

## **INDEX**

<b>TOPICS</b>	<b>Page No's</b>
➤ <b>Certificates</b>	
➤ <b>Acknowledgement</b>	
➤ <b>Abstract</b>	
➤ <b>Figures/Tables</b>	
<b>CHAPTER-1: INTRODUCTION</b>	<b>1</b>
<b>CHAPTER-2: LITERATURE SURVEY</b>	<b>2-4</b>
<b>CHAPTER-3: SYSTEM ANALYSIS</b>	
3.1 Existing System	5
3.2 Proposed System	5-6
<b>CHAPTER-4: SYSTEM REQUIREMENTS</b>	
4.1 Functional Requirement	7
4.2 Non-Functional Requirements	7-8
<b>CHAPTER-5: SYSTEM STUDY</b>	
5.1 Feasibility Study	9
5.2 Feasibility Analysis	9-10
<b>CHAPTER-6: SYSTEM DESIGN</b>	
<b>6.1 SYSTEM ARCHITECTURE</b>	<b>11</b>
<b>6.2 UML DIAGRAMS</b>	<b>12-19</b>
6.2.1 Use Case Diagram	
6.2.2 Class Diagram	
6.2.3 Sequence Diagram	
6.2.4 Collaboration Diagram	
6.2.5 Activity Diagram	
6.2.6 Component Diagram	
6.2.7 Deployment Diagram	

6.2.8 Er Diagram	
6.2.9 Data Dictionary	
<b>CHAPTER-7: INPUT AND OUTPUT DESIGN</b>	
7.1 Input Design	20
7.2 Output Design	21
<b>CHAPTER-8: IMPLEMENTATION</b>	
8.1 Modules	22
8.1.1 Module Description	22-23
<b>CHAPTER-9: SOFTWARE ENVIRONMENT</b>	
9.1 Python	24-63
9.2 Source Code	64-90
<b>CHAPTER-10: RESULTS/DISCUSSIONS</b>	
10.1 System Testing	91-94
10.1.1 Test Cases	94-97
10.2 Output Screens	97-111
<b>CHAPTER-11: CONCLUSION</b>	
11.1 Conclusion	112
11.2 Future Scope	112
<b>CHAPTER-12: REFERENCES</b>	<b>113-114</b>

## **LIST OF FIGURES**

<b>S.NO</b>	<b>TABLES/FIGURES</b>	<b>PAGE NO'S</b>
1	System Architecture	11
2	UML Diagrams	12-19
	2.1 Use Case Diagram	12
	2.2 Class Diagram	13
	2.3 Sequence Diagram	13
	2.4 Collaboration Diagram	14
	2.5 Activity Diagram	15
	2.6 Component Diagram	16
	2.7 Deployment Diagram	16
	2.8 ER Diagram	17
	2.9 Data Dictionary/Data Set	17-19
3	Python Source code with Compilation	24-63
4	Screenshots	97-111

# **STREAMLINING CREDENTIAL VERIFICATION FOR HIRING PROCESSES WITH BLOCKCHAIN TECHNOLOGY**

## **ABSTRACT**

In the current landscape, both companies and educational institutions rely heavily on manual processes to validate student academic credentials, which can be susceptible to fraudulent verification through bribery or manipulation by employees. Some universities have attempted to address this issue by maintaining student credentials in centralized servers. However, these centralized systems are not foolproof, as internal database administrators can manipulate data, leading to potential security breaches and service disruptions.

To overcome these challenges, we propose a Blockchain-based student credential verification system. Blockchain inherently supports data verification, encryption, and distributed data storage. By storing data in a decentralized manner across multiple nodes, the system ensures resilience in case of node failures, providing uninterrupted services.

Blockchain's built-in tamper-proof features involve storing each data record as a block or transaction, associating it with a unique hash code. When new records are added, the Blockchain verifies the hash codes of all previous blocks. If the data remains unchanged, the hash codes match, resulting in successful verification. Any attempt to alter data would lead to a hash code mismatch, triggering a verification failure and detection of data tampering.

# **CHAPTER-1**

## **INTRODUCTION**

In today's dynamic professional landscape, the need for reliable and efficient credential verification processes is paramount, particularly in the hiring domain. Traditional methods often fall prey to issues such as fraudulent credentials and time-consuming manual verification processes. In response to these challenges, the integration of cutting-edge blockchain technology emerges as a transformative solution. This project aims to streamline credential verification for hiring processes by harnessing the power of blockchain, ensuring a secure, transparent, and tamper-proof system.

In today's fast-paced digital era, businesses are continually seeking innovative solutions to streamline their operations, particularly in the realm of hiring processes. Traditional methods of verifying credentials often involve cumbersome and time-consuming procedures, leading to delays and inefficiencies in recruitment. However, with the emergence of blockchain technology, there exists a promising opportunity to revolutionize credential verification. Blockchain offers a decentralized and immutable ledger system that can securely store and validate credentials, ranging from academic qualifications to professional certifications. By leveraging blockchain technology, organizations can expedite the hiring process, enhance transparency, and mitigate the risk of credential fraud. This introduction sets the stage for exploring the transformative potential of blockchain in optimizing credential verification for hiring processes.

## **CHAPTER-2**

### **LITERATURE SURVEY**

**TITLE:** Cred Chain: Academic and Professional Certificate Verification System using Blockchain

**AUTHOR:** Saniyat Al Ahmed; Rifat Al Mamun Rudro; Afrina Jannat Prity

**ABSTRACT:** The integrity of academic and professional credentials is a foundation in education and the professional fields. Traditional certificate verification methods are flawed by in-efficiencies, susceptibility to fraud, and a reliance on manual processes that compromise security and expedience. To address these issues, we present CredChain, a robust Academic and Professional Certificate Verification System that harnesses the power of blockchain technology. Our system utilizes Ethereum-based Decentralized Applications (DApps) and Smart Contracts integrated with the InterPlanetary File System (IPFS) to ensure immutable, secure, and transparent verification processes. We outline the novel algorithms for essential system functions such as certificate uploading, verification, and retrieval of certificates. The CredChain system has been subjected to rigorous testing and validation, demonstrating its effectiveness in mitigating the risk of fraudulent certificates and streamlining the verification process accurately.

**TITLE:** Trustworthy Healthcare Professional Credential Verification Using Blockchain Technology

**AUTHOR:** Aysha Alnuaimi; Diana Hawashin; Raja Jayaraman

**ABSTRACT:** Healthcare credentialing plays a vital role in ensuring the competence and integrity of healthcare professionals. However, the current credentials verification process suffers from time-consuming procedures due to the large number of intermediaries, limited information access, data fragmentation and the persistent risk of fraudulent credentials, leading to delayed hiring, increased administrative burden, and loss of trust and reputation in the healthcare system. In this paper, we utilize blockchain technology to enhance the credentialing process by streamlining the verification steps, improving data security, and providing stakeholders with confidence through secure storage of credentials. In addition, we utilize advanced security techniques, such as proxy re-encryption and cryptographic algorithms, to ensure the protection of sensitive data, facilitate secure communication, and prevent

unauthorized access. We develop smart contracts which eliminate the need for intermediaries, automate the verification process, and enhance transparency and data integrity. We present system architecture, sequence diagrams, entity relationship diagrams, and the underlying algorithms of our blockchain-based solution. We discuss how our proposed solution attains the objectives outlined in the paper. We conduct cost evaluation and security analysis to validate the effectiveness of our solution. Additionally, we compare our proposed system with existing blockchain-based solutions, highlighting its novelty. The code of our smart contracts is made publicly available on GitHub.

**TITLE:** Blockchain Based Certificate Verification for Employee Hiring & Admission System

**AUTHOR:** Saber Al Sakib; Md. Rakib; Md. Monir Hossain

**ABSTRACT:** Students earn a variety of academic qualifications throughout their academic careers. The student offers their academic credentials while submitting an application for a job or scholarship. In light of the fact that cryptocurrencies are prohibited in academia, the purpose of this article is to put forth a hypothetical blockchain-based certificate verification system that runs on the cloud. In this paper, we analyze the application of Block-chain technology to address these problems. Immutability and publicly verifiable transactions may be provided by this blockchain. Additionally, these blockchain characteristics are combined to create a digital academic certificate that is easy to verify quickly and is anti-counterfeit. Furthermore, the planned "Blockchain-based Certificate Verification" demonstrated how a nation that forbade cryptocurrencies could use blockchain technology.

