**BLOCKCHAIN BASED CERTIFICATE VALIDATION**

**A PROJECT REPORT**

***Submitted by***

**PON ANITHA C 211615205065**

**POOJA SREE V 211615205067**

**POORVAJA S 211615205068**

***in partial fulfillment for the award of the degree***

***of***

**BACHELOR OF TECHNOLOGY**

**in**

**INFORMATION TECHNOLOGY**

****  ****

**RAJALAKSHMI ENGINEERING COLLEGE, CHENNAI**

**ANNA UNIVERSITY: CHENNAI 600 025**

**April 2019**

**BONAFIDE CERTIFICATE**

Certified that this project report “**BLOCKCHAIN BASED CERTIFICATE VALIDATION”** is the bonafide work of **PON ANITHA C, POOJA SREE V and POORVAJA S** who carried out the project work under my supervision.

**SIGNATURE SIGNATURE**

**Dr. L. Priya, Dr. S. Baghavathi Priya**

**HEAD OF THE DEPARTMENT, SUPERVISOR**

Professor, Professor,

Information Technology Information Technology, Rajalakshmi Engineering College, Rajalakshmi Engineering College, Rajalakshmi Nagar, Rajalakshmi Nagar, Chennai-602 105. Chennai-602 105.

This project report is submitted for viva voce examination to be held on \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_.

**Internal Examiner External Examiner**

ii

**ACKNOWLEDGEMENT**

We thank the Almight God for the successful completion of the project. Sincere thanks to our Chairman **Mr. S. Meganathan** for his sincere endeavor in educating us in his premier institution.

We heartily thank our chairperson **Dr. Thangam Meganathan** for her motivation and inspiration that paved way for the completion of our project.

We also express our gratitude to our principal **Dr. S. N. Murugesan** who helped us in providing the required facilities in completing the project.

We would like to thank our Head of Department **Dr. L. Priya** for her guidance and encouragement for completion of project.

We thank our project coordinator **Dr. (Mrs.) S. Poonkuzhali** and **Mrs. R. Anto Arockia Rosaline** for his invaluable guidance, ideas, advice and encouragement for the successful completion of this project.

We would also like to thank **Dr.** **S. Baghavathi Priya** our supervisor for constantly guiding us and motivating us throughout the course of the project. We express our gratitude to our parents and friends for extending their full support to us.

iii

**ABSTRACT**

Certificates received are a very important part of everyone’s life. It is a very long process to reapply for the certificates and procure it, if it is lost once. Also trusting someone, say a university with these certificates, again puts us in a situation where we depend on that particular organization. The certificate verification process also involves the production of the certificate and checking it manually each time. To overcome misplacing, damaging and verification problems, a certificate validation system is proposed, in which, blockchain technology is used to store the certificates in a secure format. Using SHA-265 hashing algorithm, the certificates are uploaded in the system, and can be viewed by anyone with the right credentials. Due to the usage of blockchain technology, the certificates are stored in a decentralized manner, which ensure there is no central point of failure. Also new nodes are added when certificate uploading is required, instead of modifying the past blocks.

iv

**TABLE OF CONTENTS**

v

|  |  |  |
| --- | --- | --- |
| **CHAPTER** | **TITLE** | **P.No** |
|  | Abstract | iv |
|  | List of Tables | vii |
|  | List of Figures | viii |
|  | List of Abbreviations | x |
| **1** | **INTRODUCTION** | 1 |
|  | 1.1 Blockchain based Certificate Validation System | 2 |
|  | 1.2 Objectives | 2 |
|  | 1.3 Advantage | 3 |
| **2** | **LITERATURE REVIEW** | 3 |
|  | 2.1 Overview of Existing System | 3 |
|  | 2.2 Proposed System | 6 |
| **3** | **SYSTEM REQUIREMENTS SPECIFICATION** | 7 |
|  | 3.1 Platform | 7 |
|  | 3.2 Hardware Requirements | 7 |
|  | 3.3 Software Requirements | 7 |
| **4** | **SYSTEM DESIGN** | 9 |
|  | 4.1 System Architecture | 9 |
|  | 4.2 Use Case Diagram | 11 |
|  | 4.3 Module Description | 12 |
|  | 4.3.1 Student Module | 12 |
|  | 4.3.2 Organization Module | 12 |
|  | 4.4 Algorithms Used | 13 |
|  | 4.4.1 Secure Hash Algorithm | 13 |
| **5** | **SYSTEM IMPLEMENTATION AND TESTING** | 17 |
|  | 5.1 Student Interface | 17 |
|  | 5.2 Organization Interface | 22 |
| **6** | **RESULTS AND DISCUSSION** | 25 |
|  | 6.1 Results | 25 |
| **7** | **CONCLUSION AND FUTURE ENHANCEMENTS** | 32 |
|  | **APPENDIX 1 – SAMPLE CODE** | 33 |
|  | **REFERENCES** | 40 |

|  |  |
| --- | --- |
|  |  |

vi

**LIST OF TABLES**

|  |  |  |
| --- | --- | --- |
| **Table No.** | **Description** | **P. No** |
|  |  |  |

vii

**LIST OF FIGURES**

viii

|  |  |  |
| --- | --- | --- |
| **Figure No.** | **Description** | **P. No** |
| 4.1 | Architecture Diagram of Certificate Validation System | 9 |
| 4.2 | Use case diagram of Certificate Validation System | 11 |
| 5.1 | User is asked if they are a student or an organization | 17 |
| 5.2 | Authentication page for the student user | 18 |
| 5.3 | New User or Existing User | 18 |
| 5.4 | New user registration | 19 |
| 5.5 | Login page for existing users | 20 |
| 5.6 | Choose to upload or view certificate hash | 21 |
| 5.7 | Certificate hash upload | 21 |
| 5.8 | View Uploaded hash Id | 22 |
| 5.9 | Authentication page for Organization | 23 |
| 5.10 | Student Id prompt | 23 |
| 5.11 | Student details | 24 |
| 5.12 | View Uploaded hash Id | 24 |
| 6.1 | Independent nodes in a network | 25,26 |
| 6.2 | Linking the nodes in a network through postman | 26 |
| 6.3 | Linked nodes in the same network | 27 |
| 6.4 | Pending transactions in both the nodes | 28 |
| 6.5 | Mining of a new block | 29 |
| 6.6 | Transactions added in the new block mined, in both the nodes | 30 |
| 6.7 | The end result for viewing the certificate hash | 31 |

ix

**LIST OF ABBREVIATIONS**

|  |  |
| --- | --- |
| **Abbreviation** | **Expansion** |
| SHA | Secure Hash Algorithm |

x

## **CHAPTER 1**

## **INTRODUCTION**

**1. INTRODUCTION**

The certificate verification process is an ever occurring and tedious process. You want an admission in a university, you need to verify your certificates. You want to join an organization; the certificates need to be verified. And there are many more instances where our presence needs to be verified through our certificates. Producing the original certificates each and every time for verification and then placing it back into safety is a tough process. Also, no harm or damage must befall the certificate during the process of verification. It is known that certificates literally define our lives. Once lost or damaged, it takes a long time to obtain another original certificate. 90% of the institutions in India, be it universities or schools, hold the student’s certificates until the student leaves that institution. Companies that recruit employees, double check the student’s mark by demanding their certificates. Instead of producing the certificates each and every time, or leave the certificates with any institution, a system is proposed where the certificates can be stored in a hashed format and ensured safety. The system proposed is a website, where the certificates can be stored as hashes and retrieved as and when required, and verified. The blockchain technology is used for the storage of the hashes, thus ensuring that they cannot be modified in any way. The system is available for two kinds of users, the students and any valid organizations who wish to go through the student’s certificates. The system uses the latest technology, blockchain, which stores each hash of the certificate in each block, in a private network, open to the students and organizations only. The system employs a two-step verification login, the 10th grade register number or the organization code being the first, and logging in being the second. Once logged in, the students will be able to view and upload their certificates, while the organizations will only be able to view the certificates. But the students will not be able to make any changes to the already existing certificates, once uploaded. This way, the hashes of the certificates will stay secure and unmodified.

## **BLOCKCHAIN BASED CERTIFICATE VALIDATION**

The certificate validation is mainly focused on two objectives, which are, first to store the certificate hashes in a secure way, where they cannot be modified, and second, to be able to verify the certificates during all periods of times without any failure or loss of data.

The problem with the current system is that the certificates are stored in centralized databases, where failure of the server can occur or data may be lost. Also, the security levels provided for the certificate storage is outdated, thus making it easy for anyone to modify those certificates.

* 1. **OBJECTIVES**

1. To develop a certificate validation system that aims to store the certificates in their hash format.
2. To make sure the certificate data is accessible at all times and doesn’t vanish due to some server failure.
3. To ensure the safety of the certificates, SHA-256 algorithm is implemented through blockchain technology.
4. To ensure that no malicious users use the system, a two-step verification is deployed.
   1. **ADVANTAGE**
5. The organizations can use and refer the student’s certificates through this system, which are advantageous to both the students and the organizations.
6. The organizations can also verify if the certificates uploaded are authentic by cross checking the hashes of the certificate.
7. No modification to the certificate or it’s hash can be made as they are stored using the blockchain technology.
8. The certificates are accessible at all times, as they are stored in a decentralized format.

## **CHAPTER 2**

## **LITERATURE SURVEY**

### **2.1 OVERVIEW OF EXISTING SYSTEMS**

1. **The Future of Data Storage: A Case Study with the Saudi Company**

Azzah Al GhamdI and Thomas Thomson investigates the data storage methods and future requirements for one of the largest oil companies in the world, Saudi SA Oil Company [1]. But data is stored in a centralized storage and also, backup data which doesn’t have much of permissions, is difficult to access. Along with this, security is provided through checksum.

1. **United States Patent , Borlick et al , Date of Patent : Jan . 1 , 2019 , Patent No . : US 10 , 171 , 585 B2 .**

Matthew G . Borlick, Lokesh M . Gupta, Roger G . Hathorn and Karl A . Nielsen has stated a system which stores different parts of data in various clouds [2]. Since cloud storage is used, it is unknown at what place the data is stored and if any attacks are made on the data, it is difficult to find out about the attacks and the owners are not notified. Also, the data is stored only for a particular time period.

1. **A Survey on the Security of Blockchain Systems.**

Xiaoqi Li, Peng Jiang, Ting Chen, Xiapu Luo and Qiaoyan Wen has made a systematic study on the security threats to blockchain and survey the corresponding real attacks by examining popular blockchain systems [3]. But it has been observed that the Consensus mechanism PoW is a waste of computing resources and not all of the data stored in blockchain is valid.

1. **Capacity barriers in hard disks: problems, solutions and lessons.**

Wasim Ahmad Bhat has studied and investigated the storage problems in hard disks and proposed solutions to overcome the hardships faced in hard disks [4].

1. **Digital Signature Certificate Policy for Railway Units.**

Under the Government of India, the Ministry of Railways has passed the “*Digital Signature Certificate Policy for Railway Units”,* under which it has been stated that digital signature certificates have been approved for all the railway units [5].

1. **Long-term preservation of big data: prospects of current storage technologies in digital libraries.**

Wasim Ahmad Bhat, in his paper has investigated the prospects of current storage technologies for long-term preservation of big data in digital libraries [6] and has come to the conclusion that current storage technologies might not fulfil the long-term storage of library data in the future.

1. **Document Validation and Verification System.**

Samit Shivadekar, Stephen Raj, Abraham and Sheikh Khalid has proposed an E-Governance system where 'E-Governance system' will be an online platform for deliverance of Government to Citizen Services and storage of digital certificates, documents etc. [7]. But there is the high initial cost of implementing such a system, as is typical of all computerized systems.

1. **Strongly Leakage-Resilient Authenticated Key Exchange.**

Rongmao Chen, Yi Mu, Guomin Yang, Willy Susilo and Fuchun Guo in there has presented a practical instantiation of the general framework based on the Decisional Diffie-Hellman assumption without random oracle [8]. But the output size of the leakage function and Challenge dependent leakage security of AKE protocols.

1. **MAD4SA: a prototype of document management system for small-scale archives.**

Tongjin Lee and Jun Iio has designed a prototype for document management system for small scale archives with open source software [9]. Also, evaluation of the system is made in the perspective of the administrators. But this system is designed only for small scale storage and not for huge amount of data.

1. **Designing an Automatic Web-Based Certificate Verification System for Institutions.**

Nwachukwu-Nwokeafor K.C and Igbajar Abraham has designed an online certificate verification system based on the verification process adopted by the university to verify the results [10]. But there is a high initial cost of implementation and users need to be educated about the usage of the system. The system is also vulnerable to attacks and change of data.

1. **A Survey on Digital Signatures and its Applications.**

Abhishek Roy and Sunil Karforma have made a thorough study of the industry standard digital signature schemes to obtain optimum security level for the electronic mechanisms and explored its probable applications in various domains [11].

1. **An Academic Certification Verification System Based on Cloud Computing Environment.**

Nicholas Mwankiki Musee has developed a prototype which is used as SaaS to provide certificate verification [12]. But the failure of central server will affect the whole system greatly and also the location of documents will be unknown.

1. **Verification of the integrity and legitimacy of academic credential documents in an international setting.**

George D. Gollin has studied and concluded that Public-key cryptography provides a technical solution to the problem of authenticating academic documents such as transcripts and diplomas [13]. But during that period of time it was not promoted, as Electronic Transcript Technology has not yet developed to a higher level. Also, the database to be used was not mentioned.

**2.2 PROPOSED SYSTEM**

1. The certificate validation system provides an online website to upload and retrieve the hashes of the certificates.
2. Each certificate is uploaded in its hash format and is stored using blockchain.
3. The hashes are stored in unique transactions in nodes, which are deployed on a private network.
4. Each node maintains a copy of the transactions along with the hashes, thus making it available at any time.
5. The certificates are hashed using the SHA-256 algorithm, and also, the transaction ids are obtained using the same.

# **CHAPTER 3**

# **SYSTEM REQUIREMENT SPECIFICATIONS**

## **3.1 PLATFORM**

Chapter 3 describes the requirement analysis in accordance with the input and the resources and it also describes the implementation of the project with the technology used.

