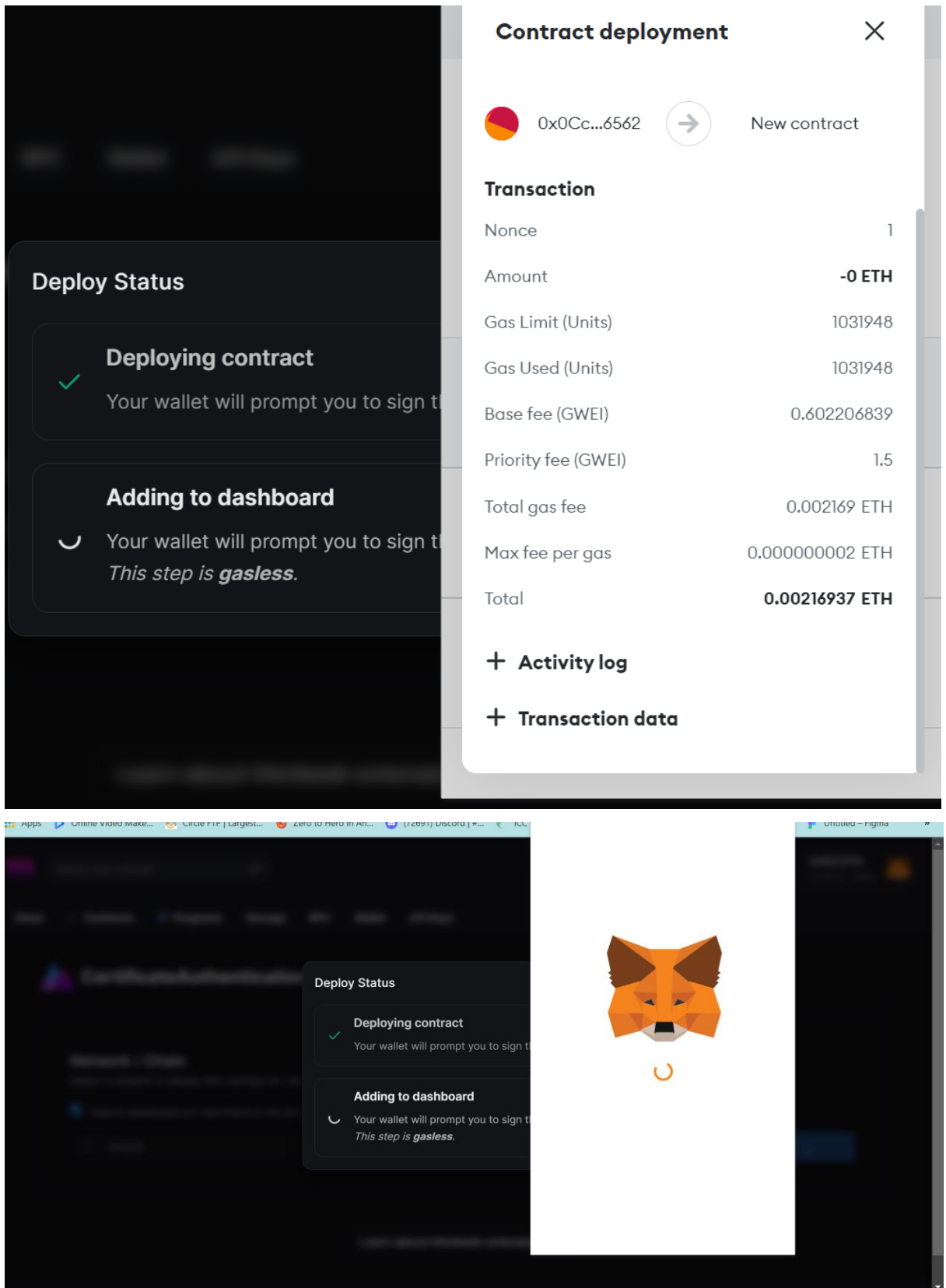


Figure: 6.3 – Confirm metamask & deploy smart contract to thirdweb page

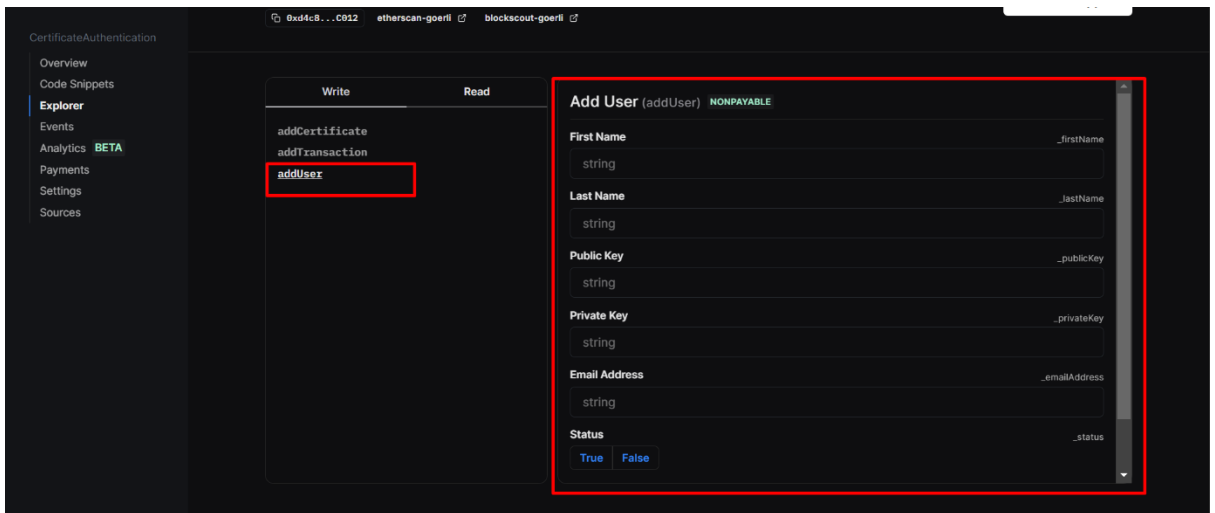


Figure: 6.4 – add User

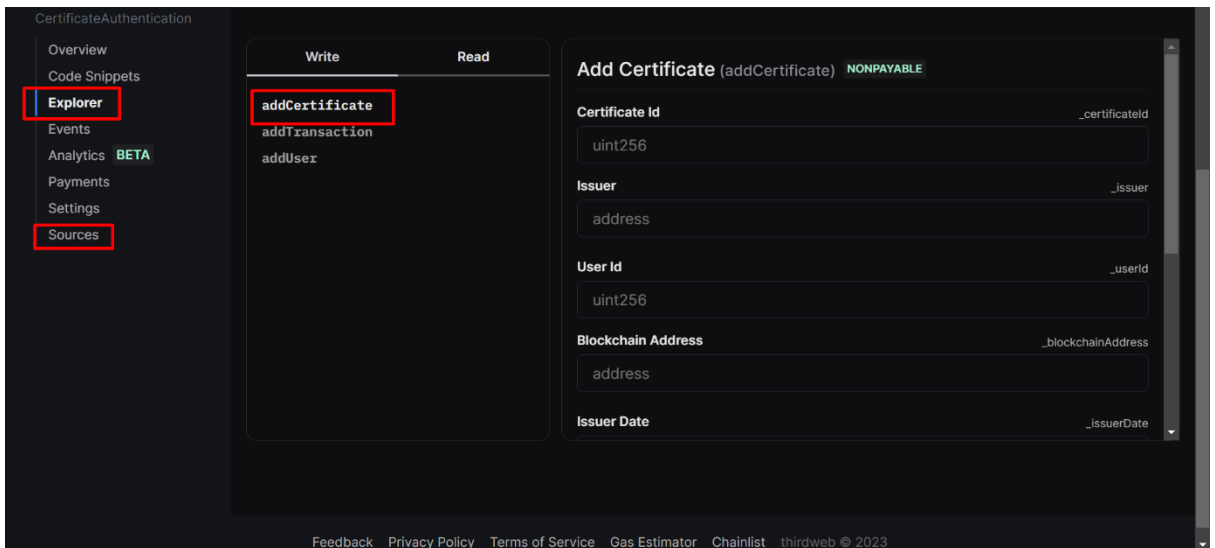


Figure: 6.5 – add Certificate

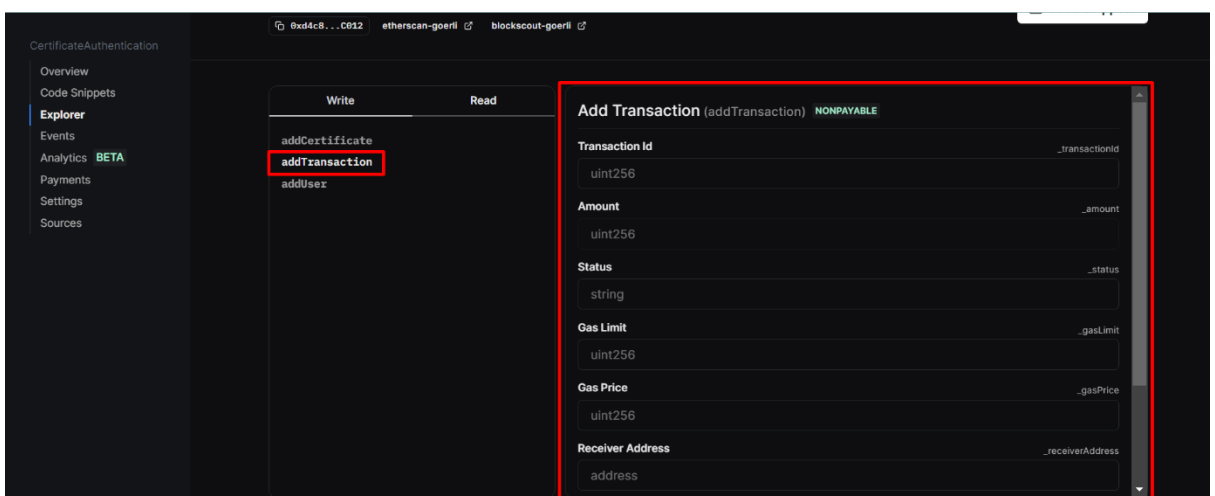


Figure: 6.6 – add Transaction

Chapter 7

Conclusion

7. Project Summary

7.1 Limitations

- Measurable concerns
- Regulatory Challenges
- Dependence on block chain infrastructure
- Transaction costs
- Immutability of data
- Smart contract security
- Privacy concerns
- Technological evolution
- User authentication
- Centralization of authority

7.2 Obstacle & Achievements

Obstacles

- Block chain technology is an unfamiliar technology and thing to me. So it was a big challenge for me to learn and implement it.

