Samsung Blockchain Capstone
Project

Product Requirements Document (PRD) for Credential Verification System using Blockchain and Decentralized Infrastructure

Submitted by

Maanik Arora
Suneel Kumar Surimani
Vinit Kumar Patra
Adil Sabih
Mohit Sinha

Samsung Research Institute

Bangalore
July, 2023

Contents

List of Figures

1 Introduction

The purpose of this project is to develop a decentralized credential verification system using blockchain technology. The system aims to address the challenges faced by various stakeholders, including regulated entities, private organizations, educational institutions, and platforms, in verifying credentials such as academic transcripts, standardized test scores, skill-based training certificates, vocational course certificates, and executive education certificates.

2 Abstract

The credential verification system using blockchain and decentralized infrastructure is designed to revolutionize the process of verifying credentials across various stakeholders, including regulated entities, private organizations, educational institutions, and platforms. The current centralized systems face challenges such as lack of transparency, inefficient verification pro- cesses, privacy and security concerns, and lack of standardization. The proposed solution leverages blockchain technology, smart contracts, and decentralized infrastructure to address these challenges.

The project involves the development and deployment of smart contracts on either the Goerli or Sepolia Ethereum Testnet Blockchain via Infura, Alchemy, QuickNode, or Moralis, or the Polygon Testnet Blockchain using a free infrastructure provider. The central smart contract is responsible for storing critical information related to students and handles the deployment and verification of credentials. Stakeholder contracts, including regulated entity, private entity, platform, and NFT contracts, are developed to cater to specific functionalities and requirements.

The regulated entity contract enables the registration of education boards, courses, and the assignment of students to courses. It also allows whitelisting of private entities for secure transcript verification and maintains IPFS Content Identifier (CID) for credential-containing PDFs. The private entity contract facilitates the creation of new private entities and maintains a roster of access requests and candidate information. The platform contract handles course creation, student registration, course progress tracking, completion milestones, and generation of NFTs as credentials. The NFT contract deploys NFTs for milestones completed by a student and commemorates the completion of all course requirements.

To ensure efficient storage and retrieval of credential PDFs, a local IPFS node is set up and the corresponding CIDs are maintained in a mapping. The system includes a user-friendly verification interface for stakeholders to access and verify credentials stored on the blockchain. APIs are provided for seamless integration with external systems and platforms.

The credential verification system aims to enhance transparency, immutability, and efficiency in the verification process while providing a standardized approach for stakeholders across different sectors. By leveraging blockchain and decentralized infrastructure, the system ensures trust, security, and ease of verification, leading to a more reliable and streamlined credential verification ecosystem.

3 Problem Statement

The current centralized credentials assignment, management, and verification systems suffer from several issues, including lack of transparency, inefficient verification processes, privacy and security concerns, and lack of standardization. The proposed solution aims to overcome these challenges by leveraging blockchain technology and decentralized infrastructure.

4 Objectives

The proposed platform aims to achieve the following objectives:

4.1 Establish a Decentralized Infrastructure

1. Set up a decentralized infrastructure using blockchain technology to ensure transparency, immutability, and security of credentials.
2. Utilize either the Goerli or Sepolia Ethereum Testnet Blockchain via Infura, Alchemy, QuickNode, or Moralis.
3. Alternatively, use the Polygon Testnet Blockchain via a free infrastructure provider.

4.2 Smart Contract Implementation

1. Develop a central smart contract with functions to handle the deployment and verification of credentials.
2. Declare necessary variables to store critical information related to students across stake- holders.
3. Define functions for deploying credentials to the blockchain and verifying them in a timely manner.
4. Implement abstract functions to be inherited and defined by stakeholder contracts.

4.3 Stakeholder Contracts

4.3.1 Regulated Entity Contract

1. Enable the registration of education boards.
2. Allow registration of new courses offered by education boards.
3. Assign students to courses.
4. Whitelist private entities for secure transcript verification.
5. Maintain IPFS CID (Interplanetary File System Content Identifier) for credential-containing PDFs.

4.3.2 Private Entity Contract

1. Allow the creation of new private entities.
2. Maintain a roster of access requests made by private entities.
3. Maintain a database of candidate information.

4.3.3 Platform Contract

1. Create new courses on the platform.
2. Register students to their respective courses.
3. Track course progress for each student.
4. Track the completion of necessary milestones for each student.
5. Call an ERC721 contract to generate Non-Fungible Tokens (NFTs) as credentials.
6. Maintain a roster of NFTs and their respective public addresses.
7. Emit the NFT address on creation for GUI or logging purposes.

4.3.4 NFT Contract

1. Deploy NFTs for each milestone completed by a student.
2. Deploy an NFT to commemorate the completion of all requirements (graded and un-graded) of a course.
3. Allow public function calls for verification of NFT details.

4.4 IPFS Integration

1. Set up a local IPFS node for efficient storage and retrieval of credential-containing PDFs.
2. Store credential PDFs via IPFS and maintain the corresponding CIDs in a mapping.

5 Deliverables

The final deliverables of the project will include:

5.1 Deployed and Functional Smart Contracts

1. Develop and deploy the central smart contract, stakeholder contracts, and NFT contract on the chosen blockchain network

2. Ensure proper access control and verification processes are implemented.

3. The **NFTContract** is an ERC-721 compliant smart contract that allows a PlatformEntity contract to mint and manage non-fungible tokens (NFTs) for student milestones and commemorative purposes. The contract stores NFT details, such as course names and commemorative status, and emits events upon NFT creation. It employs a unique NFT ID generation process using keccak256 hash with student address and course name. The contract also features a public function to verify NFT details. The PlatformEntity contract is the only authorized entity to access specific functions. The contract's goal is to facilitate the issuance of NFTs for educational achievements.

4. The abstract contract "**PlatformEntity**" facilitates the management of educational courses, student progress, and NFT generation. It includes functions to create and register students to courses, update course progress, and generate NFTs for students upon course completion. Courses are represented as a struct containing course names and milestone IDs. Student progress is tracked with a struct that records completed milestone IDs for each course. The contract relies on an external NFT contract (NFTContract) to create NFTs, with the NFT contract address set using the "setNFTContract" function. The contract is designed to be inherited and requires an implementation for the "generateNFT" function for NFT creation logic.

5. The "**PrivateEntity**" contract enables the creation and management of private entities with their respective names and addresses. It features functions to create new private entities, check the count of registered entities, and retrieve entity details by index. The contract is secured using a modifier that restricts certain functions to be accessed only by the contract owner. However, the contract mentions additional functionalities, such as maintaining a roster of accesses requested by private entities and managing a roster of candidates, which are not implemented in this contract. These functionalities can be developed in separate contracts or integrated into this contract based on specific requirements.

6. The "**RegulatedEntity**" abstract contract is designed to manage and regulate educational entities, courses, students, and their interactions. It allows regulated entities to register, create courses, enroll students, and whitelist private entities. Students can be assigned to courses, and whitelisted private entities can verify student transcripts. The contract maintains mappings to track course and student information and emits events for contract interaction tracking. The contract also defines an interface for the NFT contract to generate NFTs based on student achievements, but the NFT creation logic is left abstract for implementation in the derived contract.

5.2 Decentralized Infrastructure with IPFS (Optional)

1. Set up a local IPFS node stored and retrieve credential-containing PDFs efficiently.
2. Establish a mechanism to maintain the IPFS CIDs within the system.

5.3 Verification Interface (UI)

1. Develop an intuitive user interface for stakeholders to access and verify credentials stored on the blockchain.
2. Ensure a seamless and user-friendly experience for the verification process.

5.4 Documentation

1. Prepare comprehensive documentation covering system architecture, smart contract code, API documentation, and user guides.
2. Provide detailed instructions on how to interact with the system and perform verification processes.