### **3.2 HARDWARE REQUIREMENTS**

The hardware requirements may serve as the basis for a contract for the implementation of the system and should, therefore, be a complete and consistent specification of the whole system.

**PC Minimum Requirements**

* RAM : 4 GB
* Architecture : 32-bit
* Processor : Core 2 Duo
* Processor speed : Minimum 2 GHZ
* Hard disk : 500 GB

## **3.3 SOFTWARE REQUIREMENTS**

The software requirements are the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it.

* Operating system : Windows 7/8/10
* Browser : Google Chrome/Mozilla Firefox
* Development Environment : Atom text editor, Eclipse
* Languages : Java, Solidity
* Markup and styling : HTML 5.1 CSS 3, JavaScript

# 

# 

# 

# 

# 

# **CHAPTER 4**

# **SYSTEM DESIGN**

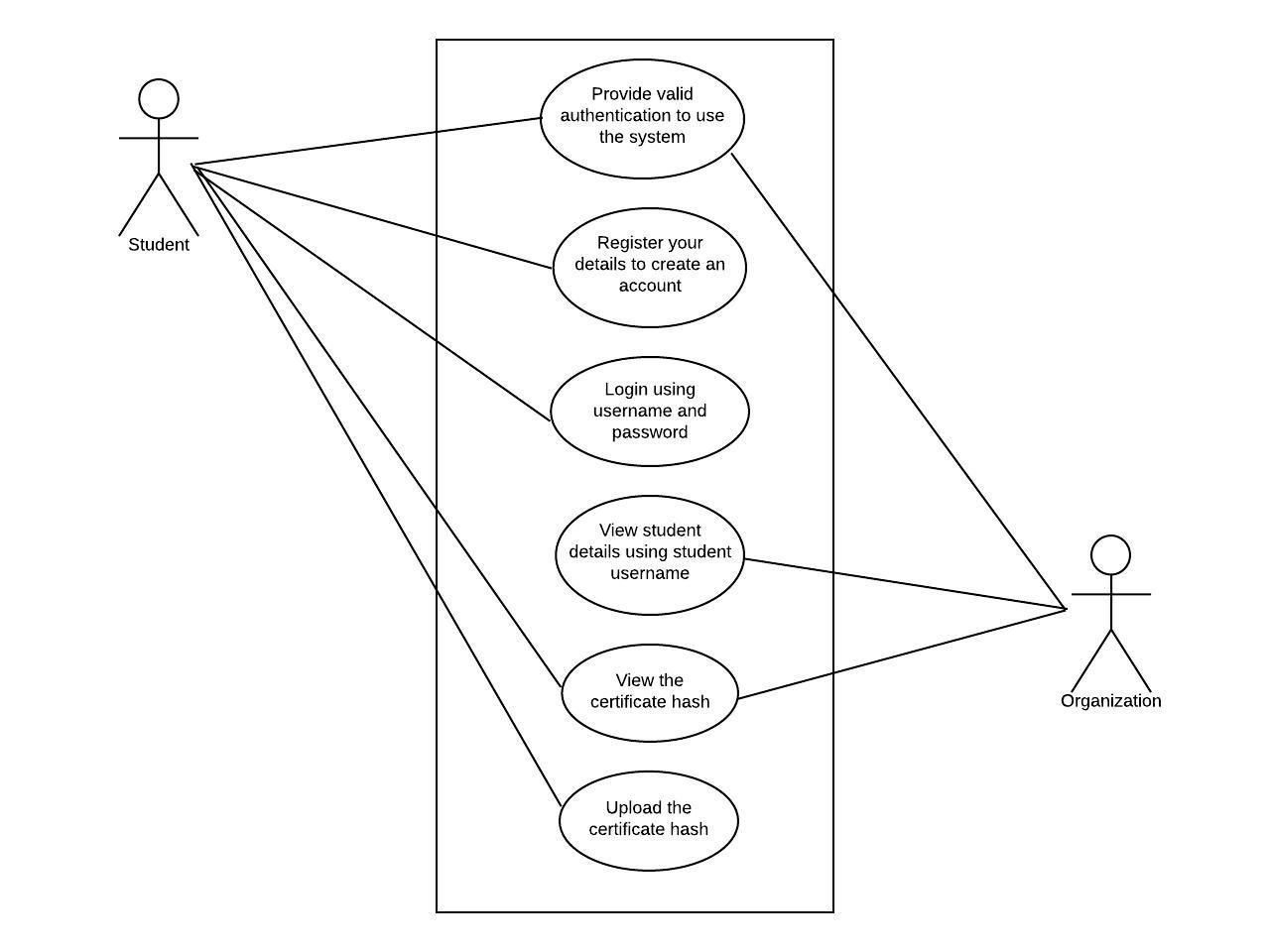
## **4.1 SYSTEM ARCHITECTURE**

**Figure 4.1 Architecture Diagram of Certificate Validation** **System**

The architecture diagram of Certificate Validation system follows the sequence of operation as elucidated below.

* The user can initially choose whether they are a student or any organization.
* The user, if a student if prompted to enter their 10th grade register number for entering into the system. If it is an organization, the code of the organization is required to enter into the system.
* The student user can either login, if they are a returning user, or can register if they are a new user. The organization is required to enter the username of the student whose certificates they wish to verify.
* Once logged in, the student user has the provision to either verify the certificate or upload a new one. The organization will be able to view the hash of the certificate and verify the same.
* The authentication, login and the register information are stored in the database, and the certificate hash information are stored in the blocks.
* There is a private network where various number of nodes are running. When blocks are mined, the transactions that contain the hash of the certificates are inserted into the blocks.
* The hash of the certificates can be viewed when the username of the student is given, which is the sender’s address in the transaction.
* The functionality in the blockchain part is as follows:
  + The transactions consist of the transaction id, the sender address which will be the student’s username and the hash of the certificate that the student wishes to upload.
  + The transactions are put in a pending list and are included in a new block when a block is mined.
  + When a user wants to view the hash of the certificate that was uploaded, they need to give the username of the student, and the hash of the certificates will be displayed.

## **4.2 USE CASE DIAGRAM**



**Figure 4.2 Use case diagram of Certificate Validation System**

The above use case diagram depicts that there are two characters(actors) involved in the system namely student and organization.

The student needs to enter their 10th grade register number to enter into the system, can login into the system if they are an existing user, register to create a new account if they are a new user, upload the hashes of new certificates and then view the hashes of the already stored certificates

The organizations need to use the organization code to enter into the system, should enter the student’s username to view the hashes of their respective certificates and verify accordingly.

## **4.3 MODULE DESCRIPTION**

### **4.3.1 STUDENT MODULE**

The student uses the web application that is integrated with blockchain to store and retrieve the hash of the certificates. Steps to be followed in this module are:

Step i) The user needs to enter their 10th grade register number into the UI to enter into the system.

Step ii) The user is then led to a register/ login page, where they can choose either option.

Step iii) The user needs to register before using the system if they are a new user, or login directly, if they are an existing user.

Step iv) On the next page, the user maybe able to the operations of either uploading the certificate hash or viewing the certificate hash.

**4.3.2 ORGANIZATION MODULE**

The organization uses the web application to view the hashes of the certificates that the students have uploaded so that they can verify if the certificates in hand are authentic. Steps to be followed in this module are:

Step i) The user should enter the organization code to enter into the system.

Step ii) After submission of the credentials, the user is directed to a page where

student’s user name is required to view the student details.

Step iii) Upon entering the student’s username, the organization will be able to verify

whether the student certificate with the organization is authentic or not.

Step iv) The user will be displayed with the student’s username, and the hash of the

certificate that the student has uploaded.

### **4.4 ALGORITHMS USED**

**4.4.1 SECURE HASH ALGORITHM**

**SHA-256** is a member of the [SHA-2](https://en.bitcoinwiki.org/wiki/SHA-2) cryptographic hash functions designed by the NSA. Cryptographic hash functions are mathematical operations run on digital data; by comparing the computed "hash" (the output from execution of the algorithm) to a known and expected hash value, a person can determine the data's integrity. A one-way hash can be generated from any piece of data, but the data cannot be generated from the hash.

This Protocol works with information broken down into pieces of 512 bits (or 64 bytes in other words). It produces its cryptographic "mixing" and then issues a 256-bit [hash](https://en.bitcoinwiki.org/wiki/Hash) code. The algorithm includes a relatively simple round, which is repeated 64 times.

In addition, SHA-256 has quite good technical parameters:

* block size indicator (byte): 64.
* maximum allowed message length (bytes): 33.
* characteristics of the message digest size (bytes): 32.
* the standard word size (bytes): 4.
* internal position length parameter (bytes): 32.
* the number of iterations in one cycle: 64.
* the speed achieved by the Protocol (MiB/s): approximately 140.

SHA-256 is used in several different parts of the Bitcoin network:

1. [Mining](https://en.bitcoinwiki.org/wiki/Mining) uses **SHA-256 as the**[**proof-of-work**](https://en.bitcoinwiki.org/wiki/Proof-of-work)**algorithm**.
2. SHA-256 is used in the creation of bitcoin addresses to improve security and privacy.

SHA-256 operates in the manner of MD4, MD5, and SHA-1:

The message to be hashed is first

(1) padded with its length in such a way that the result is a multiple of 512 bits long and then

(2) parsed into 512-bit message blocks M(1) , M(2) , …. , M(N) .

The message blocks are processed one at a time:

Beginning with a fixed initial hash value H(0) , sequentially compute

H(i) = H(i1) + CM(i) (H(i1));

where C is the SHA-256 compression function

and + means word-wise mod 232 addition. H(N) is the hash of M.

**SECURE HASH ALGORITHM STEPS:**

**1. PADDING**

To ensure that the messag1 has length multiple of 512 bits:

* first, a bit 1 is appended,
* next, k bits 0 are appended, with k being the smallest positive integer such that

l + 1 + k ≡ 448 mod 512,

where l is the length in bits of the initial message,

* finally, the length l < 2 64 of the initial message is represented with exactly 64 bits, and these bits are added at the end of the message.

The message shall always be padded, even if the initial length is already a multiple of 512.

**2.BLOCK DECOMPOSITION**

For each block M ∈ {0, 1} 512, 64 words of 32 bits each are constructed as follows:

* the first 16 are obtained by splitting M in 32-bit blocks

M = W1||W2|| · · · ||W15||W16

* the remaining 48 are obtained with the formula:

Wi = σ1(Wi−2) + Wi−7 + σ0(Wi−15) + Wi−16, 17 ≤ i ≤ 64.

**3.HASH COMPUTATION**

* First, eight variables are set to their initial values, given by the first 32 bits of the fractional part of the square roots of the first 8 prime numbers:

= 0x6a09e667 = 0xbb67ae85

= 0x3c6ef372 = 0xa54ff53a

= 0x510e527f = 0x9b05688c

= 0x1f83d9ab = 0x5be0cd19

* Next, the blocks M(1) , M(2), . . . , M(N) are processed one at a time:

For t = 1 to N

* construct the 64 blocks Wi from M(t) , as explained above
* set (a, b, c, d, e, f, g, h) = ()
* do 64 rounds consisting of:

T1 = h + Σ1(e) + Ch(e, f, g) + Ki + Wi

T2 = Σ0(a) + M aj(a, b, c)

h = g

g = f

f = e

e = d + T1

d = c

c = b

b = a

a = T1 + T2

* compute the new value of H (t) j

=+ a

= + b

= + c

= + d

= + e

= + f

= + g

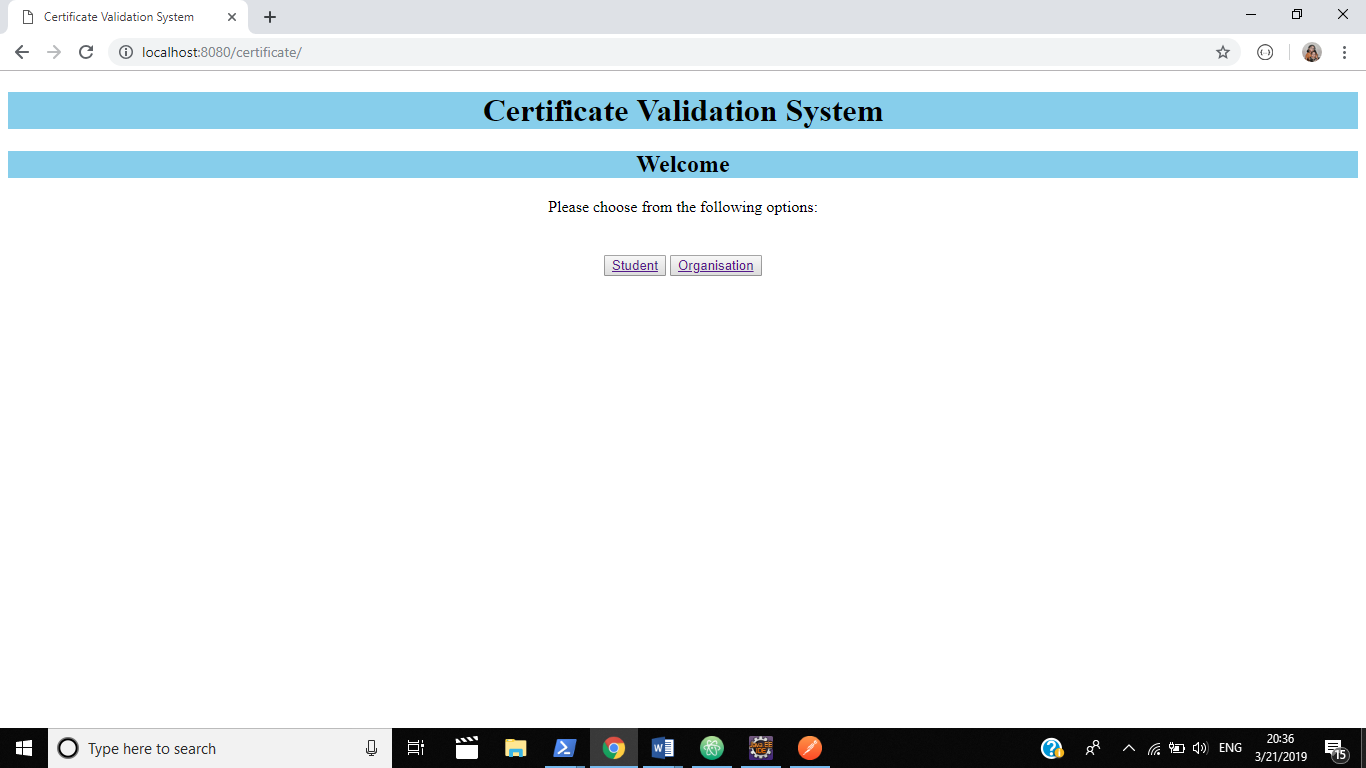
= + h

**4.END FOR**

* The hash of the message is the concatenation of the variables HN i after the last block has been processed

