A Minor Project Final Report on

Digital Certificate Verification System Using Blockchain

Submitted in Partial Fulfillment of the Requirements
for the Degree of Bachelor of **Engineering in Computer Engineering**
under Pokhara University

Submitted by:

Amrit Bhusal, 171303
Yukta Prasad Sharma, 171344
Sushant Babu Luitel, 171338
Satendra Yadav, 171351

Under the supervision of
Asst. Prof. Mahesh Neupane

Date:
03 October, 2021

Department of Computer Engineering

NEPAL COLLEGE OF INFORMATION TECHNOLOGY

Balkumari, Lalitpur, Nepal

Acknowledgement

First of all, we would like to express our sincere gratitude towards the **Department of Computer Engineering, Pokhara University** for including the 6th Semester minor project as part of the syllabus. Also, we extend our gratitude towards **Mrs. Resha Deo**, Head of Department, Computer Engineering, for assisting us in the project.

Special thanks and gratitude to our project supervisor **Asst. Prof. Mahesh Neupane** under whose guideline we were able to complete the project. We are wholeheartedly thankful to him for giving us his valuable time and attention and for providing us a systematic way for completing our project in time.

Extending thanks to the mentor **Mr. Biplav Raj Osti**, Blockchain Architect at iBriz, for his valuable time, knowledge inputs and guidance.

Our thanks goes to all our fellow friends and college staff who willingly helped us out with their abilities.

Every attempt has been made to include each and every aspect of the project in this report so that the reader can clearly understand the project. We would be pleased to get the feedback on it.

Sincerely,

Amrit Bhusal, 171303
Yukta Prasad Sharma, 171344
Sushant Babu Luitel, 171338
Satendra Yadav, 171351

Abstract

Blockchain is a new technology redesigning the Information and Technology industry with its characteristics like decentralization, immutable data storing capability, peer-to-peer connection, and distributed ledger. The primary target of Blockchain technology is to revolutionize the data storage technique providing enhanced security. There is merely a field, blockchain is not applicable to. The certification system is one of them. The traditional way of storing and verification of digital certificates had many issues such as data loss, easy data modification by unauthorized people, and reliability of third parties. In traditional techniques, verification of data takes a lot of time. Also, the advancement of IT has enabled the forgery of important documents like certificates, passports, citizenship id, etc. The primary goal of this paper is to propose a certificate verification system that can offer potential solutions for issuing and verification using blockchain technology. The blockchain technology has emerged as one of the potential ways of authenticated certificate verification and management process that combats the misuse and frauds of digital certificate credentials. The blockchain technology has several functionalities like hash generation, public/private key cryptography, peer-to-peer network, proof of work, digital signatures, etc. Using these functionalities, this system ensures trust between the parties ignoring their background and keeps digital certificates secure. Also, this system avoids dependency on third parties for digital certificate verification.

Keywords: Blockchain, digital certificate, hash, cryptography

Technical Terms: Ethereum, Distributed Ledger

Table Of Contents

List Of Figures

List of Tables

Abbreviations

DCVS	Digital Certificate Verification System
IPFS	Interplanetary File System
NCIT	Nepal College Of Information Technology
UI	User Interface
IT	Information Technology
DCC	Digital Credentials Consortium
MIT	Massachusetts Institute of Technology
API	Application Programming Interface
ERC	Ethereum Request for Comment
SDLC	Software Development Life Cycle
DAPP	Decentralised Application
TSS	Time Stamping Service
UML	Unified Modeling Language
PoW	Proof-of-Work
EVM	Ethereum Virtual Machine
CLI	Command Line Interface
HTTP	HyperText Transfer Protocol
IPC	Inter Process Communication
SQL	Structured Query Language
JSON	JavaScript Object Notation
SHA	Secure Hash Algorithm
ECDSA	Elliptic Curve Digital Signature Algorithm
ECC	Elliptic Curve Cryptography
CID	Content Identifier

1. Introduction

Certificates are documents stating the particular facts are true. One can earn a certificate after completion of a course or any other achievements. They are the most common form of data that ensures one's participation and achievement. Certificates are something that are issued by a central authority whether it is a college or an organization or a government body. University degrees (a particular type of certificate) can help you get the job you want, or prevent you from getting it if you don't have the right certificate. Ideally we should be in charge of our own credentials. But most of the time we have to rely on third parties, such as universities or employers to store, verify, and validate our credentials. Today, in every global market, the certificates issued to us are used to access our knowledge and skills that we gained through our work in that field. But the problems of certificate counterfeiting has been increasing in every nook and cranny of the world. Certificates these days are being used in printed format. Use of it in digitized form is also increasing day by day, as it is easy to operate with. However, the authenticity of digital certificates has been scrutinized as faking it has been easier. This not only affects people who are provided with a certificate but also the organization issuing it as well. Also, the certification system becomes more questionable and loses trust.

There is a certificate authority certified by the government or ministry of education and affairs. However one cannot guarantee that the issued certificate is trustworthy and shows correct results which may contain both subjective as well as objective factors. The certificate issued by these authorities is in printed format. Using digitized certificates has been becoming easier to deal and operate with. However, Due to advancement of technology, counterfeiting of certificates has become a headache and worrying sign not only for students but also for the organization and university issuing it. This not only affects people who are provided with a certificate but also the organization issuing it as well. Also, the certification system becomes more questionable and loses trust.

Blockchain is still a new technology, it has taken the tech world by storm. It is expected to modify the internet and make it more decentralized. It actually is a set of blocks, linked in the form of a chain. Each block contains sets of transactions which includes cryptocurrency transactions and the transaction data gets replicated to all the nodes to form identical transaction details in the ledger. Blockchain transactions are cryptographically sealed which ensure security. Credentials are stored in the blocks with a specific hash. It provides more security as files are stored in the block and changes in it does not hamper as files are

distributed in the network and that maintains the integrity of our credentials. It is a digitally decentralized public ledger for transactions between parties. Each block in blockchain has strong cryptography involved in it, so data inside it is safe despite being a public ledger. Because of these functionalities, issuing certificates in blockchain is highly secured.

To re-establish trust, Students or the general public need a trusted system ensuring immutability and a tamper-proof certification. Keeping this in mind DCVS, a blockchain based certification system offering both issuers and users a trusted platform of certificate verification has been developed . The “Digital Certificate Verification System Using Blockchain” not only verifies but also stores the certificate. This system is confined to reducing the involvement of third parties and certifying authorities to verify the certificates. Using it, a company/verifier makes sure the certificate submitted by the student is genuine. Since, this system stores the certificate in encrypted form, the data within the certificate remains safe. It puts oneself in charge of their own credentials and helps them verify their credentials easily to the employers when necessary.

1.1 Problem Statement

In today’s context we generally have traditional ways of certification systems where printed format of document is used. The printed format of certificate is more likely to be tampered and edited. Thus, certificates are being less authenticated and are losing their value. Printed certificates may get lost and be found by unauthorised personnel. Here, the data is easily exposed to unauthorized people and can be misused. For a verification process it needs to be checked on authorizing agencies or organizations(issuer) manually increasing workload of both verifying and the organization. There is no any means by which one can differentiate whether the certificate is original or duplicate except checking it manually by eyes. This causes problems, since the honest ones who are real and work hard put them in the same ranked category as the ones who never work hard to enhance their academic performance. Certificate verification is a major concern in academic institutes and organizations. This is a lengthy process as well. There is always a time delay in the traditional method of verification of certificates

. Similarly, this process of certification and its verification becomes more problematic when used/applied for multiple organizations.

In the context of universities there have been several cases where fake certificates are used regularly but only very few are identified. Such problems show that there is no trusted tool

that can differentiate fake certificates. There have been several cases in Nepal where fake certificates are being used and only very few of them have been caught, that too, is through the informants. Thus, one can sense that a concrete tool to distinguish the real certificates from the fake ones is lacking.

1.2 Project Objectives

This project's main goal is to store certificates digitally and verify them to redefine trust for digital certificates. However, the project has other objectives too.

The key objectives of this projects can be listed as:

1. To develop a tamper-proof, immutable and trusted certification system that can be used by educational institutes to issue certificates
2. To make the remote issuance of certificate possible more efficiently and in lesser time
3. To promote use of digital forms of certificates
4. To analyze blockchain in context of its functionalities, applications and research trends
5. To present blockchain as a platform capable of storing data preventing data leaks and data losses

1.3 Significance of the Study

The findings of this study will greatly contribute to the benefit of society as the study contributes to creating a digital society by solving problems in digital ways. The main purpose of this study was to help and give benefit to the student, issuer and university and anyone that wants to verify the authenticity of any certificate in the system. The digital certificate verification system would improve the monitoring capacities of those who maintain the system. Furthermore, the issuer would be much guided when it comes to uploading and generating the certificates of the students, letting them get details and download their certificates. The digital certificate verification system has become quite popular in the past few years, as suggested by many recent studies. As the digitized certificate verification system is more secure and has more advantages over the traditional certificate verification system. On the other hand, it has become more profitable for the

organizations as it increases the accuracy of results and it minimizes the cost and time dramatically along with proper data security and authorization.

This System is focused on developing a platform which can provide data security and an easy certificate verification platform for universities and different organizations of the country. In today's context most of the foreign universities are focused on issuance and verification of certificates digitally by developing their own certificate verification system. Through this platform, this system will be trying to give all the rights to the students to maintain privacy of their own data or certificates. Using this platform certificates are stored securely on the network without worries about data theft and tampering i.e. making certificate storing and verifying easy, fast and secure.

2. Literature Study

Certificates are needed in every step of life. They are a source to prove the authenticity of one's expertise. In this competitive society, certificates are required in more ways than just to verify context. They are extremely vital to establish the authenticity of an applicant, whether for visa application, employment or education. Considering the importance of certificates, there have been many proposed methodologies and solutions to ensure safety of one's certificate, especially in digital form. Ensuring digital certificate security using blockchain is one of the solutions. Among many ongoing projects related to this solution, below mentioned are few of them,

1) "How to Time-Stamp a Digital Document"[1]

A paper published by Stuart Haber and W. Scott Stornetta in 1990 entitled "*How to Time-Stamp a Digital Document*" is the research paper proposing ideology on how digital data and documents can be preserved and secured from being faked, copied and tampered without owners authority. In a way we can say it theorized about enforcing copyright. Though Bitcoin by Satoshi Nakamoto is considered to be the first implementation of blockchain, this paper is the origin of Blockchain technology. In this paper researchers Stuart Authors have described how digital documents like research papers, articles and other papers can be safeguarded and copyrighted using time-stamp. Time stamp is a sequence of characters or encoded information identifying when a certain event occurred These character sequences are time of day, or seconds sometimes. i.e if a paper is uploaded, the time period when it was uploaded or issued will be the exact time-stamp. Their solution takes time-stamping as a security key which cannot be altered. The solution includes improvements on hash and digital signatures. Hash was improved by the use of a family of cryptographically secure collision-free hash functions. Another improvement was made by using digital signatures which is an algorithm scheme for a party, the signer, to tag messages in a way that uniquely identifies the signer. With a secure signature that was used, when the time stamping service (TSS) receives the hash value, it appends the date and time, then signs this compound document and sends it to the client. After checking the digital signatures, the client is assured that time stamping service actually did process the request, appended hash was correctly received and tha correct time is included. Time stamping service was basically included to service records at which date and time the document was received and retains the copy of document for safekeeping and whenever the integrity of the user's document is ever challenged, it can be compared with the copy stored by TSS.

Haber and Stornetta intended to introduce a mathematically sound and computationally practical solution to the time-stamping problem. A TSS can be made trusted using Collision Free Hash and Signature.

- **Hash** : Hash functions compress bit-strings of arbitrary length to bit strings of a fixed length *l*, and it is easy to pick a member of the family at random. It is computationally infeasible, given one of these functions,

  $$h : \{0,1\}^* \rightarrow \{0,1\}^l$$

  to find a pair of distinct strings x, x' satisfying $h(x) = h(x')$. Such a pair is called collision for h. The hash functions are one way functions and it is infeasible to compute another string x' given string x with $x \neq x'$ satisfying,

  $$h(x) \neq h(x'), \quad \text{for a randomly chosen } h.$$

  Hash function in the paper has been used to send hash value, $h(x) = y$ of

  document to TSS with y as time-stamping equivalent to time-stamping x .

- **Signature** : Signature scheme is an algorithm for a party, the signer, to tag messages in a way that uniquely identifies the signer. One way functions can be used to design a signature scheme satisfying the very strong notion of security that was first defined by Goldwasser, Micali, and Rivest. TSS receives a hash value, appends data and time, then signs the document and sends it to the client using signature. TSS assures client checking the signature, ensures hash was correctly received and correct time has been included.

Since, neither the signature nor hash functions could prevent issuing false time-stamping, a further mechanism was suggested. The legitimate time-stamp of a Algorithm A with document x, and timing information T in bit string can be written as,

$$c = A(x,T)$$

The above c is to be prevented from being forged to produce same time information T, and same certificate c later. For this two approach was proposed,

i) Linking : This approach is to constrain a centralized but possibly untrustworthy TSS to produce genuine time-stamps, in such a way that fake ones are difficult to produce. If bits from the previous sequence of client requests in signed certificates are included in present

then we know time-stamp occurred after these requests. Linking leaves no room to insert a new document with forged timing information to the service, making it ineligible.

Let **σ(.)** be procedure to sign document by TSS where, it issues signed and sequentially numbered time-stamp certificates. ***H***, be a hash function, ***l***-bit string ***y*** be time-stamping request and ***ID*** as client id number. Also, client request be, ***(γn , IDn)*** , in nth request. TSS

- *Sends signed certificate*, ***S = σ(Cn)*** *where,* ***Cn = (n, tn, IDn, yn; Ln)***
  Here, ***Ln*** is the linking information, coming from certificates previously issued.

  Lo = (t(n-1), ID(n-1), γ(n-1), H(n-1)).

- *TSS sends* ***ID(n+1)*** *for the next request*.

Client checks signature ***S***, and ***ID(n+1)*** on arrival. Timestamp of the document can be verified as ***(S, ID(n+1))*** for any collision between client and TSS. The challenger of time-stamp can also call ***ID(n+1)*** to produce their time-stamp, ***(S', ID(n+2))*** and check copy of ***yn*** in ***L(n+1)*** and authenticate by including ***H(Ln)*** of linking information to verify with ***L(n+1)*** .

The paper also suggests another variation of linking, in which each request is linked to next ***k*** requests.

- *Linking information,* ***Ln*** *will be*, ***Ln = [(t(n-k), ID(n-k), γ(n-k), H(L(n-k))) …(t(n-1), γ(n-1), H(L(n-1))]***.
- *Tss sends a client list* ***(ID(n+1), …, ID(n+k))*** *after next* ***k*** *requests have been processed.*

ii) <u>Distributed Trust</u> : The distributed trust approach is to distribute the required trust among the users of service. When a client needs to time-stamp, hash value ***y*** can be used as seed for a pseudorandom generator, whose output can be interpreted in a standard as a ***k-tuple*** of client ID numbers, ***G(y) = (ID1, ID2, ..., ID(k))*** . Client then sends a request ***(y, ID)*** to each client. Sending client receives a signed message, ***σj = (t, ID, y)*** . The final time-stamp of the client now consists, ***[(y, ID), (S1, …, S(k)]***. This scheme suggests the number of dishonest clients is always less than the majority of clients and suggests choosing seeds to be impossible to find.

