CERTIFICATE VALIDATION USING BLOCK CHAIN

A project report submitted to

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT (GITAM)

A project Report submitted in partial fulfilment of the requirements for the

award of the degree of

BACHELOR OF TECHNOLOGY IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

L Mrudula	221810311014
Saadhvi Cheruku	221810311025
P Phani Satya Sai	221810311021
K.M.Deekshitha	221810311053

Under the esteemed guidance of

Mr VINAY KUMAR JOSHI

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF TECHNOLOGY AND MANAGEMENT (GITAM)

HYDERABAD CAMPUS

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING GITAM

INSTITUTE OF TECHNOLOGY

GITAM

(Deemed to be University)

DECLARATION

We, hereby declare that the internship report entitled “Certificate Validation Using Blockchain ” is an original work done in the Department of Computer Science and Engineering, GITAM Institute of Technology, GITAM (Deemed to be University) submitted in partial fulfilment of the requirements for the award of the degree of B.Tech. in Computer Science and Engineering. The work has not been submitted to any other college or University for the award of any degree or diploma.

Date: NOV 2021

Registration No(s).	**Name(s)**
221810311014	L.Mrudula
221810311025	Saadhvi Cheruku
221810311021	P.Phani Satya Sai
221810311053	K.M.Deekshitha

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF TECHNOLOGY

GANDHI INSTITUTE OF

TECHNOLOGY AND MANAGEMENT

(GITAM)

(Declared as Deemed-to-be-University u/s 3 of UGC Act 1956)

HYDERABAD CAMPUS

CERTIFICATE

This is to certify that the seminar report "WEB DEVELOPMENT" submitted by L Mrudula (221810311014), Saadhvi Cheruku (221810311025), P Phani Satya Sai (221810311021), K.M.Deekshitha (221810311053) in partial fulfilment of the requirements for the Internship in "Bachelor of Technology" in "Computer Science and Engineering".

Guided by

Mr VINAY KUMAR JOSHI

(Assistant Professor)

Head of Department

Dr. PHANI KUMAR

(professor & HOD)

ACKNOWLEDGEMENT

Our project would not have been successful without the help of several people. We would like to thank the personalities who were part of our project in numerous ways, those who gave us outstanding support from the birth of the project.

We are extremely thankful to our honourable Pro-Vice Chancellor, **Prof. N. Siva Prasad** for providing necessary infrastructure and resources for the accomplishment of our project.

We are highly indebted to **Prof. N. Seetharamaiah**, Principal, School of Technology, for his support during the tenure of the project.

We are very much obliged to our beloved **Prof. S. Phani Kumar,** Head of the Department of Computer Science & Engineering for providing the opportunity to undertake this project and encouragement in completion of this project.

We hereby wish to express our deep sense of gratitude to **Mr Vinay Kumar Joshi** Assistant Professor, Department of Computer Science and Engineering, School of Technology for the esteemed guidance, moral support and invaluable advice provided by them for the success of the project.

We are also thankful to all the staff members of Computer Science and Engineering department who have cooperated in making my project a success. We would like to thank our parents and friends who extended their help, encouragement and moral support either directly or indirectly in our project work.

Sincerely,

L.Mrudula (221810311014)

Saadhvi Cheruku (221810311025)

P.Phani Satya Sai (221810311021)

K.M.Deekshitha (221810311053)

CONTENTS

ABSTRACT

In India, about one million students graduate each year, some of them will go for higher studies or tertiary institutions, and some will be ready to enter the workplace employment. During the course of study, the student's certificates, score transcripts, etc., will become an important reference for admitting into new schools or new works. As schools make various certificates or transcripts, only the names of the schools and the students are input. Due to the lack of effective anti-forge mechanism, Counterfeit academic certificates have been a longstanding issue in the academic community. In order to solve the problem of counterfeiting certificates, the digital certificate system based on blockchain technology would be proposed. Our project will help to store the certificate in the blockchain system and provide security. First, the paper certificates are converted into digital certificates. The chaotic algorithm is used to generate the hash code value for the certificate. Then the certificates are store in blockchain. And these certificates are validated.

LIST OF FIGURES

LIST OF SCREENS

1.INTRODUCTION

Blockchain was introduced in the year 2008 by Satoshi Nakamoto.A block is an individual transaction or piece of data that is being stored within the blockchain. A blockchain is a continuously growing list (“chain”) of records (“block”), called blocks, which are linked chronologically and secured using cryptography. Blockchain is one of the online ledgers which provide decentralized and transparent data sharing. Decentralization is a concept in which participants work together to validate transactions without relying on a central authority. A block contains Timestamp, the time the block is created determines the location of it on the blockchain, Transaction Data, the information to be securely stored in the block, Hash, a unique code produced by combining all the contents within the block itself also known as a digital fingerprint, Previous Hash, each block has a reference to the block prior to its hash. This is what makes the blockchain unique because this link will be broken if a block is tampered with. Data are distributed among various nodes (the distributed data storage) and are thus decentralized. Consequently, the nodes maintain the database together. Under blockchain, a block becomes validated only once it has been verified by multiple parties. Furthermore, the data in blocks cannot be modified arbitrarily. A blockchain-based smart contract, for example, creates a reliable system because it dispels doubts about information’s veracity.

1.1 MOTIVATION

The motivation of our study is

- To eliminate the time-consuming certificate verification process, for organizations.
- To reduce the database frauds from various attacks.
- To enhance the transaction clarity.
- Improve the trust of the system

1.2 PROBLEM DEFINITION

In this project, we design an application used to provide secure verification of our certificates. In this work, we design and develop a system for secure e certificate generation system using smart contracts in a blockchain environment, illustrate our own blockchain using ethereum block chain and finally, validate and algorithm for proof of validation.

1.3 OBJECTIVE OF PROJECT

Use the unmodifiable property of blockchain to provide more security. Confidentiality is transparent with each transaction visible to all the peers. The certificate is validated rapidly. Provide accurate and reliable information.

1.4 ADVANTAGES OF THE PROJECT

- Slight Control over Fraud.
- Transparency system
- Eradicate the problem

1.5 DISADVANTAGES OF THE PROJECT

- Lengthy process.
- Cost can be high.

2. LITERATURE SURVEY

Jin-chiou et al [1] developed software in order to avoid counterfeiting certificates.

- Due to the lack of an anti-forge mechanism, the graduation certificate is to be forged. so, the decentralized application was designed based on etherum blockchain technology.
- First, generate the digital certificate for the paper certificate then hash value created for the certificate is stored in the blockchain system.
- Even it used to verify the authenticity of the certificate it required another scanning app to scan the certificate.
- The system saves on paper, prevent document forgery.
- But the QR-Code must be scanned with a smartphone and an internet connection is required.

Ze Wang et al [2] designed a blockchain-based certificate transparency and revocation transparency system.

- In this system, the certificate authority (CA) signed the certificate and the revocation status information of the respected certificates are published by the subject (Certificate Authority).
- Public logs are used to monitor the CAs operation.
- This system was implemented with Firefox and Nogix.
- This system provided the trust but Certificate validation is delayed and a false sense of security.

Madala et al [3] used the Hyper ledger Fabric blockchain platform.

- In this system, the certificates are issued by CAs only by obtaining approval from the domain owner Certificate Transparency (CT) technique, invented by Google.
- The aim to prevent SSL/TLS CA from issuing certificates for a domain without visible to the owner of the domain.
- But there was low scalability and less transaction.

Aisong Zhang et al [4] designed a system based on consortium blockchain technology.

- They used a secret sharing scheme. It can validate the digital certificate to protect the user's information and also the property of the user.
- The digital certificate revocation lists have collaborated among the CAs.
- The trust and reliable CRL(Certificate Revocation List)are more compared with the traditional system.
- If the user wants to verify the certificate, they only need to decrypt the signature with the public key.
- And the result will be compared with the hash operation of the original message.

- If the result is consistent, it proved that the digital certificate not tampered.
- But there is a false sense of security.

Macro Baldi etal[5] designed a system named certificate validation through public ledgers and blockchain.

- In this system, CRLs(Certificate Revocation List) were distributed through the use of a private blockchain, and it shared among CA(certificate authority).
- CAs are responsible for issuing certificates to requestors who meet the requirements and maintain CRLs.
- The certificate revocation list was available and authentication was provided at any time for a certificate. The certificate revocation list for a set of the certificate was maintained by the same certification authority who issued the certificates.
- CA ecosystem is fragile and prone to compromise.

3. PROBLEM ANALYSIS

Nowadays, all Graduation certificates and transcripts hold information that is easily tampered illegally by individuals and should not be easily accessible to outside entities. Hence, there is a high need for an efficient mechanism, that can guarantee the information in such certificates is original, which means the document has originated from a reliable and authorised source and is not forged.

Various systems have been designed to secure e-certificates for education institutions and to store them securely in cloud-based systems.

3.1 EXISTING SYSTEM

The certificate are stored in centralised manner and verified manually, so it takes too much time to verify. There is no safety to the certificate that are given to any private sectors (banks).But, the data may be changed, deleted or modified. Certificates are easily hacked and make duplicate of that certificate. Students bring their certificates on interview places. There is no security for certificates.

3.2 PROPOSED SYSTEM

In this proposed system the academic certificates are converted into digital certificates using sampling and quantization. Then the certificates are added with the hash values generated for the digital certificate and store it into the blocks. Blockchain is the main tool to felicitate this need and when combined with different hashing techniques, this becomes a powerful method for protecting the data. It also helps in eliminating the need for constant verification of certificates.

Blockchain technology can be used to reduce the incidence of certificate forgeries and ensure that the security, validity, and confidentiality of graduation certificates would be improved. Technologies that exist in security domains include digital signatures, which are used in digital documents to provide authentication, integrity, and non-repudiation. Also with blockchain in play, the storage of certificates are more secure.

3.3 TECHNOLOGIES ANALYSIS

With these technologies, an application created that facilitates the secure validation of digital certificates.

3.3.1 Blockchain

The concept of blockchain was proposed by Satoshi Nakamoto in 2008. Blockchain is an online ledger that provides decentralized and transparent data sharing. With distributed recordings, all transaction data (stored in nodes) are compressed and added to different blocks. Data of various types are distributed in distinct blocks, enabling verifications to be made without the use of intermediaries. All the nodes then form a blockchain with timestamps. The data stored in each block can be verified simultaneously and become inalterable once entered. The whole process is open to the public, transparent, and secure. The emergence of Ethereum Smart Contracts in 2013 boosted blockchain technology, which became blockchain 2.0. As presented in blockchain 1.0 was mainly adopted by Bitcoin to solve problems concerning cryptocurrencies and decentralized payments. Blockchain 2.0 focused on decentralizing the entire market and is employed to transform assets through smart contracts, thereby creating value through the emergence of alternatives to Bitcoin.

3.3.2 Ethereum

Ethereum is an open and decentralized platform featuring Turing completeness and supporting various derivative applications. Most smart contracts and decentralized autonomous organizations are created by using Ethereum. If the Bitcoin blockchains are considered a global payment network, Ethereum would be the global computing system. Furthermore, Ethereum is an open source platform similar to Android (developed by Google). It provides an infrastructure that enables developers to create applications. The infrastructure is developed and maintained by both Ethereum and those developers. The major characteristics of Ethereum are as follows: incorruptible: third-parties are not able to modify any data; 2) secure: errors derived from personnel factors are avoided because the decentralized applications are maintained by entities rather than individuals; permanent: blockchain does not cease to operate even if an individual computer or server crashes. 1) Ethereum Virtual Machine (EVM) The EVM is a programmable blockchain. Unlike Bitcoin, which provides a fixed set of commands, the EVM allows developers to run any programs in the manner they wish. Developers instruct the EVM to execute applications by using a high-level language called Solidity. 2) Solidity Solidity is the programming language used for implementing smart contracts and is similar to JavaScript. After a Solidity-programmed smart contract is completed, a complier called solc is

required to transform the Solidity code into contract byte code, which is then interpreted by the EVM. Next, the compiled instructions are deployed in an Ethereum blockchain. This completes the whole process

3.3.3 Smart Contracts

Smart contracts were first proposed by Nick Szabo in the early 1990s. He explained that a smart contract enabled computers to execute transaction clauses. As blockchain has become popular, smart contracts have received increased attention. Smart contracts are the main feature of Ethereum, a blockchain platform founded in 2015. A smart contract is “a digital contract that is written in source code and executed by computers, which integrates the tamper-proof mechanism of blockchain”. Smart contracts can be created using the Ethereumblockchain. Developers are able, according to their needs, to specify any instruction in smart contracts; develop various types of applications, including those that interact with other contracts; store data; and transfer Ethers. Additionally, smart contracts that are deployed in blockchains are copied to each node to prevent contract tampering. With related operations executed by computers and services provided by Ethereum, human error can be reduced to avoid disputes regarding such contracts. Smart contracts are mostly used in voting system and cryptocurrency applications. Fig. 3 depicts an example of how developers can easily deploy smart contracts for cryptocurrency transactions. The high-level programming languages used for writing smart contracts are mainly Solidity, Serpent, and LLL. Currently, most developers employ Solidity to write smart contracts and compile the instructions into bytecode for the EVM to execute. Certain costs are incurred when developers create smart contracts.