**TITLE:** Skill Check: An Incentive-based Certification System using Blockchains

**AUTHOR:** Jay Gupta; Swaprava Nath

**ABSTRACT:** Skill verification is a central problem in workforce hiring. Companies and academia often face the difficulty of ascertaining the skills of an applicant since the certifications of the skills claimed by a candidate are generally not immediately verifiable and costly to test. Blockchains have been proposed in the literature for skill verification and tamper-proof information storage in a decentralized manner. However, most of these approaches deal with storing the certificates issued by traditional universities on the blockchain. Among the few techniques that consider the certification procedure itself, questions like (a) scalability with

limited staff, (b) uniformity of grades over multiple evaluators, or (c) honest effort extraction from the evaluators are usually not addressed. We propose a blockchain-based platform named SkillCheck, which considers the questions above, and ensure several desirable properties. The platform incentivizes effort in grading via payments with tokens which it generates from the payments of the users of the platform, e.g., the recruiters and test takers. We provide a detailed description of the design of the platform along with the provable properties of the algorithm.

**TITLE:** Blockchain Powered Skill Verification System

**AUTHOR:** Radha Govindwar; Sumit Didhate; Sayali Dalal; Neha Musale

**ABSTRACT:** A skill verification system is the process of checking and verifying the skill legitimacy of a graduate student or an experienced employee. To maintain an objective, effective, transparent, and secure environment, these methods are necessary. Blockchain, a distributed digital ledger-based technology, can make it easier to implement these criteria successfully [1]. One of the most fundamental yet crucial tasks for every company is the verification of the candidate's qualifications and experience before hiring. In order to understand the present status of information technology usage in the field of human resource management and how Blockchain might help achieve smart, affordable, efficient, transparent, and secure factory management, a thorough literature research was undertaken. With an increasing number of applying candidates especially in larger firms, verifying candidate qualifications and experience is becoming very time-consuming, and as a result is overlooked many times. This problem of candidates falsifying their information is an increasing concern for the HR department and while conventional skill verification systems are useful, they are still far from perfect to handle this system [3].



## **CHAPTER-3**

### **SYSTEM ANALYSIS**

#### **3.1 EXISTING SYSTEM**

The prevailing credential verification systems heavily rely on manual processes, leading to vulnerabilities and inefficiencies. Companies and educational institutions grapple with the risks of fake verifications, bribery, and data tampering. While some have attempted to address these concerns through centralized servers, these systems are not immune to manipulation, presenting a critical need for an innovative and secure solution. The existing paradigm underscores the urgency to adopt a more robust and technologically advanced approach to credential verification.

#### **DISADVANTAGES**

**Technical Complexity** Implementing blockchain solutions requires expertise in both blockchain technology and the specific requirements of the hiring process. Developing and maintaining a blockchain network can be technically complex, requiring specialized knowledge and resources. **Scalability Concerns** Blockchain networks, especially public ones like Bitcoin or Ethereum, face scalability challenges when it comes to handling a large volume of transactions. As hiring processes involve numerous credential verifications, scalability issues may arise, leading to delays and increased costs. **Data Privacy and Security** While blockchain offers immutability and cryptographic security, ensuring data privacy can be challenging. Confidential information stored on the blockchain, such as personal credentials, must be adequately protected to prevent unauthorized access or breaches.

#### **3.2 PROPOSED SYSTEM**

The proposed system leverages the inherent features of blockchain technology to revolutionize the credential verification landscape. Blockchain, with its decentralized, tamper-proof architecture, provides an ideal framework for storing and verifying academic and professional credentials. The system consists of three interconnected modules: the University Module, empowering institutions to securely upload student certificates; the Company Module, enabling efficient access to verified credentials for hiring purposes; and the Student Module, providing students control over their own data.

Through this blockchain-based solution, each academic or professional accomplishment is stored as a unique block, associated with a hash code for tamper-proof verification. The decentralized nature of the blockchain ensures data redundancy, mitigating the risk of single-point failures. The transparency and efficiency of the proposed system significantly enhance the trustworthiness of credential verification, marking a paradigm shift in hiring processes.

The proposed system consists of three main modules:

University Module:

- Universities can sign up and log in to the application.
- After logging in, universities can enroll students in various courses and upload certificates to the Blockchain.
- Each certificate is identified by a unique hash code address.
- Universities can access a list of students enrolled in different courses.

## **ADVANTAGES**

**Enhanced Security** Blockchain employs cryptographic techniques to secure data, making it highly resistant to tampering and fraud. Once information is recorded on the blockchain, it becomes immutable, ensuring the integrity of credentials throughout the hiring process.

**Improved Efficiency** Blockchain streamlines the verification process by providing a decentralized and transparent ledger accessible to authorized parties. This eliminates the need for manual verification steps, reducing the time and resources required to authenticate credentials.

## **CHAPTER-4**

### **SYSTEM REQUIREMENTS**

#### **4.1 FUNCTIONAL REQUIREMENTS**

- **UNIVERSITY**

Designing a university module on "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" involves structuring a comprehensive curriculum that educates students about blockchain fundamentals, its application in credential verification, and its implications for hiring processes.

- **COMPANY**

Developing a company module on "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" involves tailoring the curriculum to address the specific needs and objectives of businesses interested in leveraging blockchain for credential verification.

- **STUDENT**

Designing a student module on "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" involves creating a curriculum that introduces students to blockchain fundamentals, explores its application in credential verification, and encourages critical thinking about its implications for future employment.

#### **4.2 NON-FUNCTIONAL REQUIREMENTS**

##### **4.2.1 HARDWARE REQUIREMENTS**

- |             |                             |
|-------------|-----------------------------|
| ➤ Processor | - Pentium –IV               |
| ➤ RAM       | - 4 GB (min)                |
| ➤ Hard Disk | - 20 GB                     |
| ➤ Key Board | - Standard Windows Keyboard |
| ➤ Mouse     | - Two or Three Button Mouse |

- Monitor - SVGA

#### **4.2.2 SOFTWARE REQUIREMENTS**

- ❖ **Operating system** : Windows 7 Ultimate.
- ❖ **Coding Language** : Python.
- ❖ **Front-End** : Python.
- ❖ **Back-End** : Django-ORM
- ❖ **Designing** : Html, css, javascript.
- ❖ **Data Base** : MySQL (WAMP Server).

## **CHAPTER-5**

### **SYSTEM STUDY**

#### **5.1 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

#### **5.2 FEASIBILITY ANALYSIS**

Three key considerations involved in the feasibility analysis are

- **ECONOMICAL FEASIBILITY**
- **TECHNICAL FEASIBILITY**
- **SOCIAL FEASIBILITY**

#### **ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

#### **TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement,

as only minimal or null changes are required for implementing this system.