This paper concludes by giving the idea that, if both the schemes/approaches be used

together can be a secure practical solution to time-stamping of digital documents. This way it gave rise to block chain technology

2) Digital Credentials Consortium (DCC)[2]

Digital Credentials Consortium (DCC) is an ongoing project being developed by a group of Universities including MIT, Harvard, Delft University of Technology, UC Berkeley, University of Toronto,TU Munich and others. It is a project creating a trusted, distributed, and shared infrastructure that will become the standard for issuing, storing, displaying, and verifying academic credentials digitally. The project especially focuses on University certificates and prevents them from being faked. The system being developed has the features of issuing/receiving credentials, store/manage/retrieve credential, share credentials, verify credentials, reissue credentials and onboard issuer. The “issue credentials” is initiated by the issuer which sends invitations to the students to receive digital credentials. The students are presented with the credentials they are eligible to receive (determined by the students achievement record). The students make their selection and provide the identifiers with which they want the credential to be associated. The issuer generates the credentials and sends it to the students. Students store and manage their credentials in a wallet that is on their device or in a website with storage and management features that is either hosted by universities, service providers or proficient users themselves. The system allows students to share their credentials in different contexts, in which the student wants choice over which credentials to reveal. They can share their credentials on social networking sites as well as with potential employers or other educational institutes. The verification process is initiated after a student shares their credentials with a relying party that then uses a standard complaint means of verification. A credential issued by the DCC will contain pointers to the source of trust approved by the consortium to be used during verification. Cryptographic protocol enforces that credentials may not be falsely attributed to a source of trust that is not part of it. It features reissuance of credentials which may be required for a variety of reasons, such as name change, a typo or loss of the original credentials or loss of cryptographic keys. Tasks associated with reissuance of credentials is similar to that of original issuance, except there is an additional step of revoking of original credentials. Onboard issuing process establishes the issuers credentials signing keys and synchronizes the issuers identifier and key information to a registry for use during credentials authenticity checks. DCC supports multiple issuers, different types of credentials and data standards. Open Badges, Schema.org, Embedded for Language Model (ELMo) are well known credentials. Also, it will be accepted

internationally. DCC works closely with W3C Verifiable Credential standards which is a tamper-evident credential that has authorship that can be cryptographically verified.

Being inspired by the project DCVS has been developed hoping it solves existing problems and issues in the certification and verification system of Universities in our nation. This system makes sure a faulty certification system, lengthy and tedious verification system will be replaced, preventing degradation of the reputation of respective University issued certificates.

3) Blockcerts[4]

Blockcerts is an open standard initially based on prototypes developed at the "*MIT Media Lab*"[3]and "*Learning Machine.*" It is now being developed further by "*Hyland Credentials*". Blockcerts helps in creating, issuing, viewing and verifying blockchain based certificates. The digital records are registered on a blockchain and are signed cryptographically. They aimed to enable a wave of innovation that gives the individual capacity to possess and share their own official records. Blockcerts is an open source project encouraging multiple collaborators. It represents an important step toward an open digital credentialing ecosystem and paves a path forward for further research in the field. The Blockcerts was initially built on bitcoin blockchain and later was expanded to ethereum blockchain. They have been working to deploy it across public chains and to expand it in private chains as well. The main system components of Blockcerts are issuers (school or university), recipient (students) and verifier (employer). The issuer issues the credentials and sends the recipient an invitation to receive blockchain credentials. The recipient accepts the invitation and sends the issuer response with their blockchain address. Issuer hashes the credentials onto the blockchain and the issuers send the students the blockchain credentials. Using the blockchain credentials provided by the issuer, students can have a look at their credentials and they can verify it. In case of verification by the third party, students need to send their blockchain credentials and the third party will check for the credentials on the blockchain.

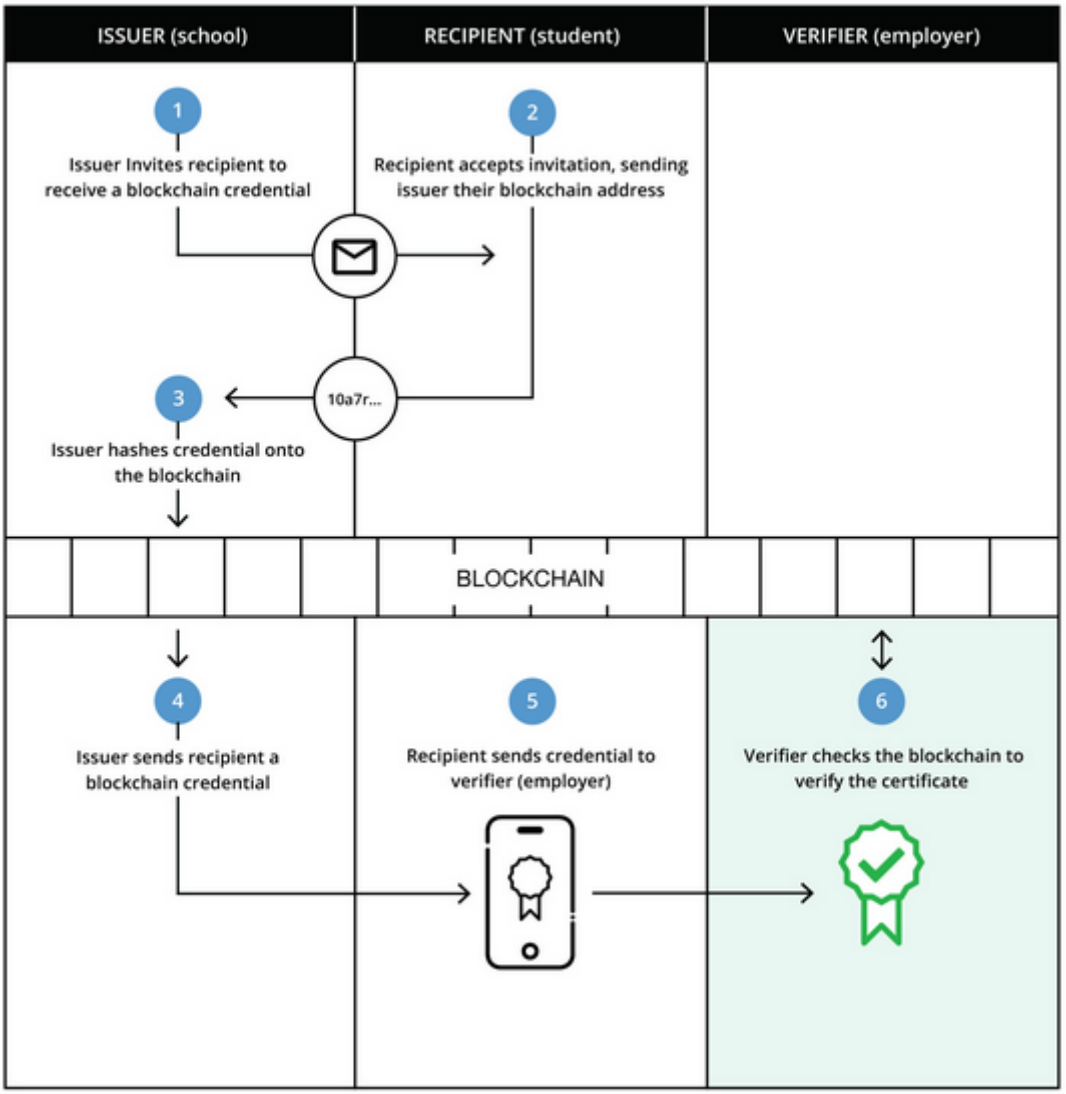

Figure 2.1. Blockcerts workflow (https://www.blockcerts.org/guide/)

The Blockcerts are aligned with various standards like, IMS Open Badges, W3C verifiable Claims, W3C Linked Data Signatures, and W3C / Rebooting Web of Trust Decentralized Identifiers. Its open source confirms the consideration of above standards. This variable system generates solutions for subtle difficulties arising in a real-world system including,

revocation of certificates, W3C / Rebooting Web of Trust Decentralized Identifiers and other lifecycle concerns, including key management.

4) OpenCerts

OpenCerts is an open and online certificate verifier developed by "*GovTech of Singapore",* Ministry of education (MOE) and *"Ngee Ann Polytechnic and SkillsFuture Singapore"*. Opencerts has been developed for creating, verifying, checking, and viewing purposes of digital credentials. OpenCerts offer an easy and reliable way to issue and validate academic

certificates that are tamper-resistant. OpenCerts aims to allow individuals to easily access and retrieve their digital certificate from a single location. Most of the universities of Singapore have implemented this system and students have been provided with blockchain-based digital certificates since 2019. In OpenCerts, data doesn't have to be stored in one location as each certificate's details are kept in an individual file. All the verification processes are done through Ethereum. When a certificate is created on OpenCerts, a unique digital code is tagged to it and this code together with information on the certificate is stored on Ethereum blockchain. The certificates published on a public ledger like Ethereum are permanent and unalterable. A cryptographic key is appended onto the certificate and sent to the individual recipient. The certificates are essentially time-stamped and assigned with cryptographic features (mainly signatures) and if any kind of modification is done to the content of the certificates, the certificates become invalidated. Once the certificates get issued to the students, they own the file but won't be able to do any kind of modification to the content of the issued file as it would void the time-stamp and cryptographic signature of the certificate. The students will be able to send their certificates to the potential employer. When an employer receives the digital certificates, they can verify the authenticity of the certificate against the public ledger using the certificate store interface. At the time of verification, certificate data is checked against its cryptographic key and time-stamp for signs of tampering and against the code on blockchain for validity.

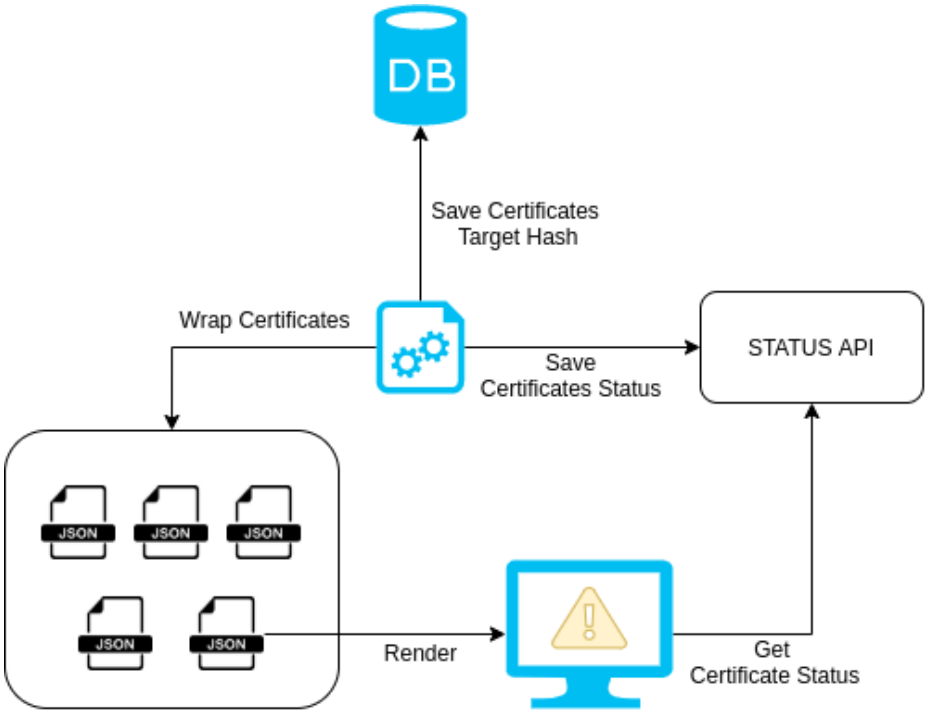

Figure 2.2. OpenCerts workflow ,(https://docs.opencerts.io/docs/multi-issuer)

3. Methodology

3.1 Context Diagram

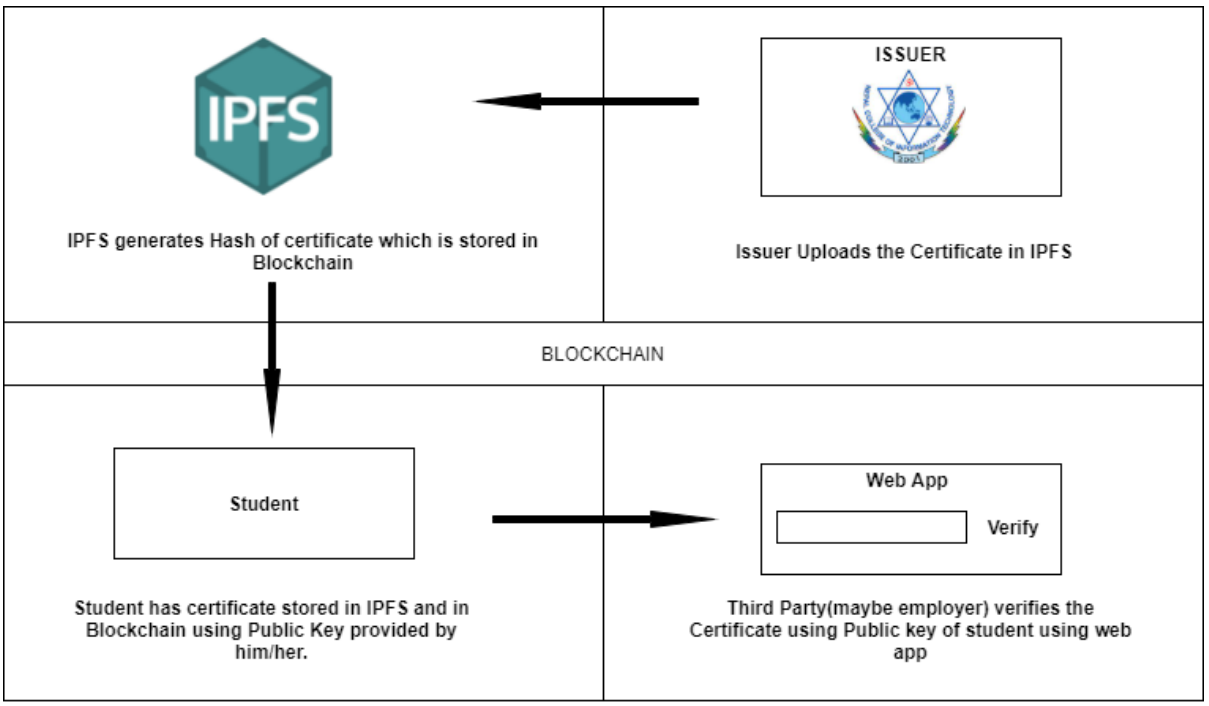

Figure 3.1. Context Diagram

Above diagram represents the overall architecture of the “Digital Certificate Verification System”. Issuer uploads the certificates in IPFS. Students In the modern global context everything is digitizing. It’s the time certificates as well as other important documents too need to be digitized. have to provide their Public Key during signup. By the use of smart contracts, data of students including public key and certificate hash are stored. Now, Students can download their certificate file using their key and also they may share it with a third party(maybe employer). Employers can verify the originality of a certificate using the web app which allows the employer to generate a hash of the file and compare it with the hash stored in the database. However, students have to be logged in using the same key via wallet to download. This prevents verifiers from downloading certificates without permission despite having a Student's public key.

3.1.1 Issuer:

Issuers or Admins are authorised actors who issue and upload the certificates in the System. They send an invitation to students to provide their credentials like their Public address. Issuer is considered as the college admin who has control over students certificate data and can take responsibilities over deployment and certification. In the issuer module, following processes are carried out:

1. Issuer is set as the contract admin. System has Address of the issuer hard coded as admin and set modifier 'onlyAdmin' so that only admin can take control over the smart contract deployment and certificate issuance.

   Here the issuer's Address is : ***0x5CA9499c3Bf66F0DfcA00c3798B8c4E40D2E4ce1***

   ```
   constructor() {
       Admin = 0x5CA9499c3Bf66F0DfcA00c3798B8c4E40D2E4ce1;
   }
   //Modifier onlyAdmin
   modifier onlyAdmin{
      require(msg.sender==Admin);
      _;
   }
   ```

2. Now the Issuer can upload the certificate in .pdf format mapped along with the Student's address through the Issuer module. i.e

   Upload → RamCertificate.pdf
   Student's Public Key : *0x92fab661Cf913be74B5098597D2DbED2A8618B00*

3. Then the system generates File Hash of the certificate using SHA256 and stores the certificate in IPFS . File Hash , IPFS hash and timestamp are shown as details..