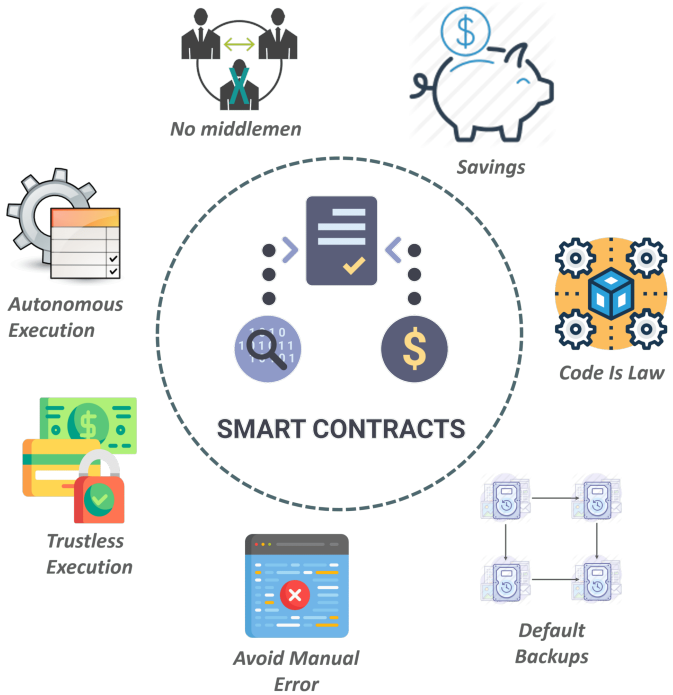


Fig 2: Uses of smart contract

4. DESIGN

4.1 SYSTEM REQUIREMENTS FOR BLOCKCHAIN TECHNOLOGY

4.1.1 Software Requirements:

The functional requirements or the overall description documents include the product perspective and features, operating system and operating environment, graphics requirements, design constraints and user documentation. The appropriation of requirements and implementation constraints gives the general overview of the project in regards to what the areas of strength and deficit are and how to tackle them.

The software requirements includes an operating system, GUI, user documentation etc

Tools going to use are :

- Operating System: Windows Xp/7/8/10
- Programming Languages: JAVA
- Tools: JDK 1.7 or Higher
- Server: Tomcat

4.1.2 Hardware Requirements:

- Minimum hardware requirements dependent on the particular software being used
- Applications that need to store large arrays/objects in memory will require more RAM, whereas applications that need to perform numerous calculations or tasks more quickly will require a faster processor.

The minimum requirements needed to install Java and associated applications:

- Modern Operating System: Windows 7 or 10
- x86 64-bit CPU (Intel / AMD architecture)
- 4 GB RAM
- 5 GB free disk space

4.2 SYSTEM DESIGN

4.2.1 Waterfall Model

In the 1970, Winston Royce proposed a model for software development , the model is called the waterfall model. Waterfall model is also called Linear-sequential life cycle model because this model specifies the software development process in a linear sequence manner. In the waterfall model, the output of the previous phase will be the input to the next phase in the software development life cycle. Each phase has contains various different tasks and different activity to complete the process. In this model, we cannot go back to the previous phase to alter any thing if we come to the next phase.

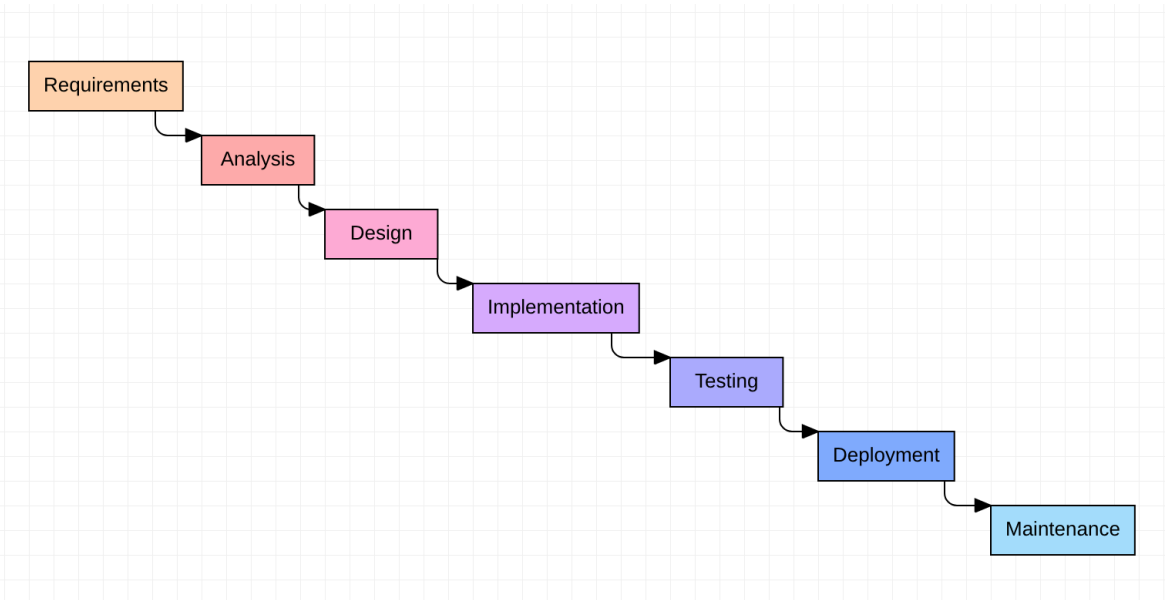


Fig 3: Waterfall Model

Requirements & Analysis:

All possible requirements of the system to be developed are captured in this phase and documented in a requirement specification document.

Design

The requirement specifications from first phase are studied in this phase and the system design is prepared. This system design helps in specifying hardware and system requirements and helps in defining the overall system architecture.

Implementation

With inputs from the system design, the system is first developed in small programs called units, which are integrated in the next phase. Each unit is developed and tested for its functionality, which is referred to as Unit Testing.

Testing

All the units developed in the implementation phase are integrated into a system after testing of each unit. Post integration the entire system is tested for any faults and failures. Deployment of system − Once the functional and non-functional testing is done; the product is deployed in the customer environment or released into the market.

Maintenance

There are some issues which come up in the client environment. To fix those issues, patches are released. Also to enhance the product some better versions are released. Maintenance is done to deliver these changes in the customer environment.

4.2.2 UML Diagrams

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group. The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems. The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems. The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects. The Primary goals in the design of the UML are as follows: Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models. Provide extendibility and specialization mechanisms to extend the core concepts.Be independent of particular programming languages and development process. Provide a formal basis for understanding the modeling language. Encourage the growth of OO tools market. Support higher level development concepts such as collaborations, frameworks, patterns and components. Integrate best practices.

Types of UML Diagrams

There are several types of UML diagrams and each one of them serves a different purpose regardless of whether it is being designed before the implementation or after.

1. Use case diagram
2. Class diagram
3. Object diagram
4. State diagram
5. Activity diagram
6. Sequence diagram
7. Collaboration diagram

4.2.2.1 Use case diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

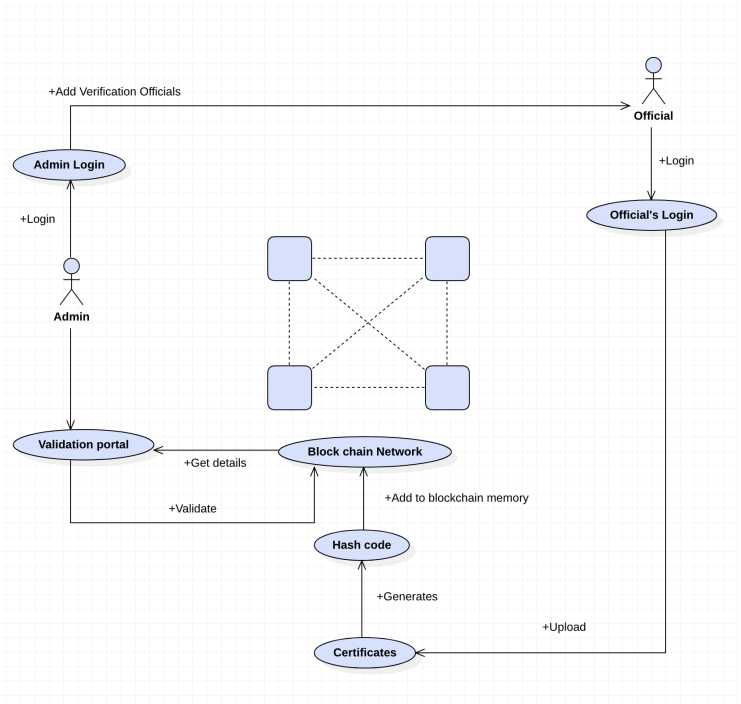


Fig 4: Use Case Diagram

4.2.2.2 Class diagram:

The class diagram is used to refine the use case diagram and define a detailed design of the system. The class diagram classifies the actors defined in the use case diagram into a set of interrelated classes. The relationship or association between the classes can be either an "is-a" or "has-a" relationship. Each class in the class diagram may be capable of providing certain functionalities. These functionalities provided by the class are termed "methods" of the class. Apart from this, each class may have certain "attributes" that uniquely identify the class.