6 Timeline

The project will be divided into the following milestones:

6.1 Milestone 1: Smart Contract Development and Deployment (Weeks 1)

1. Implement abstract functions for stakeholder contracts and define access control modifiers.
2. Develop and deploy the central smart contract with necessary variables and functions

6.2 Milestone 2: Stakeholder Contracts and IPFS Integration (Weeks 2)

1. Develop and deploy stakeholder contracts (regulated entity, private entity, platform, and NFT
2. Implement functionality for registration, assignment, whitelisting, and access control.
3. Set up a local IPFS node for efficient storage and retrieval of credential PDFs.

6.3 Milestone 3: Integration with Blockchain Networks (Weeks 3)

1. Connect the system to the chosen Ethereum testnet (Goerli or Sepolia) or Polygon testnet.
2. Deploy smart contracts on the selected blockchain network.
3. Ensure proper communication and interaction with the blockchain through APIs.

6.4 Milestone 4: Verification Interface & API Integration(Weeks 4)

1. Develop a user-friendly verification interface for stakeholders to access and verify credentials.
2. Integrate APIs to enable external systems and platforms to interact with the verification system.
3. Test and ensure seamless functionality of the verification processes.

6.5 Milestone 5: Documentation and Finalization (Weeks 5)

1. Prepare detailed documentation covering system architecture, smart contract code, API documentation, and user guides.
2. Review and finalize all deliverables.

3. Conduct thorough testing and quality assurance.

7 Methodology

7.1 Initial requirement actors

1. Student:
    - Can register for courses offered by education boards.
    - Can submit credentials for verification.
    - Can view the status of their submitted credentials.
    - Can access and share verified credentials.
2. Regulated Entity (Education Board):
    - Can register as an education board.
    - Can register courses offered by the education board.
    - Can assign students to courses.
    - Can whitelist private entities for transcript verification.
    - Can securely verify transcripts of students.
3. Private Entity:
    - Can create a new private entity.
    - Can request access to verify transcripts of students.
    - Can view the list of candidates for verification.
    - Can securely verify the transcripts of candidates.
4. Platform:
    - Can create and manage new courses.
    - Can register students to their respective courses.
    - Can track the course progress of students.
    - Can track the completion of necessary milestones for each student.
    - Can generate Non-Fungible Tokens (NFTs) as credentials for students.
    - Can emit the NFT address on creation for GUI or logging purposes.
5. NFT Contract:
    - Represents the Non-Fungible Token (NFT) contract.
    - Deploys NFTs for each milestone completed by a student.
    - Deploys an NFT to commemorate the completion of all course requirements.
    - Provides public functions for verification of NFT details.

6. Central Smart Contract:

    • Stores critical information related to students.

    • Handles the deployment and verification of credentials.

    • Declares basic variables for storing student information.

    • Defines functions for deploying credentials and verifying them.

    • Declares abstract functions to be inherited and defined by stakeholder contracts.

7. IPFS (Interplanetary File System):

    • Provides a decentralized storage system.

    • Generates and maintains IPFS Content Identifiers (CIDs) for PDFs.

8. Blockchain Network:

    • Provides the underlying blockchain infrastructure for storing and validating smart contracts and transactional data.

    • Enables interaction with the deployed smart contracts.

9. User Interface (UI):

    • Provides an intuitive interface for stakeholders to interact with the system.

    • Allows students to register for courses, submit credentials, and view verification status.

    • Allows regulated entities to manage courses, assign students, and verify transcripts.

    • Allows private entities to request access, view candidates, and verify transcripts.

    • Provides access to view and share verified credentials.

10. API:

    • Enables integration with external systems and platforms for seamless verification processes

    • Allows other applications to interact with the credential verification system.

11. Database:

    • Stores and manages data related to stakeholders, courses, credentials, verification status, and NFTs.

    • Provides persistent storage for the system’s data.