3.1.2 Students:

Students are the owners of the certificate issued to them by the University. Students should Sign Up to the system providing their Public Address, so that the issuer can upload his/her certificate in the blockchain network this is done outside the system. In Students module , following work is done :

1. The issuer maps the certificate with the public address given by the students.
2. Now the students can download their certificates by signing up and logging-in in the system with correct credentials.
3. Download is only possible if the current Wallet Address of the student matches the address to which the certificate is mapped after signing in to the system.
4. Students can share their certificates to third parties or with anyone else according to their interest for verification.

3.1.3 Third Party/Verifier:

There is no need of carrying physical documents to the third party, may be an employer who wishes to check the authenticity and validation of the certificates . For verification, users/students may send the pdf file to the employer or third party for verification. In the verification module following processes are carried out:

1. Third Party or Verifier can verify certificate independently . First they upload the certificate provided by the students to the system along with the Public Address given by the student.
2. Hash Of uploaded certificate is generated :

   Generated file Hash :

   576630746b627f139ebe450d35e69c6bd5dba9813b44225857ed44f3ede09160

3. Hash of certificate mapped to the Public address is generated (original document hash uploaded to the blockchain network):

   Mapped hash to the Public address:

   576630746b627f139ebe450d35e69c6bd5dba9813b44225857ed44f3ede09160

4. Comparison of hashes shows the integrity of the document. If hashes matched, the document is verified as original, else not.

3.2 UML Diagram:

3.2.1 Use Case Diagram:

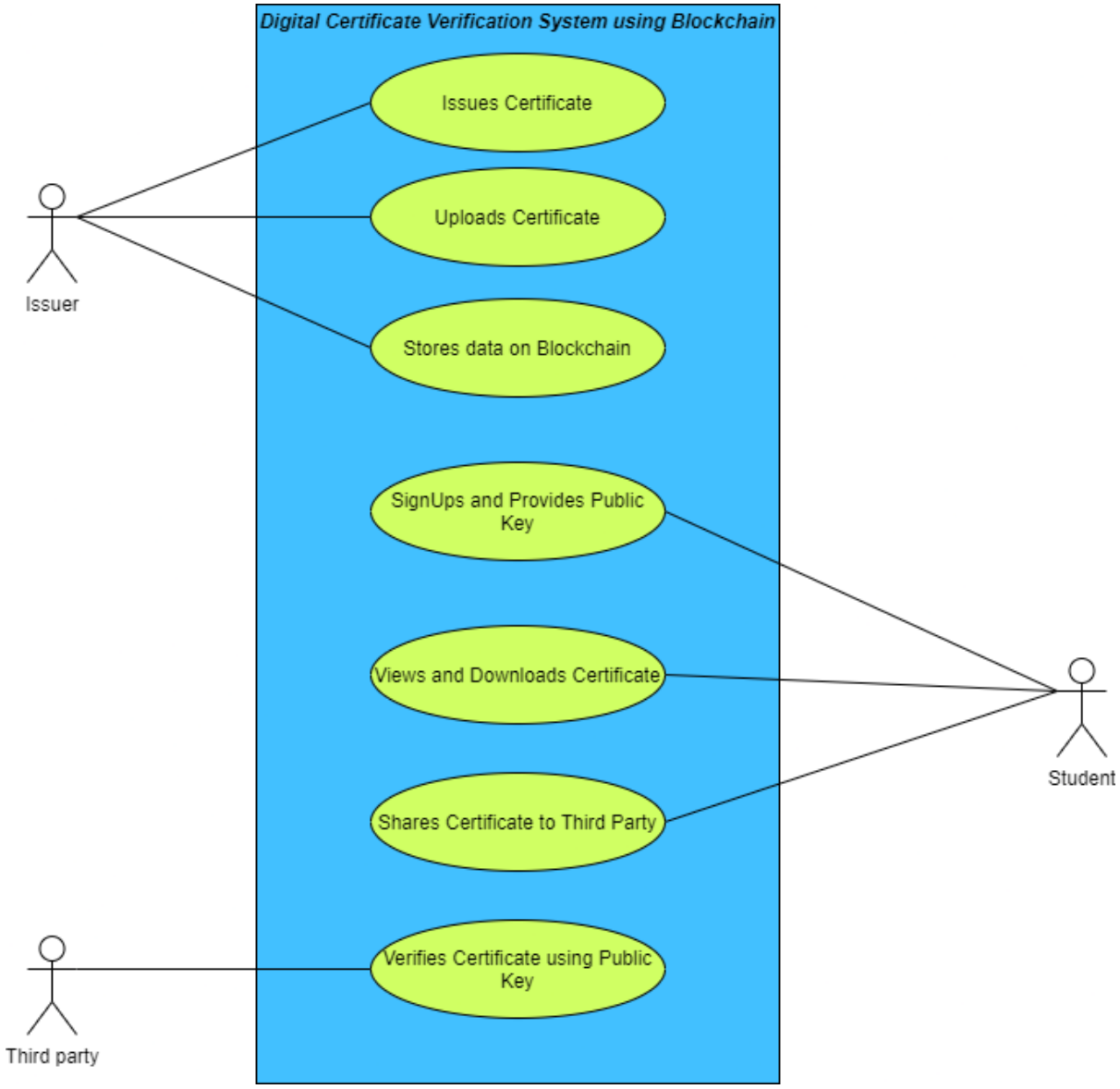

Figure 3.2.1. System Use-Case Diagram

Issuer is an organization (institution) providing the certificates. After issuing the certificates to students, it uploads the certificate in pdf form to a decentralized storage network(IPFS). Students can now download certificates using the key. Students also may apply (share certificate) to a third party(employer). The Third Party verifies the certificate in the Digital Certificate Verification System using certificate hash and Public Key provided by the student.

3.2.2 Activity Diagram

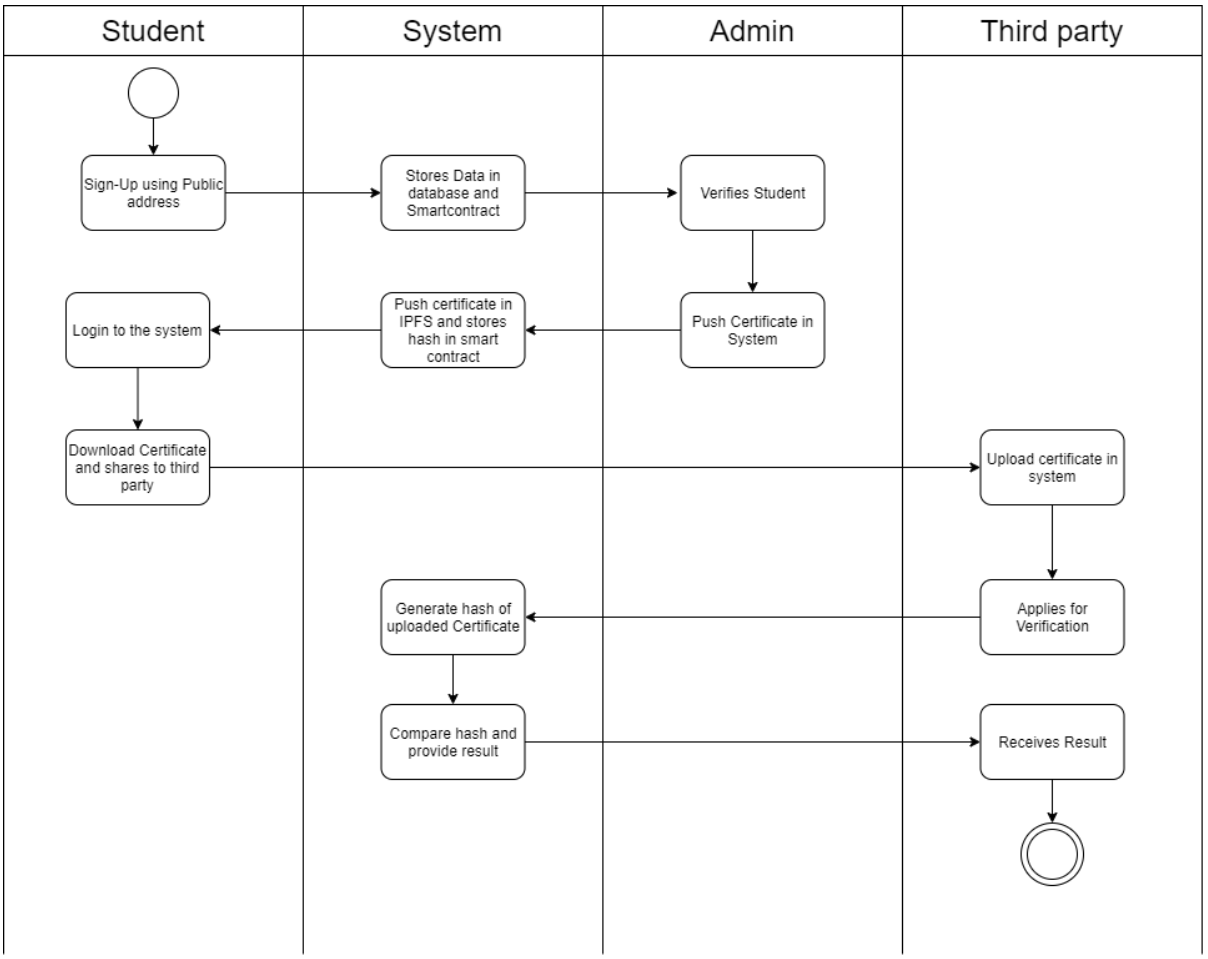

Figure 3.2.2. Activity diagram of the DCVS system

Considering the smart contract already deployed by Issuer, the activity diagram can be summarized as above. Here the students sign up with their details and public address. System stores the student's details in the database and Smart contract. Issuer verifies the students sign up details and issuer pushes the certificates in the system and system stores the certificates in IPFS and Smart Contract. Students need to log-in in the system to download and to get the details of the certificate from the system. Students can also share their certificates to the third party with whom they wish. In order to verify the certificates, a third party uploads the certificates in the system and applies for verification. System generates the hash of the certificates and compares it with the hash of the certificates in the Smart Contract and provides the result to the third party whether the certificates are original or counterfeit.

3.2.3 ER Diagram

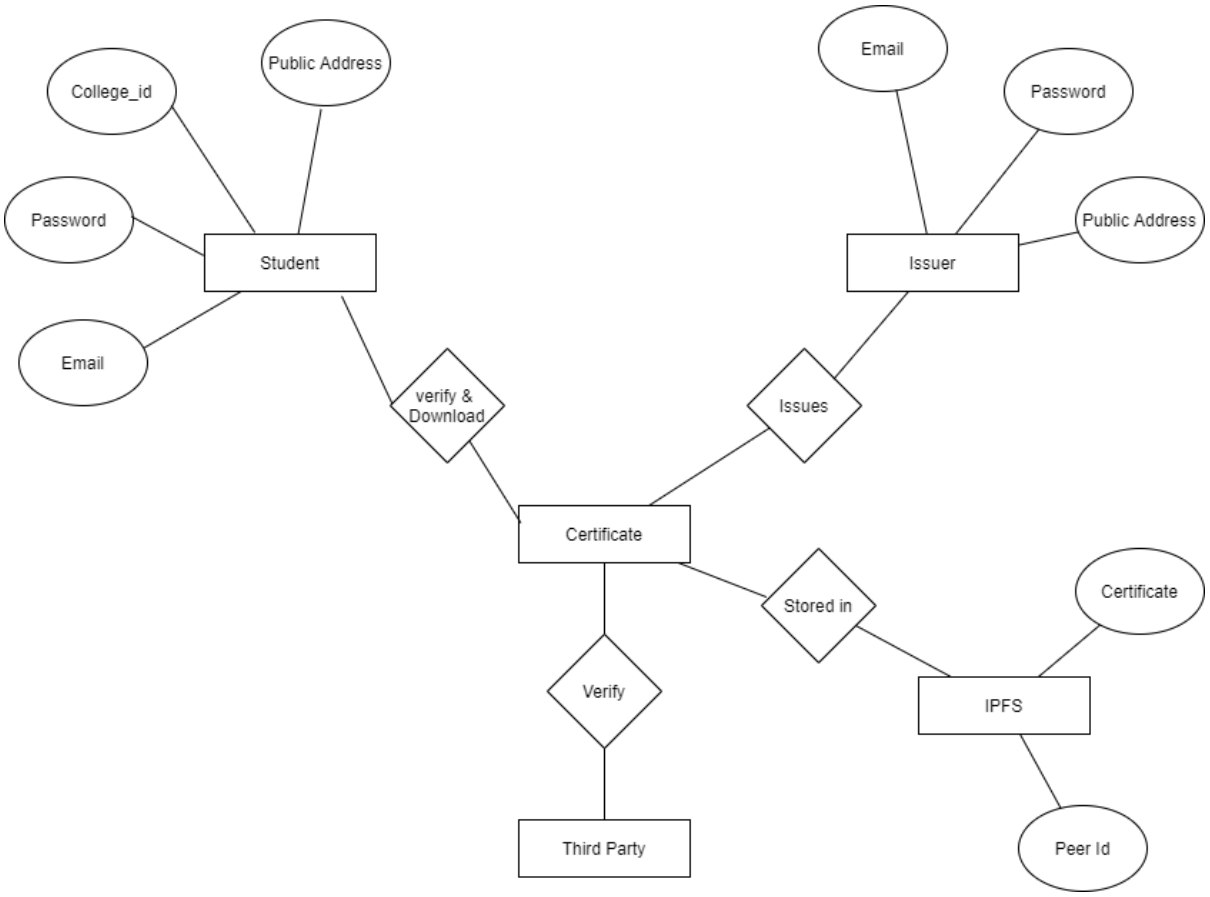

Figure 3.2.3. ER diagram

The main entities of the systems are Students, Issuers and Third Party as in figure above. The Issuer issues the certificates which are stored in the IPFS with certificate details, certificate hash and peer id. Students can verify and download their certificate stored in IPFS. Third parties can verify the certificate provided by the students.