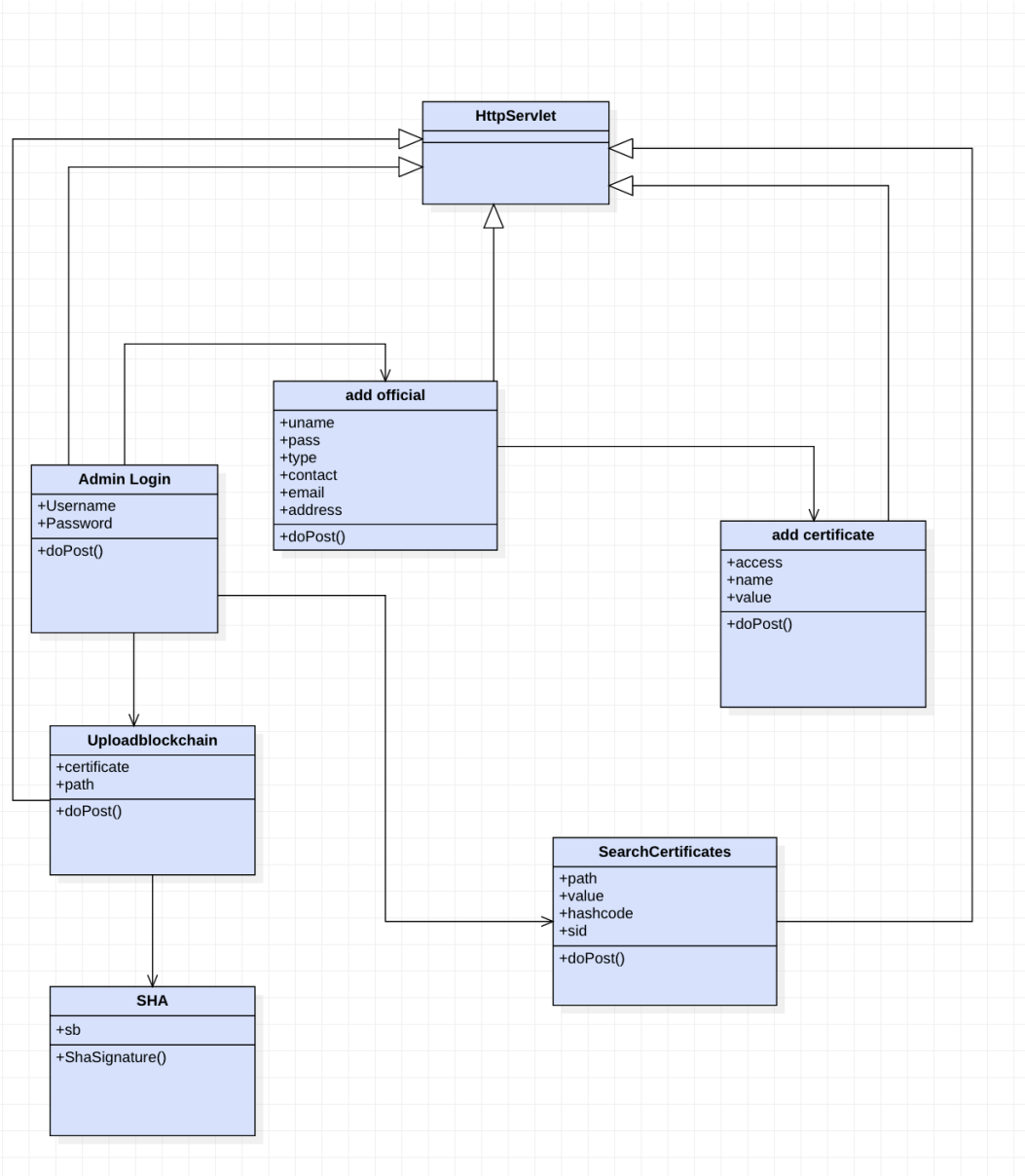


Fig 5: Class Diagram

4.2.2.3 Object diagram:

The object diagram is a special kind of class diagram. An object is an instance of a class. This essentially means that an object represents the state of a class at a given point of time while the system is running. The object diagram captures the state of different classes in the system and their relationships or associations at a given point of time.

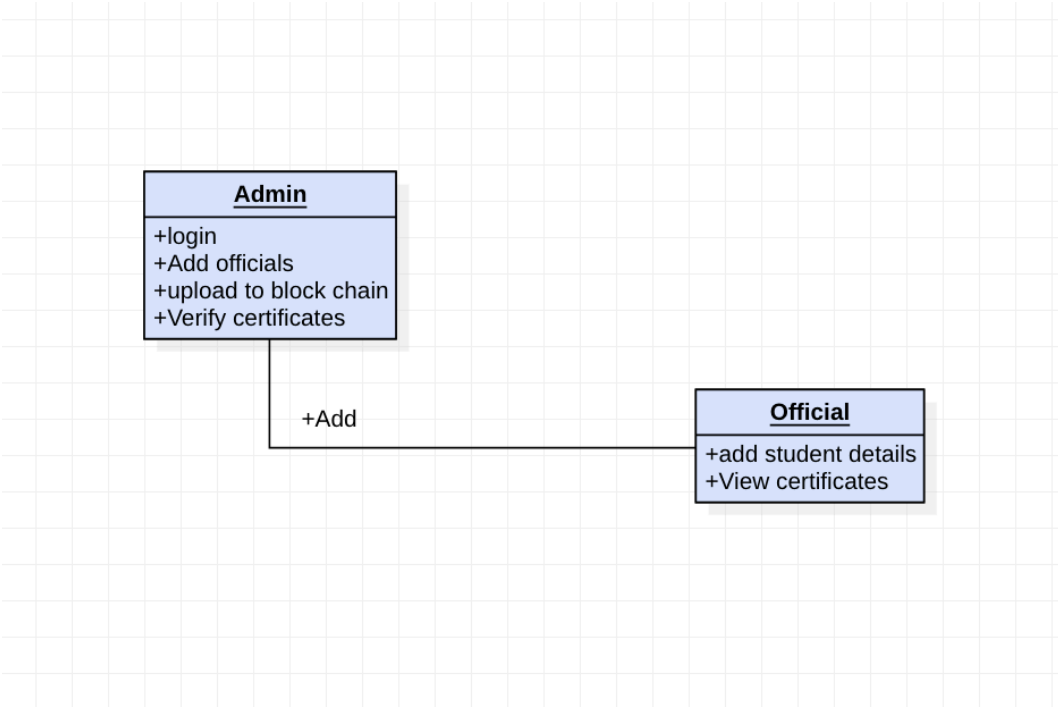


Fig 6: Object Diagram

4.2.2.4 State diagram:

A state diagram, as the name suggests, represents the different states that objects in the system undergo during their life cycle. Objects in the system change states in response to events. In addition to this, a state diagram also captures the transition of the object's state from an initial state to a final state in response to events affecting the system.

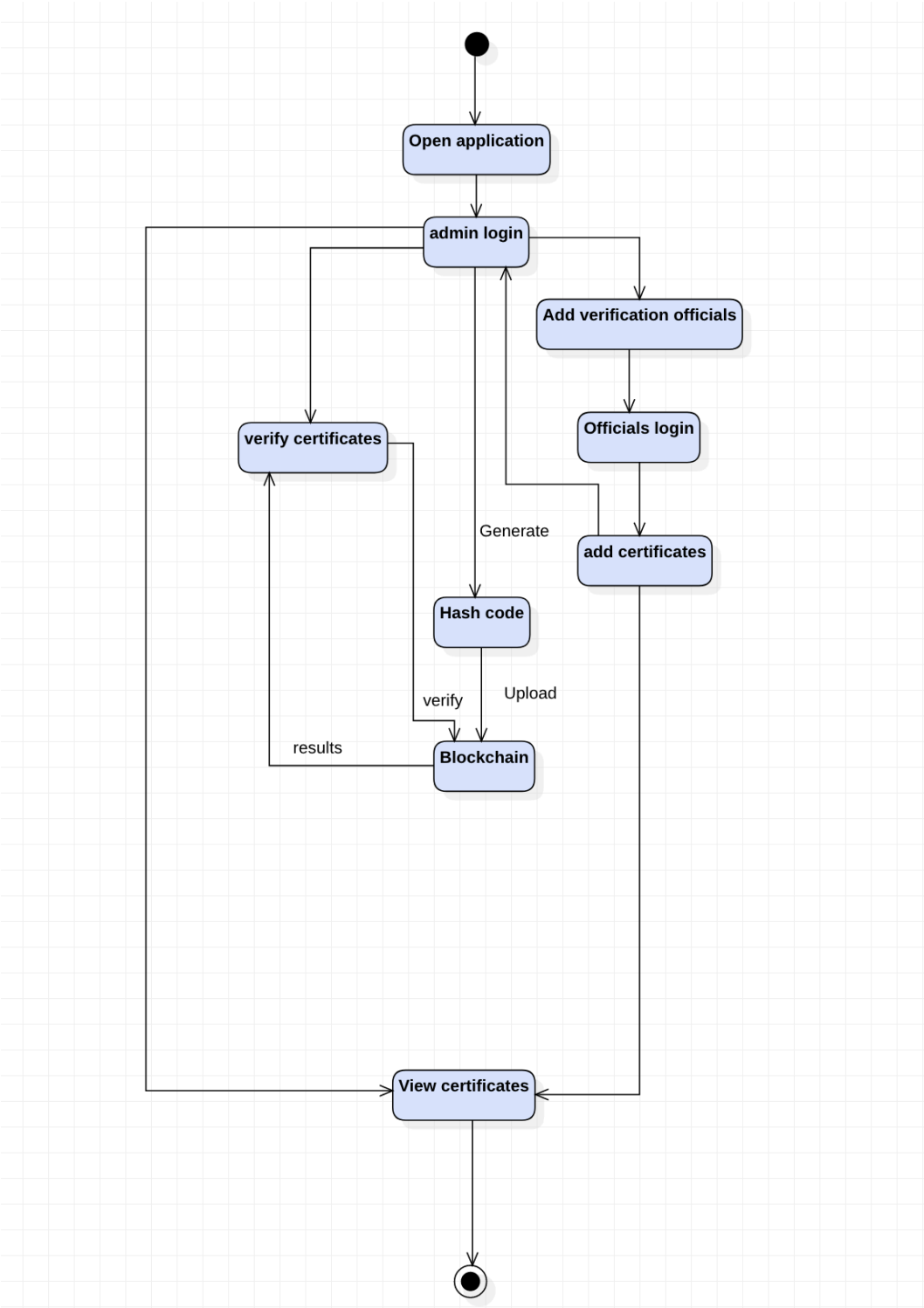


Fig 7: State Diagram

4.2.2.5 Activity diagram:

The process flows in the system are captured in the activity diagram. Similar to a state diagram, an activity diagram also consists of activities, actions, transitions, initial and final states, and guard conditions.

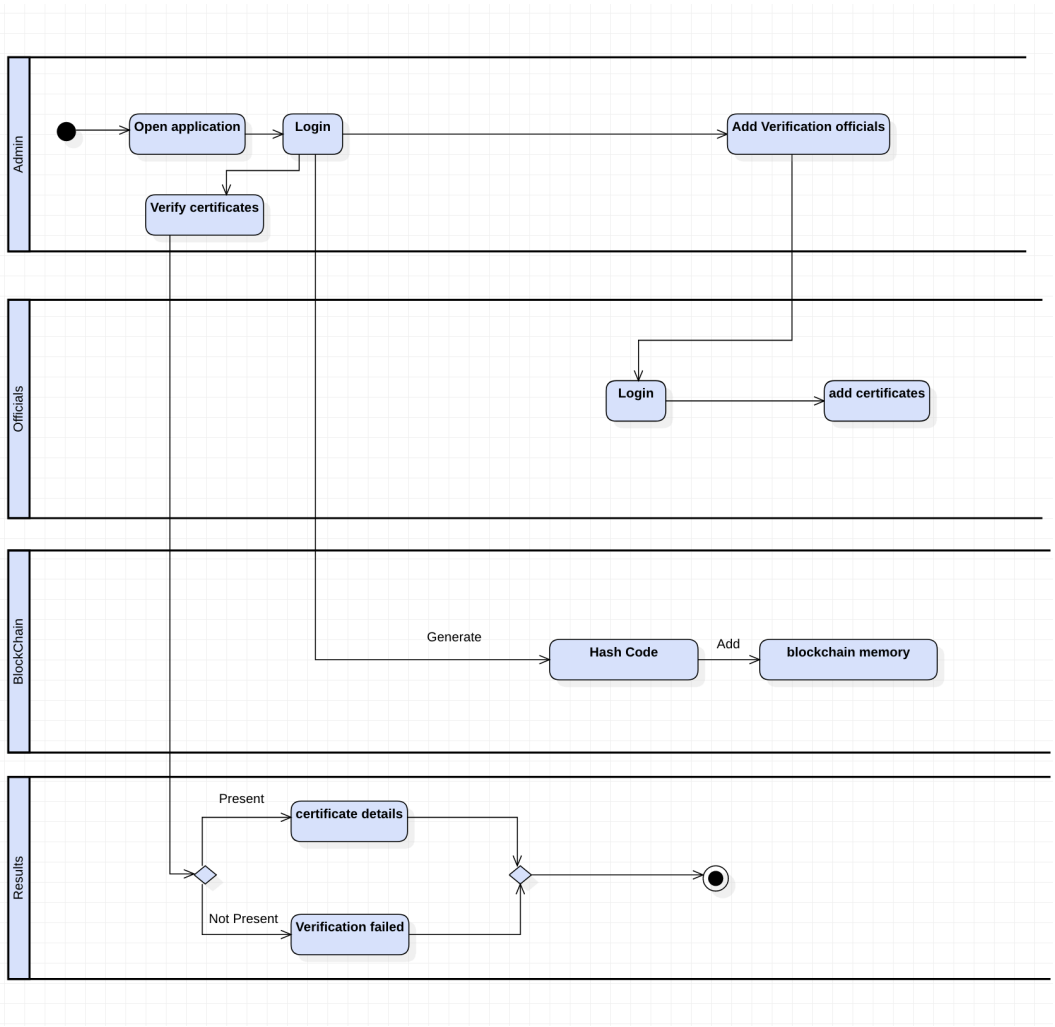


Fig 8: Activity Diagram

4.2.2.6 Sequence diagram:

A sequence diagram represents the interaction between different objects in the system. The important aspect of a sequence diagram is that it is time-ordered. This means that the exact sequence of the interactions between the objects is represented step by step. Different objects in the sequence diagram interact with each other by passing "messages".