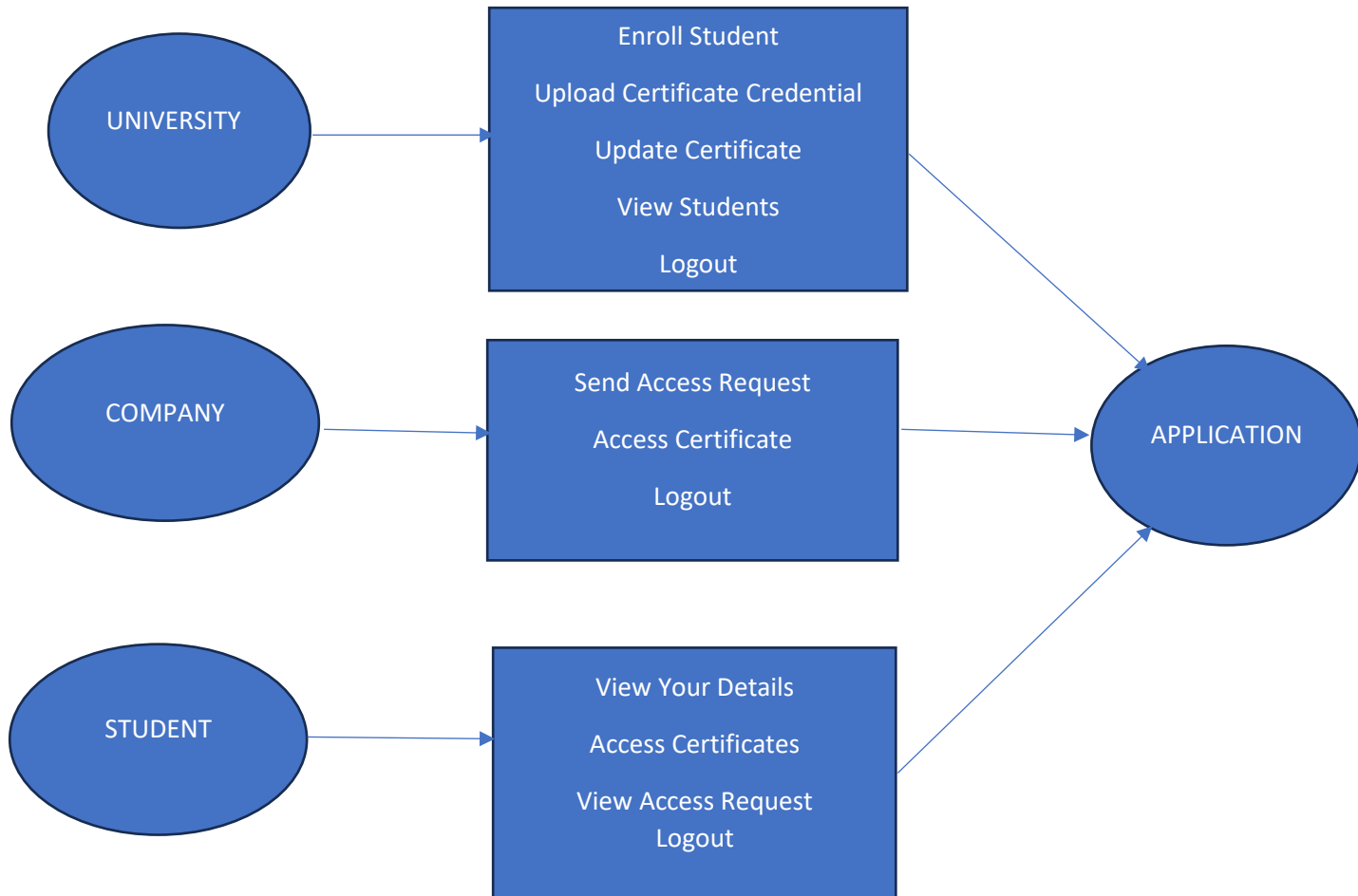
## **SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