3.2.4 Class Diagram

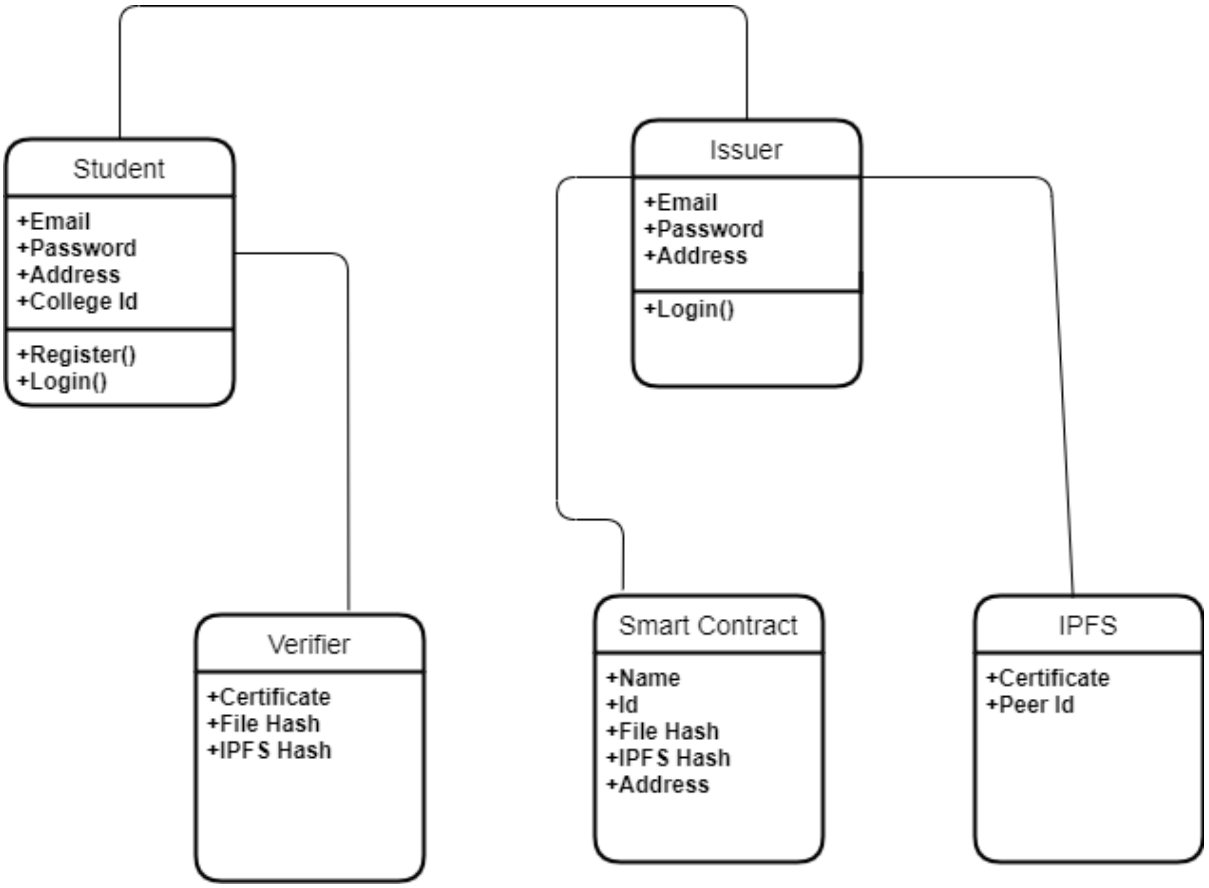

Figure 3.2.4. Class Diagram

The system has the class Students, Issuers, Verifier, Smart Contract and IPFS. Students with required info, registers and then log-in into the system. The Issuer logs into the system with and stores students' certificates in IPFS with its peer id and in Smart Contract. Verifier checks for validation of certificates provided by the students.

3.3 Architecture Overview:

Digital Certificate Verification System ensures certificate integrity and helps Issuer Organizations, Students, Verifier and every other parties related to the certificate remain away from forgery and frauds. The Issuer issues certificate, Student Downloads or Verifies and Verifier/3rd Party verifies certificate provided by Students. This general and simple workflow is possible due to communication between various components within the system.

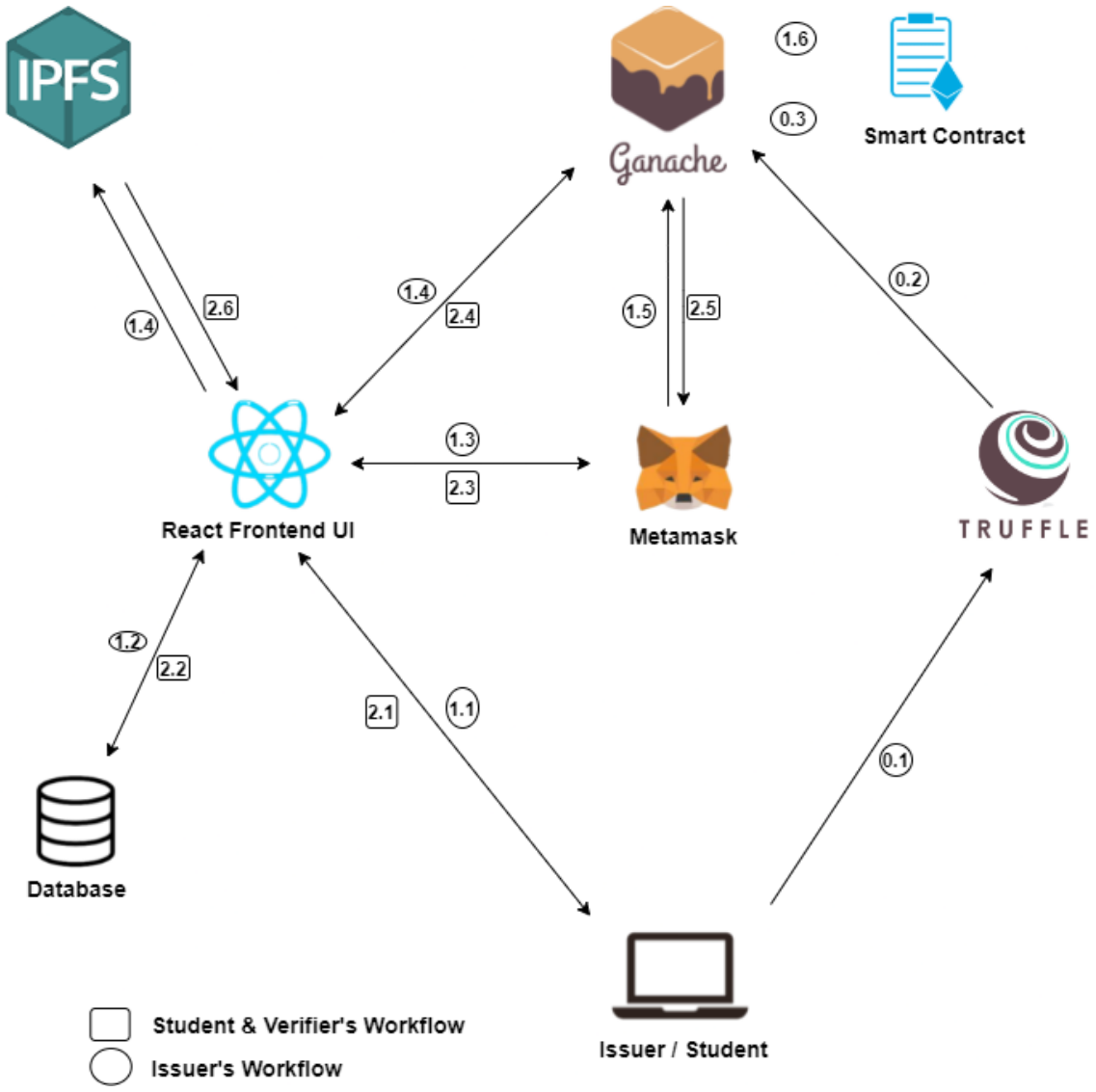

Figure 3.3. Architecture of system

The figure Fig: 9 shows the overall architecture of the DCVS system. In the system components like IPFS, Metamask, Ganache, Truffle, Database, SmartContract and Frontend UI communicate depending upon the action by Issuer or Students.

The Architecture flow with respect to Issuer to deploy Smart Contract is,

- ➔ 0.1, Issuer develops Smart Contract with Truffle Suite
- ➔ 0.2, With the help of Truffle Suite, Smart Contract is migrated to Ganache
- ➔ 0.3, Finally Smart Contract is deployed to blockchain network

Architecture flow with respect to Issuer to store Issued Certificate in system is,

- ➔ 1.1, React Frontend UI receives data from Issuer for Signing In
- ➔ 1.2, Sign In data is authenticated from Database and signed in if matches
- ➔ 1.3, React Frontend connects to Metamask Wallet
- ➔ 1.4, File is sent to IPFS to store a file which sends CID back. Now the CID along with File Hash is sent to Ganache.
- ➔ 1.5, Ganache confirms the transaction via Metamask
- ➔ 1.6, Intended datas are stored in Smart Contract on respective Public Address of Student

The task at 1.6 in Fig 3.234344 indicates the completion of Issuing certificate by Issuer. Prior to Issuer uploading certificate, Students have to Sign Up providing the Public Address on which certificate is to be issued.

Architecture Flow with respect to Students for Downloading certificate is,

- ➔ 2.1, Student provides Log In data,
- ➔ 2.2, React verifies if it is valid credentials
- ➔ 2.3, Metamask connection is established
- ➔ 2.4, UI requests Ganache to send data stored on respective Public Address
- ➔ 2.5, Ganache confirms transaction via Metamask and sends data
- ➔ 2.6, Considering IPFS Hash in Smart Contract, IPFS is requested to provide the certificate.

Now, Students can download their certificate.

Architecture Flow with respect to Student & Third Party for Certificate Verification,

For verification of Certificate signing in is not required. Also, Third Party may or may not Sign Up. However, a certificate to be verified along with Public Address is to be

provided. After receiving this data React UI extracts file hash stored in smart contract as steps briefed above from 2.1 to 2.5 and hashes it using SHA hashing algorithm. Finally, the hash of the file uploaded and the hash stored in the smart contract is compared. If they match, then the certificate is verified and if not the certificate is said to be tampered.

3.4 Tools And Technology Used:

3.4.1 Blockchain:

Blockchain is a decentralized, distributed and digital ledger consisting of records called blocks. The blockchain data structure is a time-stamped list of blocks, which records and aggregates data that have ever occurred within the blockchain network. The blockchain provides an immutable data storage container, which only allows inserting data without updating or deleting any existing data on the blockchain to prevent tampering. Every data recorded in the blockchain can be separately verified without any interruption by using its hash value, since it is open, can be publicly verified and the data once entered cannot be altered which helps in preventing forgery and misuse. In blockchain every block of data is linked to the previous block with the help of the hash value of the preceding block. The only genesis block which is the initial block does not have a hash of the preceding block. Blockchain has widely been popularized by crypto currencies like Bitcoin and people have misunderstood that Blockchain is Bitcoin. The fact is Bitcoin is only a use case of Blockchain. Blockchain is far more beyond crypto currencies and is expected to replace WEB at present with WEB 3.0 . It could be of 3 types based on permission to access the network.

- **Public Blockchain,** Public blockchain is an open network which can be accessed by every general public. Ethereum and Bitcoin are the best examples of it. Anyone participating in the network can read, write, and audit the activities.
- **Private Blockchain,** Private blockchain are generally used within an organization preventing others from accessing the network. Permissions are kept centralized. It lets network administrators (organizations) design and change network rules, architecture based on their necessity. Ripple, Multichain are examples of it.
- **Consortium Blockchain,** Consortium Blockchain are designed to have both Public and Private networks. Here, instead of a single organization, multiple organizations collaborate to take part in the network. It is a permissioned blockchain. Rubix and Hyperledger Fabric are some of its examples.

3.4.2 Ethereum:

Ethereum is an open-source, decentralized public blockchain with smart contract features. It was proposed by Vitalik Buterin in 2013 and was released initially in 2015. However, a stable version was released on 5th August 2021. It is a public blockchain network that emphasizes on running any kind of programming code of DApps. It helps in sharing information across the globe that cannot be manipulated or altered. The blockchain developers can build, deploy, monetize, and use applications on the platform, and use its Ether as payment. Ether is its native decentralized currency. It was created to assist developers to build and publish smart contract and decentralized applications that can be used without the risks of downtime, fraud, or interference from a third party. Ethereum is also called "the world's programmable blockchain". It distinguishes itself from Bitcoin as a programmable public network that serves as an decentralized app, marketplace for financial services, and games, all of which can be paid for in Ether cryptocurrency and are safe from all kinds of frauds and theft.

Ethereum is a distributed public blockchain network similar to that of bitcoin. There are some significant technical differences between them and out of them the most important distinction between Bitcoin and Ethereum differ substantially in purpose and capability. Bitcoin offers a peer to peer electronic cash system that enables online Bitcoin payments. Bitcoin blockchain tracks owners' claim of digital currency (bitcoins), Ethereum focuses on running the program code of any decentralized application. In the Ethereum blockchain, users work to earn Ether, a crypto token that fuels the network. Beyond a tradeable cryptocurrency, test or mainnet Ether is also used by application developers to pay for transaction fees and services on the Ethereum network.

3.4.3 Smart Contract and Solidity

Smart Contract is an agreement between two parties which is self enforcing and is deployed in a blockchain network. It is a code generally written using Solidity , Python , C# etc which run on top of blockchain networks. These smart contracts are known as a set of rules that govern the digital assets of the system. Their main purpose is to remove centralized entities to ensure trust and agreement among anonymous parties.

Smart Contract is a centerpiece of the project and main thrust of the Ethereum blockchain. Smart Contract is an agreement between two parties which is self enforcing and is deployed in a blockchain network. It is a code generally written using Solidity , Python , C# etc which

run on top of blockchain networks. These smart contracts are known as a set of rules that govern the digital assets of the system. Their main purpose is to remove centralized entities to ensure trust and agreement among anonymous parties.nSmart contract works with the application-specific semantics and constraints of the transaction and verifies, validates, and executes them. Since smart contracts are deployed on the blockchain network, the smart contract grips the immutable records. Smart contracts once deployed to the network cannot be changed. Smart contracts can store variables in it called state variables. We could redeem how these variables change over the blocks. Contract in the Ethereum blockchain contains a pragma directive, name of the contract, data or the state variable defining the state of contract, and a collection of functions that carry out the intent of a smart contract. The Ethereum project introduced the idea of decoupling the contract layer from the blockchain layer and by decoupling the smart contract layer from the blockchain layer. Ethereum aims to provide a more flexible and scalable development environment than that of the Bitcoin blockchain. This piece of code controls all the digital assets implementing arbitrary rules on the top of the blockchain network.

Solidity is an object oriented and high level programming language which is used to develop Smart Contracts. Solidity is influenced by Python, C++ and JavaScript and is designed to target Ethereum Virtual Machine (EVM). Solidity supports inheritance, libraries and complex user-defined types and is statically typed.

3.4.4 Truffle Suite:

Truffle suite is a development environment or development tool for testing framework and asset pipeline for blockchains using the Ethereum Virtual Machine (EVM), that is used for built-in Smart Contracts compilation, linking, deployment and binary management. With the help of Truffle, Smart Contracts can be compiled and deployed, inject them into web apps and develop front-end for DApps. Truffle suite provides automated contract testing for rapid development and scriptable, extensible deployment and migration framework. It manages networks for deploying to any number of private and public networks. It gives interactive consoles for direct contract communication. It is provided with a configurable build pipeline with support for tight integration. Truffle Suite mainly comprises three core elements. They are Truffle, Ganache and Drizzle. Drizzle is an assortment of front-end libraries that offers useful components for developing useful web applications that can seamlessly connect with Smart Contracts.

3.4.5 Ganache:

Ganache is a personal blockchain for rapid ethereum and Corda distributed application development. It is provided by the Truffle Suite. Ganache can be used across the entire development cycle; enabling us to develop, deploy, and test system DApps in a safe and deterministic environment. Ganache comes in two flavours: UI and CLI. Ganache UI is a desktop application supporting both Ethereum and Corda technology. The command-line tool, ganache-cli, is available for Ethereum development. Ganache can be used for setting up personal Ethereum Blockchain for testing Solidity Smart Contract. It provides more features compared to remix (platform for developing Smart Contract). Ganache provides features like; displaying blockchain log output, provides advanced mining control, built-in block explorer, Ethereum blockchain environment. Also, it provides 15 ethereum accounts with 100 Eth on each address. This 100 Eth can be used to pay while transacting in the local network.

3.4.6 Web3 JS:

Web3 is a collection of libraries which allows developers to interact with a local and/or remote ethereum node using WebSocket, IPC and HTTP. Web JS is a Ethereum Javascript API which connects to Generic JSON-RPC. It has libraries with many modules containing functionality for the ethereum ecosystem. web3-eth, web3-shh, web3-utils, web3-bzz are libraries of web3. Web3 is also available for other languages like Python, Scala, Java, PHP etc. Frontend of Dapp communicates with blockchain networks via this API.