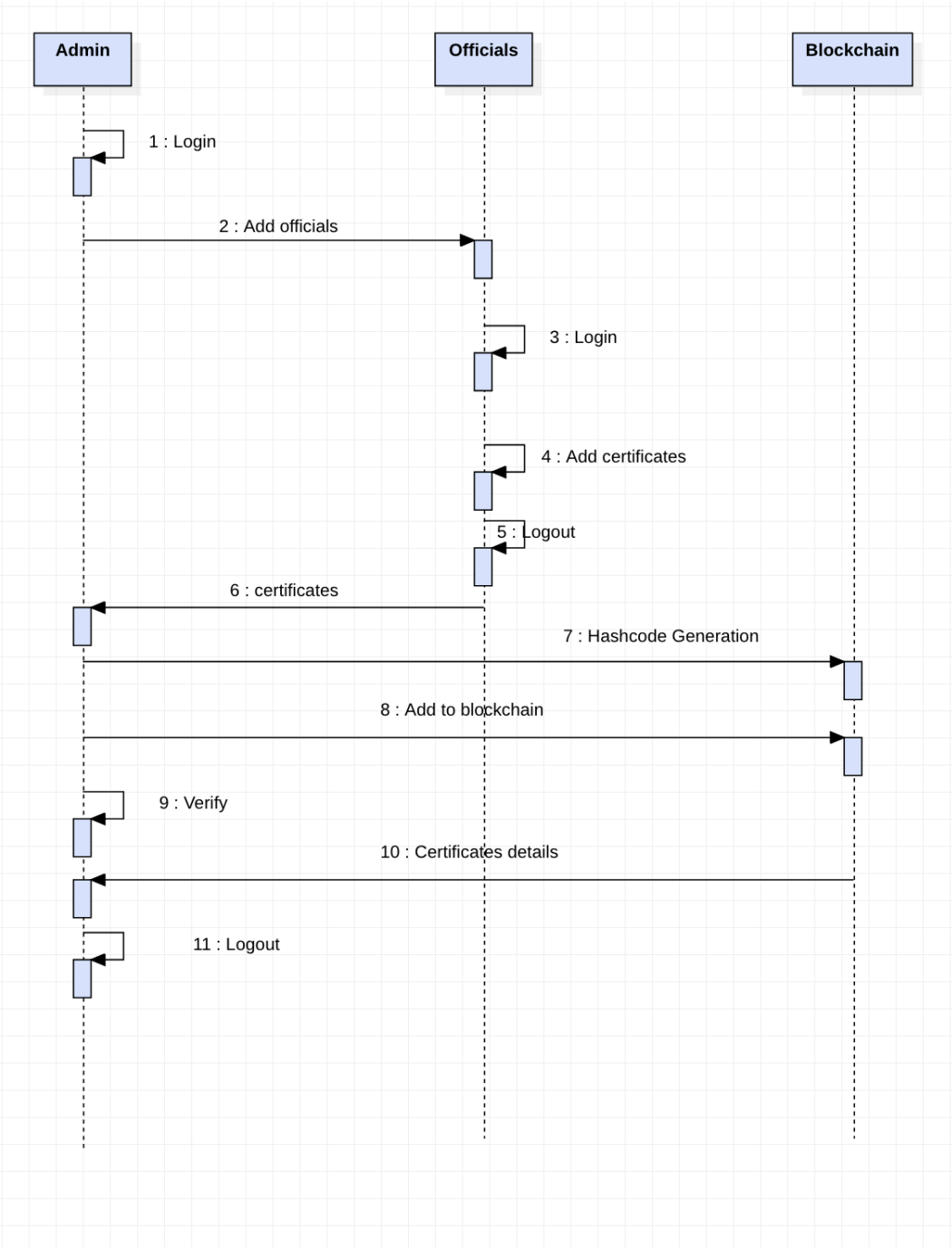


Fig 9: Sequence Diagram

4.2.2.7 Collaboration diagram:

A collaboration diagram groups together the interactions between different objects. The interactions are listed as numbered interactions that help to trace the sequence of the interactions. The collaboration diagram helps to identify all the possible interactions that each object has with other objects.

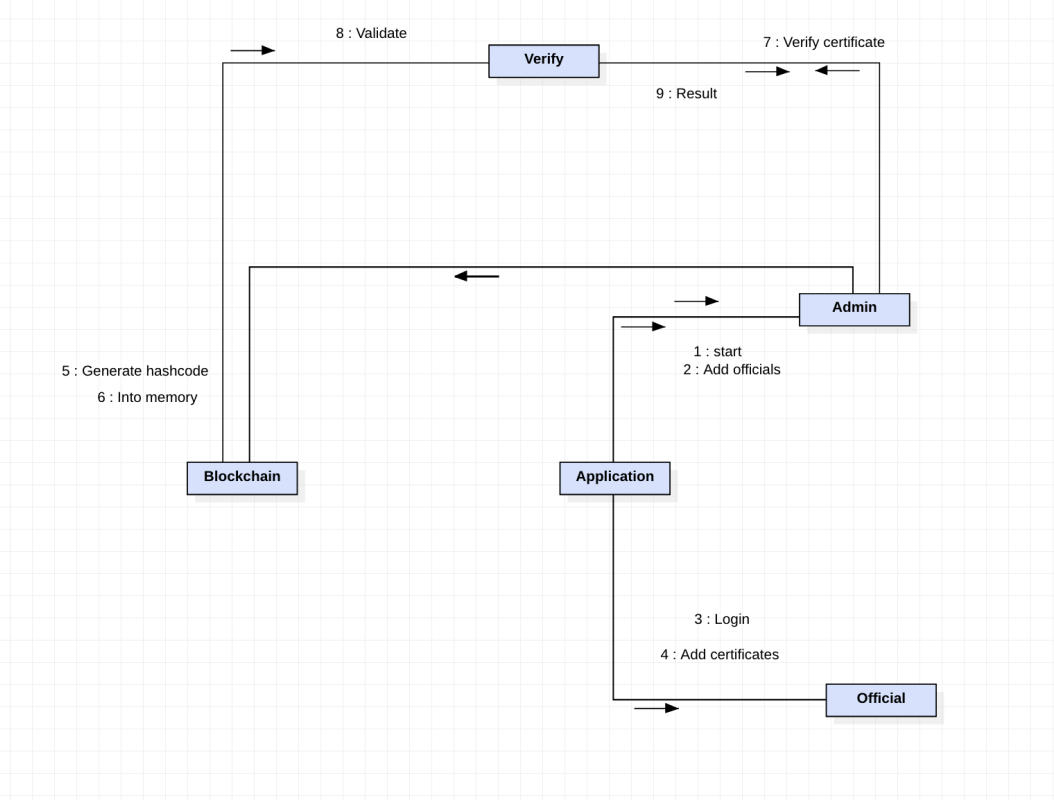


Fig 10: Collaboration Diagram

5. IMPLEMENTATION

Blockchain Model To Track & Verify Official Certificates

In this project we are using Blockchain technology to store academic certificates to avoid certificate counterfeit as lots of fake certificates are using to get jobs and there is no current technology to detect such fake certificates and to prevent such forgery we are using Blockchain based certificate verification.

5.1 MODULES

This project consists of following modules

5.1.1 Admin Module

Admin can login to system by using username as ‘admin’ and password as ‘admin’ and then admin will perform following actions

- Add Verification Officials: Using this module admin can add various verifier officials such as SSC verifier, Intermediate, PHD and PG.
- Upload Verified Certificates to Blockchain: using this module admin will upload verified certificates to Blockchain. For each certificate image one hash code will be generated and this hash code will be stored at Blockchain memory. Whenever we upload any test certificate then Blockchain verify test certificate hash code with stored hash code and if matched found then certificated will be authenticated and certificate owner details will be fetched and display otherwise certificate authentication will be failed.
- Search Verified Certificates: In this module admin will upload test certificate and then application generate hash code from certificate and matched with Blockchain store images and if matched found then certificate will be authenticated.

5.1.2 Verify Officials Module

Using this module officials will login to application by using username and password given by Admin user and then will upload and view certificates.

5.2 ALGORITHMS

5.2.1 SHA Algorithm

SHA stands for secure hashing algorithm. SHA is a modified version of MD5 and used for hashing data and certificates. A hashing algorithm shortens the input data into a smaller form that cannot be understood by using bitwise operations, modular additions, and compression functions. Hashing is similar to **encryption**, the only difference between hashing and encryption is that hashing is one-way, meaning once the data is hashed, the resulting hash digest cannot be cracked, unless a brute force attack is used.

5.2.1.1 Different SHA Forms

When learning about SHA forms, several different types of SHA are referenced. Examples of SHA names used are SHA-1, SHA-2, SHA-256, SHA-512, SHA-224, and SHA-384, but in actuality there are only two types: SHA-1 and SHA-2. The other larger numbers, like SHA-256, are just versions of SHA-2 that note the bit lengths of the SHA-2. SHA-1 was the original secure hashing algorithm, returning a 160-bit hash digest after hashing.

5.2.1.2 What SHA is used for and Why

As previously mentioned, Secure Hashing Algorithms are required in all digital signatures and certificates relating to SSL/TLS connections, but there are more uses to SHAs as well. Applications such as **SSH**, S-MIME (Secure / Multipurpose Internet Mail Extensions), and IPSec utilize SHAs as well. SHAs are also used to hash passwords so that the server only needs to remember hashes rather than passwords. In this way, if an attacker steals the database containing all the hashes, they would not have direct access to all of the plaintext passwords, they would also need to find a way to crack the hashes to be able to use the passwords. SHAs can also work as indicators of a file's integrity. If a file has been changed in transit, the resulting hash digest created from the hash function will not match the hash digest originally created and sent by the file's owner.

5.2.1.3 SHA1

In our project we used SHA1 for generating the hash code for the certificates. The code is as follows:

Code:

```
package com;
import java.security.MessageDigest;
public class SHA{
public static String ShaSignature(byte img[]){
    StringBuilder sb = new StringBuilder();
    try{
        MessageDigest mDigest = MessageDigest.getInstance("SHA1");
        byte[] result = mDigest.digest(img);
        for (int i = 0; i < result.length; i++) {
            sb.append(Integer.toString((result[i] & 0xff) + 0x100, 16).substring(1));
        }
    }catch(Exception e){
        e.printStackTrace();
    }
    return sb.toString();
}
}
```

5.2.2 Smart contract

Smart contracts are simply programs stored on a blockchain that run when predetermined conditions are met. They typically are used to automate the execution of an agreement so that all participants can be immediately certain of the outcome, without any intermediary's involvement or time loss. They can also automate a workflow, triggering the next action when conditions are met.

Smart contracts work by following simple "if/when…then…" statements that are written into code on a blockchain. A network of computers executes the actions when predetermined conditions have been met and verified. These actions could include releasing funds to the appropriate parties, registering a vehicle, sending notifications, or issuing a ticket. The blockchain is then updated when the transaction is completed. That means the transaction cannot be changed, and only parties who have been granted permission can see the results.

Within a smart contract, there can be as many stipulations as needed to satisfy the participants that the task will be completed satisfactorily. To establish the terms, participants must determine how transactions and their data are represented on the blockchain, agree on the "if/when...then…" rules that govern those transactions, explore all possible exceptions, and define a framework for resolving disputes.

Then the smart contract can be programmed by a developer – although increasingly, organizations that use blockchain for business provide templates, web interfaces, and other online tools to simplify structuring smart contracts.