# CHAPTER-6

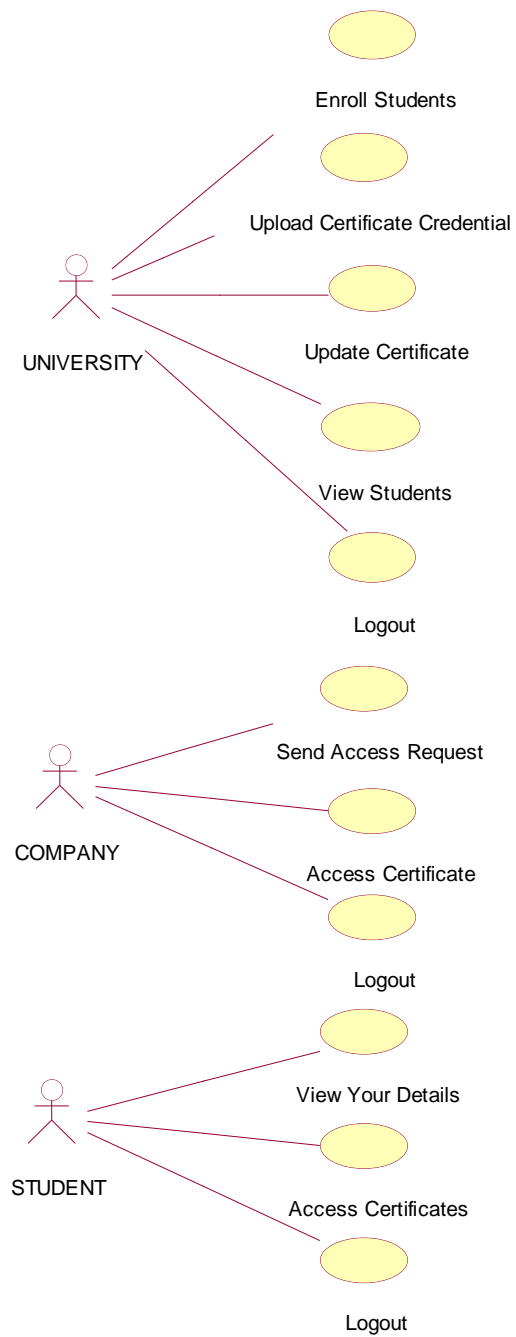
## SYSTEM DESIGN

### 6.1 SYSTEM ARCHITECTURE



## 6.2 UML DIAGRAMS

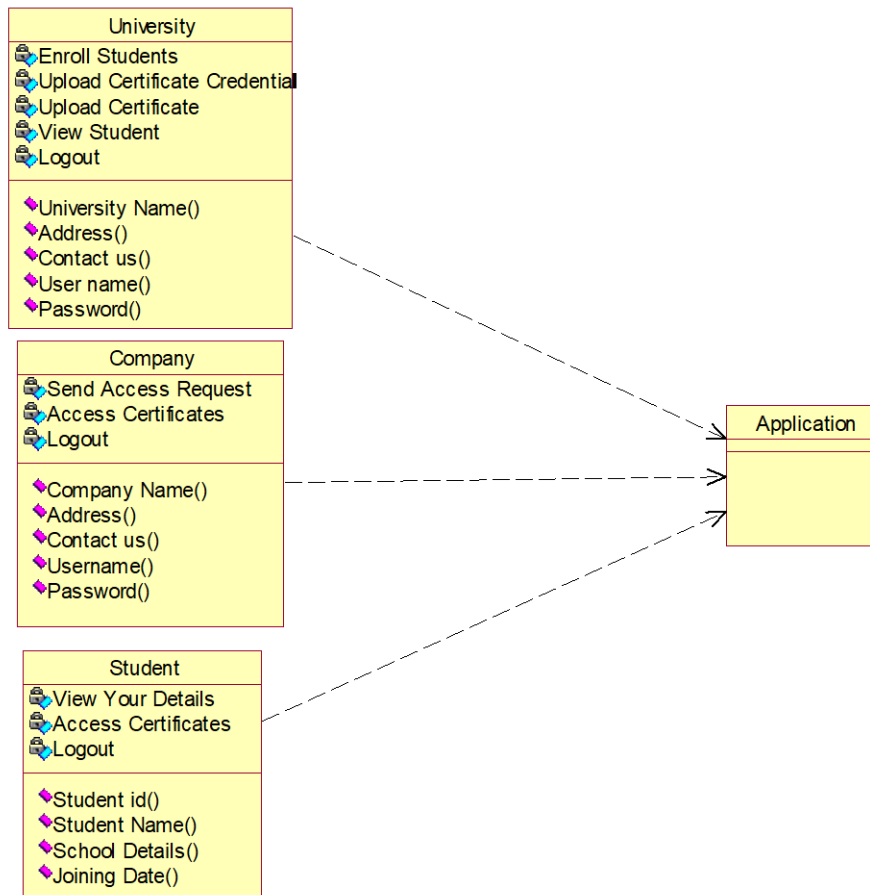
### 6.2.1 USE CASE DIAGRAM



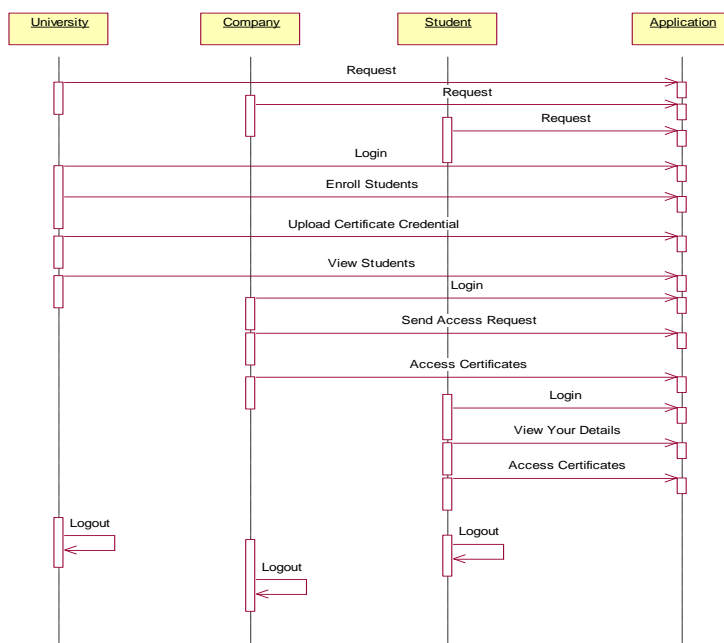
\



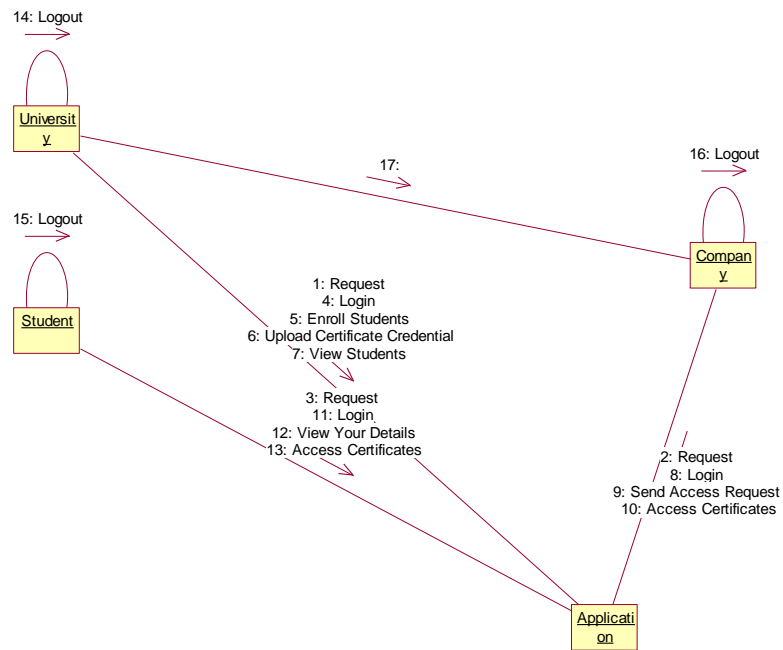
## 6.2.2 CLASS DIAGRAM



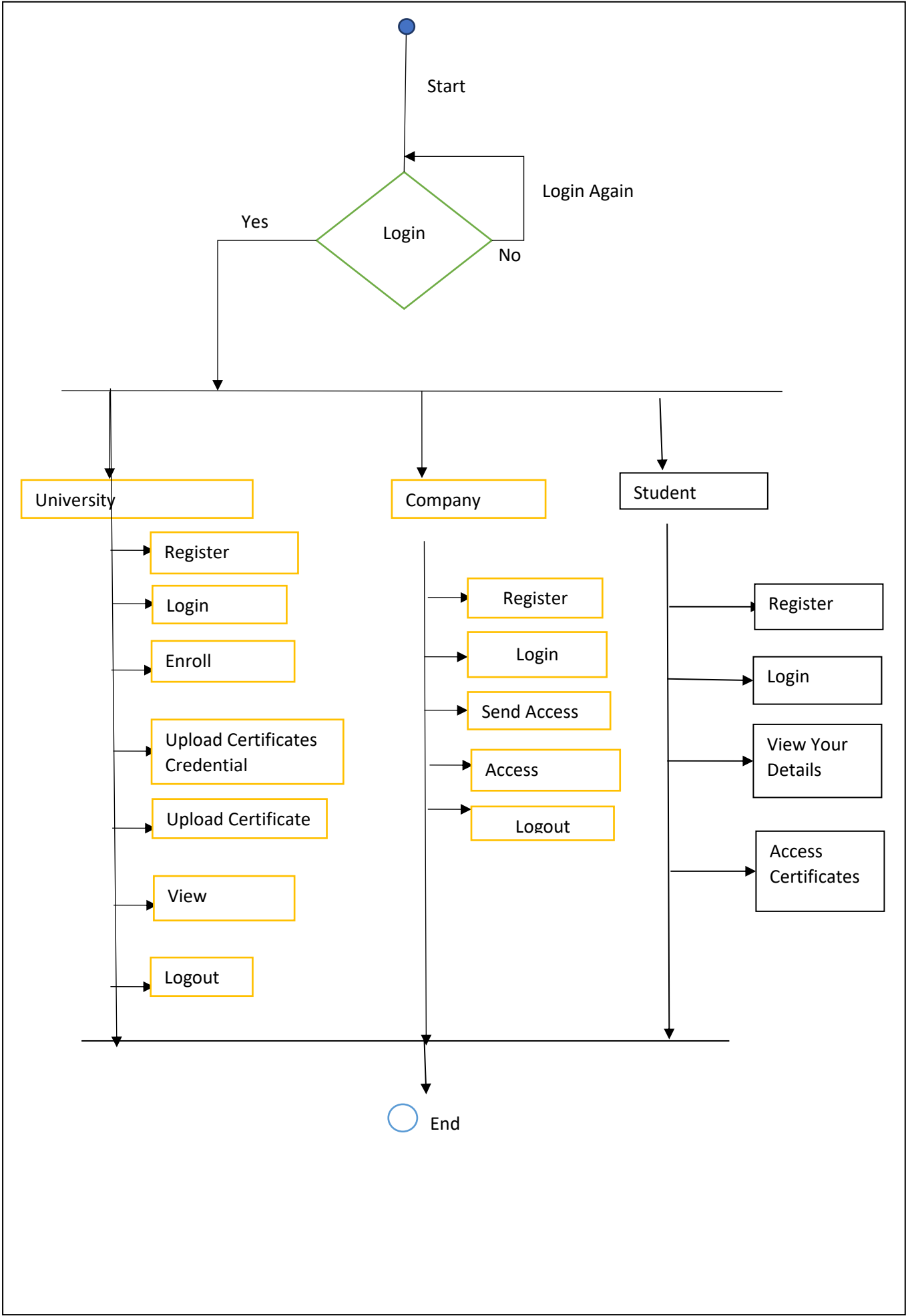
## 6.2.3 SEQUENCE DIAGRAM



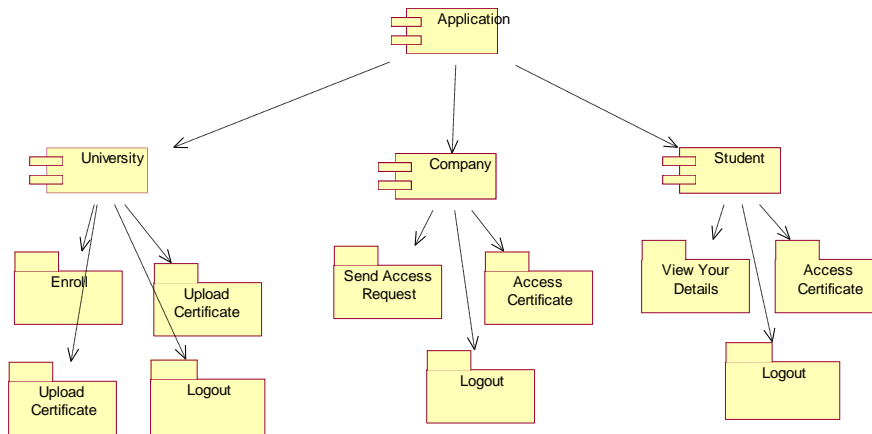
## 6.2.4 COLLABRATION DIAGRAM



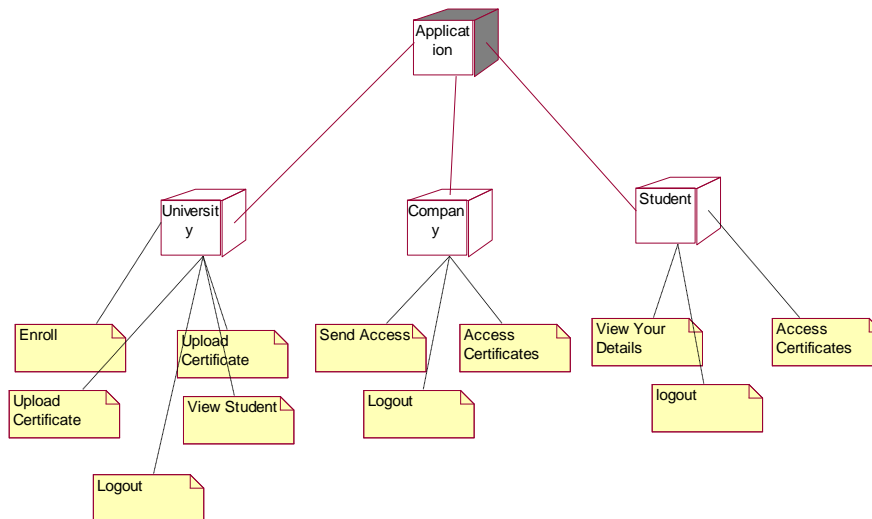
## 6.2.5 ACTIVITY DIAGRAM



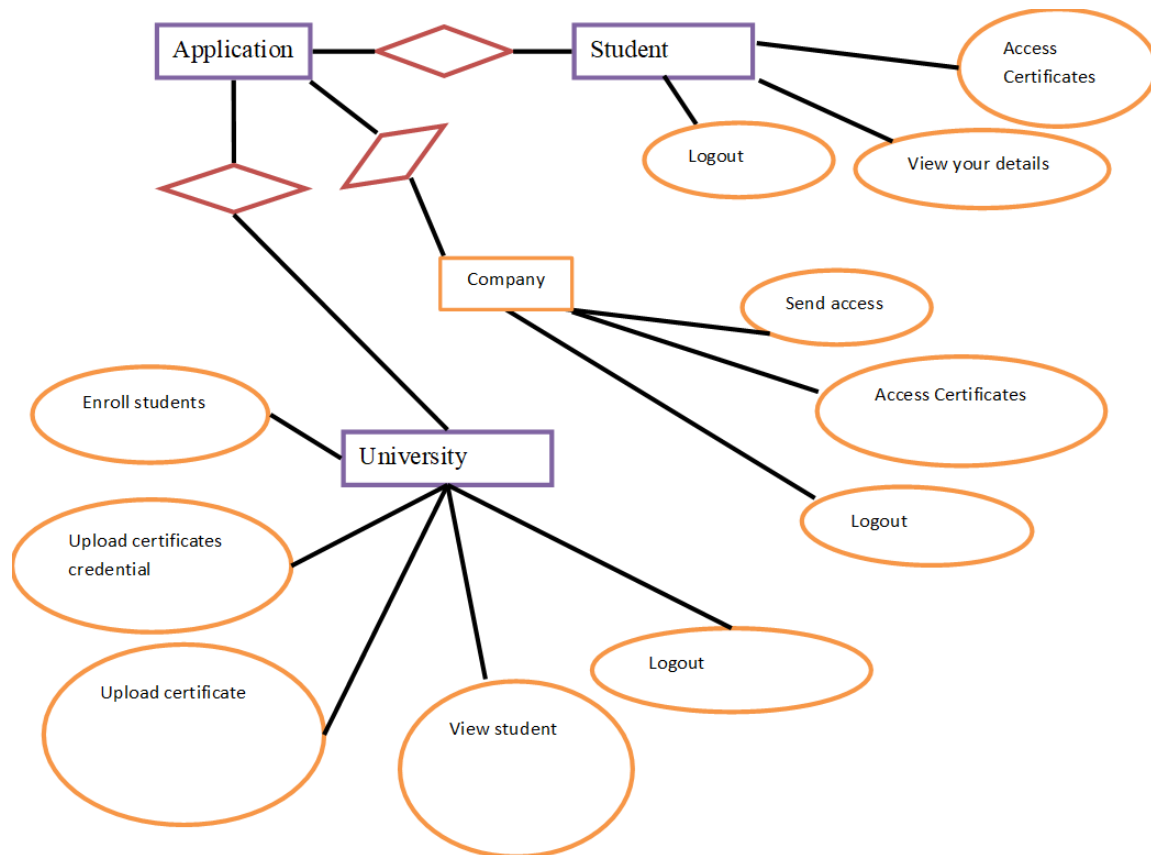
## 6.2.6 COMPONENT DIAGRAM



## 6.2.7 DEPLOYMENT DIAGRAM



## 6.2.8 ER DIAGRAM



## 6.2.9 DATA DICTIONARY

**Database:** streamlining credential verification for hiring processes with blockchain technology

methods

Table name: auth\_group

Column	Data Type	Constraints	Description
--------	-----------	-------------	-------------

id	Int(11)	Primary key	Unique Identifier
name	Varchar(1000)	Not null	name

Table Name: Auth\_group\_Permission

Table Name: auth\_permission

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
Group id	Int(11)	Primary Key	Unique Identifier
Permission_id	Int(11)	Primary Key	Unique Identifier
Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
Name	Int(255)	Not Null	name
Content_type_id	Int(11)	Primary Key	Unique Identifier
Code name	Int(100)	Not Null	name

Table Name: auth\_User

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
Password	varchar(128)	Not Null	password
Last_Login	Datetime(6)	Not Null	Last login
Is_superuser	tinyint(1)	Not Null	Name
username	Varchar(150)	Not Null	Username
lastname	Varchar(30)	Not Null	Lastname
email	Varchar(150)	Not Null	Email id
Is_staff	Tinyint(1)	Not Null	Staff
Is_active	Tinyint(1)	Not Null	Active
Date_joined	Datetime(6)	Not Null	Date and time

Table Name: auth\_user\_groups

Column	Data Type	Constraints	Description
id	Int(11)	Primary key	Unique Identifier
User_id	Int(11)	Primary Key	Unique Identifier
Group_id	Int(11)	Primary Key	Unique Identifier

# **CHAPTER-7**

## **INPUT AND OUTPUT DESIGN**

### **7.1 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

- What data should be given as input?
- How the data should be arranged or coded?
- The dialog to guide the operating personnel in providing input.
- Methods for preparing input validations and steps to follow when error occur.