H =

**CHAPTER 5** **SYSTEM IMPLEMENTATION AND TESTING**



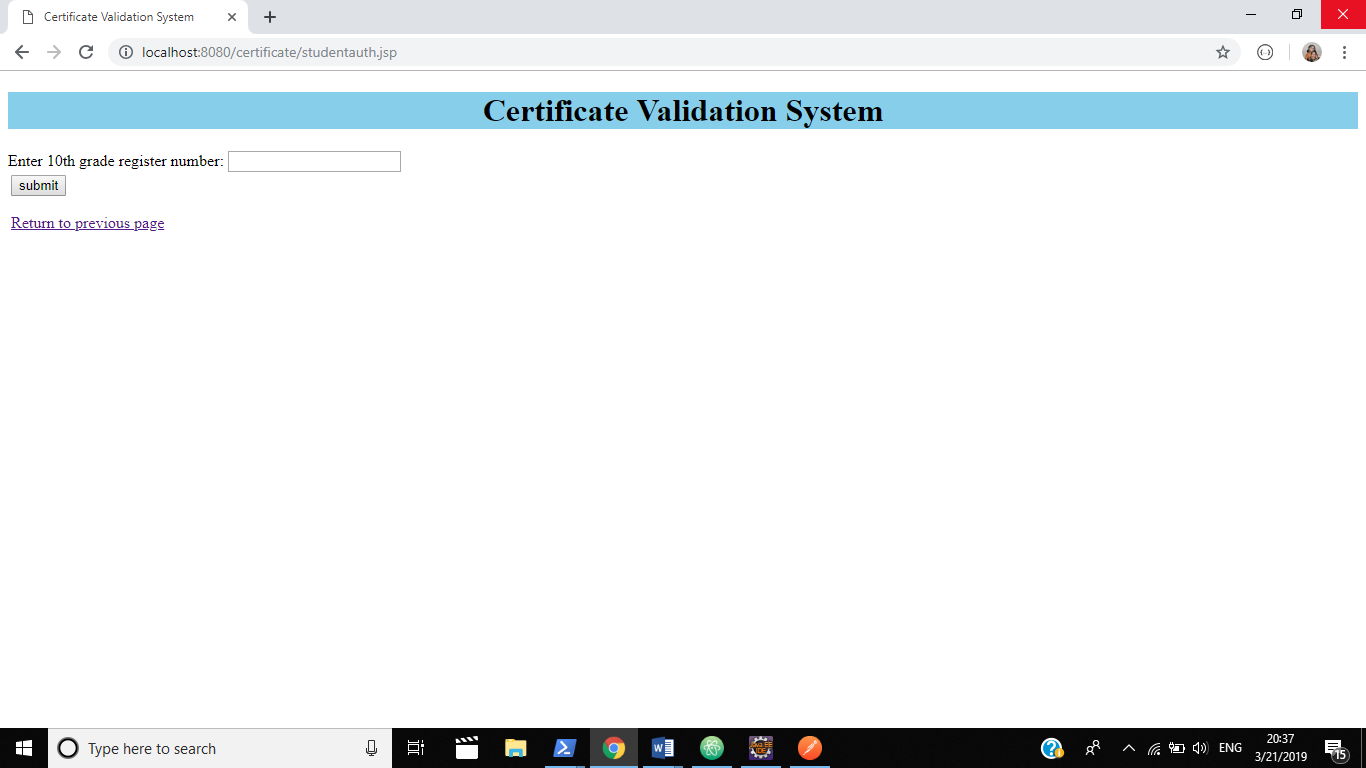
**Fig 5.1 User is asked if they are a student or an organization**

When a user enters into the system, they are displayed with the page shown in Fig5.1, where the user is prompted to choose if they are a student or an organization. Once an appropriate option is chosen, the user is directed towards the respective page.

**5.1 STUDENT INTERFACE**

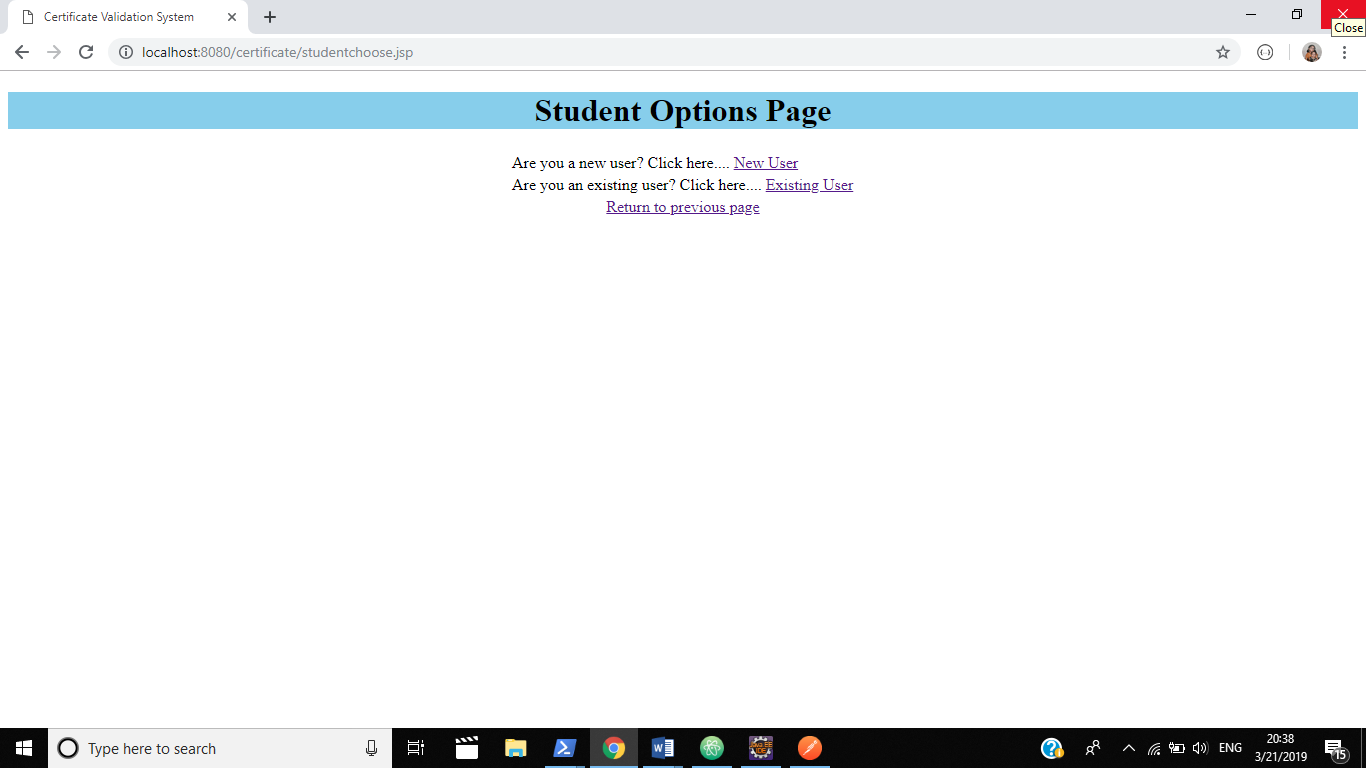
The student interface provides the student with an authentication page initially, after which the option of registering as a new user or a returning user. The student is then able to login into the system and view the certificates or upload them.

The implementation screenshots of implementation procedure are shown below:



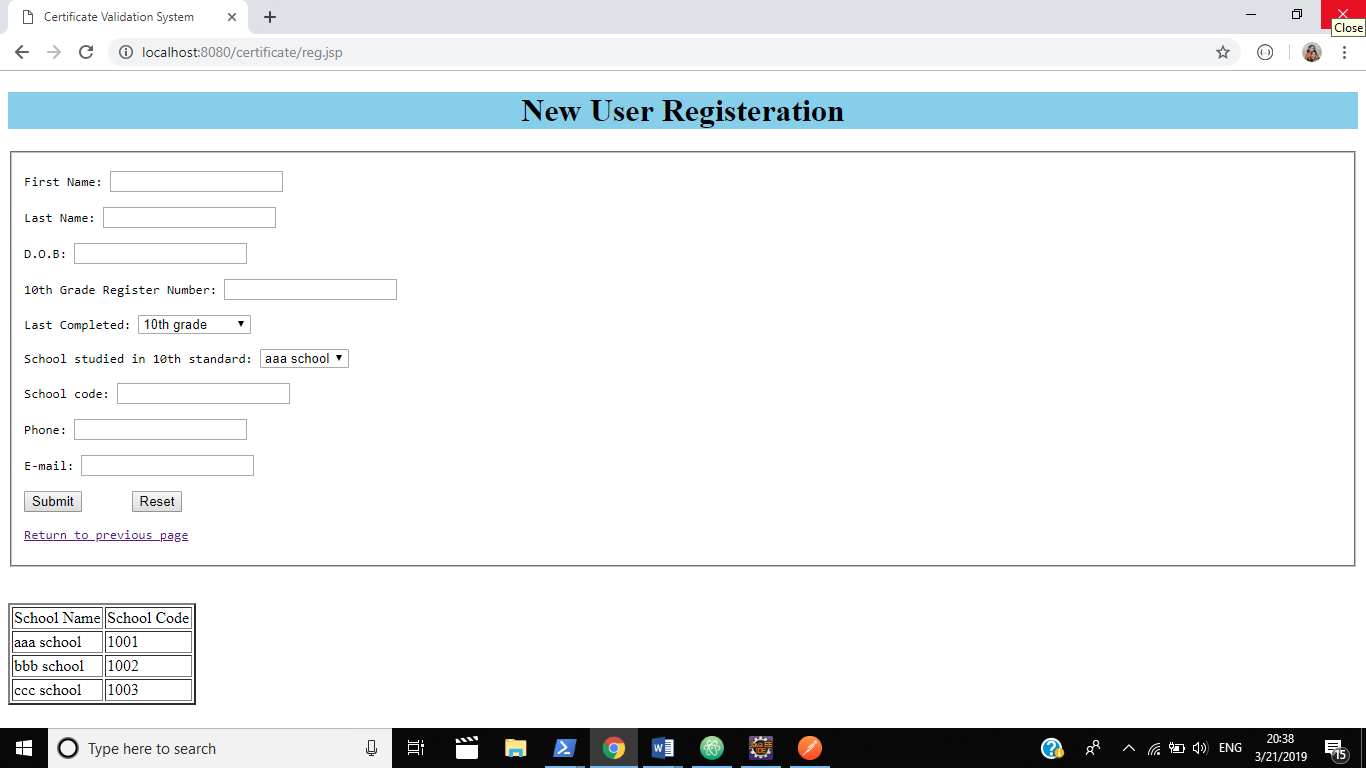
**Figure 5.2 Authentication page for the student user**

The student interface prompts the student to enter their 10th grade register number, so that it can be assured that only valid students will be able to enter into the system.



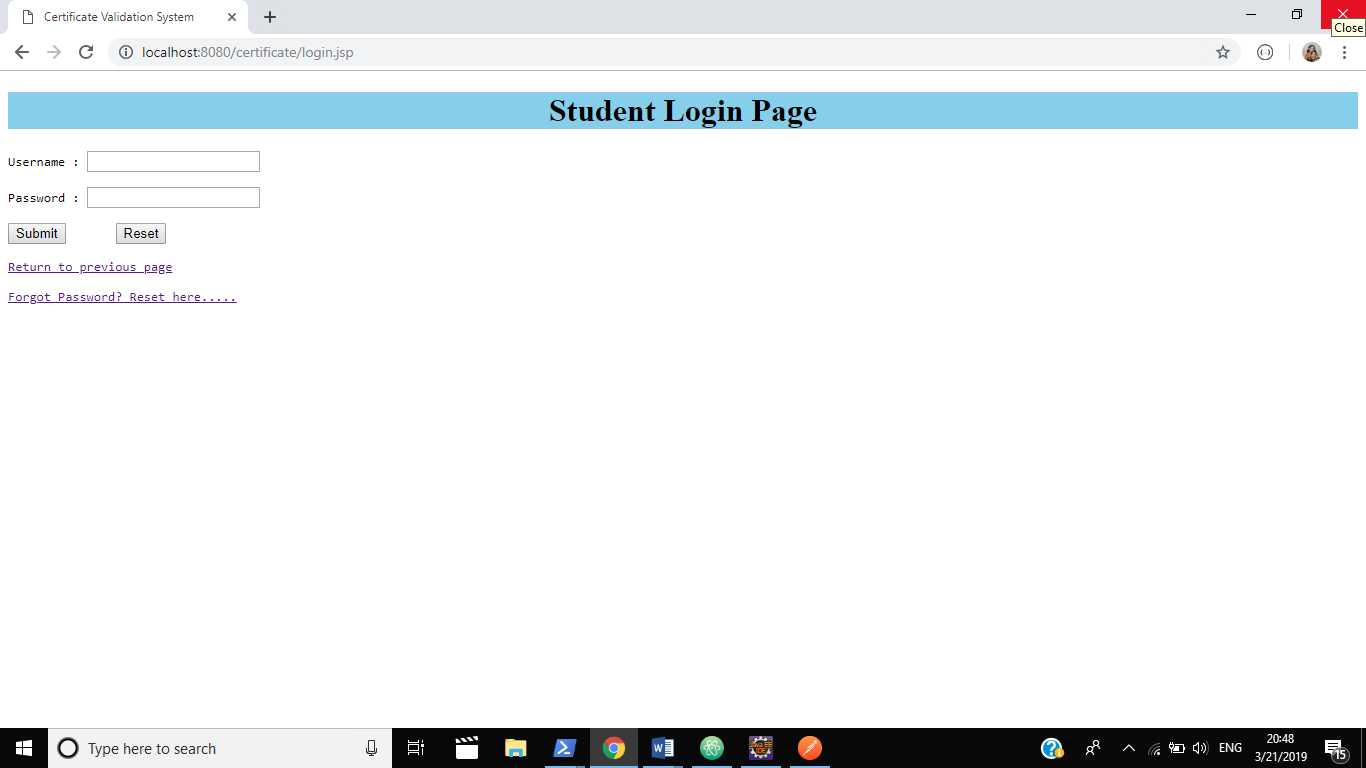
**Figure 5.3 New User or Existing User**

The user will be directed to a page where they are provided with an option of new user or an existing user. Based on the option choses, the user is directed to the next page.