3.4.7 Metamask Wallet:

Metamask is a cryptocurrency wallet that interacts with Ethereum Blockchain Network. It's a browser extension or application that lets users connect to a network via web browsers. It also stores accounts and amounts and has a mobile app developed as well. Metamask is a well known wallet used for storing Ethereum or Ethereum based ERC-20 Tokens. Through Metamask ,transactions (Send and receive) of ethereum based tokens and cryptocurrencies are possible. It was developed by ConsenSys Software Inc. and it was initially released in 2016.

3.4.8 SHA-256:

Figure 3.4.8. SHA-256 Encryption

The SHA (Secure Hash Algorithm) is one of a number of cryptographic hash functions. A cryptographic hash or called digest is a kind of signature for data. SHA256 is a hashing algorithm created by the National Security Agency and is one of the strongest hash functions available. SHA-256 algorithm generates an almost unique, fixed size 256-bit (32-byte) hash. SHA-256 is one of the successor hash functions to SHA-1 (collectively referred to as SHA-2) and one of the strongest hash functions available. Hash is a one way function i.e, once hashed it cannot be decrypted back. This makes it suitable for password validation, anti-tamper, and digital signatures. The features provided by SHA-256 algorithm are: message length, digest length and irreversibility. Message on plain text when feeded to Hash Algorithm, gives out cryptographically hashed value as in fig 3.4.8 above.

3.4.9 Digital Signature

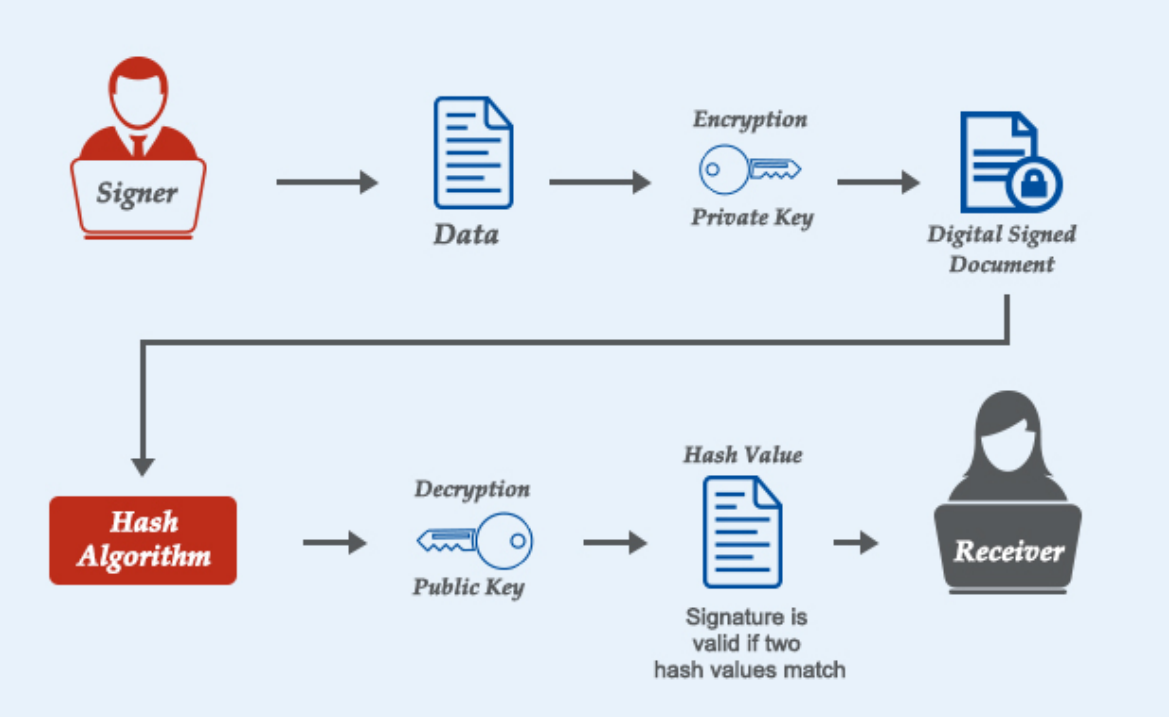

Figure 3.4.9. Working Principle of digital signature

Digital signature is a cryptographic method for verifying authenticity of digital data. It along with authentication validates data integrity. It is the digital equivalent of a handwritten signature. Digital Signatures are the building blocks of blockchain in which they are primarily used to verify the authenticity of transactions. When users submit transactions, they must prove to every node in the system that they are authorized to use those funds while preventing others from using those funds. Every node in the network verifies the condition of the submitted transaction and checks all the nodes to agree on the correct state. Digital Signature works on the principle of mutually authentication of two cryptographic key pairs, that is public key and private key. These public keys and private keys can be generated by mathematical algorithms with the help of public key algorithms. Both the public keys and private keys can be generated from one another as they are related mathematically. Signers sign the digital message with the help of their private keys. A person can encrypt their signature related digital data with the help of their private keys. The private keys should not be shared with anyone and should always be kept with the person who wants to create the digital signatures. The only way to decrypt the data is the public key of the signers. Signers can share their public key with anyone they want to. Recipients can use signer's public key to decrypt the data and verify the authentication of signatures of Signers. RSA

(Rivest-Shamir-Adleman) , DSA (Digital Signature Algorithm), ECDSA(Elliptic Curve DSA) etc are some widely used digital signature algorithms.

3.5 Data Storage

3.5.1 IPFS:

Interplanetary file system(IPFS) is a protocol that stores and shares the file in a distributed file system using peer to peer network connection. IPFS was initially designed by Juan Benet, and is available as an open-source project developed with help from the community. Its file extension is ipfs.io.

In general it is a protocol that stores and shares the file in a distributed file system using peer to peer network connection..

Figure 3.5.1. Distributed file storage through IPFS

As shown in fig 3.5.1 IPFS stores the data by dividing it into small chunks and storing them into different nodes. A block of IPFS can only contain 256 Kb of file so the files are divided into multiple chunks of 256 Kb and are stored. As shown above, IPFS divides files into 4 small chunks and stores them into 4 different nodes called Links/0, Links/1, Links/2, Links/3. All the nodes have their respective addresses and these nodes can be located in different parts of the world. Thus to receive a file, CID stores the address of all the nodes that contain chunks of file and calls them to retain the file. CIDs are labels used to point material in IPFS. When any data is stored in IPFS it chunks if needed, stored in nodes and returns CID, which can be used to retain certificates.

3.5.2 MongoDB

MongoDB is an open source document database and NoSQL database. It is used to store data in a key-value pair. It's working is based on the concept of document and collection. MongoDB uses JSON-like documents with optimal and defined schemas which makes MongoDB so scalable as well as flexible. NoSQL database means a database which does not use query languages and schemas is employed for managing the massive collection of unstructured data and when our data is not piled up in tabular format or relation like that in relational database. MongoDB provides high performance, scalability as well as availability to manage the database. MongoDB is beneficial for highly elastic scalability, and is valuable and reliable for big data and is cheaper and follows agile data models.

3.6 NodeJS and React:

NodeJS is an open source, cross platform, back-end JavaScript runtime environment that runs on V8 engine and executes JavaScript codes outside a web browser. NodeJS allows developers to run JavaScript on the server. NodeJS eliminates the waiting for execution, and simply continues with the next request execution. NodeJS runs single-threaded event, non-blocking execution and asynchronous programming where sequential execution is not carried out, which is very memory efficient.

React is a very popular free and open source front-end JavaScript library for building user interface and UI components. React is used for webapp development, frontend design and mobile app development. React library is flexible and allows developers to compose complex UIs from small and isolated pieces of code called components(class and functional).

Tools and Dependencies and Their Versions

Table 1. Tools and dependencies and their versions

S.N	Dependencies/Tools	Version	Used For
1	Solidity	>0.6.10	Smart Contract
2	Hashing algorithm	SHA-256	Data Encryption
3	MongoDB	2.0.12	Database
4	React	17.0.2	Frontend
5	NodeJS	14.17.5	Programming Language
6	Truffle	5.4.9	Smart Contract Development Framework
7	Ganache	2.5.4.0	Local Blockchain
8	IPFS	0.9.1	File Storage
9	Metamask	10.1.1	Wallet
10	Web3 JS	1.5.2	API

4. Application Areas

4.1 Certificate Storage

The traditional way of storing our certificates is not that good enough to prevent our certificates from getting damaged or tampered or being stolen. DCVS provides a platform for storing our certificates where it neither gets tampered nor has a chance of getting damaged as certificates are stored in a decentralised network- IPFS. Thus, DCVS can be used in every area requiring certificate storage.

4.2 Certificate Verification

As certificates are used in every organization and every field they need to be verified if someone has tampered. This system not only stores but verifies the data making certificate verification isan another application of it.

4.3 Data and File Storage

DCVS both stores and verifies the certificate. It preserves the concept of data integrity. It accepts any .pdf data, so it can also be used in storage of data and files. Storing data with this system lets you verify the data if needed.

Figure 5.1. Gantt-Chart

5. Conclusion

In the modern global context everything is digitizing. It's the time certificate as well as other important documents that need to be digitized. The digitized certificate with cryptography makes the certificate safe and it being stored in modern tech decentralized databases prevents its losses. According to literature studies above, certificates stored in decentralized databases and blockchain are more secure than traditional techniques. The sharing and verification becomes much quicker and they can be stored for a longer period of time. Evolvement of technologies and the internet has also many sorts of drawbacks like data loss and hacking may cause problems while storing data in the internet digitally but the features of blockchain like tamper proof, robust and immutable assets are of great value for storing certificates digitally in blockchain decentralized networks. As the perks of blockchain have been increasing heavily these days, it has overshadowed the cons of storing certificates digitally. It completely eliminates the role of third parties for verification purposes and issuance.

Therefore, with this system plans to introduce a norm where all the credentials handling is done digitally rather than the traditional way and inspire not only the university as well as government agencies to follow the digital certificate storage and verification.

6. Limitations and Future Works

DCVS have tried to eliminate the limitations as much as possible and tried to develop this application as a suitable solution to the currently existing certification and verification process of certificates. However, there are some limitations that still prevail and will be worked on in the future. Some considerable limitations of DCVS are :

1. System needs a working and stable network to function properly.
2. System application requires a good internet connection to operate properly. Some features like making transactions and fetching and dumping data to ipfs may not work properly in slow internet connection.
3. Since the system uses a decentralized network, availability may be an issue. They may not be available all the time.
4. It has been assumed a public ledger containing the issuer and public key mapping that is publicly available and maintained with high security and reliability. This might not seem a friendly approach currently but as the tradition of digital certificates will evolve these sorts of practices will also be a general thing.
5. Currently the system works on a public blockchain network but will be working on deploying the system in a private blockchain network in future.

7. Bibliography

[1] Stuart Haber, W. Scott Stornetta, “How to timestamp a Digital Document", *Advances in Cryptology* - CRYPT0 ‘90, LNCS 537, pp. 437-455, 1991. 0 *Springer-Verlag Berlin Heidelberg 1991*

[2] MIT Media Lab,Digital Credential Consortium(DCC),“Building Digital Credential InfrastructureFuture”,*https://digitalcredentials.mit.edu/wp-content/uploads/2020/02/white-paper-building-digital-credential-infrastructure-future.pdf*

[3] Philipp Schmidt, Blockcerts-An Open Infrastructure for Academic Credentials on the Blockchains,March2021,[Online].*https://medium.com/mit-media-lab/blockcerts-an-Openinfrastructure-for-academic-credentials-on-the-blockchain-899a6b880b2f*

[4] Blockcerts, [Online], Available: *https://www.blockcerts.org/*

[5] OpenCerts, [Online], Available: *https://www.opencerts.io/*

[6] Guendalina Capece *, Nathan Levialdi Ghiron and Francesco Pasquale, “Blockchain Technology: Redefining Trust for Digital Certificates”; *Published: 28 October 2020*

[7] Ayush S. Ghanghoria1, A Sahaya Anto Raja, Vivek J. Bachche, Ms. Neha Rathi, “Secure E-Documents Storage using Blockchain”, *International Research Journal of Engineering and Technology(IRJET)*, Volume: 07 Issue: 03 | *Mar 2020*

[8] Antonopoulos, Andreas M. *Mastering Bitcoin*. 2nd ed., O'REILLY, 2017.

[9] Sang, Jimmy. *Programming Bitcoin*. 1st ed., vol. 1, O'REILLY, 2019.

[10] Prusty, Narayan. *Building Blockchain Projects*. 1st ed., Birmingham, Packet>, 2017.

[11] Buterin, Vitalik. “Ethereum Whitepaper.” *Ethereum Whitepaper*, 2013, https://ethereum.org/en/whitepaper/. Accessed 02 March 2021.

[12] Kasireddy, Preethi. “How does Ethereum work, anyway?” *Medium*, 27 07 2017, https://preethikasireddy.medium.com/how-does-ethereum-work-anyway-22d1df506369. Accessed 15 April 2021.

Appendix A

MATHEMATICS OF ECDSA (ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM)

ECDSA is a cryptographically secure digital signature algorithm. It is based on elliptic curve cryptography (ECC).

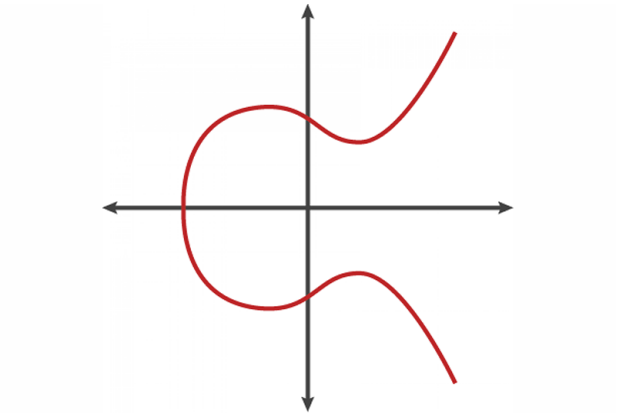

Figure 7. Elliptic Curve

Elliptic curves, used in cryptography define,

- **Generator point G**, used for scalar multiplication on the curve (multiply integer by EC point)
- **Order *n*** of the subgroup of EC points, generated by **G**, which defines the length of the private keys (e.g. 256 bits)

- **Key Generation**

  ECDSA has key pairs as,
  > private key (integer) : **Pk**, a randomly generated integer in the range **[0 … n-1]**
  > public key (EC point) : ***Pu = Pk * G*** , **G** is generator point of subgroup of large prime order **n**.

- **ECDSA Sign**
  Signing algorithm takes input message (**msg**) and **Pk** to produce **signature**. Signature consists of a pair of integers **{r, s}**. ECDSA signing algorithm works as,

  i) Calculate message hash using hash functions like SHA-256,
  h = hash(msg)
  ii) Generate a random number **k** in range **[1 …...n-1]**

ii) Calculate signature proof:

$$s = k^{-1} * (h + r * Pk) \pmod n,$$

$\mathbf{k^{-1}}$ **(mod n)** is an integer satisfying $k * k^{-1} \equiv 1 \pmod n$

iii) Return signature **{r, s}.**

Calculated signature **{r, s}** encodes random point **R = k*G,** with proof **s,** confirming the signer knows the message and private key. Proof **s,** is verifiable using corresponding **Pu.**

- **ECDSA Verification**

Verification algorithm takes signed message **msg** , signature **{r, s}** produced from signing algorithm and **Pu** corresponding to signer's **Pk**.

i) Calculate message hash, with hash function used during signing,

$$h - hash(msg)$$

ii) Calculate modular inverses of signature proof,

$$s^1 = s^{-1} \pmod n$$

iii) Recover random point used during signing:

$$R' = (h * s^1) * G + (r * s^1) * Pu$$

iv) Take from R' its x-coordinate: $r' = R'.x$

v) Calculate the signature validation result by comparing as, $r' == r$

In general, verification recovers R' using **Pu** and check with **R**

- **Public Key Recovery**

With message **m** and signature **r, s** public key can be recovered.

i) Verify **r** and **a** are integers in **[1, n-1]**

ii) Calculate curve point, $R = (x1, y1)$ where **x1** is one of **r, r+n, r+2n ……**

iii) Calculate $h = hash(msg)$

iv) Calculate $u1 = -z\, r^{-1} \pmod n$ and $u2 = sr^{-1} \pmod n$ **,** where z = leftmost bit of h

v) Calculate curve point

$$Pu = (xA, yA) = u1 * G + u2 * R$$

Pu is the recovered public key.