#### **7.1.1 OBJECTIVES**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in



maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

## **7.2 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

- Convey information about past activities, current status or projections of the
- Future.
- Signal important events, opportunities, problems, or warnings.
- Trigger an action.
- Confirm an action.

# **CHAPTER-8**

## **IMPLEMENTATION**

### **8.1 MODULES**

- University
- Company
- Student

#### **8.1.1 MODULE DESCRIPTION**

##### **UNIVERSITY**

This module provides an in-depth exploration of how blockchain technology can revolutionize the credential verification process in hiring. As traditional methods often suffer from inefficiencies, lack of transparency, and susceptibility to fraud, blockchain offers promising solutions by providing a decentralized, immutable ledger for storing and verifying credentials. Through a combination of theoretical understanding, practical applications, and case studies, students will gain insights into the potential of blockchain to streamline hiring processes, enhance security, and foster trust between employers and candidates.

##### **COMPANY**

This module delves into the application of blockchain technology to streamline credential verification, mitigating common challenges such as data inaccuracies, delays, and fraud. Participants will gain a comprehensive understanding of blockchain fundamentals, exploring its decentralized architecture, cryptographic security, and immutable ledger capabilities. Through case studies, practical exercises, and discussions, participants will learn how blockchain can revolutionize credential verification, offering enhanced security, efficiency, and transparency in hiring processes.

##### **STUDENT**

This module explores how blockchain offers innovative solutions by providing a decentralized, transparent, and immutable ledger for storing and verifying credentials. Through theoretical

insights, practical applications, and interactive discussions, students will learn about blockchain fundamentals, including its decentralized architecture, cryptographic security, and consensus mechanisms. They will also examine real-world use cases of blockchain in credential verification and explore the potential implications of blockchain adoption on future employment trends.