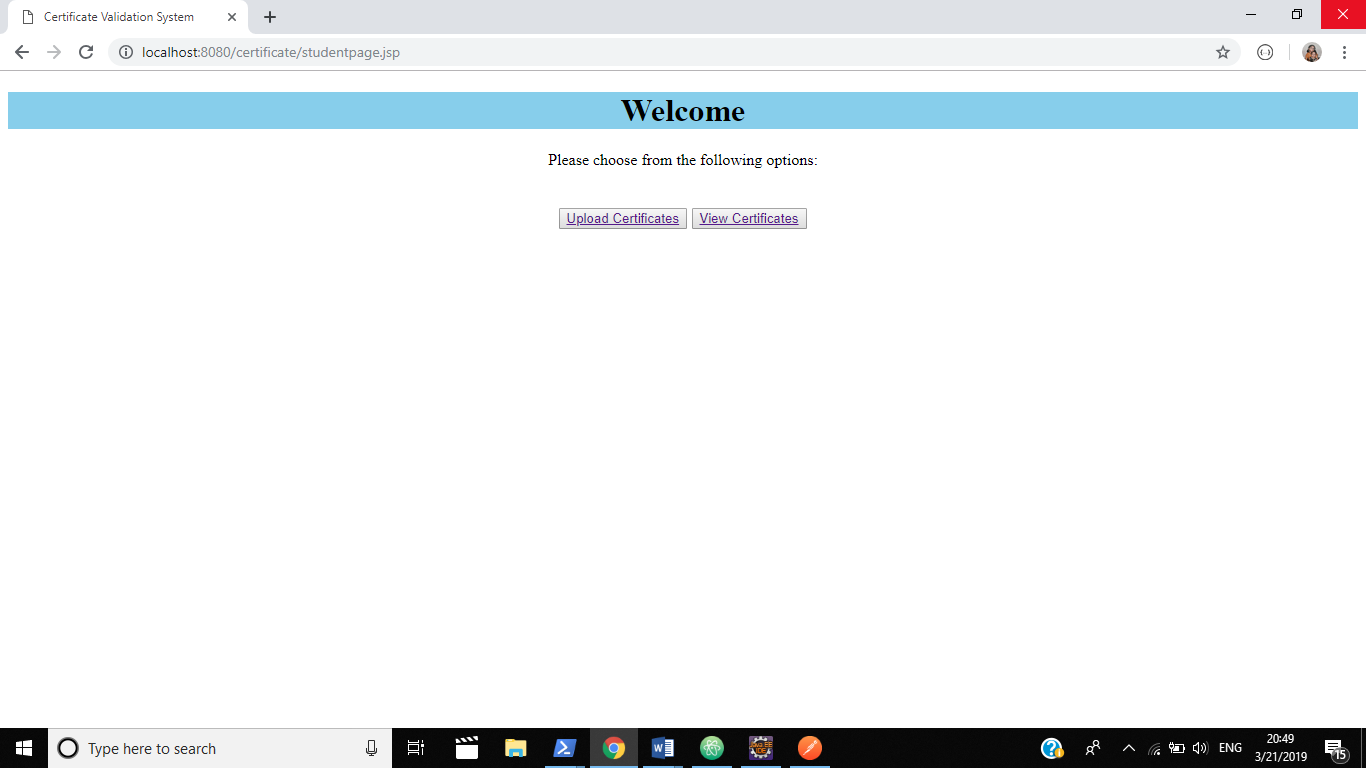
**Figure 5.4 New user registration**

On clicking the new user option, the user is led to a registration page where they are required to input information regarding their name, age, school they studied in, their phone number, and their e-mail. The school code must be entered based upon the school studied by the student in their 10th grade. The given table can be referred to for the school code.



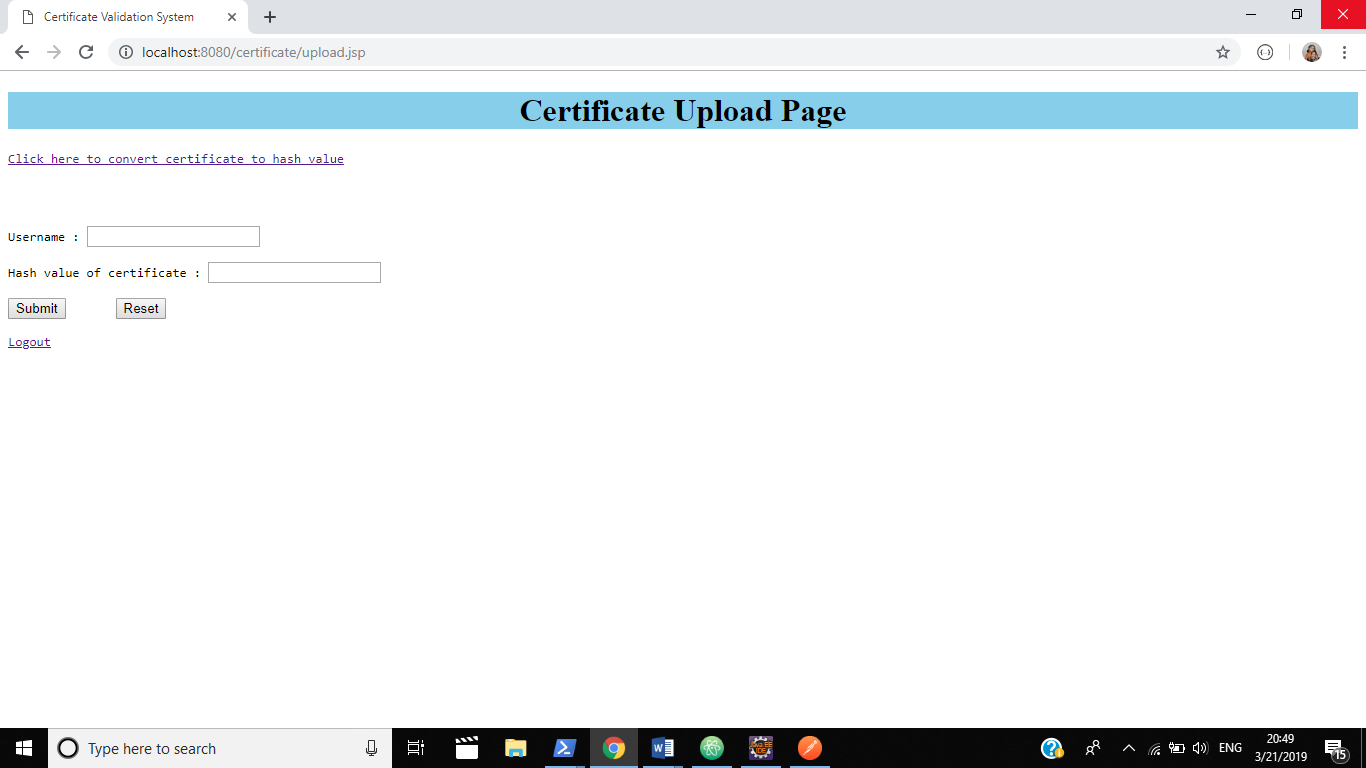
**Figure 5.5 Login page for existing users**

On clicking the existing user option, the student is directed to the login page as shown in Fig 5.5. The student needs to enter their username and password. Initially, the username and password are set as a concatenation of the school code and the student’s phone number. The password is required to be changed by the student.



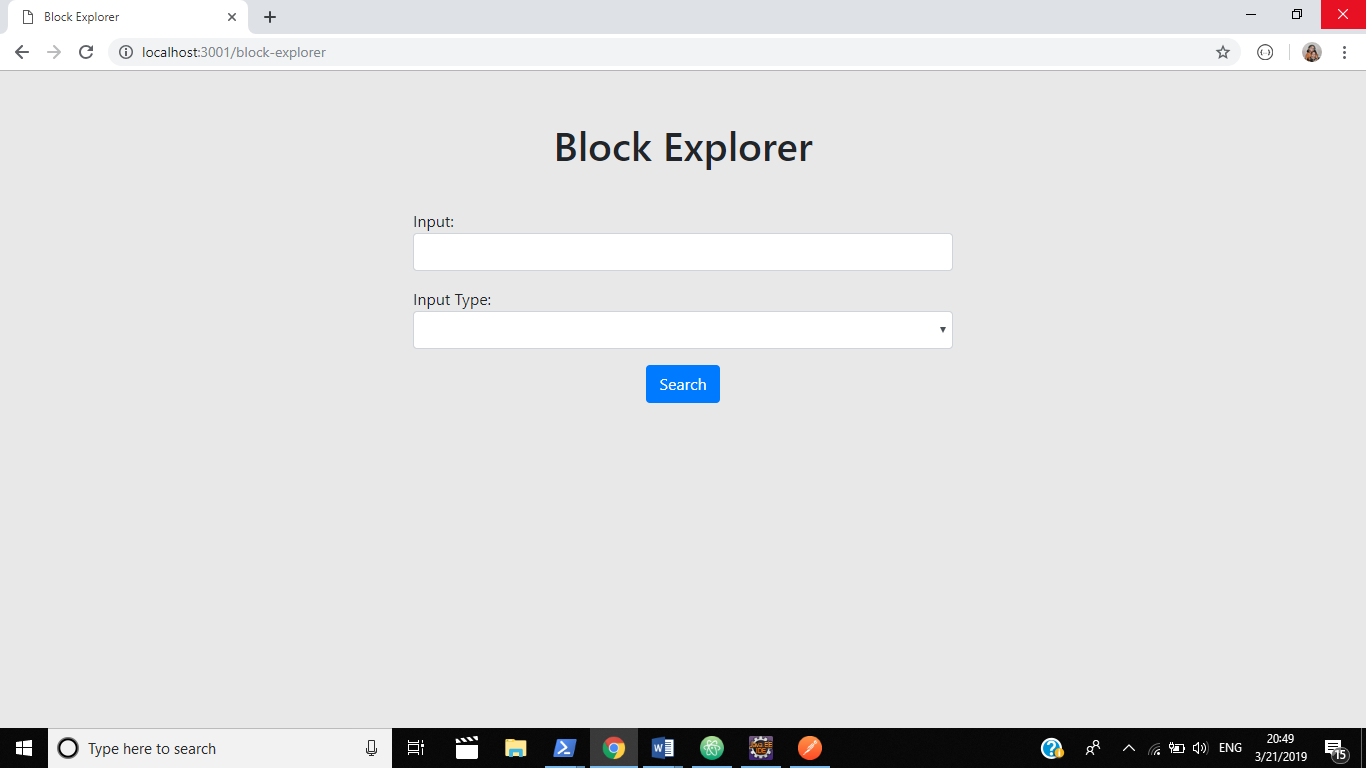
**Figure 5.6 Choose to upload or view certificate hash**

The student, once logged in, is prompted whether they want to upload their certificate hash, or view the already uploaded hashes.



**Figure 5.7 Certificate hash upload**

The student, on clicking the upload option is directed to the page shown in Fig 5.7. The student, in this page is required to input their username along with the hash id of the certificate which they want to upload.