The code for smart contract is as follows:

Code:

```
package com;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.web3j.crypto.Credentials;
import org.web3j.crypto.WalletUtils;
import org.web3j.protocol.Web3j;
import org.web3j.protocol.core.methods.response.TransactionReceipt;
import org.web3j.protocol.http.HttpService;
import org.web3j.tx.Contract;
import org.web3j.tx.ManagedTransaction;
import org.web3j.tx.Transfer;
import org.web3j.utils.Convert;
import org.web3j.protocol.core.RemoteCall;
import java.math.BigDecimal;
import java.math.BigInteger;
import java.util.List;
import java.io.BufferedReader;
import java.io.FileReader;
public class AccessSmartContract {
    public static Web3j web3j;
    public static Credentials credentials;
    public static String address;
public static void setup() {
    try{
        if(web3j == null){
            web3j = Web3j.build(new HttpService());
            credentials = WalletUtils.loadCredentials("erum","C:/ETH/data-private/keystore/
UTC--2020-07-02T14-07-15.546319700Z--f5c454e64f35c894e3e4e9dc4d637aeb25176167");
            TransactionReceipt transferReceipt = Transfer.sendFunds(web3j,
credentials,"f5c454e64f35c894e3e4e9dc4d637aeb25176167", BigDecimal.valueOf(100),
Convert.Unit.ETHER).sendAsync().get();
            BufferedReader br = new BufferedReader(new FileReader(Path.getPath()));
            address = br.readLine();
            address = address.trim();
            br.close();
            System.out.println("Transaction complete : "+ transferReceipt.getTransactionHash()+"
"+address);
        }
    }catch(Exception e){
        e.printStackTrace();
    }
}
public static String getRecord(){
    String result = "none";
    try{
        StringBuilder sb = new StringBuilder();
```

```
        setup();
        UserList ul = UserList.load(address, web3j, credentials, ManagedTransaction.GAS_PRICE,
Contract.GAS_LIMIT);
        String data = ul.getPatients().send();
        result = data;

    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}
public static String createRecord(String sid,String hashcode){
    String result = "none";
    try{
        setup();
        System.out.println("Address "+address);
        UserList ul = UserList.load(address, web3j, credentials, ManagedTransaction.GAS_PRICE,
Contract.GAS_LIMIT);
        String certificate = ul.getPatients().send();
        if(certificate.length() > 0) {
            certificate = certificate+"#"+sid+","+hashcode;
        } else {
            certificate = sid+","+hashcode;
        }
        System.out.println(certificate);
        ul.createPatient(certificate).send();
        result = "success";
    }catch(Exception e){
        e.printStackTrace();
    }
    return result;
}
}
```

6. TESTING & VALIDATION

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

6.1 TYPES OF TESTS

6.1.1 Unit testing

Unit testing involves the design of test cases that validate that the internal
program logic is functioning properly, and that program inputs produce valid outputs. All
decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

6.1.2 Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

6.1.3 Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.
Functional testing is centered on the following items:
Valid Input : identified classes of valid input must be accepted.
Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures : interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.1.4 System Test

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

6.1.5 White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

6.1.6 Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

Test objectives

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

Features to be tested

- Verify that the entries are of the correct format No duplicate entries should be allowed
- All links should take the user to the correct page.

6.1.7 Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

7. RESULT ANALYSIS

Result analysis is mainly used in explaining the results of our project comparing it to other different methods as below.

7.1 PREPARING YOUR DEVELOPMENT COMPUTER

Step 1: You might need to install the Java JDK, if you don't have it already.

System Requirements for Installing the JDK on 64-Bit Windows Platform : For supported processors and browsers, see Oracle JDK Certified Systems Configurations.

JDK Installation Instructions for Windows:

- Downloading the JDK Installer
- Running the JDK Installer
- Installing the JDK Silently
- Setting the PATH Environment Variable

1. Downloading the JDK Installer:

Download the file jdk-13.interim.update.patch_windows-x64_bin.exe.

2. Running the JDK Installer:

You must have administrator privilege to install the JDK on Microsoft Windows.

To run the JDK installer:

- Start the JDK 13 installer by double-clicking the installer's icon or file name in the download location.
- Follow the instructions provided by the Installation wizard.
- After the installation is complete, delete the downloaded file to recover the disk space.

3. Installing the JDK Silently

Instead of double-clicking or opening the JDK installer, you can perform a silent, non interactive, JDK installation by using command-line arguments.

Install JDK in silent mode using the command:

jdk.exe /s

4. Setting the PATH Environment Variable

It is useful to set the PATH variable permanently for JDK 13 so that it is persistent after rebooting.

If you do not set the PATH variable, then you must specify the full path to the executable file every time that you run it. For example:

C:\> "C:\Program Files\Java\jdk-13\bin\javac" MyClass.java

To set the PATH variable permanently, add the full path of the jdk-13\bin directory to the PATH variable. Typically, the full path is:

C:\Program Files\Java\jdk-13\bin

To set the PATH variable on Microsoft Windows:

1. Select Control Panel and then System.
2. Click Advanced and then Environment Variables.
3. Add the location of the `bin` folder of the JDK installation to the `PATH` variable in **System Variables**.

The following is a typical value for the `PATH` variable:

C:\WINDOWS\system32;C:\WINDOWS;"C:\Program Files\Java\jdk-13\bin”

Step 2: To work with blockchain install Ethereum tool

To install, double click on ‘geth-windows-amd64-1.8.22-7fa3509e.exe’ file and click on next buttons and complete installation. After installation put ETH folder inside C directory of your system.

Step 3: Connecting to Local Host

In computer networking, localhost is a hostname that refers to the current device used to access it. It is used to access the network services that are running on the host via the loopback network interface. To connect to local host download tomcat server. The Apache Tomcat® software is an open source implementation of the Jakarta Servlet, Jakarta Server Pages, Jakarta Expression Language, Jakarta WebSocket, Jakarta Annotations and Jakarta Authentication specifications.

Version: Tomcat 8

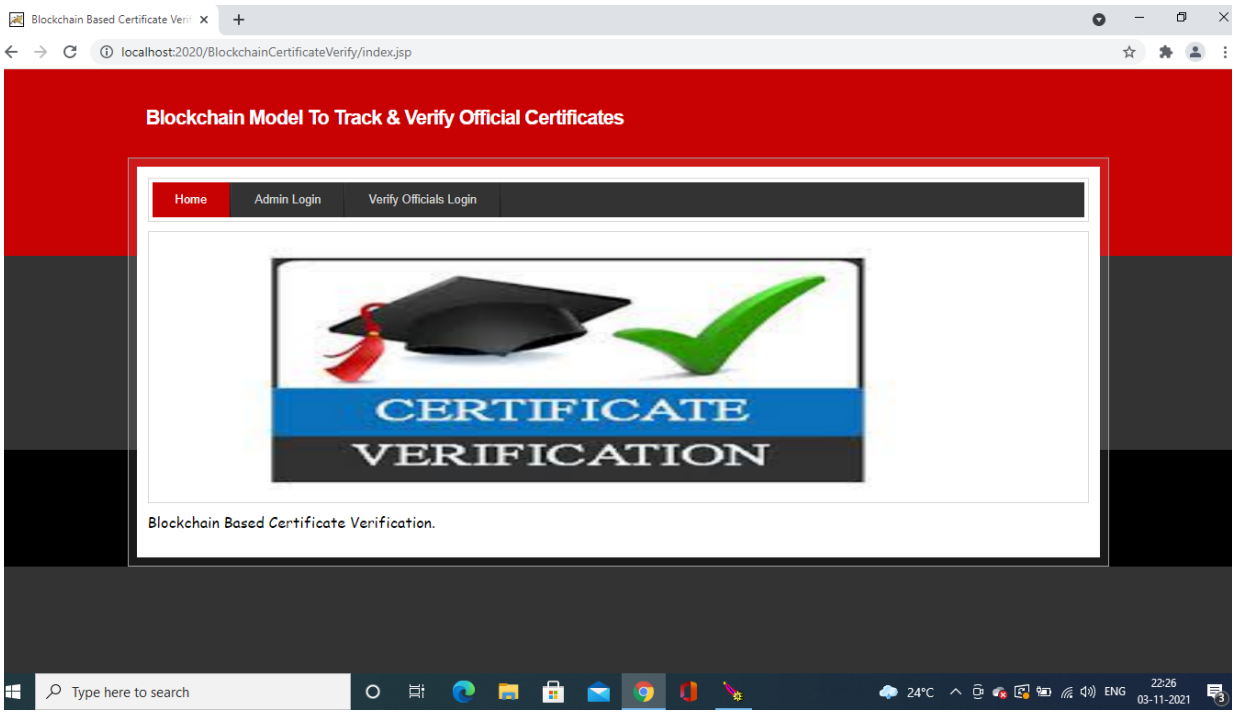
File: 32-bit/64-bit Windows Service Installer (pgp, sha512)

Step 4:

Now put ‘BlockchainCertificateVerify’ folder inside tomcat web directory and then go to WEB-INF/classes folder and double click on ‘start_blockchain_server.bat’ file to start ethereum tool wait for some time and double click on ‘‘initialize_eth.bat’ file. In above screen if we are not getting any exception then open tomcat server and then open browser and enter URL as http://localhost:2020/ BlockchainCertificateVerify/ Index.html and then press enter key to get home page