# CHAPTER-9

## SOFTWARE ENVIRONMENT

### 9.1 PYTHON

Python is a **high-level, interpreted, interactive and object-oriented scripting language**. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

- **Python is Interpreted:** Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- **Python is Interactive:** You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
- **Python is Object-Oriented:** Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
- **Python is a Beginner's Language:** Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

### HISTORY OF PYTHON

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

## PYTHON FEATURES

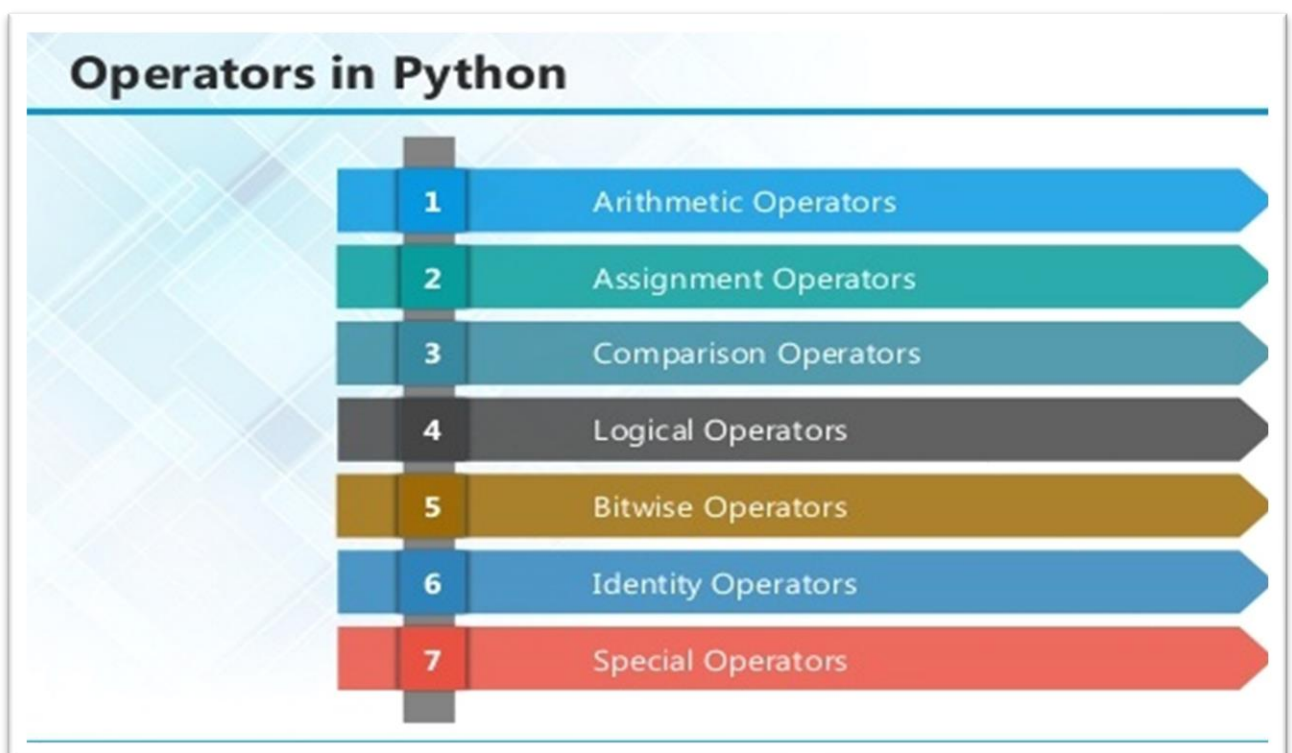
### PYTHON'S FEATURES INCLUDE:

- **Easy-to-learn:** Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
- **Easy-to-read:** Python code is more clearly defined and visible to the eyes.
- **Easy-to-maintain:** Python's source code is fairly easy-to-maintain.
- **A broad standard library:** Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
- **Interactive Mode:** Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
- **Portable:** Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
- **Extendable:** You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
- **Databases:** Python provides interfaces to all major commercial databases.
- **GUI Programming:** Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
- **Scalable:** Python provides a better structure and support for large programs than shell scripting.

Python has a big list of good features:

- It supports functional and structured programming methods as well as OOP.

- It can be used as a scripting language or can be compiled to byte-code for building large applications.
- It provides very high-level dynamic data types and supports dynamic type checking.
- IT supports automatic garbage collection.
- It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.



## ARITHMETIC OPERATORS`

Operator	Description	Example
+ Addition	Adds values on either side of the operator.	$a + b = 30$
- Subtraction	Subtracts right hand operand from left hand operand.	$a - b = -10$
* Multiplication	Multiplies values on either side of the operator	$a * b = 200$
/ Division	Divides left hand operand by right hand operand	$b / a = 2$
% Modulus	Divides left hand operand by right hand operand and returns remainder	$b \% a = 0$
** Exponent	Performs exponential (power) calculation on operators	$a ** b = 10$ to the power 20
//	Floor Division - The division of operands where the result is the quotient in which the digits after the decimal point are removed. But if one of the operands is negative, the result is floored, i.e., rounded away from zero (towards negative infinity):	$9 // 2 = 4$ and $9.0 // 2.0 = 4.0$ , - $11 // 3 = -4$ , -

		11.0//3 = -4.0
--	--	-------------------

## ASSIGNMENT OPERATOR

Operator	Description	Example
=	Assigns values from right side operands to left side operand	c = a + b assigns value of a + b into c
+= Add AND	It adds right operand to the left operand and assign the result to left operand	c += a is equivalent to c = c + a
-= Subtract AND	It subtracts right operand from the left operand and assign the result to left operand	c -= a is equivalent to c = c - a
*= Multiply AND	It multiplies right operand with the left operand and assign the result to left operand	c *= a is equivalent to c = c * a



<code>/=</code> Divide AND	It divides left operand with the right operand and assign the result to left operand	<code>c /= a</code> is equivalent to <code>c = c / a</code> <code>ac /= a</code> is equivalent to <code>c = c / a</code>
----------------------------	--	---

<code>%=</code> Modulus AND	It takes modulus using two operands and assign the result to left operand	<code>c %= a</code> is equivalent to <code>c = c % a</code>
<code>**=</code> Exponent AND	Performs exponential (power) calculation on operators and assign value to the left operand	<code>c **= a</code> is equivalent to <code>c = c ** a</code>
<code>//=</code> Floor Division	It performs floor division on operators and assign value to the left operand	<code>c //= a</code> is equivalent to <code>c = c // a</code>

## IDENTITY OPERATOR

Operator	Description	Example
<code>is</code>	Evaluates to true if the variables on either side of the operator point to the same object and false otherwise.	<code>x is y</code> , here <code>is</code> results in 1 if <code>id(x)</code> equals <code>id(y)</code> .

is not	Evaluates to false if the variables on either side of the operator point to the same object and true otherwise.	x is not y, here <b>is not</b> results in 1 if id(x) is not equal to id(y)
--------	---	--

## COMPARISON OPERATOR

Operator	Description	Example
& Binary AND	Operator copies a bit to the result if it exists in both operands	(a & b) (means 0000 1100)
Binary OR	It copies a bit if it exists in either operand.	(a   b) = 61 (means 0011 1101)
^ Binary XOR	It copies the bit if it is set in one operand but not both.	(a ^ b) = 49 (means 0011 0001)
~ Binary Ones Complement	It is unary and has the effect of 'flipping' bits.	(~a) = -61 (means 1100 0011 in 2's complement form due to

		a signed binary number.
<< Binary Left Shift	The left operands value is moved left by the number of bits specified by the right operand.	$a \ll 2 = 240$ (means 1111 0000)
>> Binary Right Shift	The left operands value is moved right by the number of bits specified by the right operand.	$a \gg 2 = 15$ (means 0000 1111)

## LOGICAL OPERATOR

Operator	Description	Example
and Logical AND	If both the operands are true then condition becomes true.	(a and b) is true.
or Logical OR	If any of the two operands are non-zero then condition becomes true.	(a or b) is true.
not Logical NOT	Used to reverse the logical state of its operand.	Not(a and b) is false.

## MEMBERSHIP OPERATORS

Operator	Description	Example
in	Evaluates to true if it finds a variable in the specified sequence and false otherwise.	x in y, here in results in a 1 if x is a member of sequence y.
not in	Evaluates to true if it does not finds a variable in the specified sequence and false otherwise.	x not in y, here not in results in a 1 if x is not a member of sequence y.

## PYTHON OPERATORS PRECEDENCE

Operator	Description
**	Exponentiation (raise to the power)
~ + -	Complement, unary plus and minus (method names for the last two are +@ and -@)
* / % //	Multiply, divide, modulo and floor division
+ -	Addition and subtraction
>> <<	Right and left bitwise shift

&	Bitwise 'AND'
^	Bitwise exclusive 'OR' and regular 'OR'
<= < > >=	Comparison operators
<> == !=	Equality operators
= %= /= //= -= += *= **=	Assignment operators
is is not	Identity operators
in not in	Membership operators
not or and	Logical operators

## LIST

The list is a most versatile data type available in Python which can be written as a list of comma-separated values (items) between square brackets. Important thing about a list is that items in a list need not be of the same type.