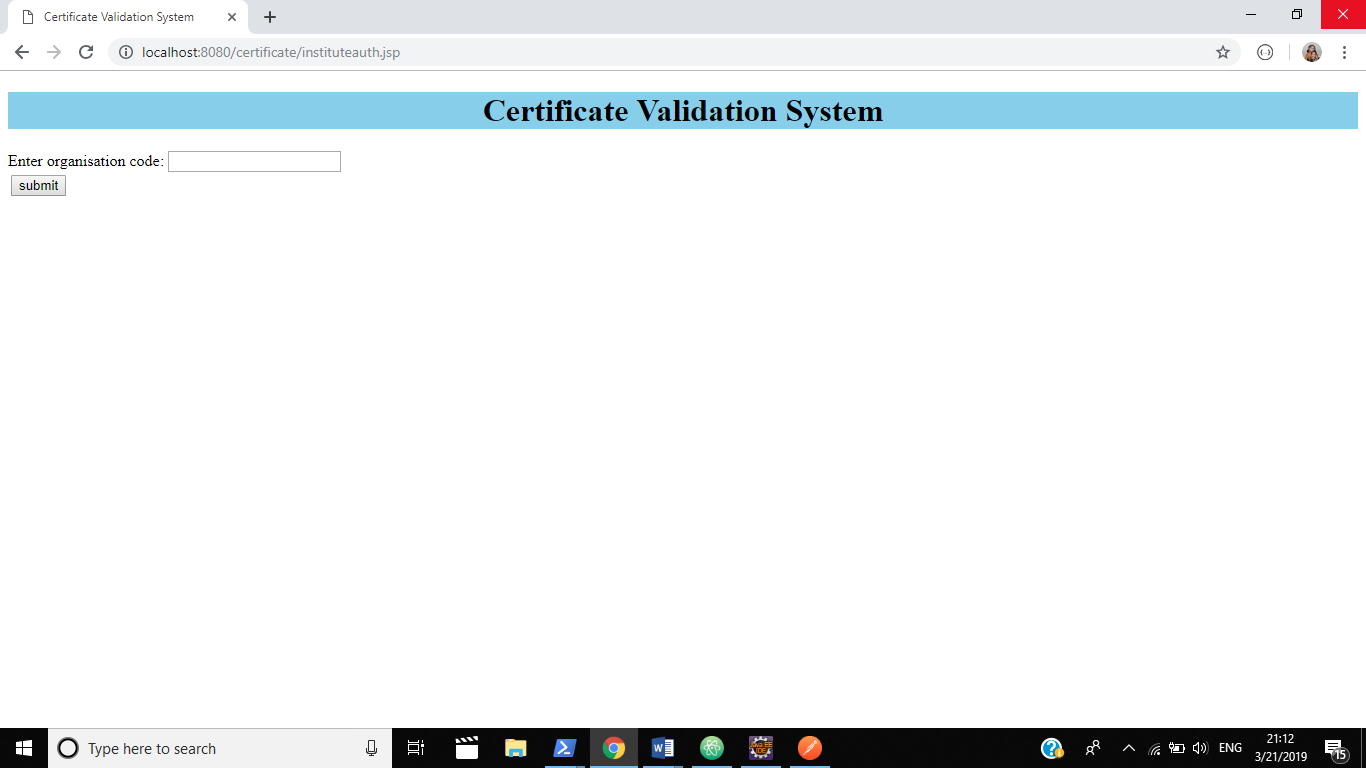
**Figure 5.8 View Uploaded hash Id**

The student, on clicking the view tab, is directed to a page, where they are prompted for their username, so that the respective hash ids may be displayed.

**5.2 ORGANIZATION INTERFACE**

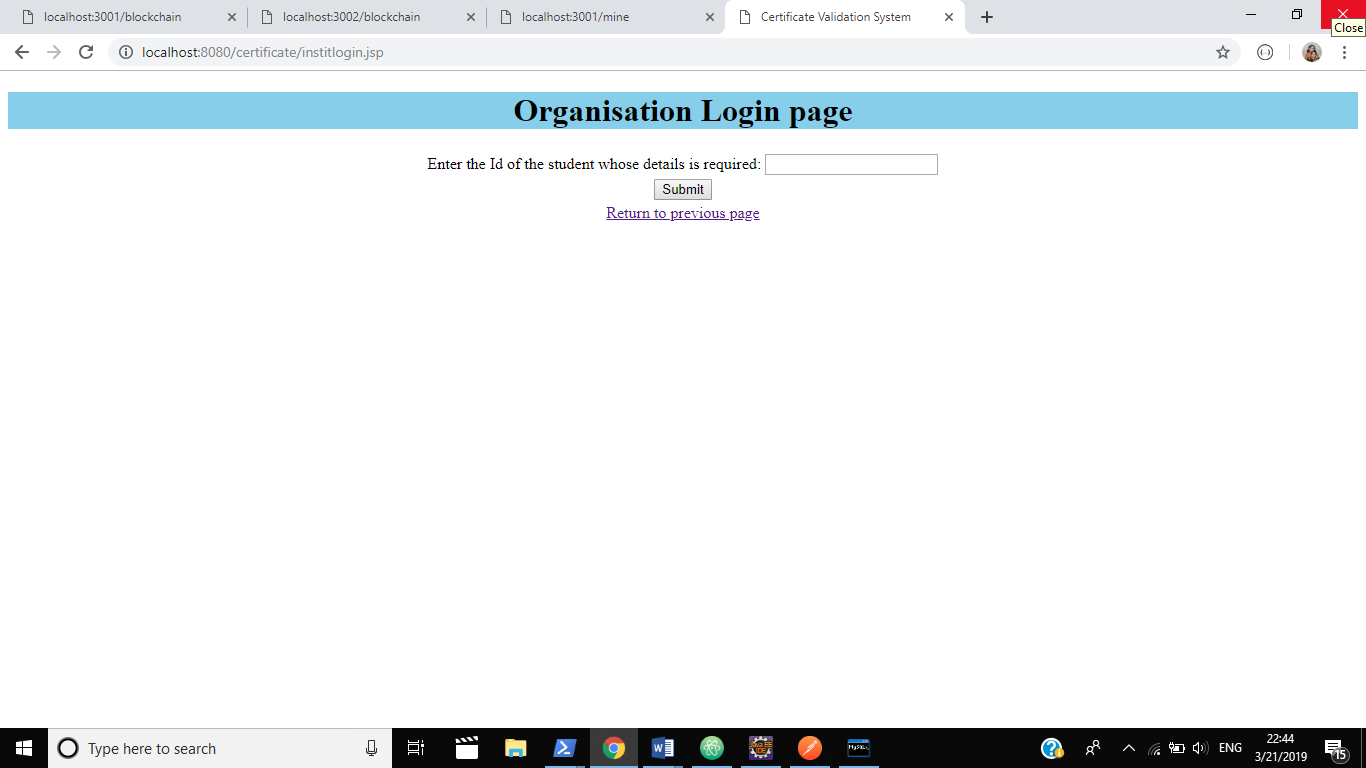
The organization interface can be accessed when the user chooses the organization tab in the welcome page. In this interface, the organization may be able to view the uploaded hashes of the students, by inputting the rite student username. The organization needs to enter the organization code to enter into the system.

The implementation screenshots of the organization interface are below:



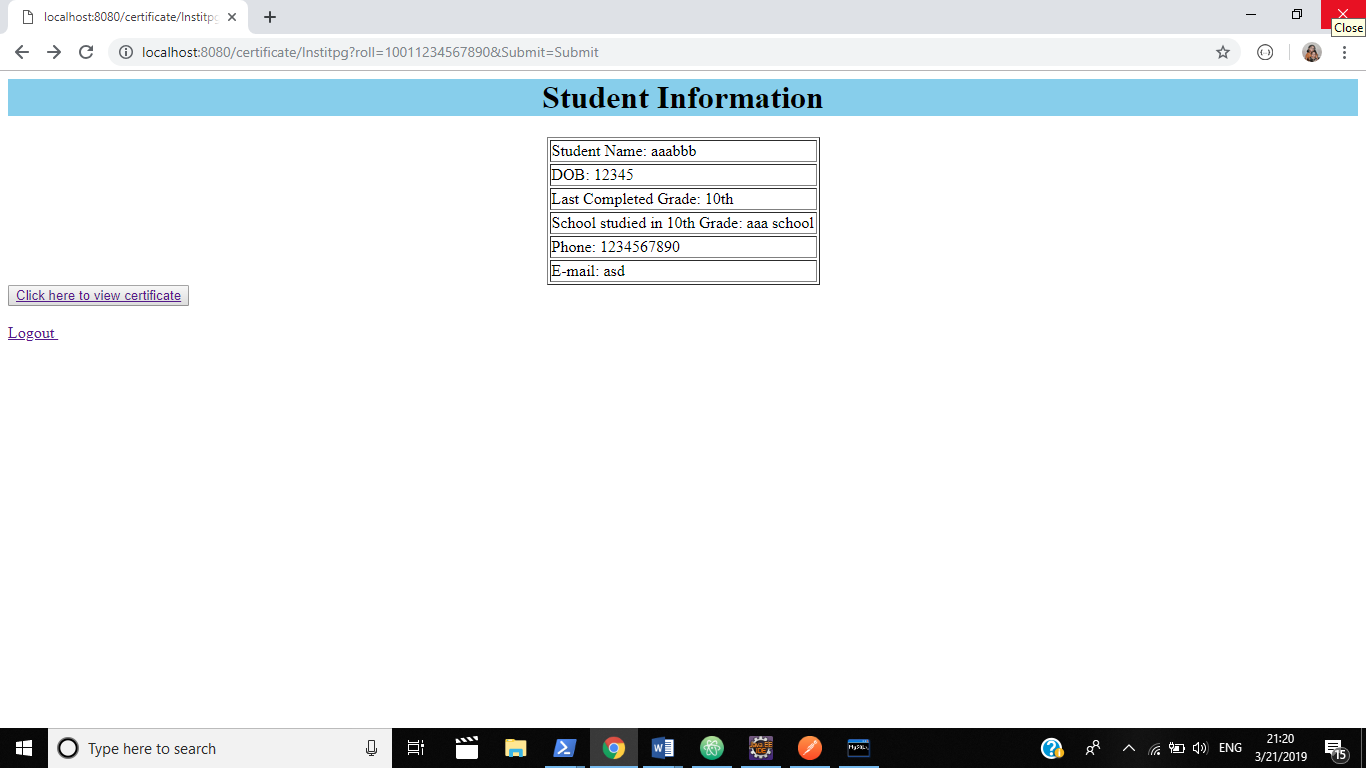
**Figure 5.9 Authentication page for Organization**

The organization needs to input their organization code to enter into the system. Without this code, the organization won’t be able to use the system or view the hashes uploaded by the students.



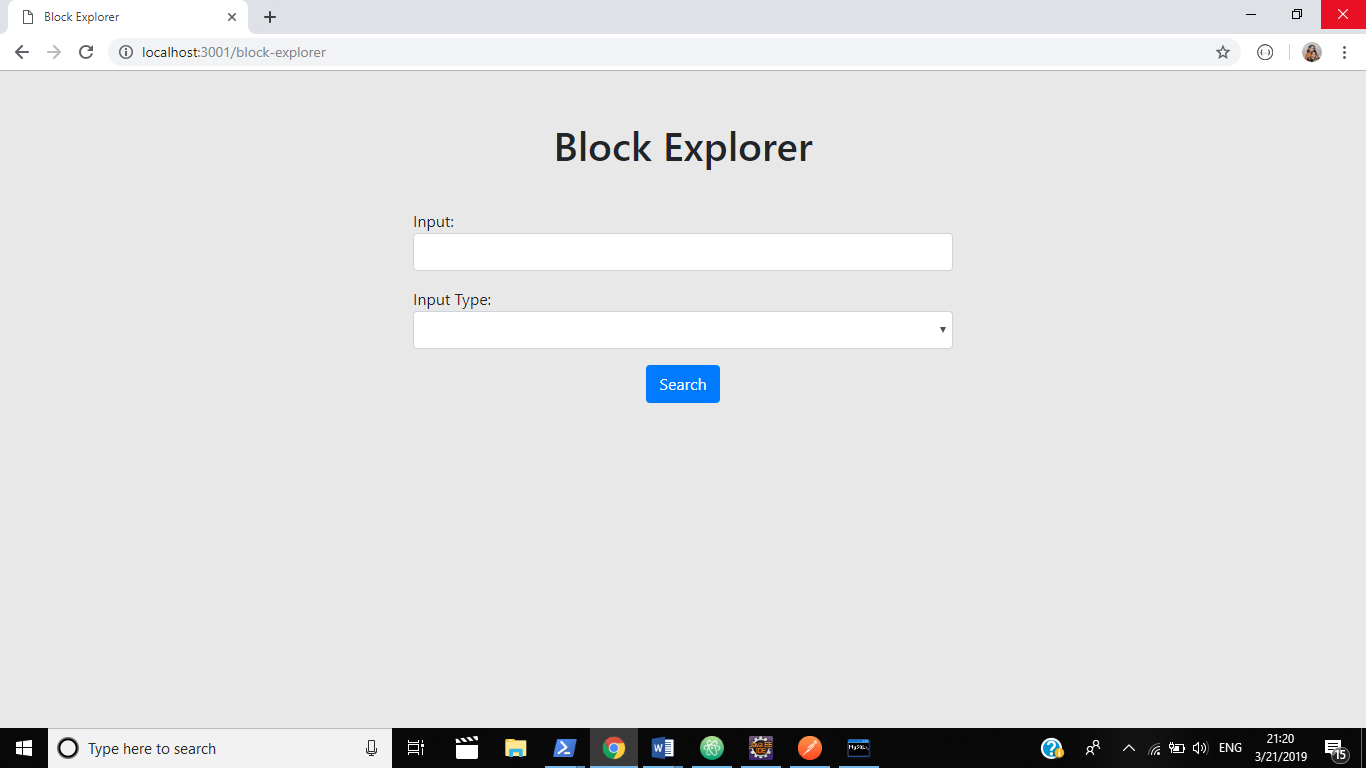
**Figure 5.10 Student Id prompt**

On entering the organization code, the user is led to a page where they are prompted to enter the student username Id.



**Figure 5.11 Student details**

The details of the student, whose username the organization has entered is displayed in the page, as shown in Fig 5.10.