7.2 Block Diagram

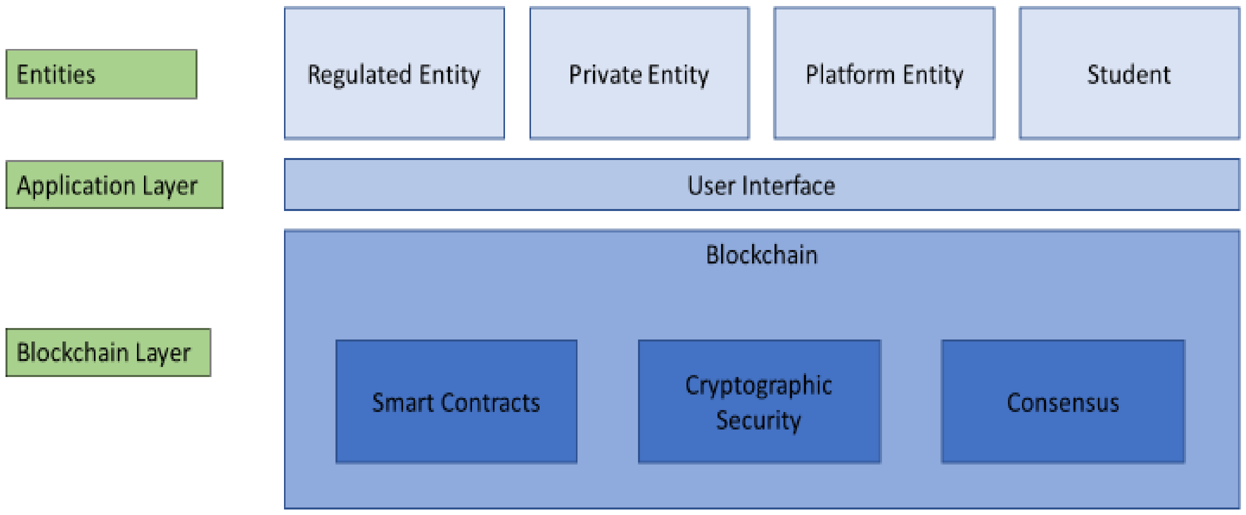


Fig: Block diagram highlighting the principal parts and functions of the system

7.3 Flowchart Diagram

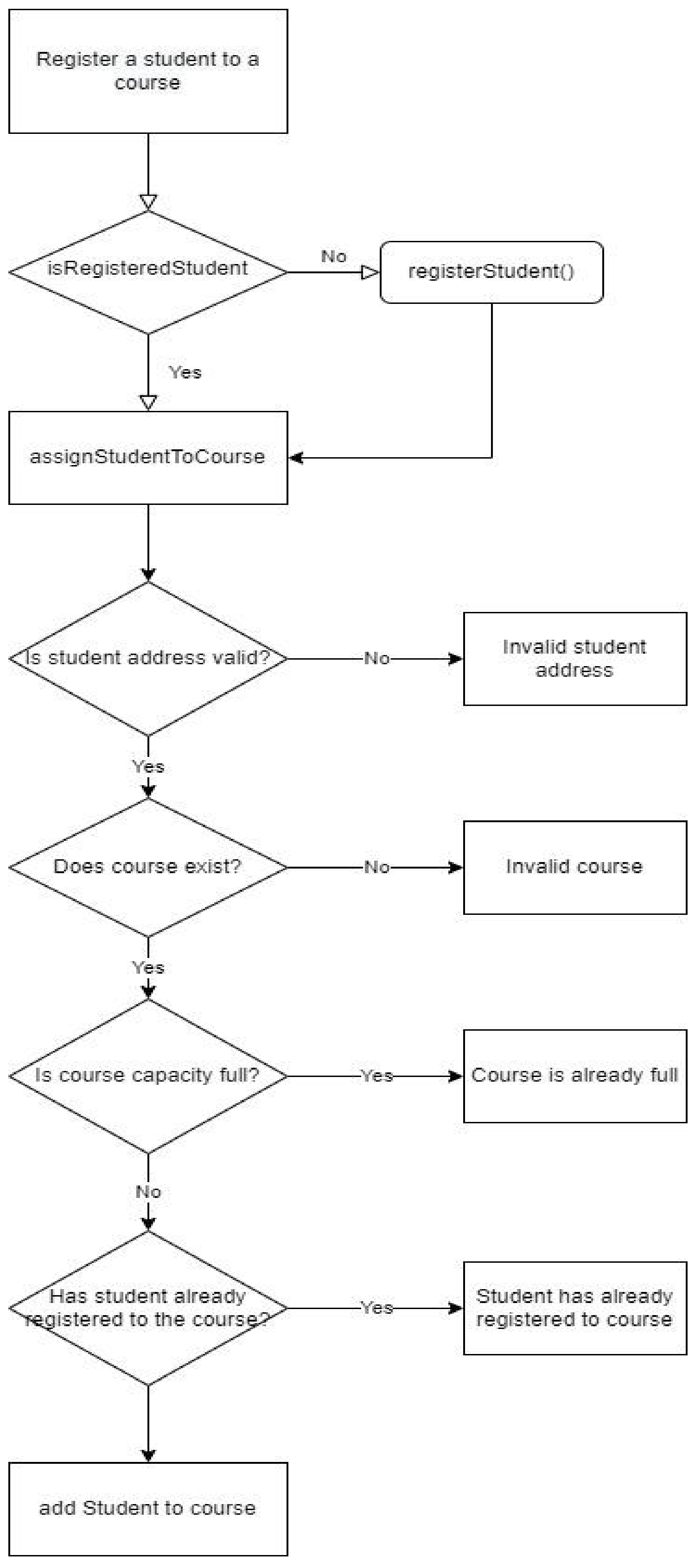


Fig: Register a student to a course

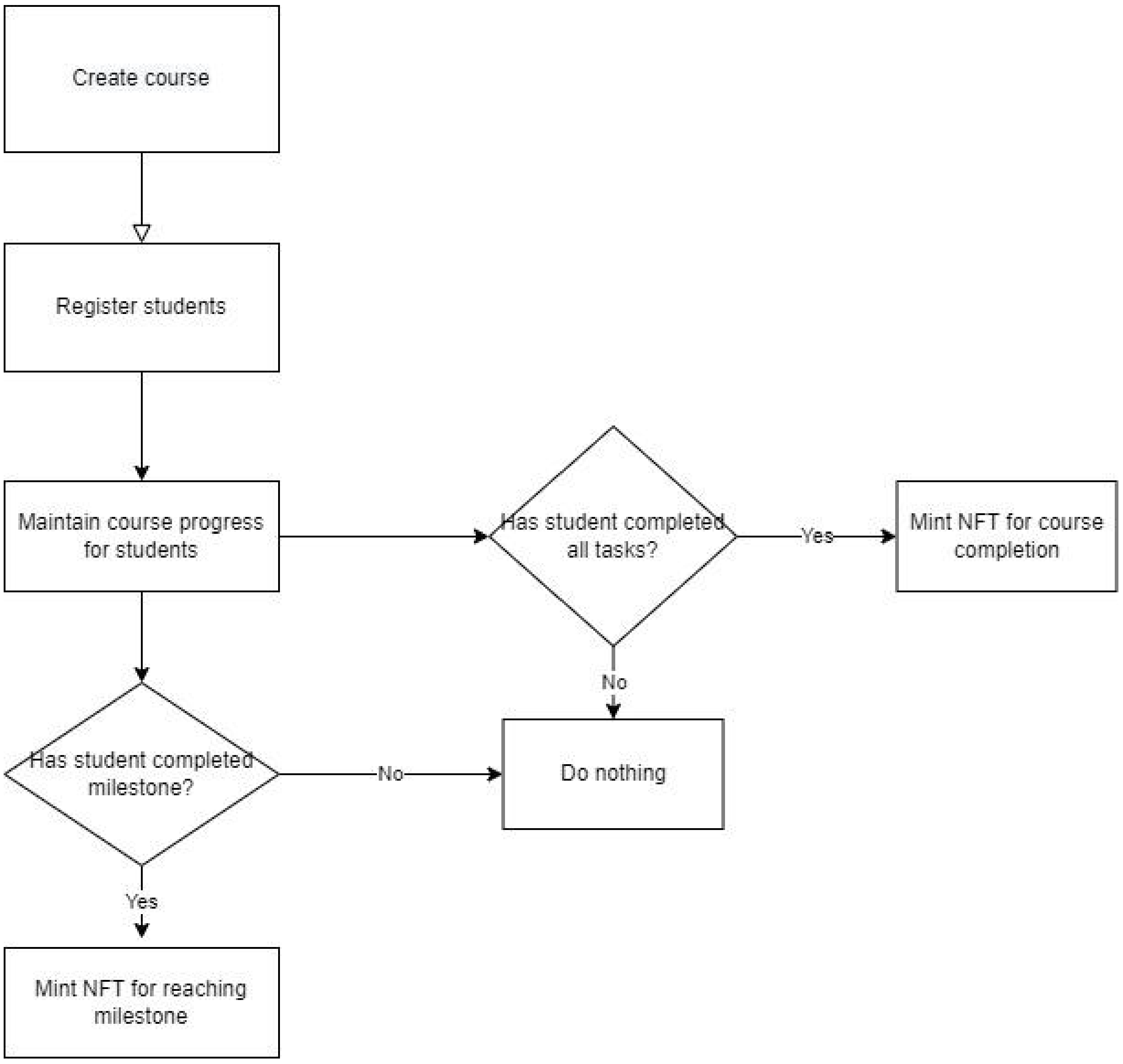

Create course
Register students
Maintain course progress for students
Has student completed all tasks?
Yes
Mint NFT for course completion
No
Has student completed milestone?
No
Do nothing
Yes
Mint NFT for reaching milestone