Creating a list is as simple as putting different comma-separated values between square brackets. For example –

```
list1 = ['physics', 'chemistry', 1997, 2000];
list2 = [1, 2, 3, 4, 5];
list3 = ["a", "b", "c", "d"]
```

## BASIC LIST OPERATIONS

Lists respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

Python Expression	Results	Description
<code>len([1, 2, 3])</code>	3	Length
<code>[1, 2, 3] + [4, 5, 6]</code>	<code>[1, 2, 3, 4, 5, 6]</code>	Concatenation
<code>['Hi!'] * 4</code>	<code>['Hi!', 'Hi!', 'Hi!', 'Hi!']</code>	Repetition
<code>3 in [1, 2, 3]</code>	True	Membership
<code>for x in [1, 2, 3]: print x,</code>	1 2 3	Iteration

## BUILT-IN LIST FUNCTIONS & METHODS:

Python includes the following list functions –

SN	Function with Description
1	<u>cmp(list1, list2)</u>  Compares elements of both lists.
2	<u>len(list)</u>  Gives the total length of the list.
3	<u>max(list)</u>  Returns item from the list with max value.
4	<u>min(list)</u>  Returns item from the list with min value.
5	<u>list(seq)</u>  Converts a tuple into list.

Python includes following list methods

SN	Methods with Description
1	<u><code>list.append(obj)</code></u>  Appends object obj to list
2	<u><code>list.count(obj)</code></u>  Returns count of how many times obj occurs in list
3	<u><code>list.extend(seq)</code></u>  Appends the contents of seq to list
4	<u><code>list.index(obj)</code></u>  Returns the lowest index in list that obj appears
5	<u><code>list.insert(index, obj)</code></u>  Inserts object obj into list at offset index
6	<u><code>list.pop(obj=list[-1])</code></u>  Removes and returns last object or obj from list
7	<u><code>list.remove(obj)</code></u>  Removes object obj from list
8	<u><code>list.reverse()</code></u>  Reverses objects of list in place



9	<u>list.sort([func])</u>  Sorts objects of list, use compare function if given
---	--

## TUPLES

A tuple is a sequence of immutable Python objects. Tuples are sequences, just like lists. The differences between tuples and lists are, the tuples cannot be changed unlike lists and tuples use parentheses, whereas lists use square brackets.

Creating a tuple is as simple as putting different comma-separated values. Optionally we can put these comma-separated values between parentheses also. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5 );
tup3 = "a", "b", "c", "d";
```

The empty tuple is written as two parentheses containing nothing –

```
tup1 = ();
```

To write a tuple containing a single value you have to include a comma, even though there is only one value –

```
tup1 = (50,);
```

Like string indices, tuple indices start at 0, and they can be sliced, concatenated, and so on.

- Accessing Values in Tuples:

To access values in tuple, use the square brackets for slicing along with the index or indices to obtain value available at that index. For example –

```
tup1 = ('physics', 'chemistry', 1997, 2000);
tup2 = (1, 2, 3, 4, 5, 6, 7 );
print "tup1[0] : ", tup1[0]
```

```
print "tup2[1:5] : ", tup2[1:5]
```

When the code is executed, it produces the following result –

```
tup1[0]: physics  
tup2[1:5]: [2, 3, 4, 5]
```

## UPDATING TUPLES:

Tuples are immutable which means you cannot update or change the values of tuple elements. We are able to take portions of existing tuples to create new tuples as the following example demonstrates –

```
tup1 = (12, 34.56);  
tup2 = ('abc', 'xyz');  
tup3 = tup1 + tup2;  
print tup3
```

When the above code is executed, it produces the following result –

```
(12, 34.56, 'abc', 'xyz')
```

## DELETE TUPLE ELEMENTS

Removing individual tuple elements is not possible. There is, of course, nothing wrong with putting together another tuple with the undesired elements discarded.

To explicitly remove an entire tuple, just use the **del** statement. For example:

```
tup = ('physics', 'chemistry', 1997, 2000);  
print tup  
del tup;  
print "After deleting tup : "  
print tup
```

### Basic Tuples Operations:

Python Expression	Results	Description
<code>len((1, 2, 3))</code>	3	Length
<code>(1, 2, 3) + (4, 5, 6)</code>	<code>(1, 2, 3, 4, 5, 6)</code>	Concatenation
<code>('Hi!') * 4</code>	<code>('Hi!', 'Hi!', 'Hi!', 'Hi!')</code>	Repetition
<code>3 in (1, 2, 3)</code>	True	Membership
<code>for x in (1, 2, 3): print x,</code>	1 2 3	Iteration

### Built-in Tuple Functions

SN	Function with Description
1	<b>cmp(tuple1, tuple2)</b> :Compares elements of both tuples.
2	<b>len(tuple)</b> :Gives the total length of the tuple.
3	<b>max(tuple)</b> :Returns item from the tuple with max value.
4	<b>min(tuple)</b> :Returns item from the tuple with min value.
5	<b>tuple(seq)</b> :Converts a list into tuple.

## DICTIONARY

Each key is separated from its value by a colon (:), the items are separated by commas, and the whole thing is enclosed in curly braces. An empty dictionary without any items is written with just two curly braces, like this: {}.

Keys are unique within a dictionary while values may not be. The values of a dictionary can be of any type, but the keys must be of an immutable data type such as strings, numbers, or tuples.

### Accessing Values in Dictionary

To access dictionary elements, you can use the familiar square brackets along with the key to obtain its value. Following is a simple example –

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

print "dict['Name']: ", dict['Name']
print "dict['Age']: ", dict['Age']
```

Result –

```
dict['Name']: Zara
dict['Age']: 7
```

### Updating Dictionary

We can update a dictionary by adding a new entry or a key-value pair, modifying an existing entry, or deleting an existing entry as shown below in the simple example –

```
dict = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}

dict['Age'] = 8; # update existing entry
dict['School'] = "DPS School"; # Add new entry
print "dict['Age']: ", dict['Age']
print "dict['School']: ", dict['School']
```

Result –

```
dict['Age']: 8  
dict['School']: DPS School
```

## Delete Dictionary Elements

We can either remove individual dictionary elements or clear the entire contents of a dictionary. You can also delete entire dictionary in a single operation.

To explicitly remove an entire dictionary, just use the **del** statement. Following is a simple example –

```
dict = { 'Name': 'Zara', 'Age': 7, 'Class': 'First' }  
  
del dict[ 'Name' ]; # remove entry with key 'Name'  
  
dict.clear();      # remove all entries in dict  
  
del dict ;         # delete entire dictionary  
  
print "dict['Age']: ", dict['Age']  
print "dict['School']: ", dict['School']
```

Built-in Dictionary Functions & Methods –

Python includes the following dictionary functions –

SN	Function with Description
1	<u>cmp(dict1, dict2)</u>  Compares elements of both dict.
2	<u>len(dict)</u>  Gives the total length of the dictionary. This would be equal to the number of items in the dictionary.

3	<u>str(dict)</u>  Produces a printable string representation of a dictionary
4	<u>type(variable)</u>  Returns the type of the passed variable. If passed variable is dictionary, then it would return a dictionary type.

Python includes following dictionary methods –

SN	Methods with Description
1	<b>dict.clear():</b> Removes all elements of dictionary <i>dict</i>
2	<b>dict. Copy():</b> Returns a shallow copy of dictionary <i>dict</i>
3	<b>dict.fromkeys():</b> Create a new dictionary with keys from seq and values <i>set</i> to <i>value</i> .
4	<b>dict.get(key, default=None):</b> For <i>key</i> key, returns value or default if key not in dictionary
5	<b>dict.has_key(key):</b> Returns <i>true</i> if key in dictionary <i>dict</i> , <i>false</i> otherwise
6	<b>dict.items():</b> Returns a list of <i>dict</i> 's (key, value) tuple pairs
7	<b>dict.keys():</b> Returns list of dictionary <i>dict</i> 's keys

8	<b>dict.setdefault(key, default=None):</b> Similar to <code>get()</code> , but will set <code>dict[key]=default</code> if <i>key</i> is not already in dict
9	<b>dict.update(dict2):</b> Adds dictionary <i>dict2</i> 's key-values pairs to <i>dict</i>
10	<b>dict.values():</b> Returns list of dictionary <i>dict</i> 's values

A function is a block of organized, reusable code that is used to perform a single, related action. Functions provide better modularity for your application and a high degree of code reusing. Python gives you many built-in functions like `print()`, etc. but you can also create your own functions. These functions are called user-defined functions.

## DEFINING A FUNCTION

Simple rules to define a function in Python.