**Figure 5.12 View Uploaded hash Id.**

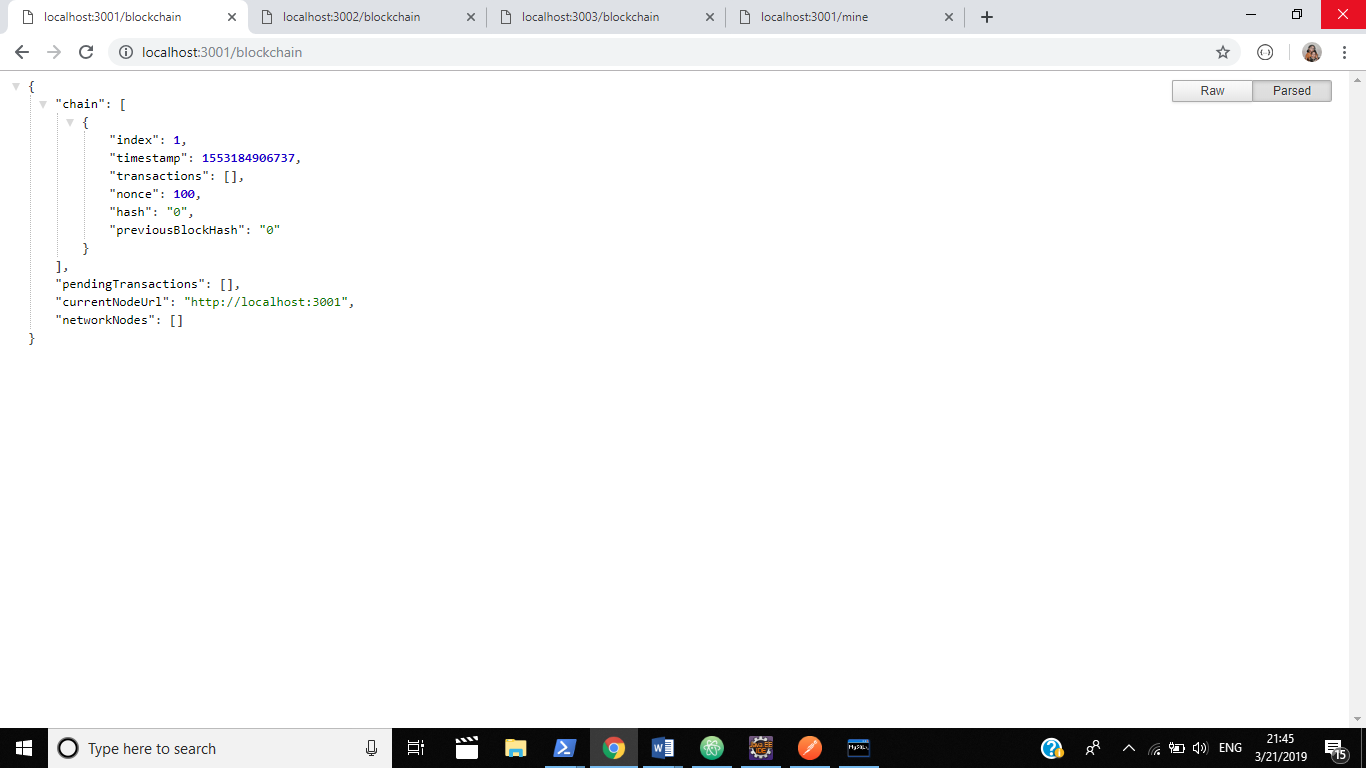
On clicking the view certificate hash tab, the user is directed toward the block explorer page where they will be able to view the hashes of the certificates uploaded by the students.

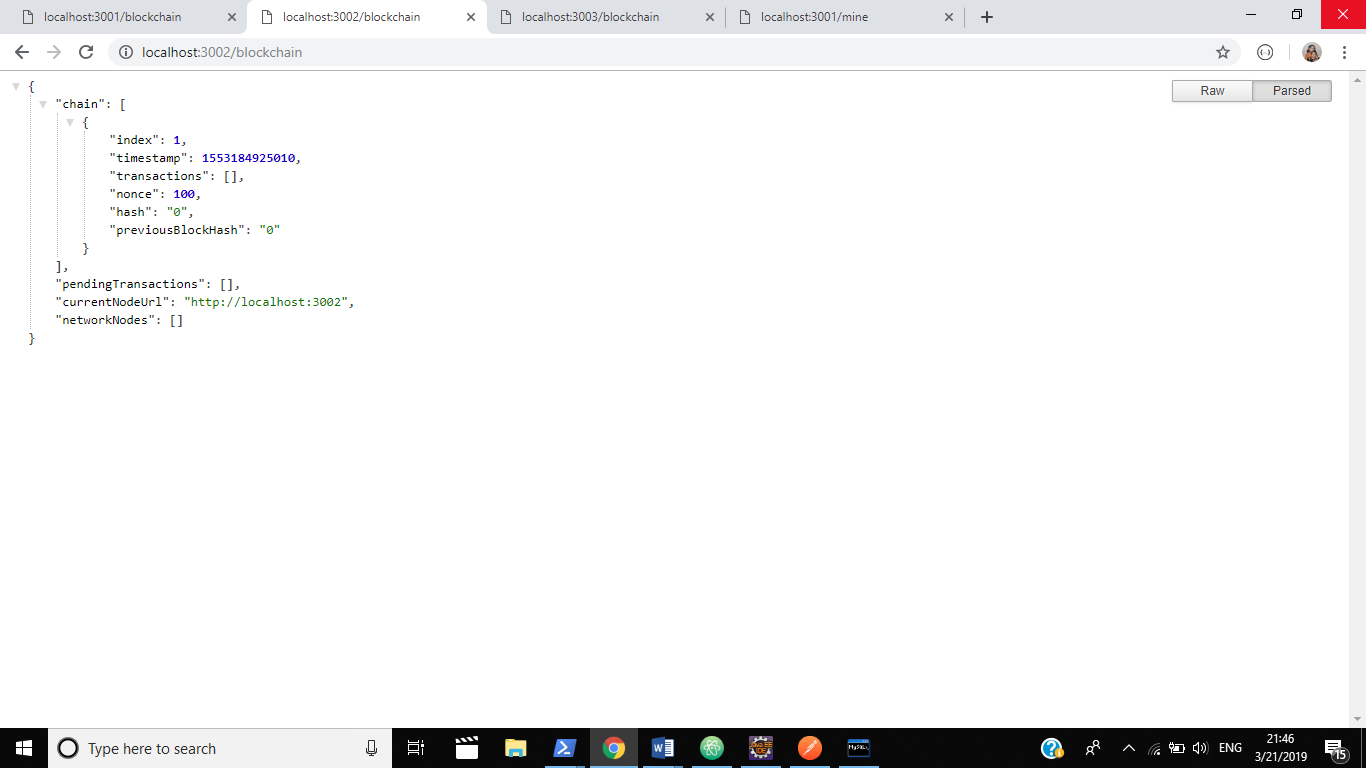
## **CHAPTER 6**

## **RESULTS AND DISCUSSION**

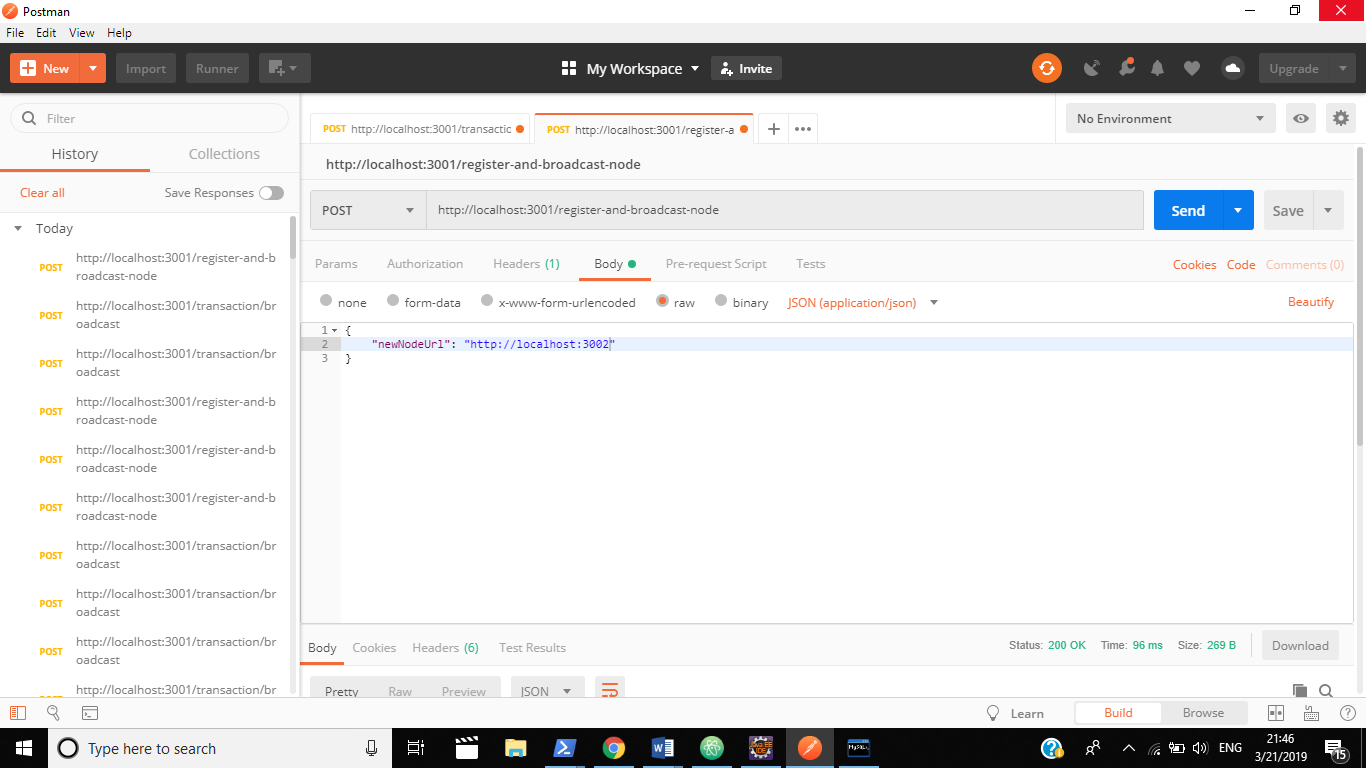
## **6.1 RESULTS**

The creation of a network and nodes, along with linking the nodes in the network, creating transactions and mining the blocks are shown in the following figures.



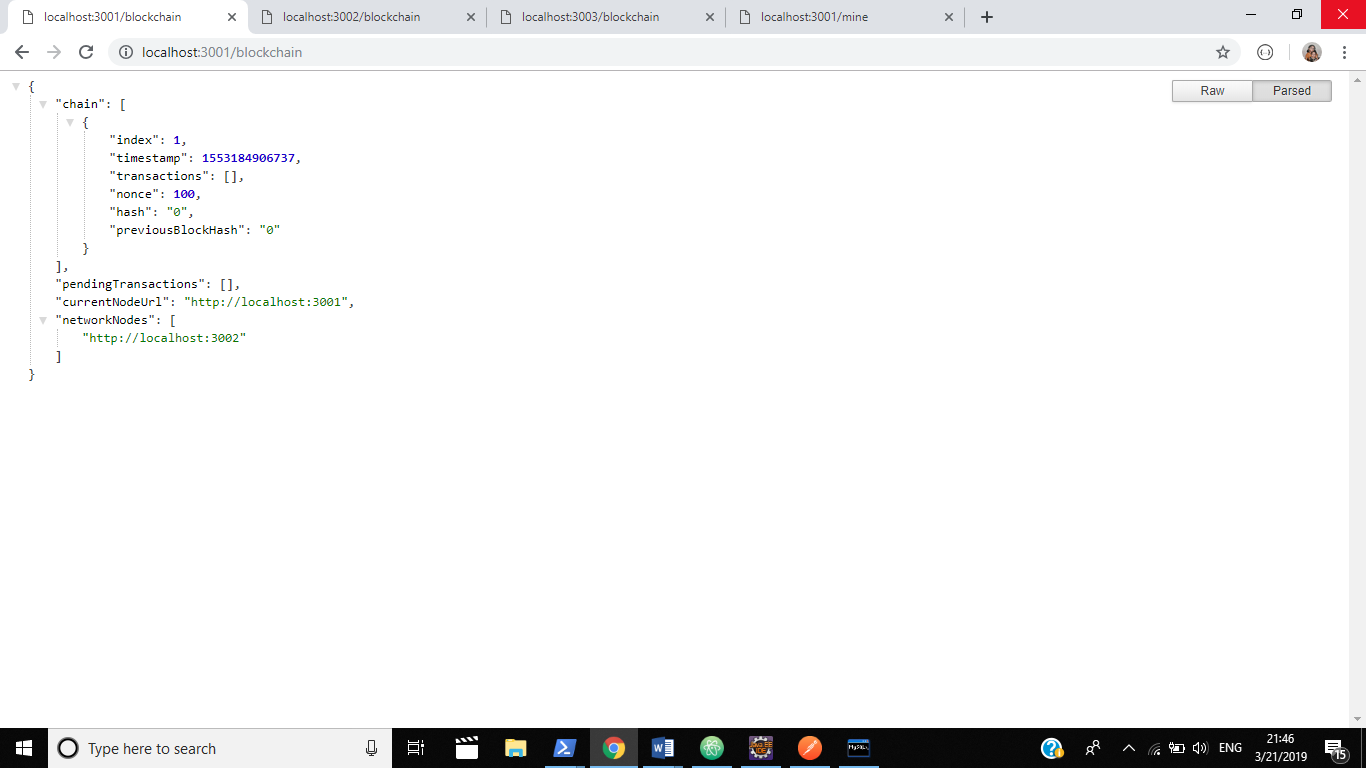


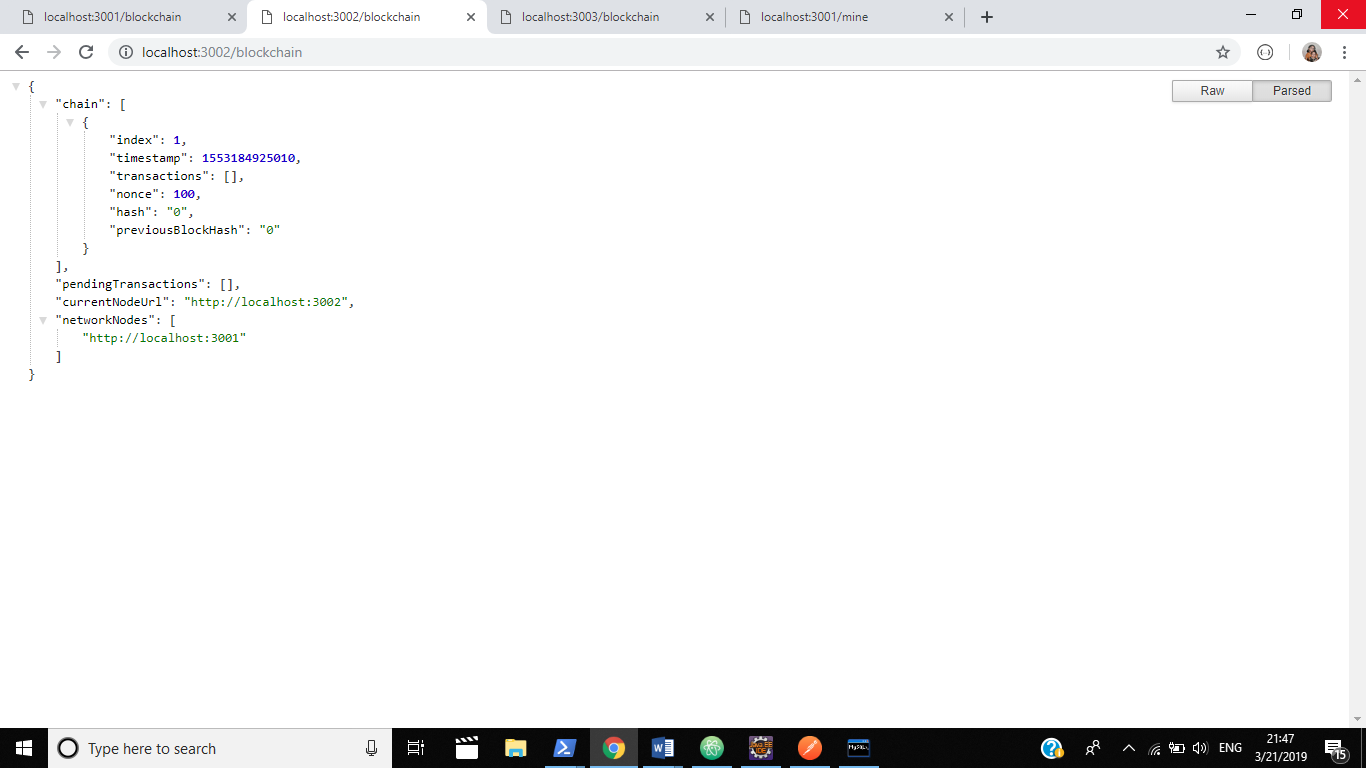
**Figure 6.1 Independent nodes in a network**