Fig: Course completion cycle for Platform entities

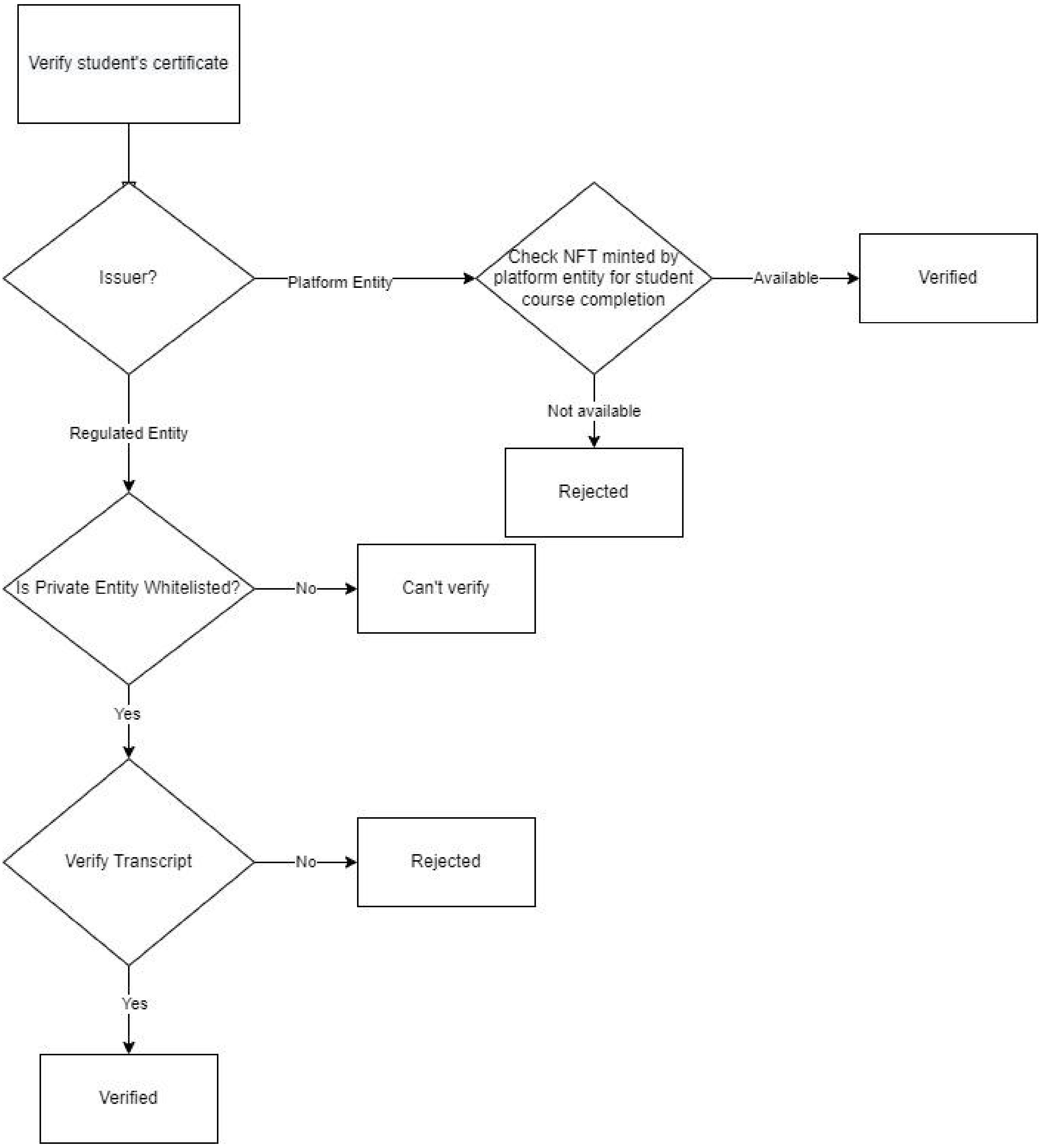


Fig: Verifying a student's certificate by a private entity

7.4 Class UML Diagram

Class UML Diagram Link

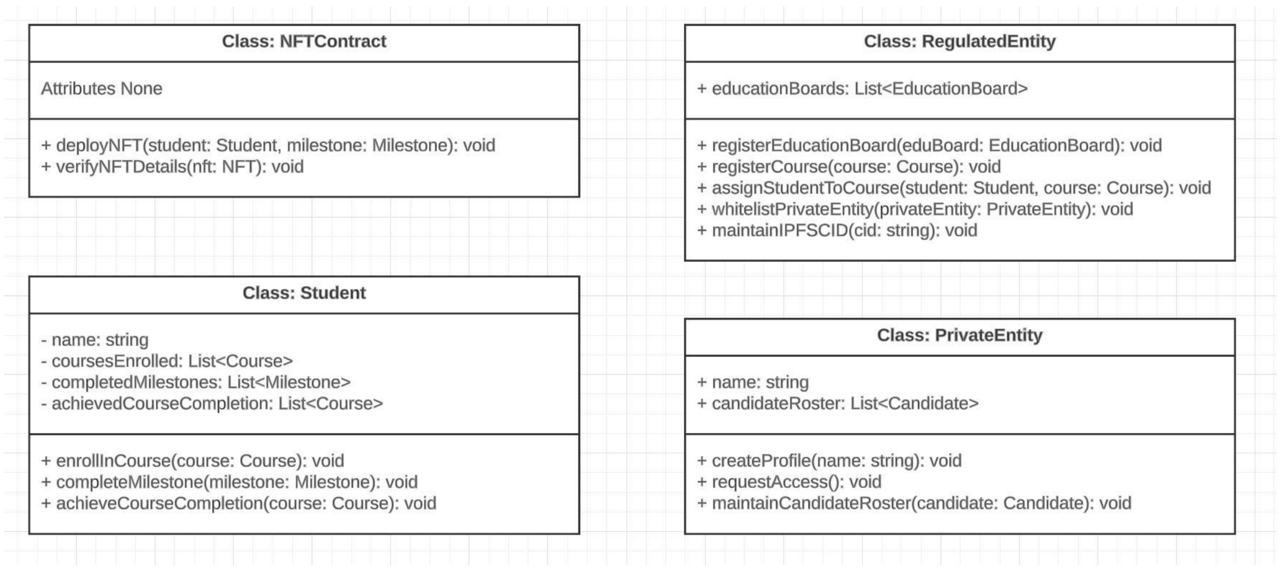

Class: NFTContract
Attributes None
+ deployNFT(student: Student, milestone: Milestone): void
+ verifyNFTDetails(nft: NFT): void
Class: RegulatedEntity
+ educationBoards: List<EducationBoard>
+ registerEducationBoard(eduBoard: EducationBoard): void
+ registerCourse(course: Course): void
+ assignStudentToCourse(student: Student, course: Course): void
+ whitelistPrivateEntity(privateEntity: PrivateEntity): void
+ maintainIPFSCID(cid: string): void
Class: Student
- name: string
- coursesEnrolled: List<Course>
- completedMilestones: List<Milestone>
- achievedCourseCompletion: List<Course>
+ enrollInCourse(course: Course): void
+ completeMilestone(milestone: Milestone): void
+ achieveCourseCompletion(course: Course): void
Class: PrivateEntity
+ name: string
+ candidateRoster: List<Candidate>
+ createProfile(name: string): void
+ requestAccess(): void
+ maintainCandidateRoster(candidate: Candidate): void

Figure 2: Class UML Part 1

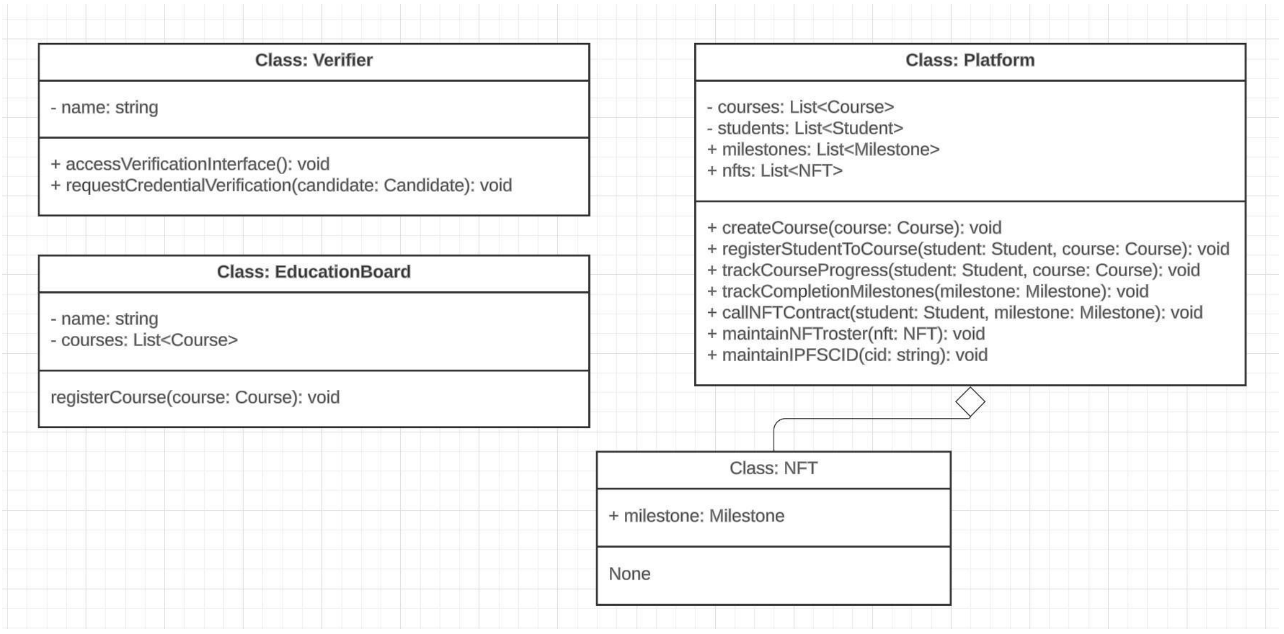

Class: Verifier
- name: string
+ accessVerificationInterface(): void
+ requestCredentialVerification(candidate: Candidate): void
Class: EducationBoard
- name: string
- courses: List<Course>
registerCourse(course: Course): void
Class: Platform
- courses: List<Course>
- students: List<Student>
+ milestones: List<Milestone>
+ nfts: List<NFT>
+ createCourse(course: Course): void
+ registerStudentToCourse(student: Student, course: Course): void
+ trackCourseProgress(student: Student, course: Course): void
+ trackCompletionMilestones(milestone: Milestone): void
+ callNFTContract(student: Student, milestone: Milestone): void
+ maintainNFTroster(nft: NFT): void
+ maintainIPFSCID(cid: string): void
Class: NFT
+ milestone: Milestone
None