Achievements

- Learn Block chain Technology
- Continuous Improvement
- Community Engagement
- Got clear idea about concord algorithm
- Learn Smart contract & front end frame work

- Learn Project Management
- Efficient certificate verification
- Transparent and auditable transactions
- Smart contract integrity
- Compliance with regulations
- Cost-effective transactions

7.3 Future Work

Although the system came out as very refined work, later alterations will be done in major changes as,

Distribute it across the Ethereum network for use.

Connect backend with developed front to deploy.

Be exact when working on user roles at the front end.

Interoperability with Other Blockchains

Machine Learning for Fraud Detection

Smart Contract Audits

Research on Quantum-Resistant Algorithms

Governance Framework

Research on Privacy-Preserving Technologies

REFERENCE

- **Research Paper** : 1st Irawan Afrianto Informatics Engineering Department Universitas Komputer Indonesia Bandung, Indonesia
2nd Yayan Heryanto Informatics Engineering Department Universitas Komputer Indonesia Bandung, Indonesia
- <https://www.researchgate.net/publication/371884440> **A Systematic Literature Review on Blockchain-Based Systems for Academic Certificate Verification**
- <https://www.researchgate.net/publication/352441968> **Blockchain Technology and Academic Certificate Authenticity-Review**
- **Blockchain Platform:** <https://ethereum.org/en/>
- **Transaction:** <https://goerli.etherscan.io/>
- **Thirdweb:** <https://thirdweb.com/>
- **Smart Contract Implement:** <https://remix.ethereum.org/>
- **Diagram:** <https://www.figma.com/>