**Figure 6.2 Linking the nodes in a network through postman**

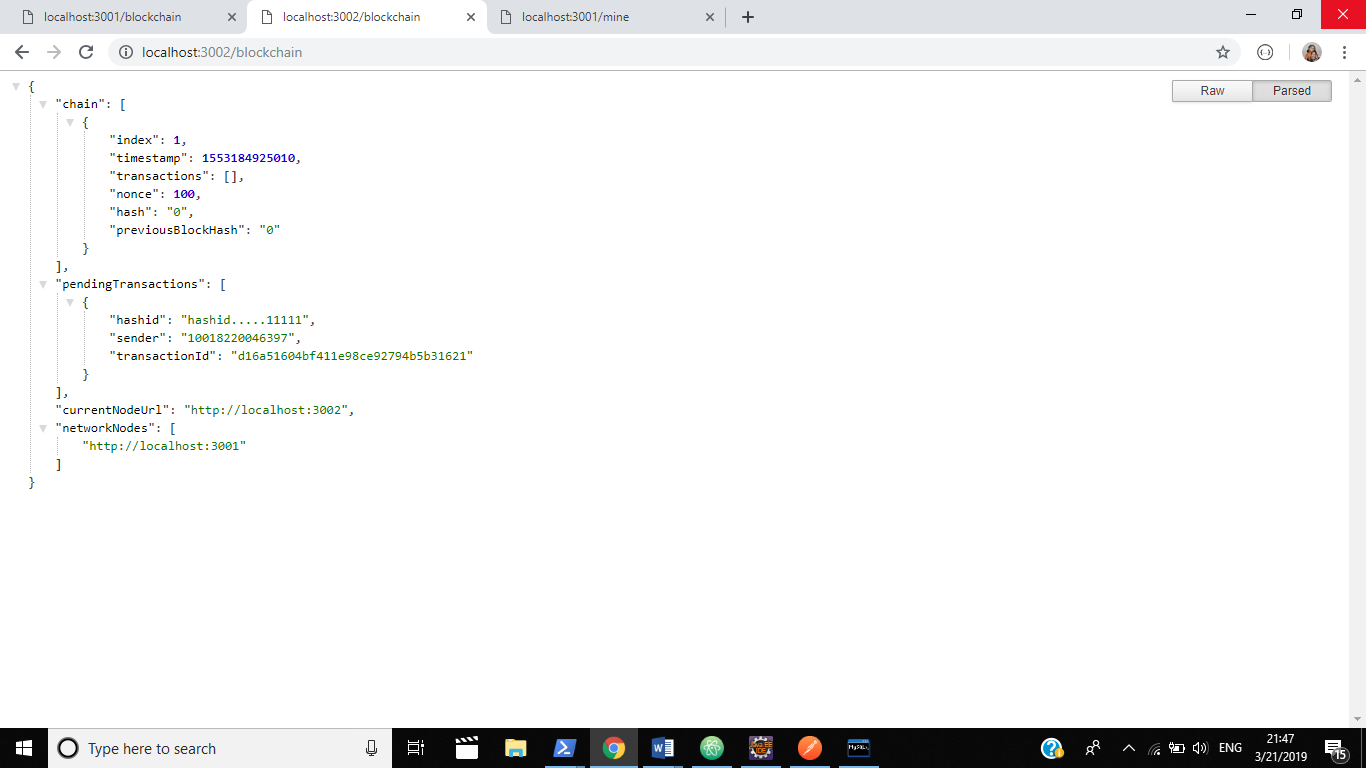
The independent nodes are linked together in the same network with the help of an application named Postman.





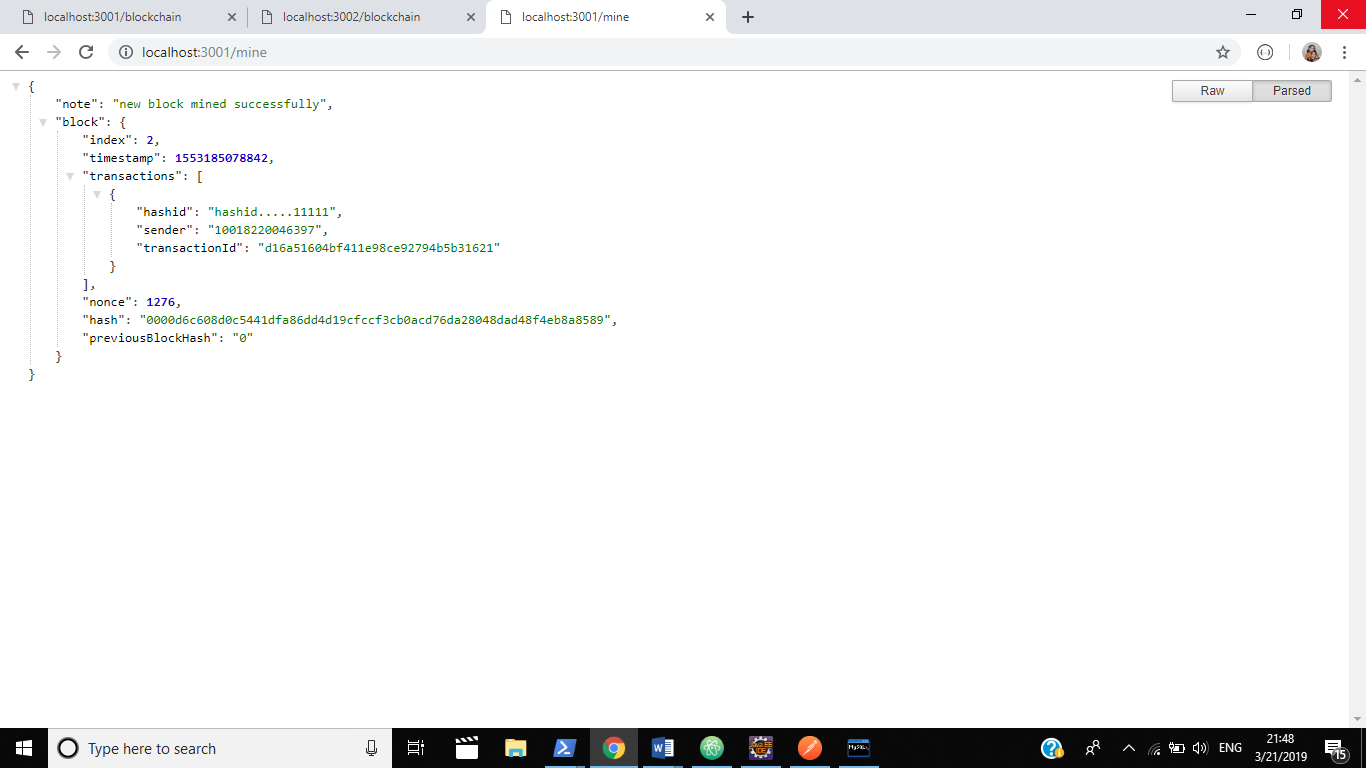
**Figure 6.3 Linked nodes in the same network**

## 



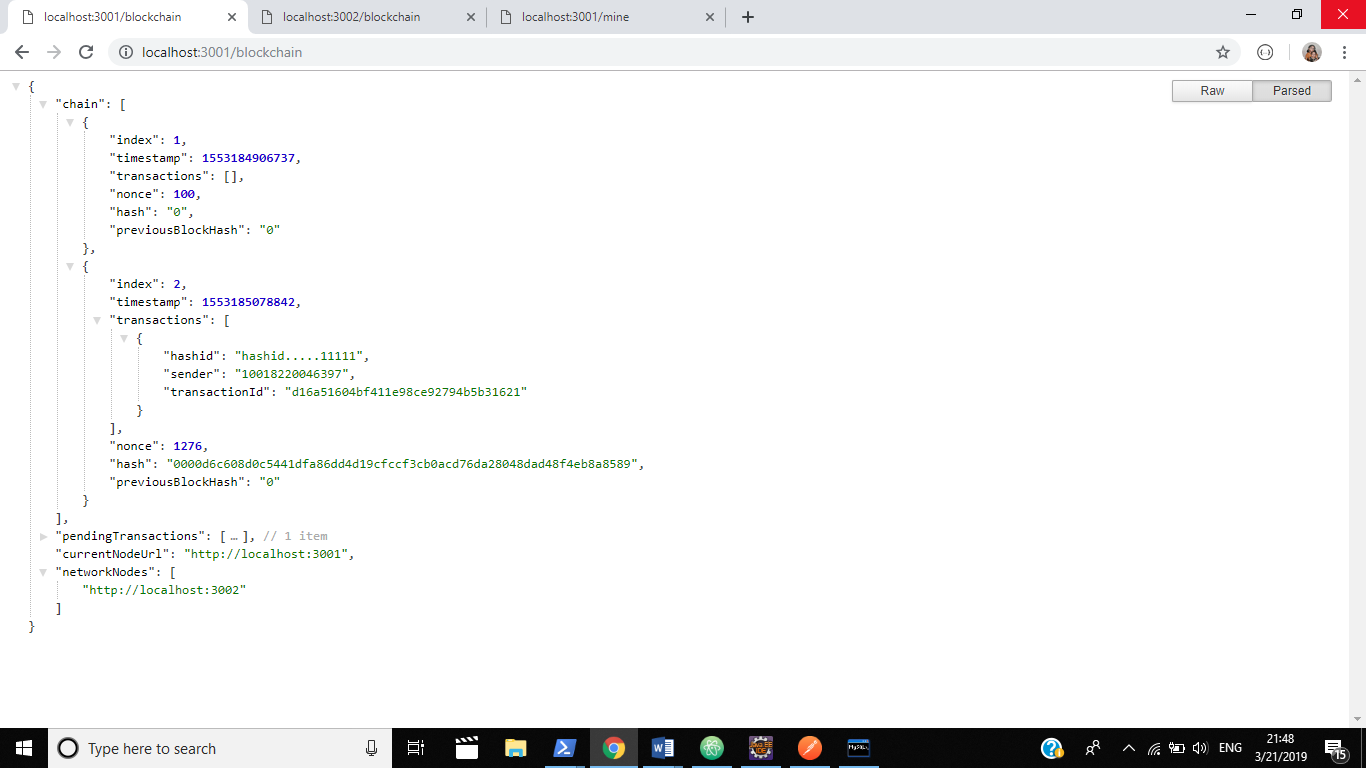
**Figure 6.4 Pending transactions in both the nodes.**

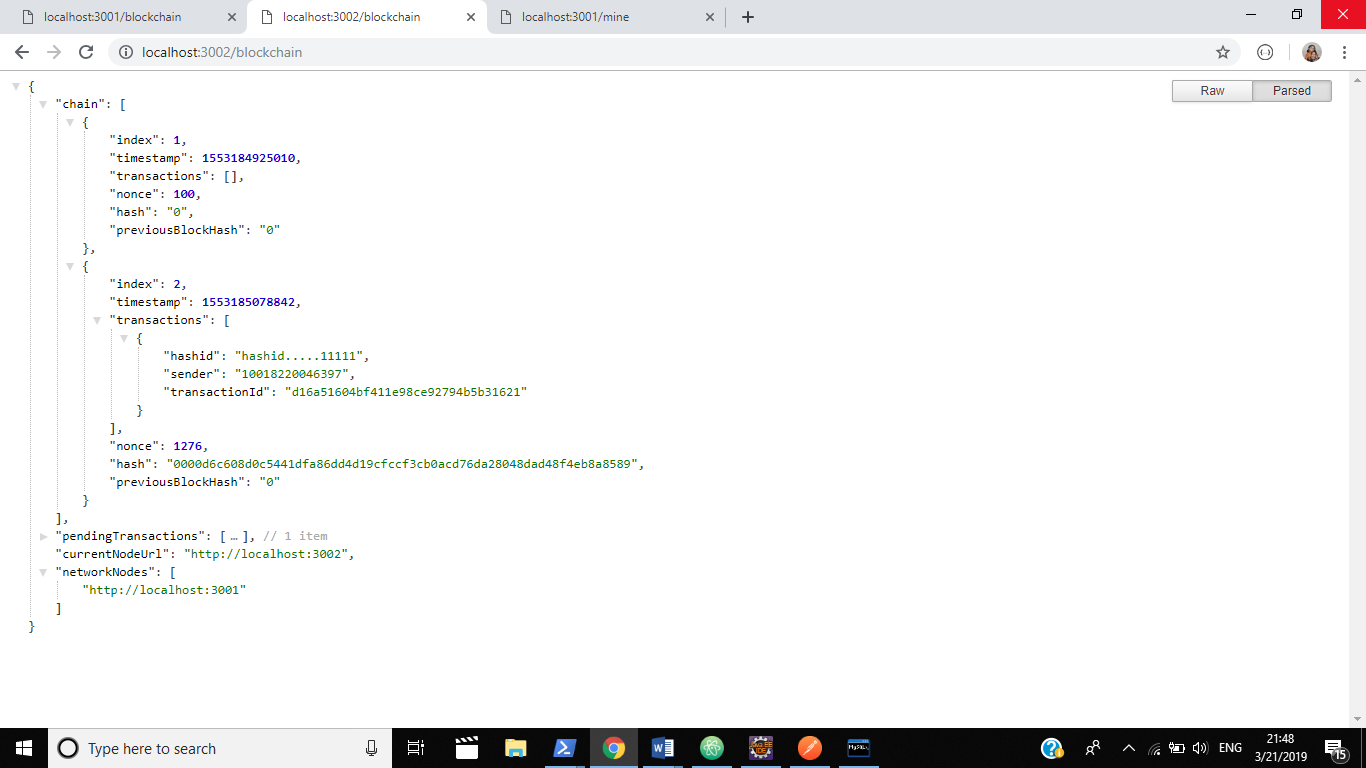
The transactions pending in the nodes are displayed. The transactions are generated by the students, who upload their certificate hashes.