Figure 3: Class UML Part 2

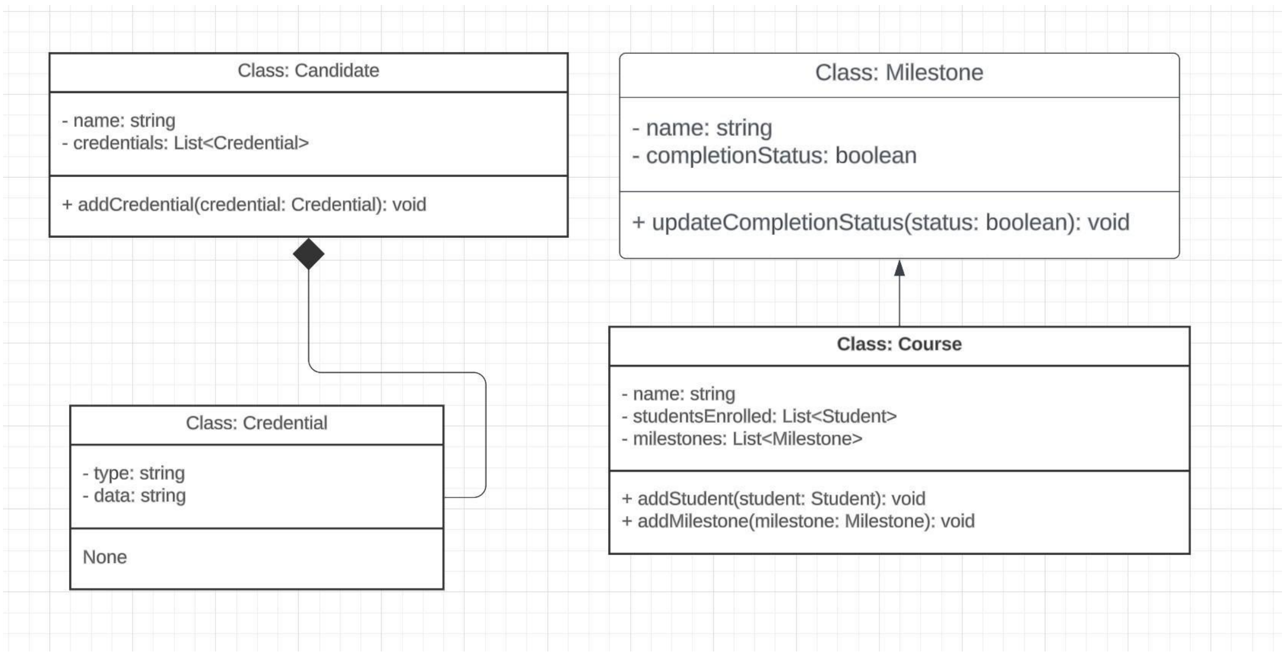

Class: Candidate
- name: string
- credentials: List<Credential>
+ addCredential(credential: Credential): void
Class: Milestone
- name: string
- completionStatus: boolean
+ updateCompletionStatus(status: boolean): void
Class: Credential
- type: string
- data: string
None
Class: Course
- name: string
- studentsEnrolled: List<Student>
- milestones: List<Milestone>
+ addStudent(student: Student): void
+ addMilestone(milestone: Milestone): void

Figure 4: Class UML Part 3

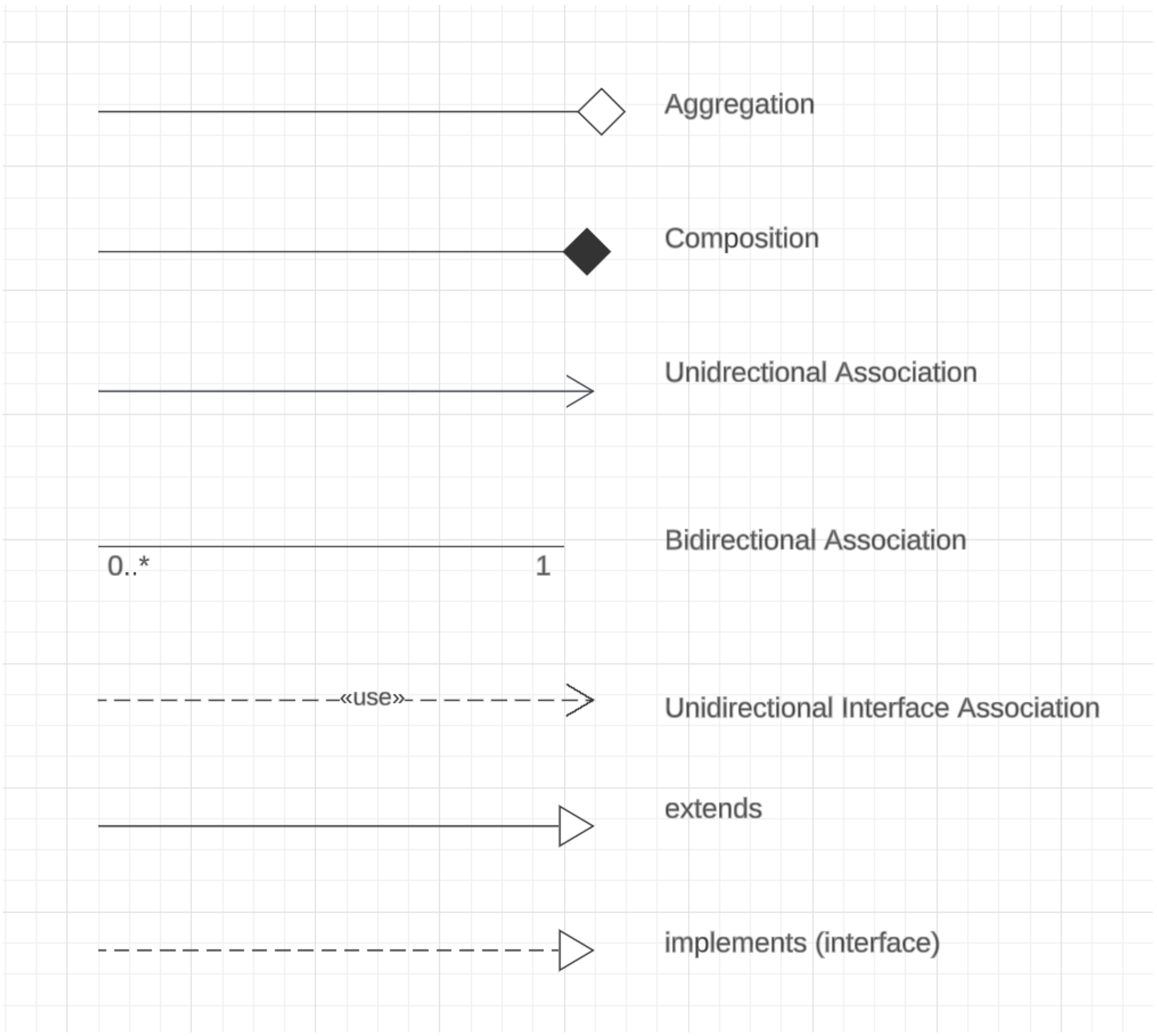

Aggregation
Composition
Unidrectional Association
Bidirectional Association
0..*
1
«use»
Unidirectional Interface Association
extends
implements (interface)