This way ECDSA works to provide double key encryption.

Appendix B

SHA-256 , HASH GENERATING ALGORITHM

Hashing algorithm takes an input to return hashes or message digest. Let's consider the message '**msg**' being inputted to the hash algorithm. The hash generation steps for **msg** can be summarized as,

- **Pre-Processing,**

  i) Message imputed is converted to binary using ASCII**.**

  'DCVS' = 01000100 01000011 01010110 01010011

  ii) **1** is appended to resultant data at the end.

  phrase = 01000100 01000011 01010110 01010011 1

  iii) Phrase size is calculated and converted to binary, **41 = 101001**

  iv) Phrase is the padded up with '**0**' and phrase size is appended at the end to make size **512 bit** long,

  010001000100001101010110010100111000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000000000000000000000000000000000000000000
  00000000000000000000000000000101001

  v) Block is then divided to **16** words of **32-bits** each,

  0) 01000100010000110101011001010011
  1) 10000000000000000000000000000000
  .
  .
  .
  15) 00000000000000000000000000101001

Let's represent this block as,

block 1 = M(1)

and, word as,

M(i)(b) where, **b** is block number and **i** refers to the specific in the block and **i** refers to word.

- **Initial Hash Values**

8 words of length **32-bit** Initial hash values **(H^0)** are defined after generating from the **first 8 primes.** These hashes should remain the same. Primes are firstly **square rooted** and **modulus 1** is taken and the result is is then multiplied by 16^8 and rounded down to nearest integer.

p = 2,3,5,7,11,13,17,19

$$initial\ hash\ value = int((\sqrt{p} \bmod 1) * 16^8$$

Results from above is converted to hexadecimal form, giving **8** words assigned from **a-h**

$$a = H_0^{(0)} = 6a09e667$$
$$b = H_1^{(0)} = bb67ae85$$
$$c = H_2^{(0)} = 3c6ef372$$
$$d = H_3^{(0)} = a54ff53a$$
$$e = H_4^{(0)} = 510e527f$$
$$f = H_5^{(0)} = 9b05688c$$
$$g = H_6^{(0)} = 1f83d9ab$$
$$h = H_7^{(0)} = 5be0cd19$$

The process continues for every block.

- **Hash Computation**

64 words are required thus, 48 more words have to be computed resulting in 4 blocks. The words can be generated using the equation,

$$W_t = RotL\ (\sigma_1(W_{t-2}) + W_{t-7} + \sigma_0(W_{t-15}) + W_{t-16})\ \ where,$$

$$\sigma_0(x) = RotR^7(x) \oplus RotR^{18}(x) \oplus SHR^{10}(x)$$
$$\sigma_0(x) = RotR^{17}(x) \oplus RotR^{19}(x) \oplus SHR^{10}(x)$$

$$t=16\ for\ 17th\ word$$

Equation finds word **15** places back **(W(1))**, and **2** copies made. First copy is Right Rotated by **7** places **(RotR^7(x))**, i.e. each digit is moved to the right one place **7** times and moved front after number falls to end.

Now, original is right shifted by **3** $(SHR^3(x))$. If the number falls off the end it is replaced by zeros at the beginning of the word. Then these **3** words and $\sigma_0(W_1)$ computed as,

$$\sigma_0(x) = RotR^7(x) \oplus RotR^{18}(x) \oplus SHR^3(x) \text{ where,}$$

$$x = W_{t-15} = W_1 \text{ and } \oplus = \text{bitwise XOR}$$

Result is then converted to hexadecimal form. The process is repeated until all **64** words are computed.

Final Hash is then generated by the equation below:

$$\text{For } t = 0 \text{ to } 63:$$

$$\{$$

$$T_1 = h + \Sigma_1^{\{256\}}(e) + Ch(e,f,g) + K_t^{\{256\}} + W_t$$

$$T_2 = \Sigma_1^{\{256\}}(a) + Maj(a,b,c)$$

$$h = g$$

$$g = f$$

$$f = e$$

$$e = d + T_1$$

$$d = c$$

$$c = b$$

$$b = a$$

$$a = T_1 + T_2$$

$$\}$$

where,

$$Ch(e, f, g) = (e \wedge f) \oplus (\neg e \wedge g)$$

$$Maj(a,b,c) = (a \wedge b) \oplus (a \wedge c) \oplus (b \wedge c)$$

$$\Sigma_1^{\{256\}}(e) = RotR^6(e) \oplus RotR^{11}(e) \oplus RotR^{25}(e)$$

$$\Sigma_1^{\{256\}}(a) = RotR^2(a) \oplus RotR^{13}(e) \oplus RotR^{22}(a)$$

$$K_t^{\{256\}} = \text{A set constant in the Bitcoin code}$$

$$\neg \text{ , Bitwise NOT}$$

$$\wedge \text{ , Bitwise AND}$$

Values generated after every **63** iterations, intermediate hash value **H(i) , ith** intermediate hash can be generated as,

$$H_0^i = a + H_0^{(0)}$$
$$H_1^i = b + H_1^{(0)}$$
$$H_2^i = c + H_2^{(0)}$$
$$H_3^i = d + H_3^{(0)}$$
$$H_4^i = e + H_4^{(0)}$$
$$H_5^i = f + H_5^{(0)}$$
$$H_6^i = g + H_6^{(0)}$$
$$H_7^i = h + H_7^{(0)}$$

a-h are values from final iteration **(t(63))** and **H(0)(0)** to **H(7)(0)** are initial hash values.

These obtained intermediate hash values intare converted into hexadecimal form, giving final message digest. Appending values together final message digest is obtained.

$$M = H_0^N \,||\, H_1^N \,||\, H_2^N \,||\, H_3^N \,||\, H_4^N \,||\, H_5^N \,||\, H_6^N \,||\, H_7^N$$

This way data is cryptographically hashed using SHA-256 hashing algorithm.