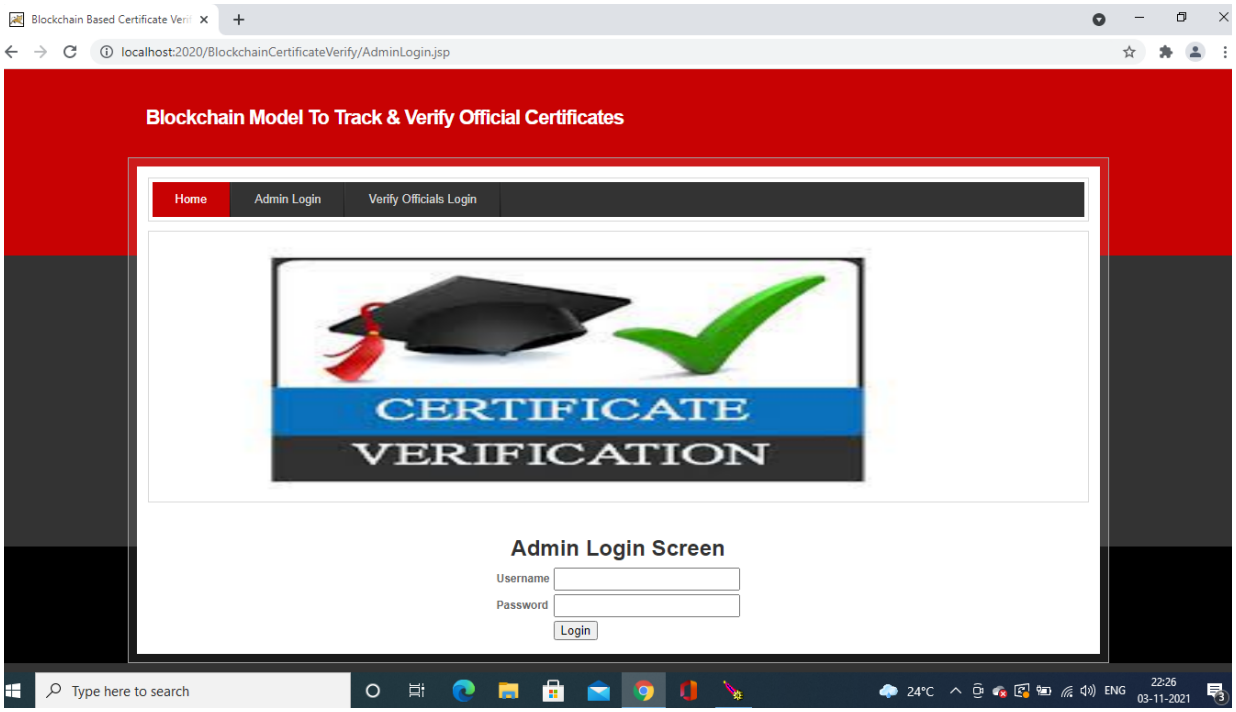
7.2 OUTPUT SCREEN CAPTURES

Home page



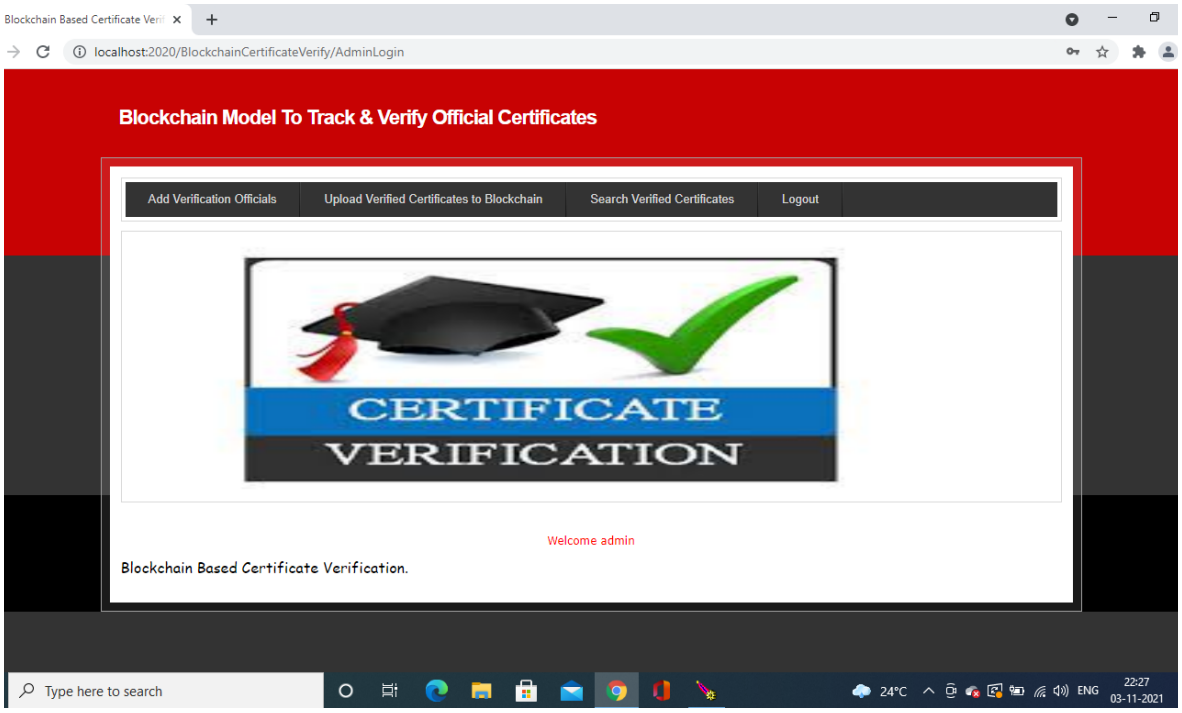
Screen 1: Home page

After clicking on admin login from the above screen, Admin Login page will be displayed.



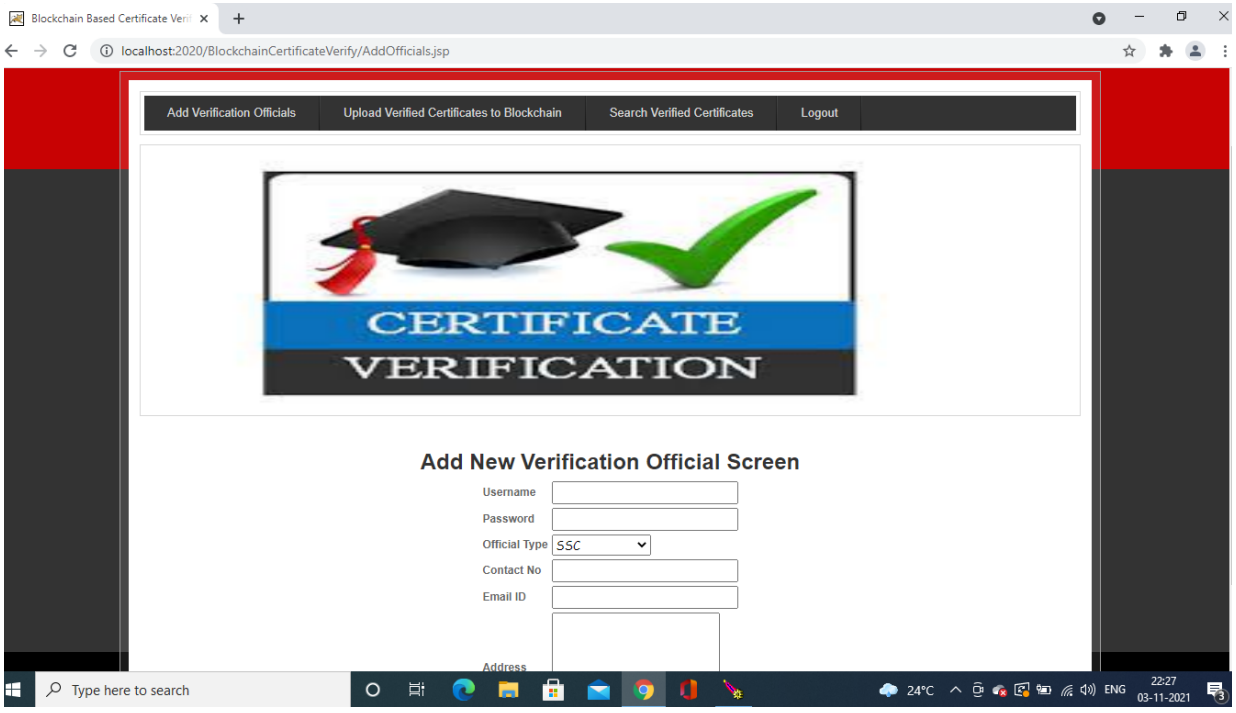
Screen 2: Admin Login page

After logging in using ‘admin’ as username and ‘admin’ as password, below screen will be displayed



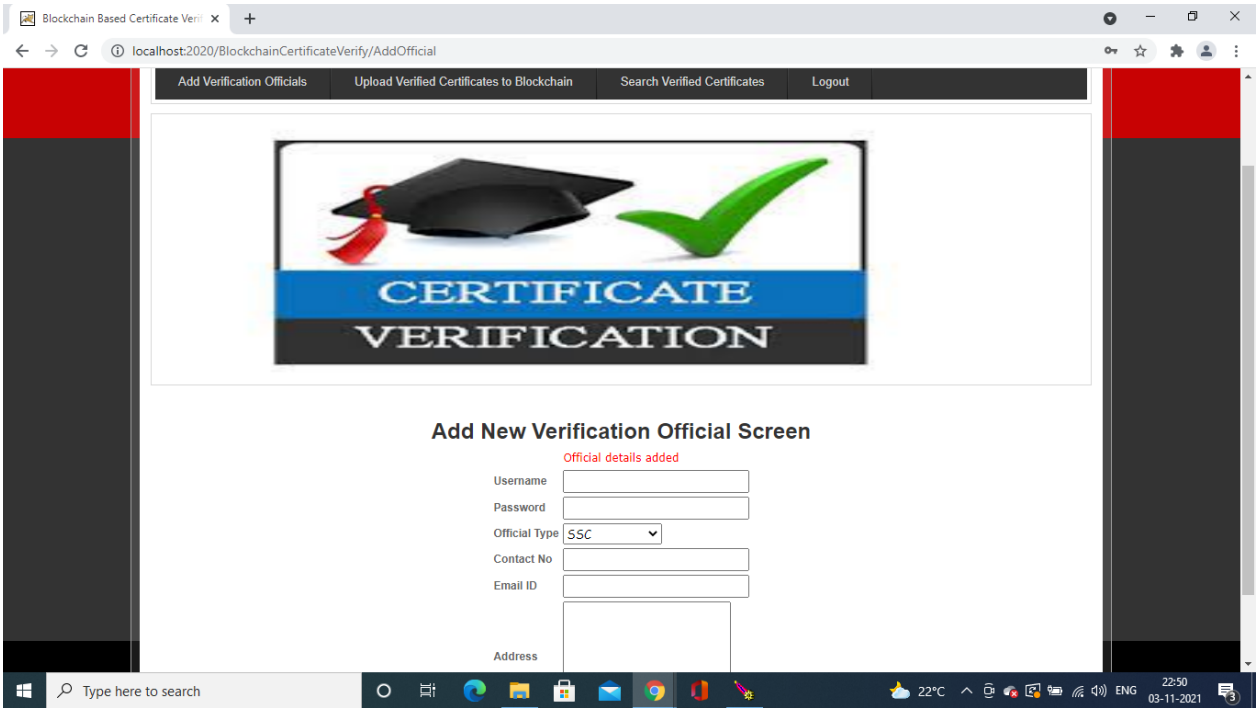
Screen 3: Welcome Admin page

In above screen admin can click on ‘Add Verification Officials’ link to add officials verifier



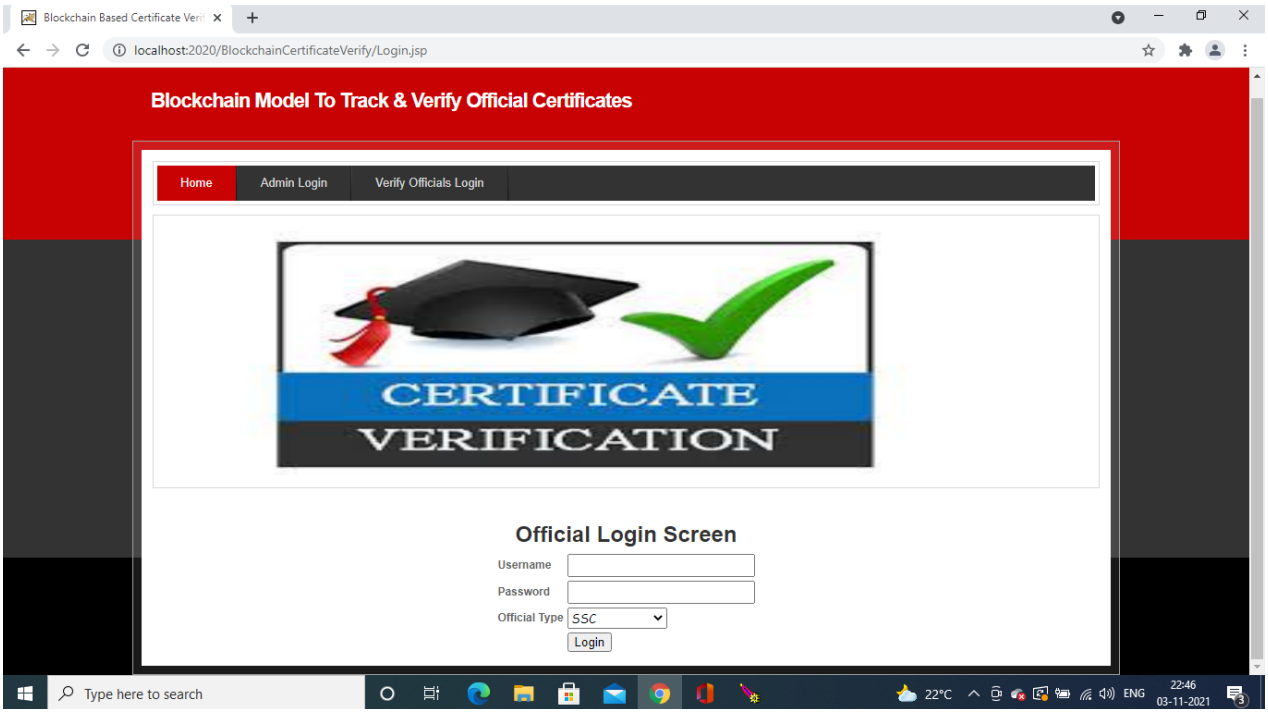
Screen 4: Add Officials page

In above screen admin add official employee details, and click on submit.