Figure 5: Class UML Part 4

7.5 Sequence Flow

1. Student Enrollment Sequence:
    - The student sends a request to the platform to enroll in a specific course.
    - The platform validates the request and registers the student for the course.
    - The platform updates the student's enrollment status and notifies the student about successful enrollment.
2. Milestone Completion Sequence:
    - The student completes a milestone within the enrolled course.
    - The platform receives the milestone completion notification from the student.
    - The platform checks the completion status of all milestones for the student's course.
    - If all milestones are completed, the platform initiates the generation of an NFT as a credential.
3. NFT Generation Sequence:
    - The platform calls the NFT contract and provides the necessary information, including the student's details and completion of all course requirements.
    - The NFT contract deploys an NFT for the student, representing their course completion.
    - The NFT contract generates a unique public address for the NFT and associates it with the student's details.
    - The NFT contract notifies the platform about the successful generation of the NFT.
4. Credential Verification Sequence:
    - A verifier accesses the system's verification interface and provides the candidate's details for credential verification.
    - The system retrieves the candidate's information from the blockchain, including their enrolled courses and associated NFTs.
    - The system verifies the authenticity and validity of the NFTs and their associated details.
    - If the verification is successful, the system sends a confirmation to the verifier indicating the credentials are valid.
5. Whitelisting Request Sequence:
    - A private entity sends a request to the regulated entity for whitelisting.
    - The regulated entity receives the request and verifies the credentials and legitimacy of the private entity.
    - If the verification is successful, the regulated entity adds the private entity to the whitelist for secure transcript verification
6. Transcript Verification Sequence:
    - A private entity requests access to verify a candidate's transcript.
    - The private entity sends a verification request to the regulated entity.
    - The regulated entity validates the request and checks if the private entity is whitelisted.
    - If the private entity is whitelisted, the regulated entity retrieves the candidate's transcript information and shares it securely with the private entity..
    - The private entity verifies the transcript and provides the verification result to the regulated entity.

8 Software Workings and Details

8.1 IPFS Database

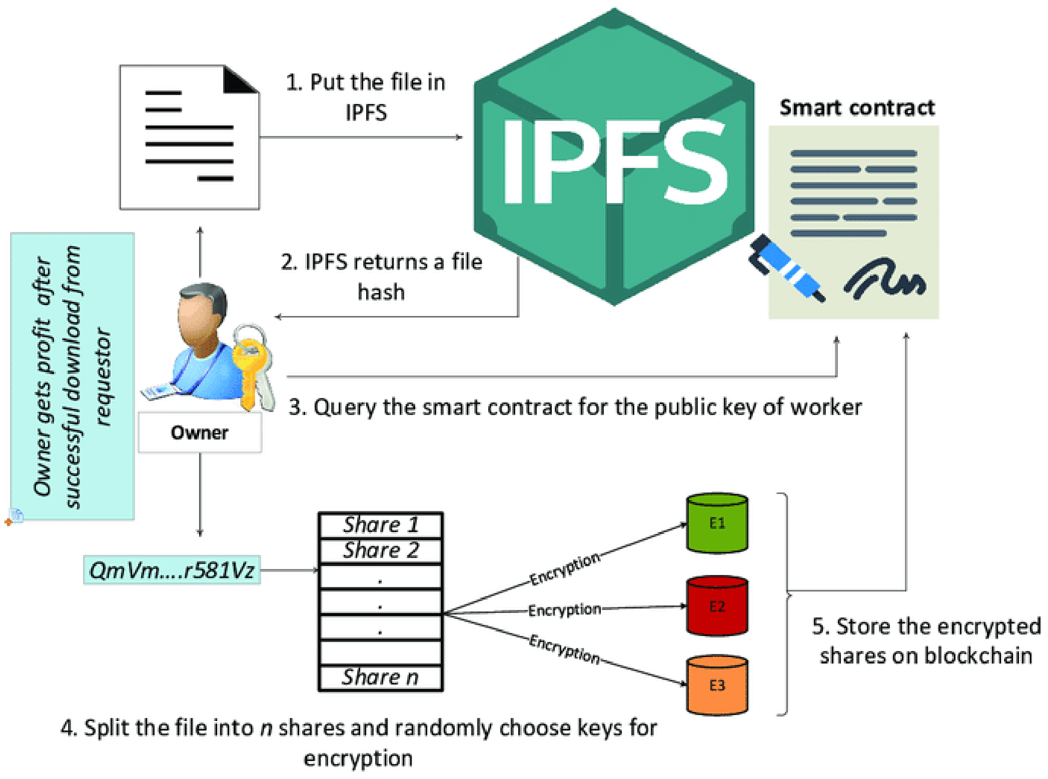

1. Put the file in IPFS
IPFS
Smart contract
2. IPFS returns a file hash
Owner gets profit after successful download from requestor
Owner
3. Query the smart contract for the public key of worker
QmVm....r581Vz
Share 1
Share 2
Share n
Encryption
Encryption
Encryption
E1
E2
E3
5. Store the encrypted shares on blockchain
4. Split the file into n shares and randomly choose keys for encryption

Figure 6: IPFS Database

IPFS is a distributed file system that acts as a decentralized database in our credential verification system. Here's how it works:

1. IPFS CID: IPFS uses Content Identifiers (CIDs) to uniquely identify content. In our system, the IPFS CID is used to reference and retrieve the credential-containing PDFs.

2. Local IPFS Node: The system sets up a local IPFS node, which acts as a gateway to the IPFS network and stores the credential PDFs efficiently.

3. Storage and Retrieval: When a user uploads a credential PDF, the system stores it in the IPFS network through the local node. The file is broken down into smaller chunks, encrypted, and distributed across multiple IPFS nodes.

4. CID Mapping: The system maintains a mapping that associates the credential data with the corresponding IPFS CID. This mapping allows for easy retrieval and verification of credentials based on the provided CID.