**Figure 6.5 Mining of a new block**

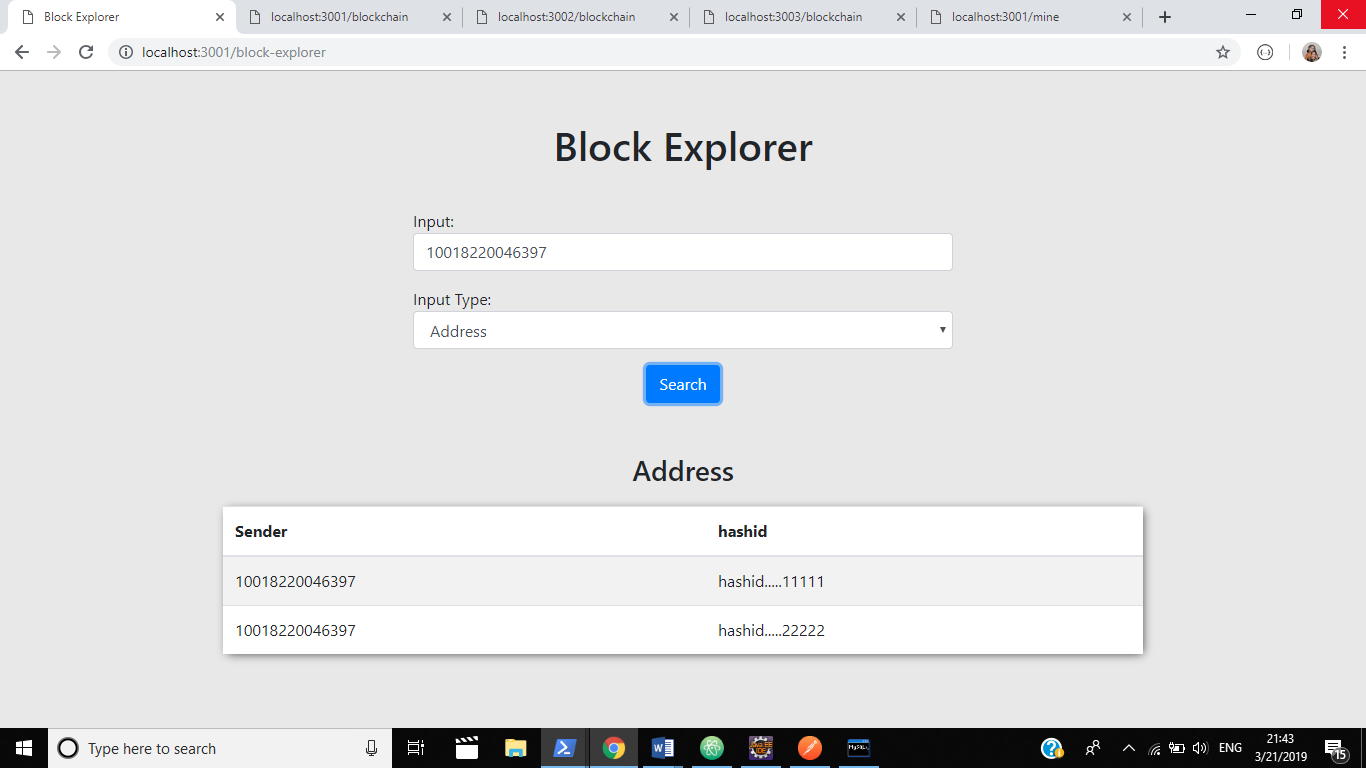
The pending transactions are maintained in an array, and when a new block is mined, the transactions are all authenticated and added in the block that was mined.





**Figure 6.6 Transactions added in the new block mined, in both the nodes**

When a new block is mined, the changes resulting are reflected in both the nodes. Thus, the blocks in both the nodes are changed, and transactions are added to the respective blocks.



**Figure 6.7 The end result for viewing the certificate hash**

The result that is displayed whenever a user wants to view the certificate hashes is shown in Fig 6.7. The username of the student, along with the hashes uploaded by the student in different blocks is displayed.

## **CHAPTER 7**

## **CONCLUSION AND FUTURE ENHANCEMENTS**

## **7.1 CONCLUSION**

We have developed a web application for the Certificate Validation system using Blockchain. The system helps the user in verifying a certificates authenticity by comparing it with the uploaded hash. Also, the system provides the provision of uploading the certificates hash. The experiments conducted for the sample data set with the proposed method produced good and the expected results.

**7.2 FUTURE ENHANCEMENTS**

* Uploading of the certificates directly along with their hashes.
* Improvement in the retrieval of data from the blocks.
* Use of different consensus algorithms.

**APPENDIX 1**

**SAMPLE CODE**

**Blockchain.js Code**

const sha256 = require('sha256');

const currentNodeUrl = process.argv[3];

const uuid = require('uuid/v1');

function Blockchain() {

this.chain = [];

this.pendingTransactions = [];

this.currentNodeUrl = currentNodeUrl;

this.networkNodes = [];

this.createNewBlock(100, '0', '0');

}

Blockchain.prototype.createNewBlock = function( nonce, previousBlockHash, hash) {

const newBlock = {

index: this.chain.length + 1,

timestamp: Date.now(),

transactions: this.pendingTransactions,

nonce: nonce,

hash: hash,

previousBlockHash: previousBlockHash

};

this.pendingTransactions = [];

this.chain.push(newBlock);

return newBlock;

};

Blockchain.prototype.getLastBlock = function() {

return this.chain[this.chain.length - 1];

};

Blockchain.prototype.createNewTransaction = function(hashid, sender, recipient) {

const newTransaction = {

hashid: hashid,

sender: sender,

recipient: recipient,

transactionId: uuid().split('-').join('')

}

return newTransaction;

};

Blockchain.prototype.addTransactiontoPendingTransactions = function(transactionObj) {

this.pendingTransactions.push(transactionObj);

return this.getLastBlock()['index'] + 1;

} ;

Blockchain.prototype.hashBlock = function(previousBlockHash, currentBlockData, nonce) {

const dataAsString = previousBlockHash + nonce.toString() + JSON.stringify(currentBlockData);

const hash = sha256(dataAsString);

return hash;

};

Blockchain.prototype.ProofOfWork = function(previousBlockHash, currentBlockData) {

let nonce = 0;

let hash = this.hashBlock( previousBlockHash, currentBlockData, nonce);

while (hash.substring(0, 4)!== '0000') {

nonce++;

hash = this.hashBlock(previousBlockHash, currentBlockData, nonce);

}

return nonce;

};

Blockchain.prototype.chainIsValid = function(blockchain) {

let validChain = true;

for (var i = 1; i < blockchain.length; i++) {

const currentBlock = blockchain[i];

const prevBlock = blockchain[i - 1];

const blockHash = this.hashBlock(prevBlock['hash'], { transactions: currentBlock['transactions'], index: currentBlock['index'] }, currentBlock['nonce']);

if (blockHash.substring(0, 4) !== '0000') validChain = false;

if (currentBlock['previousBlockHash'] !== prevBlock['hash']) validChain = false;

};

const genesisBlock = blockchain[0];

const correctNonce = genesisBlock['nonce'] === 100;

const correctPreviousBlockHash = genesisBlock['previousBlockHash'] === '0';

const correctHash = genesisBlock['hash'] === '0';

const correctTransactions = genesisBlock['transactions'].length === 0;

if (!correctNonce || !correctPreviousBlockHash || !correctHash || !correctTransactions) validChain = false;

return validChain;

};

Blockchain.prototype.getBlock = function(blockHash) {

let correctBlock = null;

this.chain.forEach(block => {

if(block.hash === blockHash) correctBlock = block;

});

return correctBlock;

};

Blockchain.prototype.getTransaction = function(transactionId) {

let correctTransaction = null;

let correctBlock = null;

this.chain.forEach(block => {

block.transactions.forEach(transaction => {

if(transaction.transactionId === transactionId) {

correctTransaction = transaction;

correctBlock = block;

};

});

});

return {

transaction: correctTransaction,

block: correctBlock

};

};

Blockchain.prototype.getAddressData = function(address) {

const addressTransactions = [];

this.chain.forEach(block=> {

block.transactions.forEach(transaction => {

if(transaction.sender === address || transaction.recipient === address) {

addressTransactions.push(transaction);

};

});

});

let balance = 0;

addressTransactions.forEach(transaction => {

if(transaction.recipient === address) balance += transaction.hashid;

else if(transaction.sender === address) balance -= transaction.hashid;

});

return {

addressTransactions: addressTransactions,

addressBalance: balance

}

};

module.exports = Blockchain;

**Package.json code**

{

"name": "project1",

"version": "1.0.0",

"description": "",

"main": "index.js",

"scripts": {

"test": "echo \"Error: no test specified\" && exit 1",

"node\_1": "nodemon --watch dev -e js dev/networkNode.js 3001 http://localhost:3001",

"node\_2": "nodemon --watch dev -e js dev/networkNode.js 3002 http://localhost:3002",

"node\_3": "nodemon --watch dev -e js dev/networkNode.js 3003 http://localhost:3003",

"node\_4": "nodemon --watch dev -e js dev/networkNode.js 3004 http://localhost:3004",

"node\_5": "nodemon --watch dev -e js dev/networkNode.js 3005 http://localhost:3005"

},

"author": "",

"license": "ISC",

"dependencies": {

"body-parser": "^1.18.3",

"express": "^4.16.4",

"nodemon": "^1.18.10",

"request": "^2.88.0",

"request-promise": "^4.2.4",

"sha256": "^0.2.0",

"uuid": "^3.3.2"

}

}

## **REFERENCES**

1. Azzah Al Ghamdi and Thomas Thomson, “*The Future of Data Storage: A Case Study with the Saudi Company*”, published in Journal of Electrical and Electronic Engineering in 2018.
2. Borlick et al. United States Patent “Patent No.: US 10, 171, 585 B2” Date of Patent: Jan. 1, 2019.
3. Xiaoqi Lia, Peng Jiang, Ting Chen, Xiapu Luo, Qiaoyan Wen, *“A Survey on the Security of Blockchain Systems”*, in arXiv:1802.06993v1 [cs.CR] 20 Feb 2018.
4. Wasim Ahmad Bhat*,“Capacity barriers in hard disks: problems, solutions and lessons”* in 2018. <https://doi.org/10.1007/s41870-018-0144-x> .
5. Government of India, Ministry of Railways, Railway Board has passed the *“Digital Signature Certificate”- No. 2017.RBCC/7/10/e-Office/DSC Policy for IR,* Policy on 21.07.2017.
6. Wasim Ahmad Bhat, (2018) "*Long-term preservation of big data: prospects of current storage technologies in digital libraries*", Library Hi Tech, Vol. 36 Issue: 3, pp.539-555, https://doi.org/10.1108/ LHT-06-2017-0117.
7. Samit Shivadekar, Stephen Raj Abraham, Sheikh Khalid*, “Document Validation and Verification System”* in International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 5 Issue 3, March 2016 .
8. Chen R., Mu Y., Yang G., Susilo W., Guo F. (2016) *“Strongly Leakage-Resilient Authenticated Key Exchange”*. In: Sako K. (eds) Topics in Cryptology - CT-RSA 2016. CT-RSA 2016. Lecture Notes in Computer Science, vol 9610. Springer, Cham.
9. Tongjin Lee and Jun Iio. *“MAD4SA: a prototype of document management system for small-scale archives”*, in Int. J. Space-Based and Situated Computing, Vol. 6, No. 2, 2016 Copyright © 2016 Inderscience Enterprises Ltd.
10. Nwachukwu-Nwokeafor K.C and Igbajar Abraham *“Designing An Automatic Web-Based Certificate Verification System For Institutions”*, in Journal of Multidisciplinary Engineering Science and Technology (JMEST) ISSN: 3159-0040 Vol. 2 Issue 12, December – 2015 www.jmest.org JMESTN42351206 3393
11. Abhishek Roy and Sunil Karforma, “*A Survey on Digital Signatures and its Applications*”, in J. of Comp. and I.T. Vol. 3(1&2), 45-69 (2012).
12. “*An Academic Certification Verification System Based on Cloud Computing Environment*” by Nicholas Mwankiki Musee in University of Nairobi. P58/63483/2011
13. George D. Gollin ,“*Verification of the integrity and legitimacy of academic credential documents in an international setting*” available in the site <http://www.hep.uiuc.edu/home/g-gollin/CUJ8404AR_Gollin.pdf> .