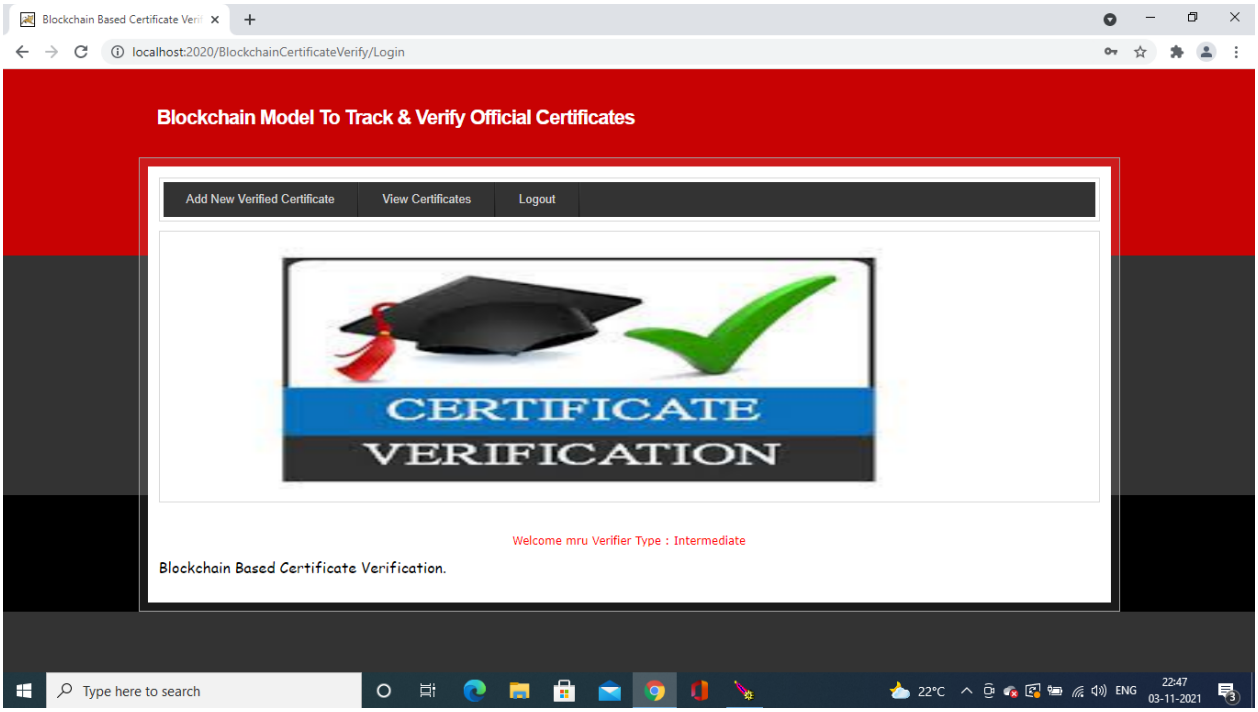
Screen 5: Official added page

In above screen official details added and now logout and login as official to add certificate details



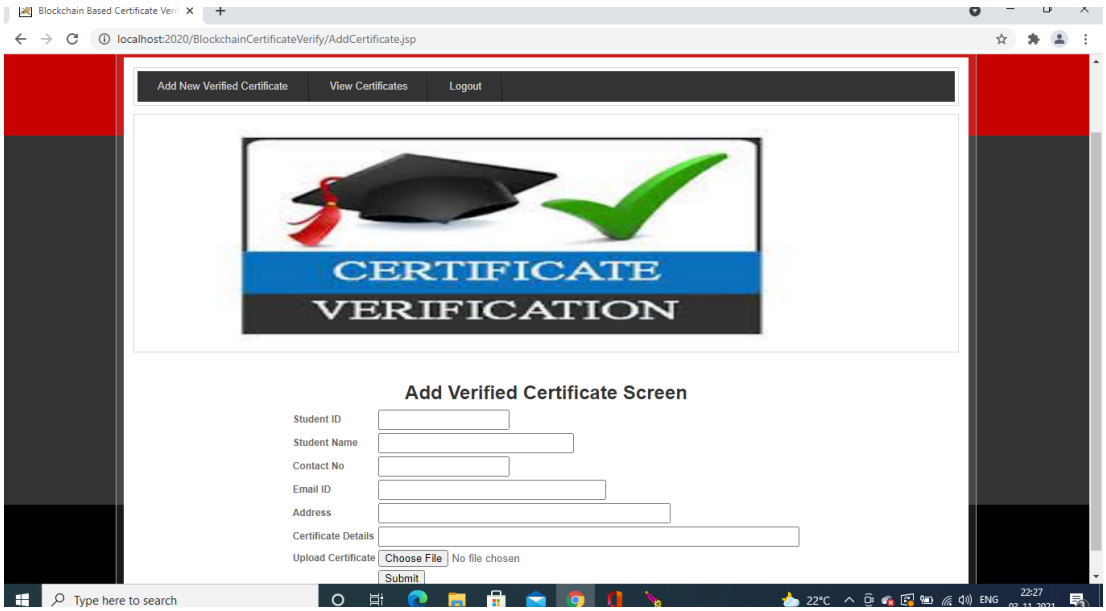
Screen 6: Official login page

official can login using the above screen and after logging in the below screen is displayed



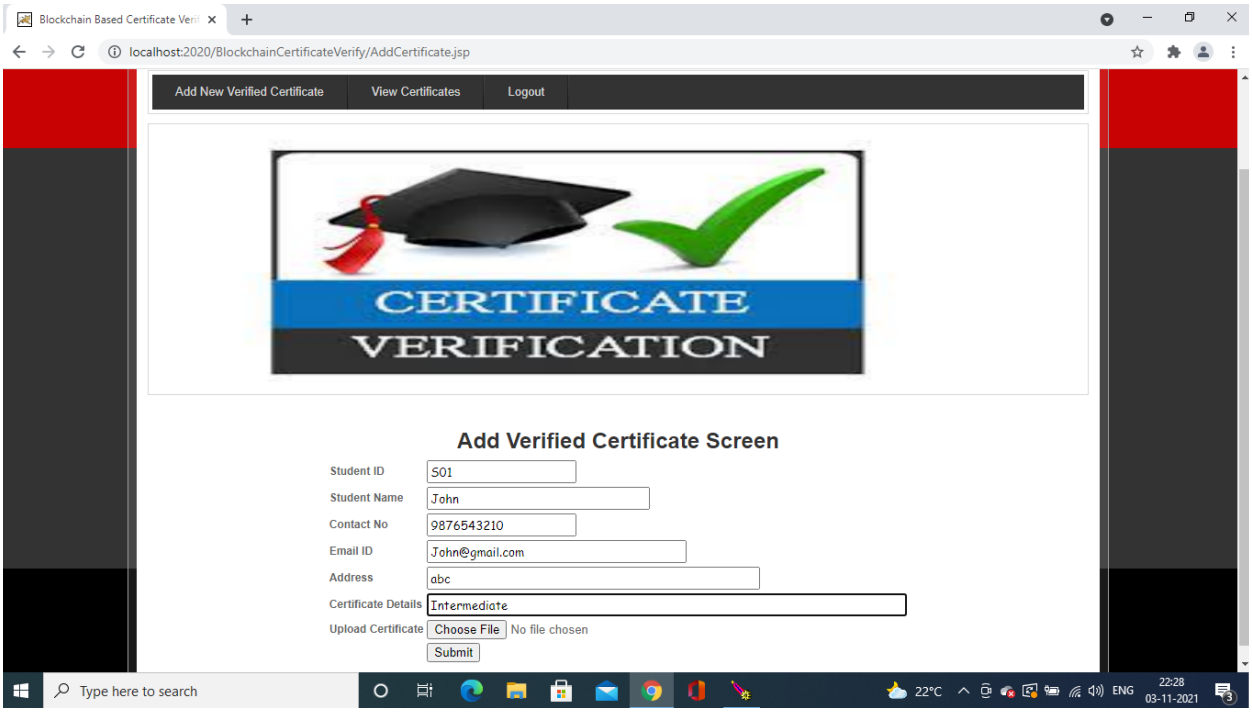
Screen 7: Welcome Official page

In above screen official can click on ‘ADD New Verified Certificate’ link to add certificate details

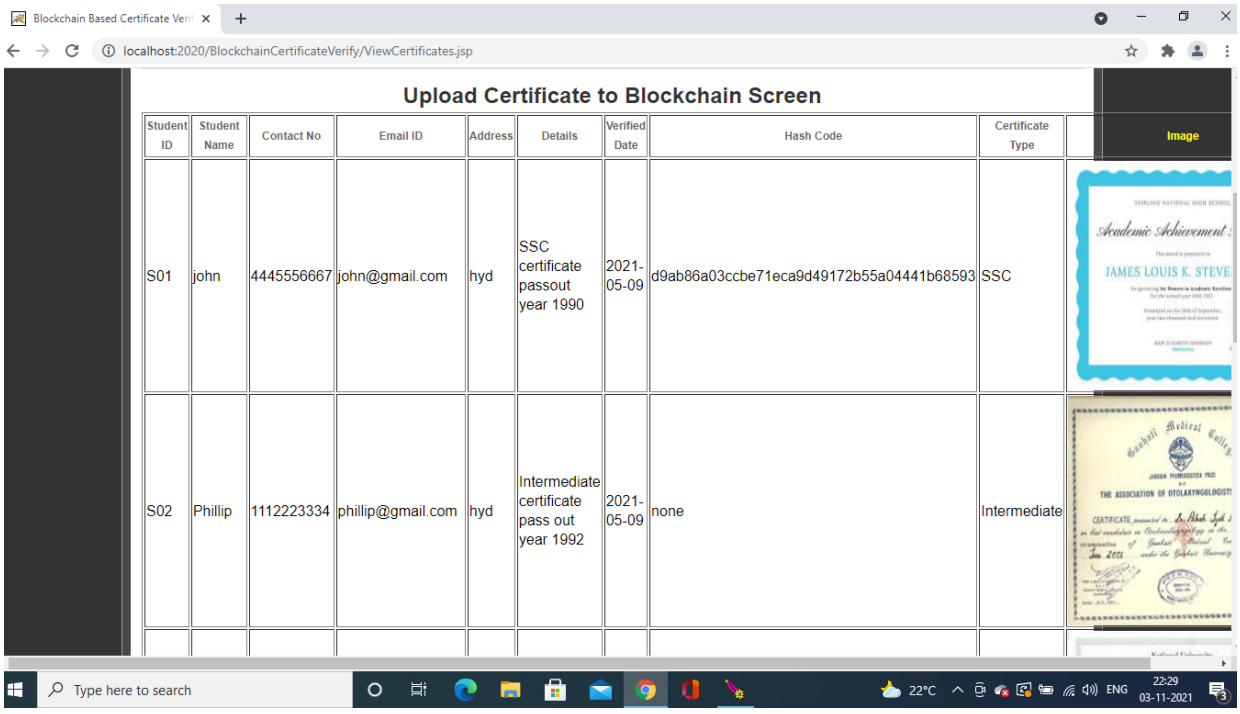


Screen 8: Add certificate details

In above screen official added certificate details of student and then click on choose file to add certificate image.