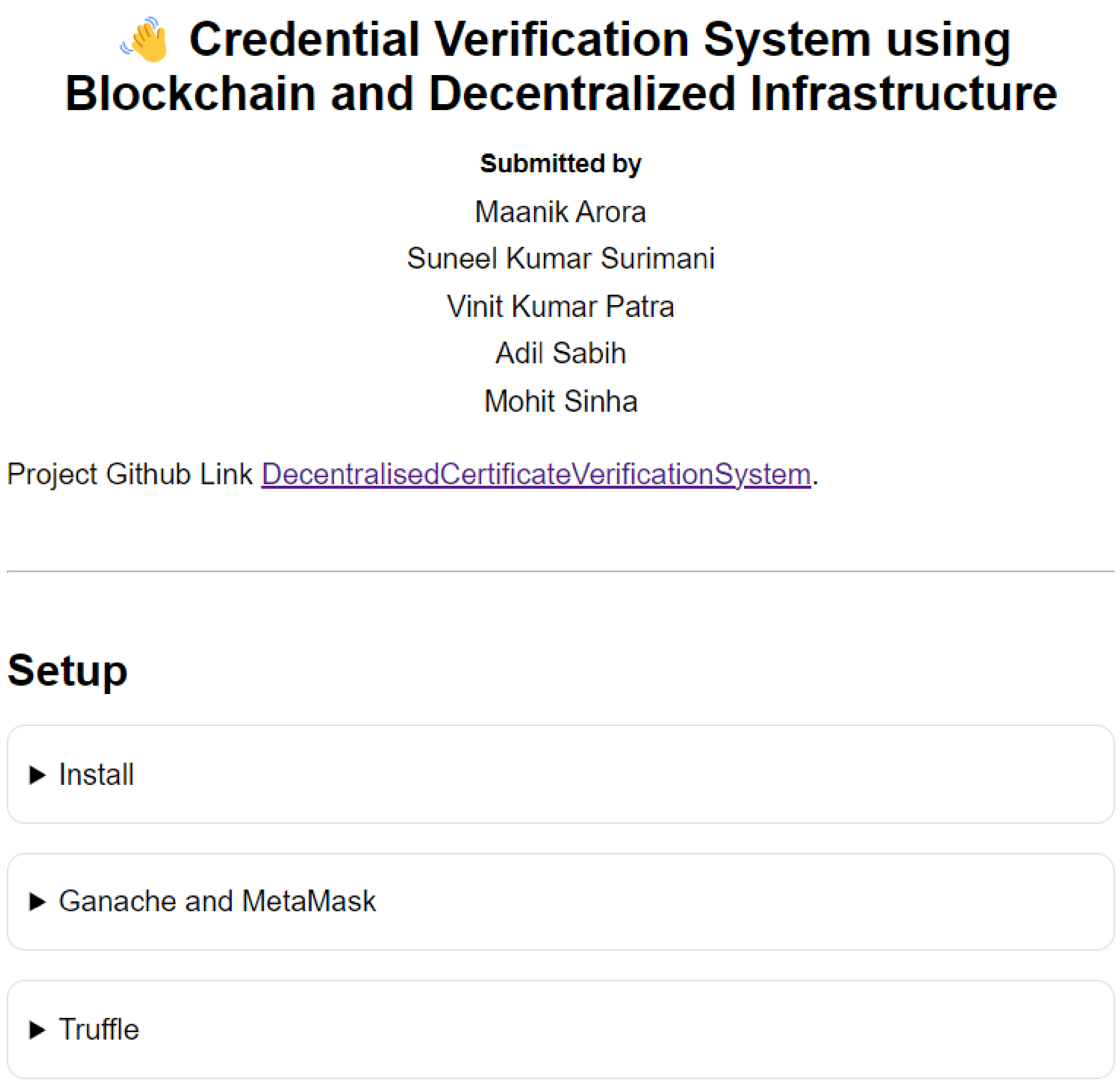
5. Efficient Retrieval: To retrieve a credential PDF, the system uses the CID to request the file from the IPFS network. The distributed nature of IPFS ensures efficient retrieval, as the file can be obtained from any node that has a copy.

6. Decentralized Storage: IPFS provides decentralized storage, eliminating reliance on a central server. This enhances security, as the data is distributed across multiple nodes, reducing the risk of a single point of failure or data loss.

The integration of IPFS as a database in the credential verification system ensures secure and efficient storage of credential PDFs. By leveraging its decentralized architecture, IPFS enhances data availability, reliability, and resilience, making it an ideal choice for storing sensitive credentials while maintaining data integrity and privacy. It’s important to note that the database in this system also may include traditional relational or NoSQL databases for storing other system-related data, such as user information, course details, and NFT metadata. The choice of database technology would depend on specific requirements, scalability needs, and data modeling considerations.

8.2 WebApp GUI

Some features of WebApp interface are as follows:



👋 Credential Verification System using Blockchain and Decentralized Infrastructure

Submitted by

Maanik Arora

Suneel Kumar Surimani

Vinit Kumar Patra

Adil Sabih

Mohit Sinha

Project Github Link DecentralisedCertificateVerificationSystem.

Setup

▶ Install

▶ Ganache and MetaMask

▶ Truffle

Figure 7: WebApp GUI

Setup

▼ Install

Install Truffle and Ganache globally.

```
$ npm install -g truffle ganache
```

▼ Ganache and MetaMask

Open a terminal and run Ganache, a simulated Ethereum blockchain on your machine.