Appendix C

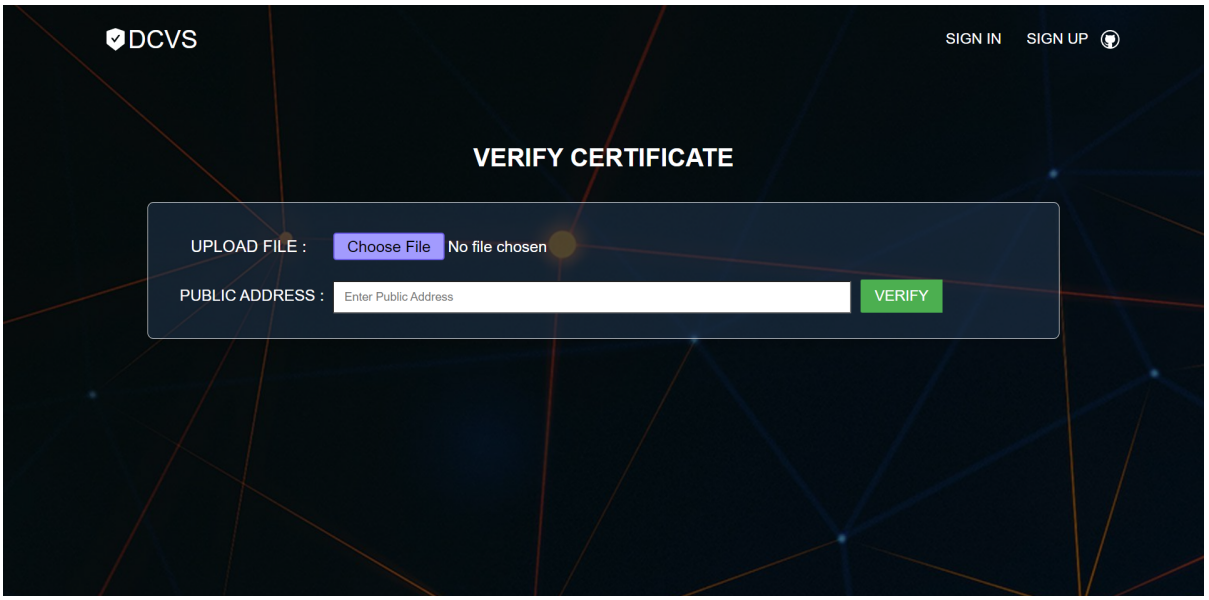

Figure 1. Homepage

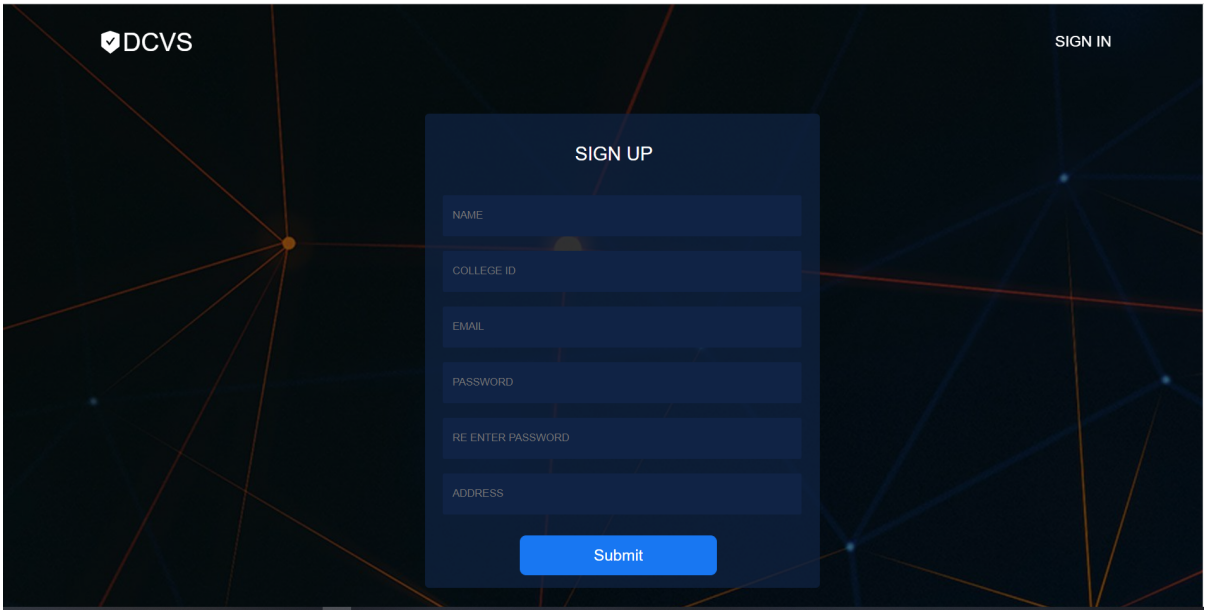

Figure 2. SignUp Page

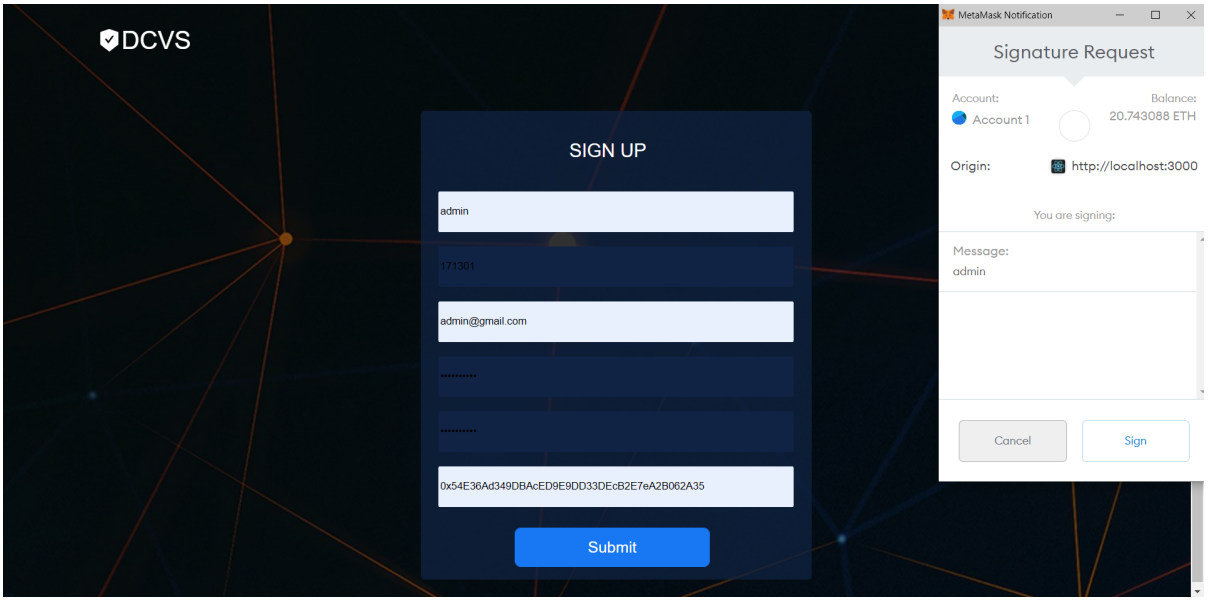

Figure 3. Conforming Address through metamask

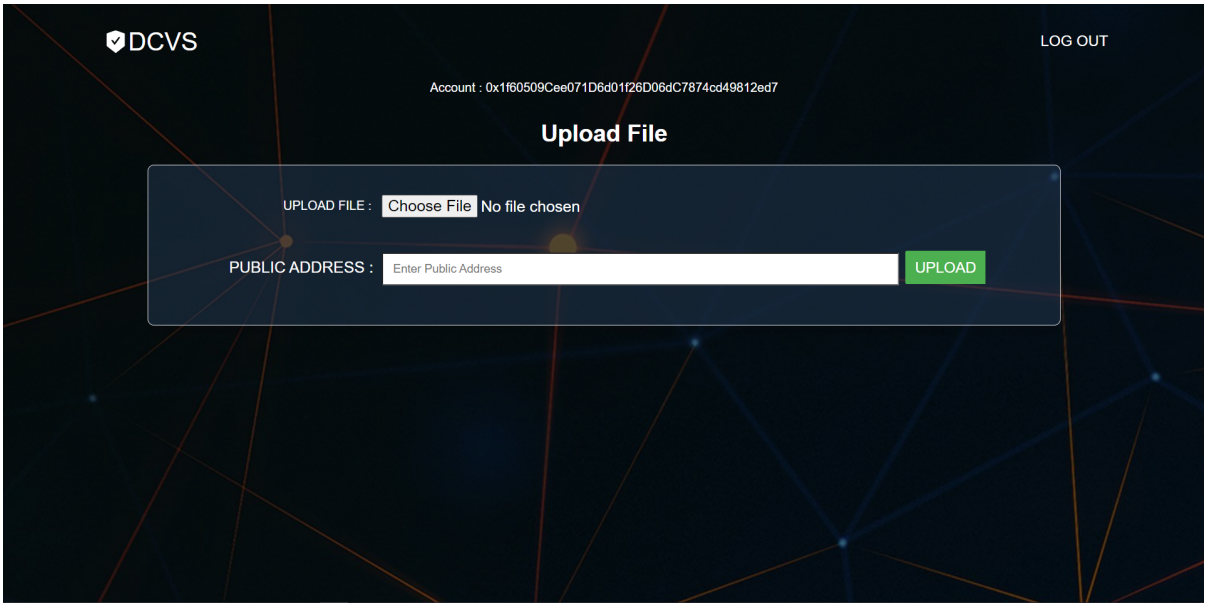

Figure 4. Issuance of certificate

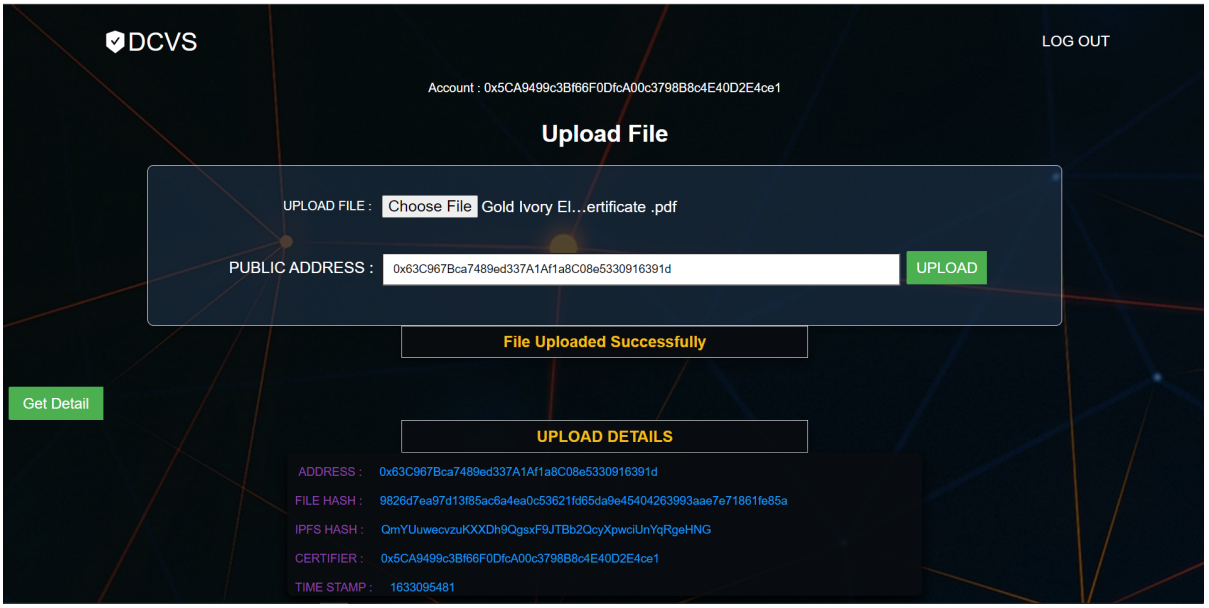

Figure 5. Uploaded certificate details

\

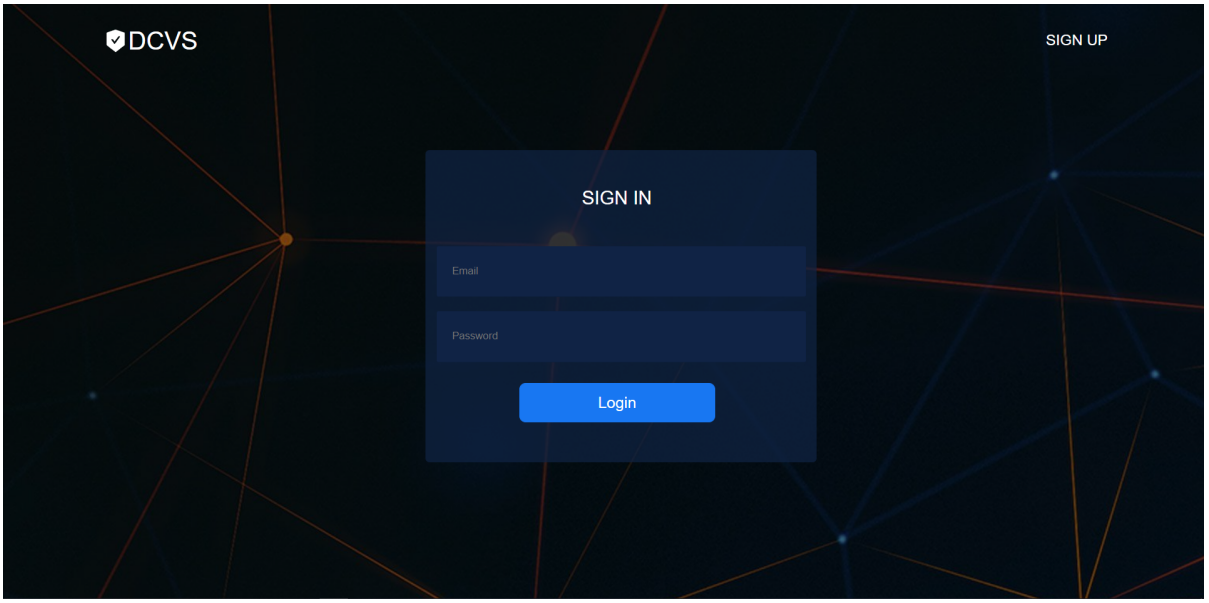

Figure 6. SignIn Page

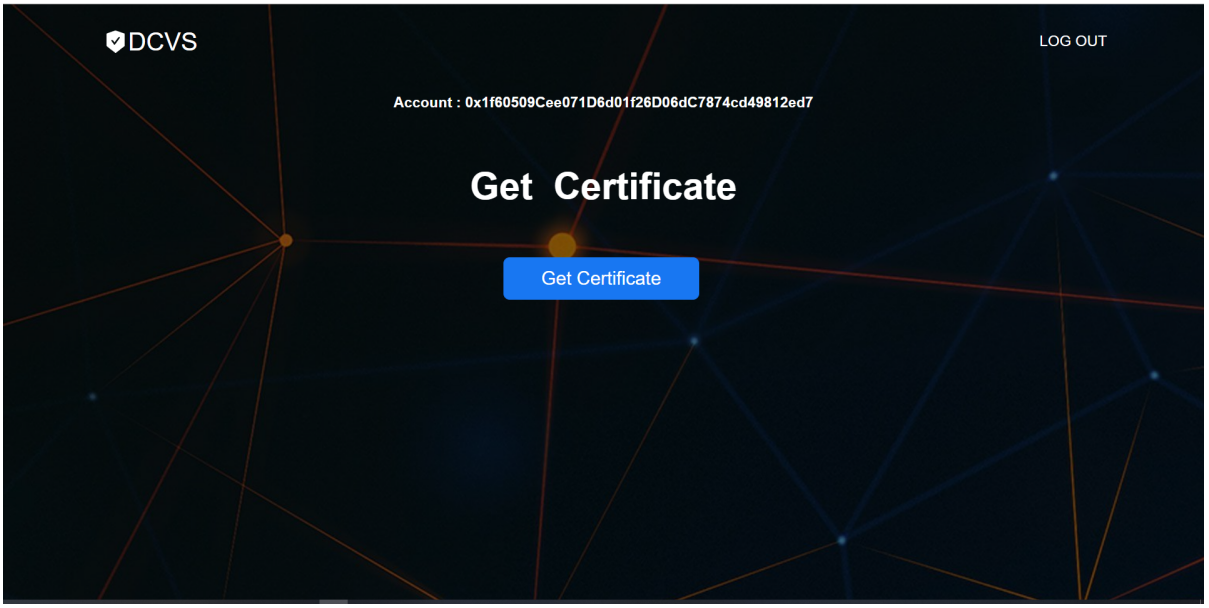

Figure 7. Download Certificate.