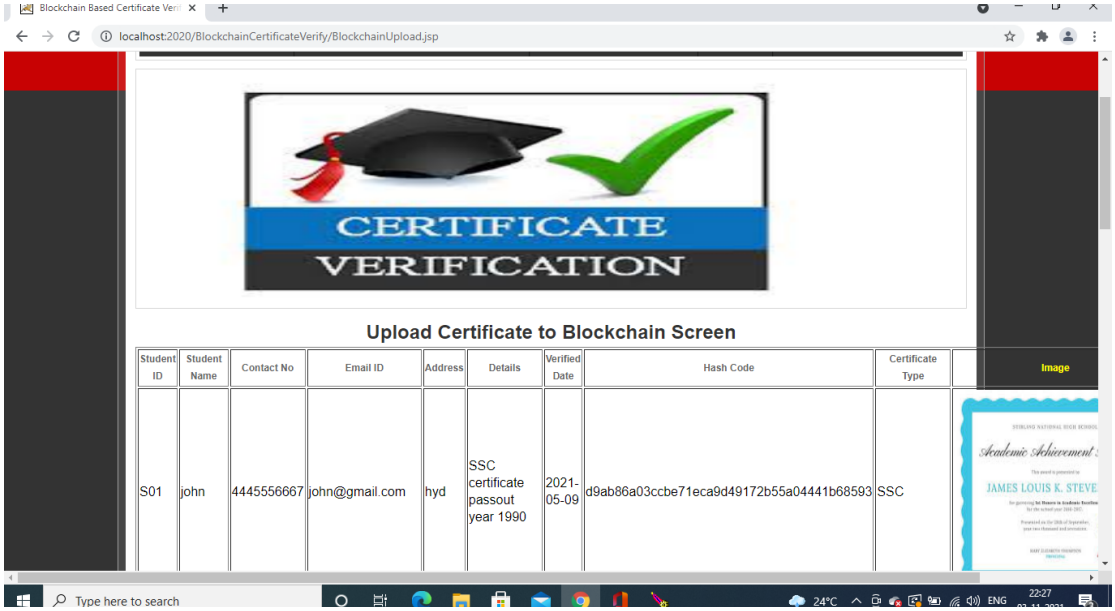


Screen 9: Student details



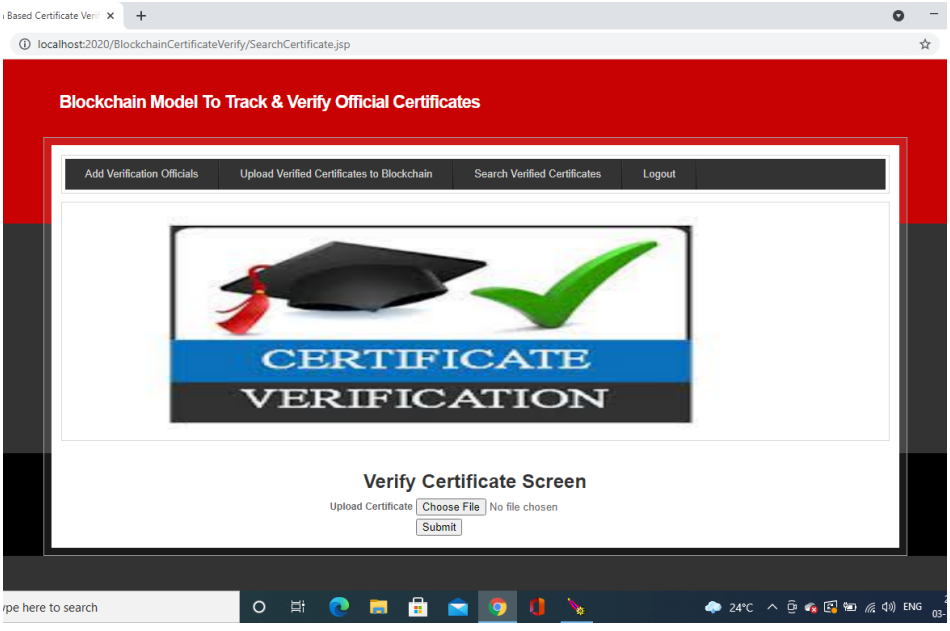
Screen 10: View certificates

In above screen verifier can view all certificates stored in Blockchain and now official logout and then admin logged in to upload above certificates to Blockchain



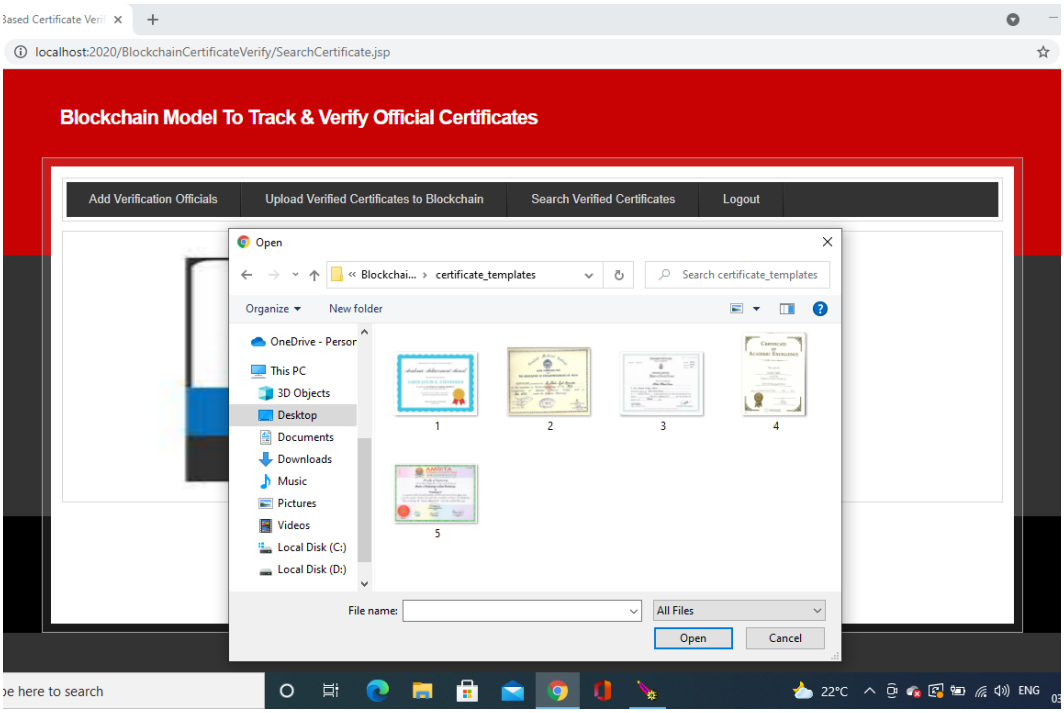
Screen 11: Upload to blockchain

In above screen admin can view all uploaded certificates and if certificate already stored in Blockchain then last column show values as ‘Done’ and if not stored then will get value as ‘Click Here’ and admin can Click on ‘Click Here’ link to store that certificate in Blockchain.



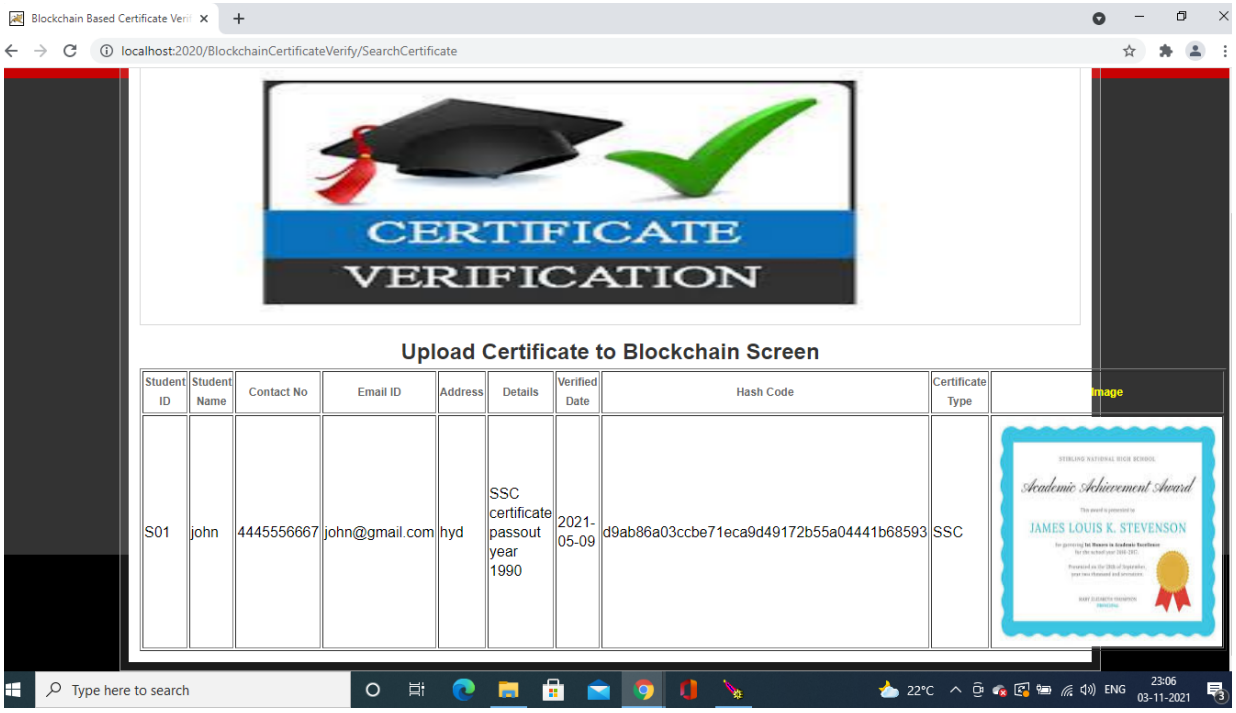
Screen 12: Verification

Using the above page, admin can validate if a certificate is original or fake



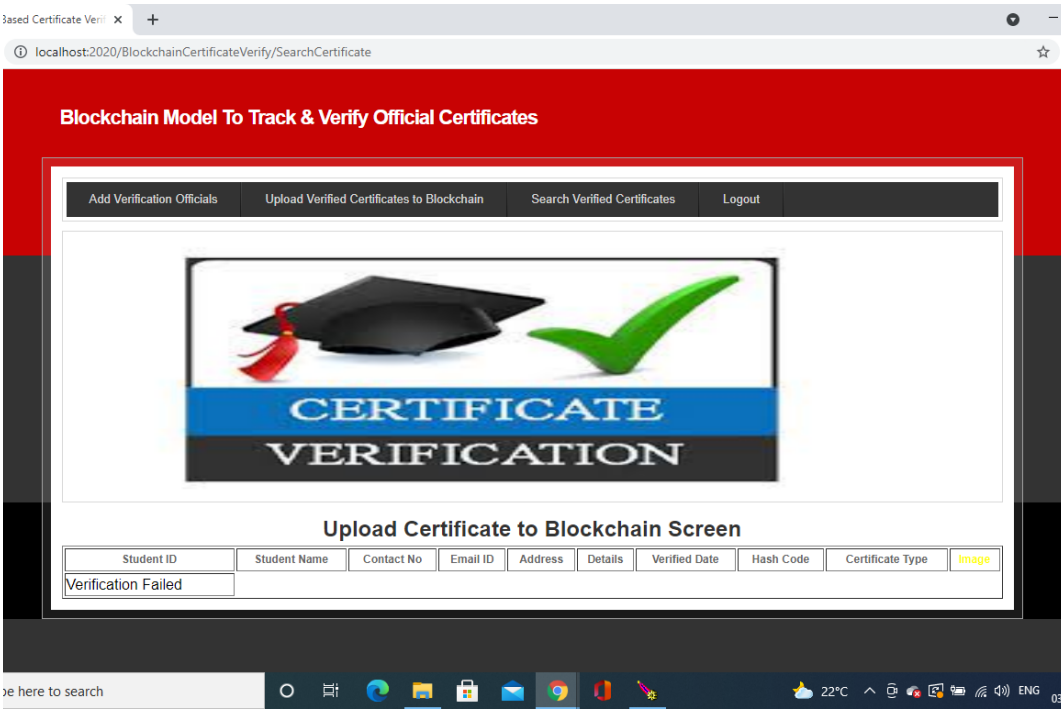
Screen 13: Certificate validation

In above screen admin can upload test certificate and then click on 'Submit' button to see the results.



Screen 14: Verification Success

If the certificate is original the above screen would be displayed.If the certificate is fake the below screen would be displayed



Screen 15: Verification failed

8. CONCLUSION

In this paper, we proposed a solution to the problem of certificate forgery based on blockchain technology. Providing security to the data is very important. By using the unchallengeable property of blockchain, we can provide more security for data and reduce the certificate forgery. The application can allow the user to view and validate the certificate. This system guarantees information accuracy and security and easy for people to manage digital certificates.

This research is very useful for the implementation of certificate validation using blockchain technology. Future research will focus on this overall scalability and speed over time to improve the user experience.

REFERENCES

Journals:

[1] Jiin-Chiou Cheng; Narn-Yih Lee; Chien Chi; Yi-Hua Chen, "Blockchain and Smart Contract for Digital Certificate" IEEE International Conference on Applied System Invention (ICASI),2018.

[2] Wang Z., Lin J., Cai Q., Wang Q., Jing J., Zha D. (2019) Blockchain-Based Certificate Transparency and Revocation Transparency. In: Zohar A. et al. (eds) Financial Cryptography and Data Security. FC 2018. Lecture Notes in Computer Science, vol 10958. Springer, Berlin, Heidelberg.

[3] D. S. V. Madala, M. P. Jhanwar, and A. Chattopadhyay, "Certificate Transparency Using Blockchain," 2018 IEEE International Conference on Data Mining Workshops (ICDMW), Singapore, Singapore, 2018, pp. 71-80, doi: 10.1109/ICDMW.2018.00018.

[4] Aisong Zhang and Xinxin Ma, "Decentralized Digital Certificate Revocation System Based on Blockchain", Journal of Physics: Conference Series, Volume 1069, 3rd Annual International Conference on Information System and Artificial Intelligence (ISAI2018) 22–24 June 2018, Suzhou.

[5] Marco Baldi, Franco Chiaraluce, Emanuele Frontoni, Giuseppe Gottardi, Daniele Sciarroni, and Luca Spalazzi "Certificate Validation through Public Ledgers and Blockchains In Proceedings of the First Italian Conference on Cybersecurity.

Websites:

1. https://www.codecademy.com/learn
2. https://www.blockchain.com/
3. https://www.geeksforgeeks.org/