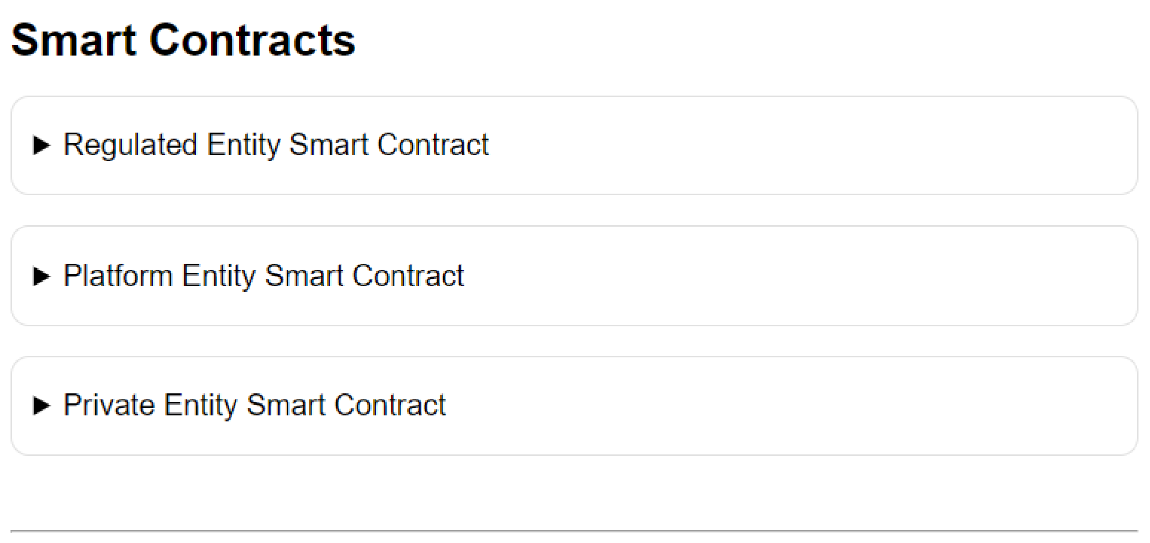
```
$ ganache
```

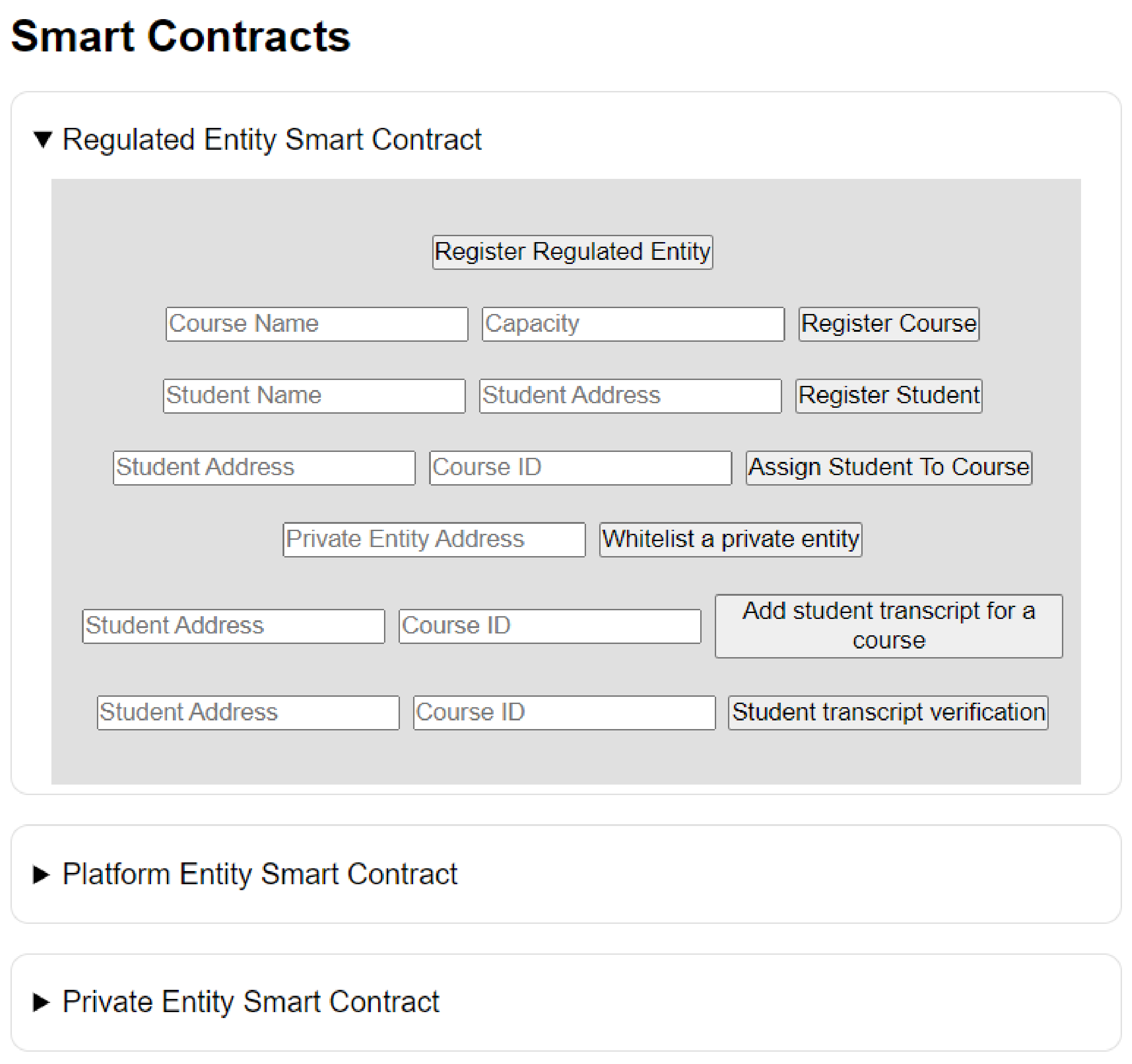
From the list of generated private keys, import the first one to MetaMask.

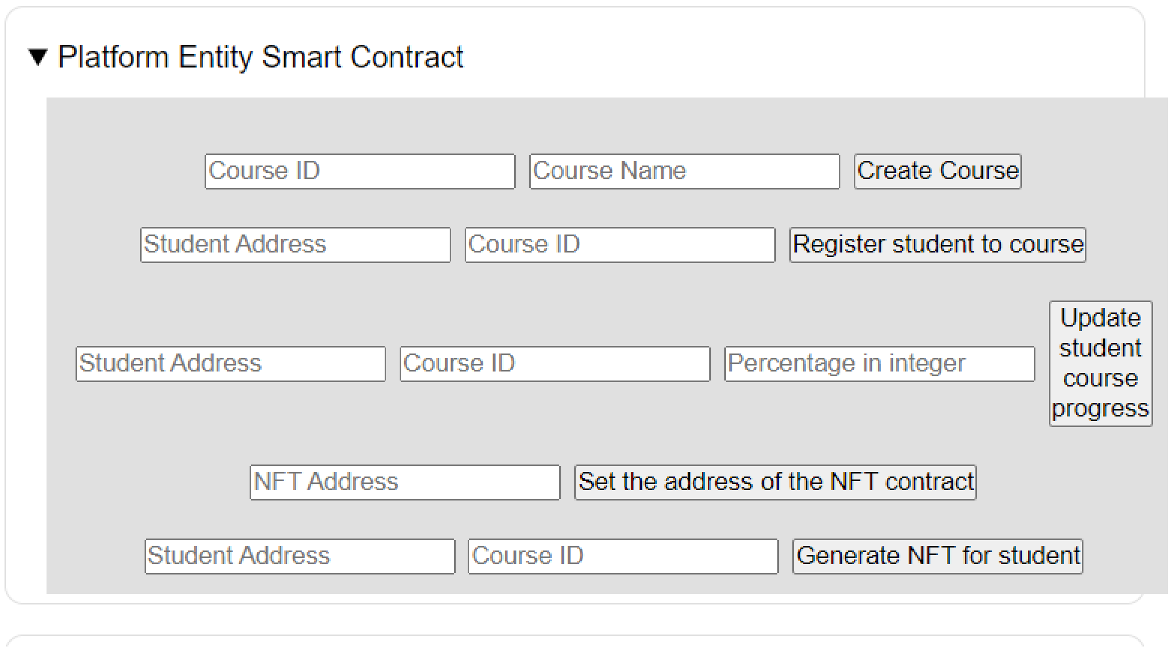
▼ Truffle

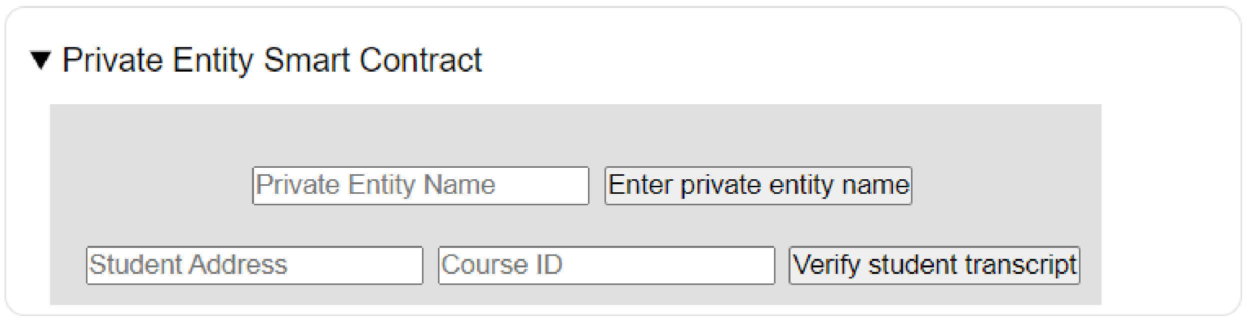
Keep Ganache running and open another terminal. Let's compile and deploy our contracts to Ganache.

```
$ cd truffle
$ truffle migrate --network development
# The `development` network points to Ganache, it's configured in
truffle/truffle-config.js on line 45.
```


Smart Contracts
▶ Regulated Entity Smart Contract
▶ Platform Entity Smart Contract
▶ Private Entity Smart Contract


Smart Contracts
▼ Regulated Entity Smart Contract
Register Regulated Entity
Course Name
Capacity
Register Course
Student Name
Student Address
Register Student
Student Address
Course ID
Assign Student To Course
Private Entity Address
Whitelist a private entity
Student Address
Course ID
Add student transcript for a course
Student Address
Course ID
Student transcript verification
▶ Platform Entity Smart Contract
▶ Private Entity Smart Contract


▼ Platform Entity Smart Contract
Course ID
Course Name
Create Course
Student Address
Course ID
Register student to course
Student Address
Course ID
Percentage in integer
Update student course progress
NFT Address
Set the address of the NFT contract
Student Address
Course ID
Generate NFT for student


▼ Private Entity Smart Contract
Private Entity Name
Enter private entity name
Student Address
Course ID
Verify student transcript

9 Conclusion

The proposed credential verification system aims to revolutionize the current centralized approach by leveraging blockchain technology and decentralized infrastructure. By ensuring trans- parency, immutability, and efficient verification processes, the system will provide a reliable and standardized method for verifying credentials across various stakeholders.

References

[1] Nakaomoto, Satashi. “Bitcoin, A Peer-to-Peer Electronic Cash System.” https://bitcoin.org/bitcoin.pdf

[2] Ethereum Foundation. “ethereum: Blockchain Platform.” https://www.ethereum.org/