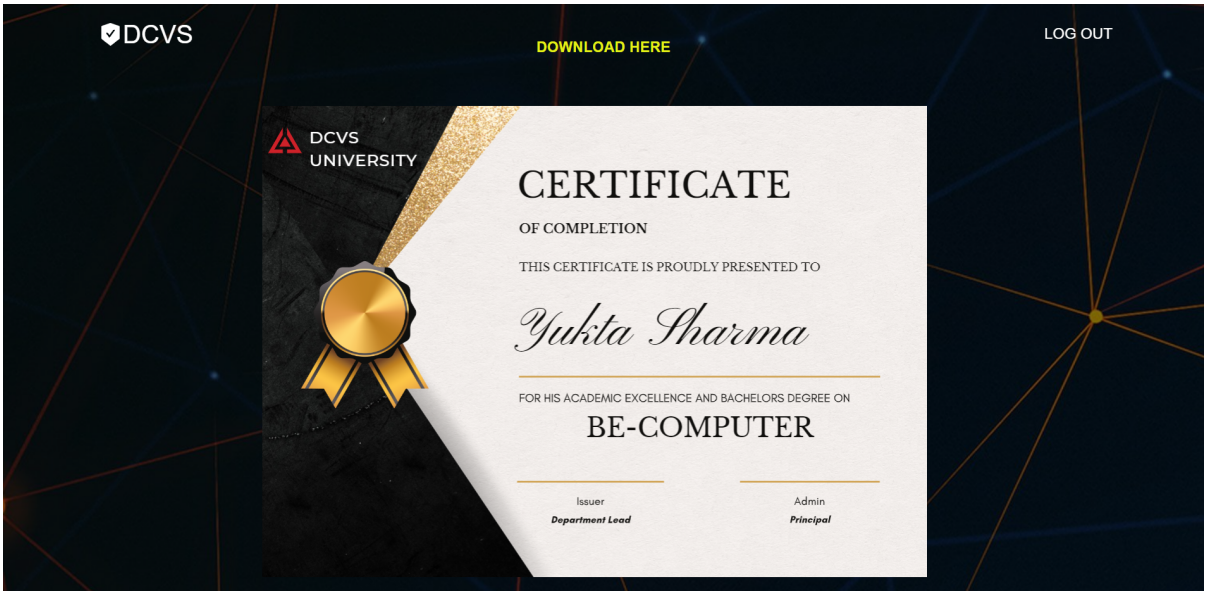

Figure 8. Downloaded certificate

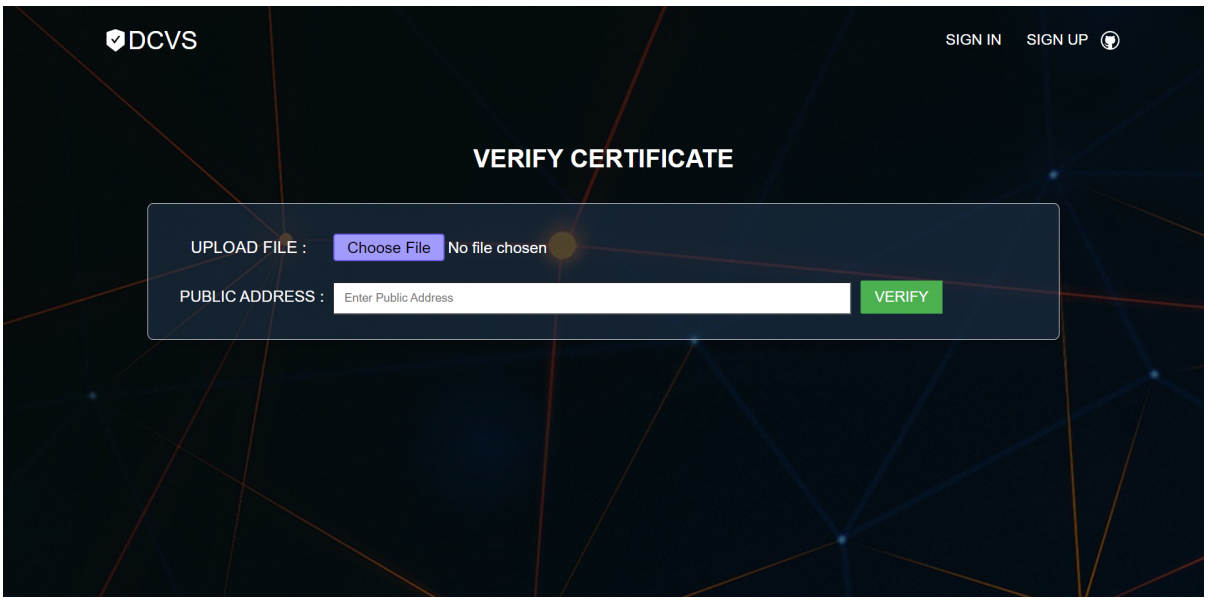

Figure 9. Verifying certificate

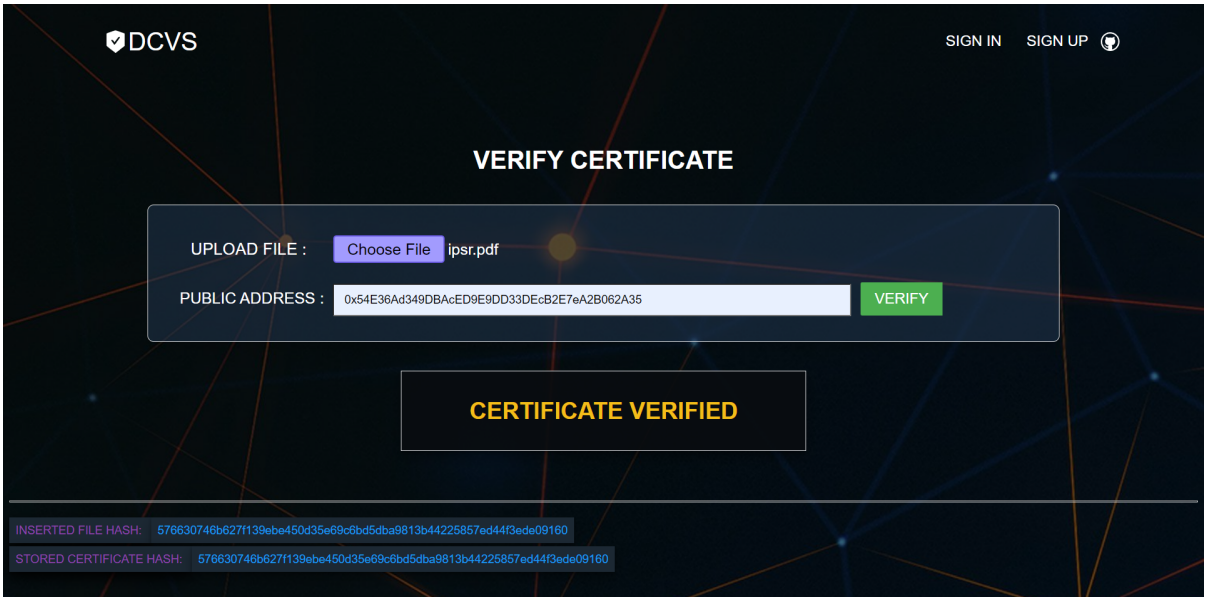

Figure 10. Verified certificate

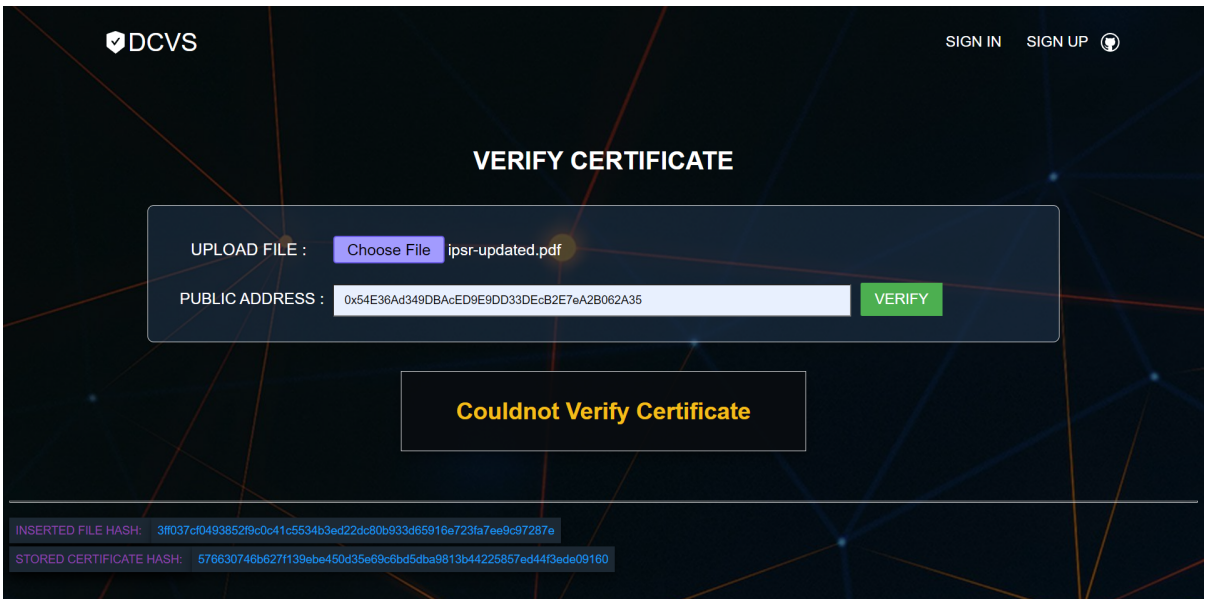

Figure 11. UnverifiedCertificate

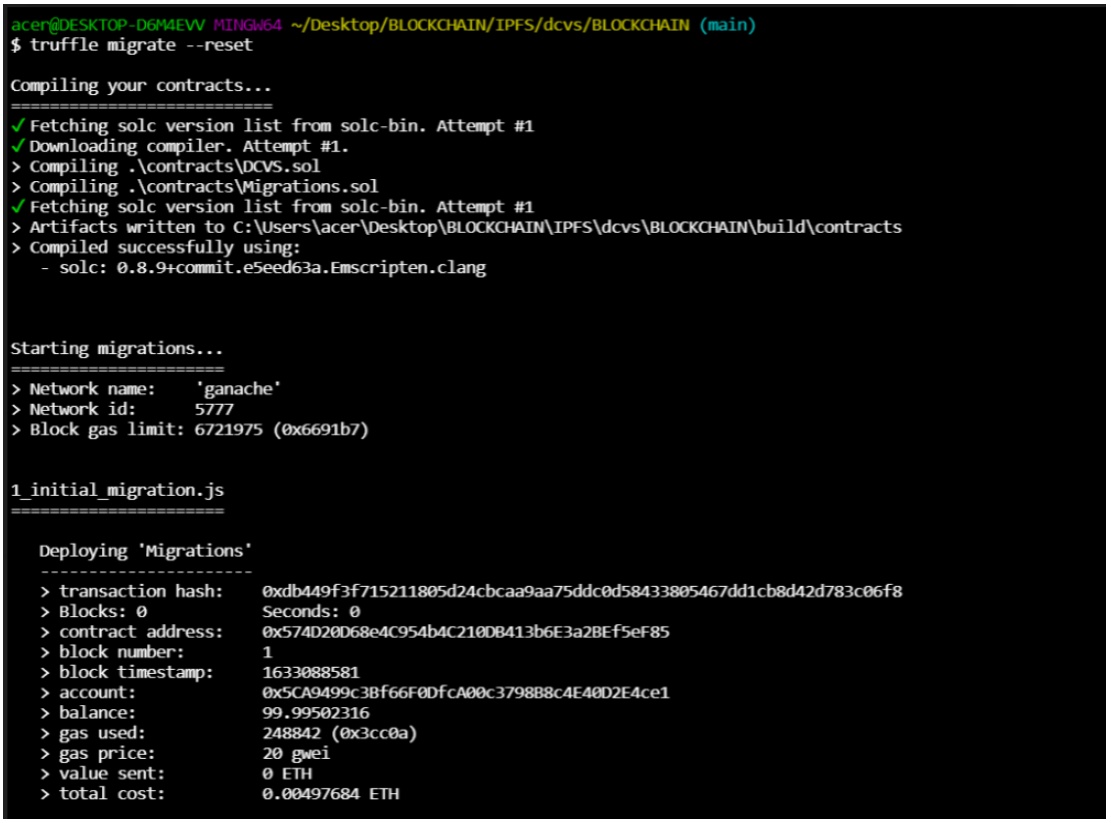

Figure 12. Deploying Contracts

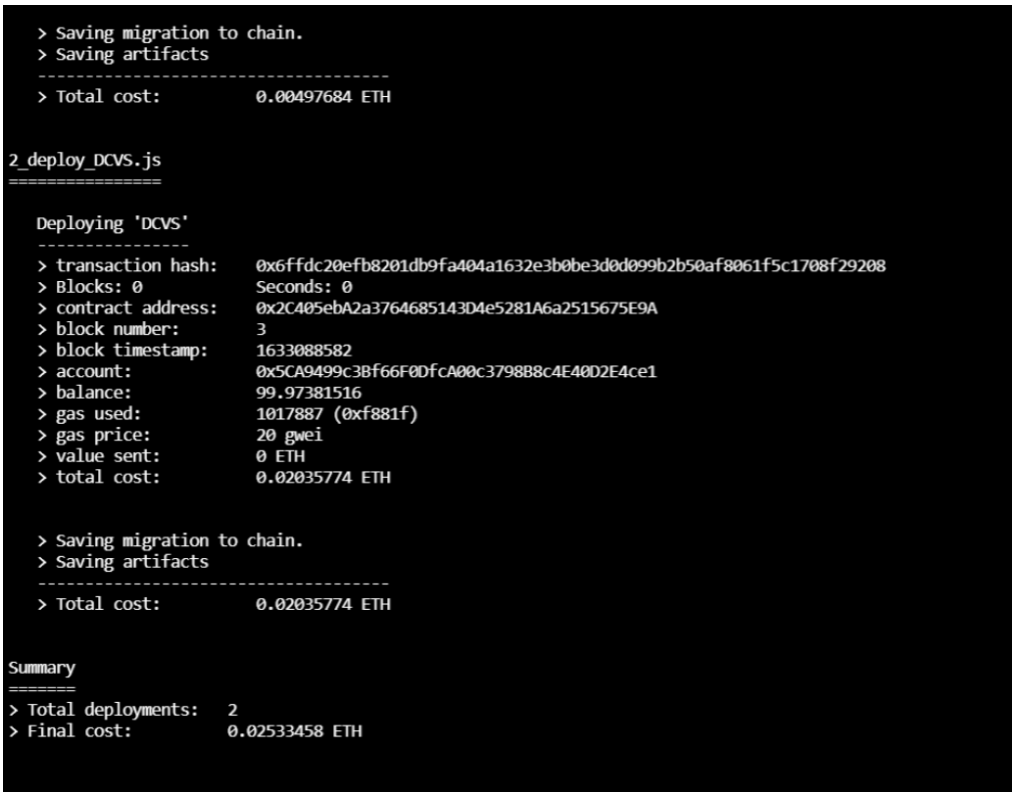

Figure 13. Contract deployment details

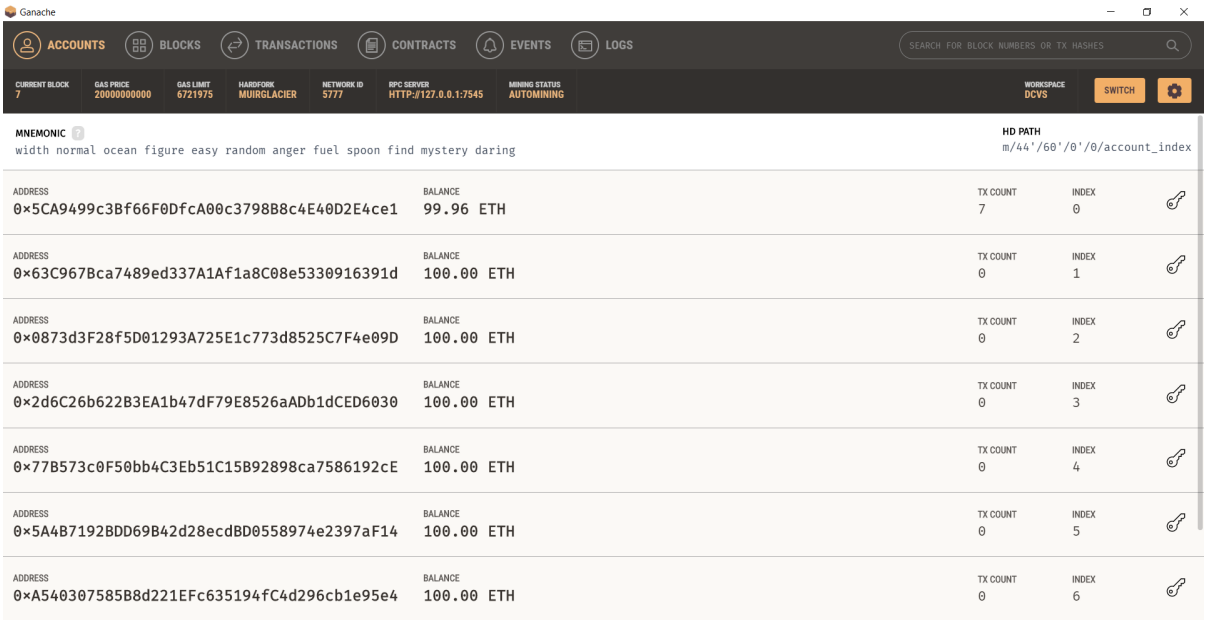

Figure 14. Ganache Accounts

Figure 15. Transaction to blockchain network

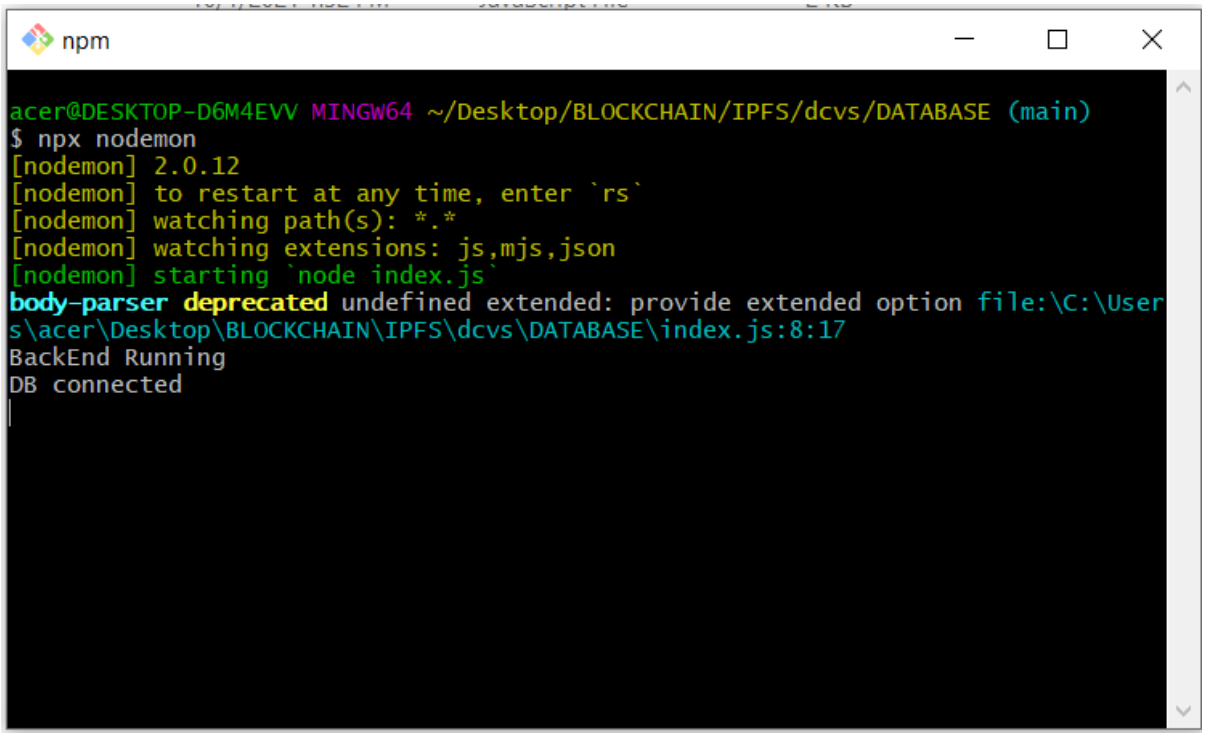

Figure 16. Database connected and running

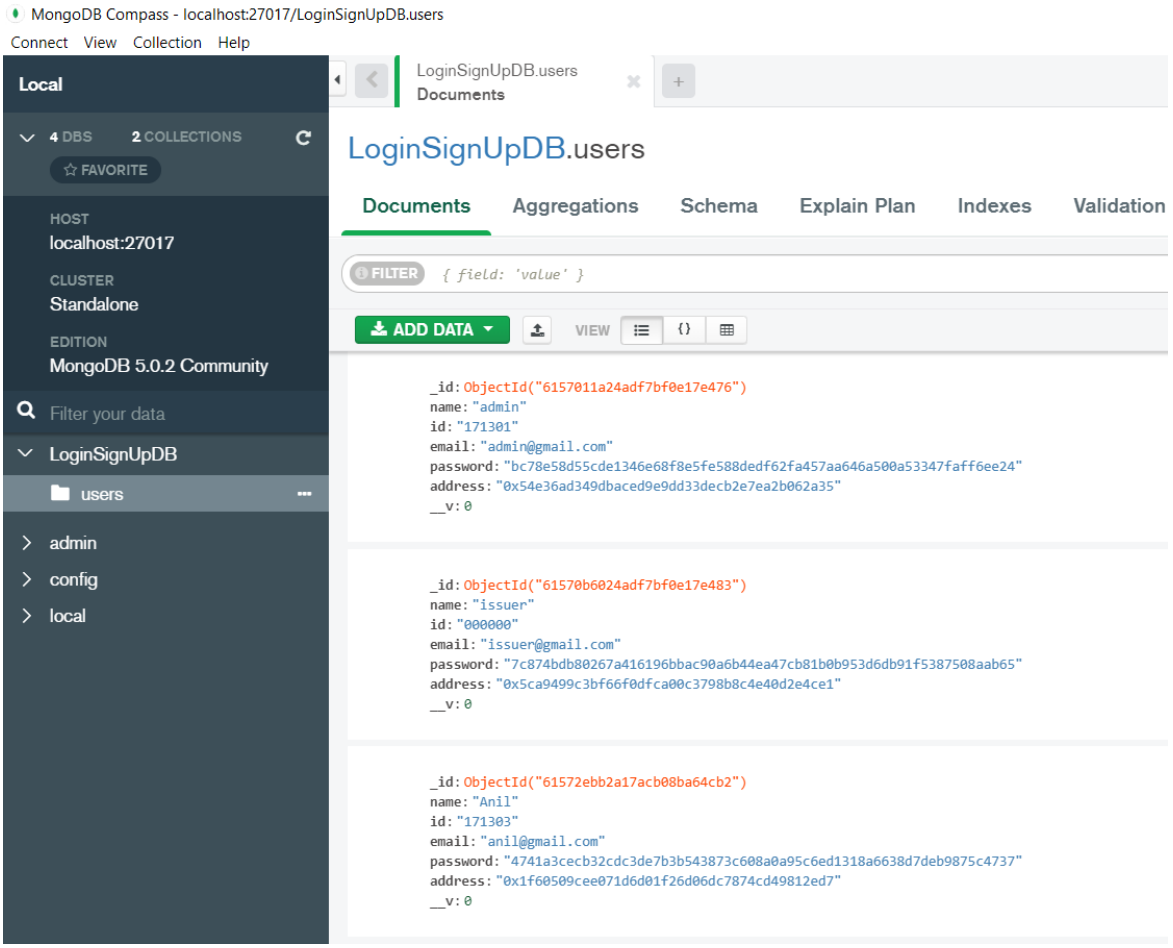

Figure 17. MongoDB