- Function blocks begin with the keyword `def` followed by the function name and parentheses `( )`.
- Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.
- The first statement of a function can be an optional statement - the documentation string of the function or *docstring*.
- The code block within every function starts with a colon `(:)` and is indented.
- The statement `return [expression]` exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as `return None`.

```
def functionname( parameters ):
    "function_docstring"
    function_suite
    return [expression]
```

## CALLING A FUNCTION

Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt. Following is the example to call `printme()` function –

```
# Function definition is here
def printme( str ):
    "This prints a passed string into this function"
    print str
    return;

# Now you can call printme function
printme("I'm first call to user defined function!")
printme("Again second call to the same function")
```

When the above code is executed, it produces the following result –

```
I'm first call to user defined function!
Again second call to the same function
```

### Function Arguments

You can call a function by using the following types of formal arguments:

- Required arguments
- Keyword arguments
- Default arguments
- Variable-length arguments



## SCOPE OF VARIABLES

All variables in a program may not be accessible at all locations in that program. This depends on where you have declared a variable.

The scope of a variable determines the portion of the program where you can access a particular identifier. There are two basic scopes of variables in Python –

Global variables

Local variables

Global vs. Local variables

Variables that are defined inside a function body have a local scope, and those defined outside have a global scope.

This means that local variables can be accessed only inside the function in which they are declared, whereas global variables can be accessed throughout the program body by all functions. When you call a function, the variables declared inside it are brought into scope. Following is a simple example –

```

total = 0; # This is global variable.

# Function definition is here
def sum( arg1, arg2 ):
    # Add both the parameters and return them."
    total = arg1 + arg2; # Here total is local variable.
    print "Inside the function local total : ", total
    return total;

sum( 10, 20 );
print "Outside the function global total : ", total

```

#### Result –

```

Inside the function local total : 30
Outside the function global total : 0

```

A module allows you to logically organize your Python code. Grouping related code into a module makes the code easier to understand and use. A module is a Python object with arbitrarily named attributes that you can bind and reference. Simply, a module is a file consisting of Python code. A module can define functions, classes and variables. A module can also include runnable code.

Example:

The Python code for a module named *aname* normally resides in a file named *aname.py*. Here's an example of a simple module, support.py

```

def print_func( par ):
    print "Hello : ", par
    return

```

## The *import* Statement

The *import* has the following syntax:

```
import module1[, module2[,... moduleN]
```

When the interpreter encounters an import statement, it imports the module if the module is present in the search path. A search path is a list of directories that the interpreter searches before importing a module. For example, to import the module `support.py`, you need to put the following command at the top of the script –

A module is loaded only once, regardless of the number of times it is imported. This prevents the module execution from happening over and over again if multiple imports occur.

## Packages in Python

A package is a hierarchical file directory structure that defines a single Python application environment that consists of modules and sub packages and sub-sub packages.

Consider a file *Pots.py* available in *Phone* directory. This file has following line of source code –

```
def Pots():  
    print "I'm Pots Phone"
```

Similar way, we have another two files having different functions with the same name as above –

- *Phone/Isdn.py* file having function `Isdn()`
- *Phone/G3.py* file having function `G3()`

Now, create one more file `__init__.py` in *Phone* directory –

- *Phone/\_\_init\_\_.py*

To make all of your functions available when you've imported Phone, to put explicit import statements in `__init__.py` as follows –

```
from Pots import Pots
from Isdn import Isdn
from G3 import G3
```

After you add these lines to `__init__.py`, you have all of these classes available when you import the Phone package.

```
# Now import your Phone Package.
import Phone
Phone.Pots()
Phone.Isdn()
Phone.G3()
```

RESULT:

```
I'm Pots Phone
I'm 3G Phone
I'm ISDN Phone
```

In the above example, we have taken example of a single functions in each file, but you can keep multiple functions in your files. You can also define different Python classes in those files and then you can create your packages out of those classes.

This chapter covers all the basic I/O functions available in Python.

#### Printing to the Screen

The simplest way to produce output is using the *print* statement where you can pass zero or more expressions separated by commas. This function converts the expressions you pass into a string and

writes the result to standard output as follows –

```
print "Python is really a great language,","isn't it?"
```

Result:

```
Python is really a great language, isn't it?
```

### Reading Keyboard Input

Python provides two built-in functions to read a line of text from standard input, which by default comes from the keyboard. These functions are –

- `raw_input`
- `input`

#### The `raw_input` Function

The `raw_input([prompt])` function reads one line from standard input and returns it as a string (removing the trailing newline).

```
str = raw_input("Enter your input: ");  
print "Received input is : ",str
```

This prompts you to enter any string and it would display same string on the screen. When I typed "Hello Python!", its output is like this –

```
Enter your input: Hello Python  
Received input is : Hello Python
```

#### The `input` Function

The `input([prompt])` function is equivalent to `raw_input`, except that it assumes the input is a valid Python expression and returns the evaluated result to you.

```
str = input("Enter your input: ");  
print "Received input is : ",str
```

This would produce the following result against the entered input –

```
Enter your input: [x*5 for x in range(2,10,2)]
```

```
Recieved input is : [10, 20, 30, 40]
```

## Opening and Closing Files

Until now, you have been reading and writing to the standard input and output. Now, we will see how to use actual data files.

Python provides basic functions and methods necessary to manipulate files by default. You can do most of the file manipulation using a **file** object.

### The *open* Function

Before you can read or write a file, you have to open it using Python's built-in *open()* function. This function creates a **file** object, which would be utilized to call other support methods associated with it.

### Syntax

```
file object = open(file_name [, access_mode][, buffering])
```

Here are parameter details:

- **file\_name:** The `file_name` argument is a string value that contains the name of the file that you want to access.
- **access\_mode:** The `access_mode` determines the mode in which the file has to be opened, i.e., read, write, append, etc. A complete list of possible values is given below in the table. This is optional parameter and the default file access mode is read (r).
- **buffering:** If the buffering value is set to 0, no buffering takes place. If the buffering value is 1, line buffering is performed while accessing a file. If you specify the buffering value as an integer greater than 1, then buffering action is performed with the indicated buffer size. If negative, the buffer size is the system default(default behavior).

Here is a list of the different modes of opening a file –

Modes	Description
r	Opens a file for reading only. The file pointer is placed at the beginning of the file. This is the default mode.
rb	Opens a file for reading only in binary format. The file pointer is placed at the beginning of the file. This is the default mode.
r+	Opens a file for both reading and writing. The file pointer placed at the beginning of the file.
rb+	Opens a file for both reading and writing in binary format. The file pointer placed at the beginning of the file.
w	Opens a file for writing only. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
wb	Opens a file for writing only in binary format. Overwrites the file if the file exists. If the file does not exist, creates a new file for writing.
w+	Opens a file for both writing and reading. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.
wb+	Opens a file for both writing and reading in binary format. Overwrites the existing file if the file exists. If the file does not exist, creates a new file for reading and writing.

a	Opens a file for appending. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
ab	Opens a file for appending in binary format. The file pointer is at the end of the file if the file exists. That is, the file is in the append mode. If the file does not exist, it creates a new file for writing.
a+	Opens a file for both appending and reading. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.
ab+	Opens a file for both appending and reading in binary format. The file pointer is at the end of the file if the file exists. The file opens in the append mode. If the file does not exist, it creates a new file for reading and writing.

#### The *file* Object Attributes

Once a file is opened and you have one *file* object, you can get various information related to that file.

Here is a list of all attributes related to file object:

Attribute	Description
file.closed	Returns true if file is closed, false otherwise.
file.mode	Returns access mode with which file was opened.
file.name	Returns name of the file.



file.softspace	Returns false if space explicitly required with print, true otherwise.
----------------	--

### Example

```
# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
print "Closed or not : ", fo.closed
print "Opening mode : ", fo.mode
print "Softspace flag : ", fo.softspace
```

This produces the following result –

```
Name of the file: foo.txt
Closed or not : False
Opening mode : wb
Softspace flag : 0
```

### The `close()` Method

The `close()` method of a *file* object flushes any unwritten information and closes the file object, after which no more writing can be done. Python automatically closes a file when the reference object of a file is reassigned to another file. It is a good practice to use the `close()` method to close a file

### Syntax

```
fileObject.close();
```

### Example

```
# Open a file
fo = open("foo.txt", "wb")
print "Name of the file: ", fo.name
# Close opened file
fo.close()
```

Result –

Name of the file: foo.txt

## Reading and Writing Files

The *file* object provides a set of access methods to make our lives easier. We would see how to use *read()* and *write()* methods to read and write files.

### The *write()* Method

The *write()* method writes any string to an open file. It is important to note that Python strings can have binary data and not just text. The *write()* method does not add a newline character ('\n') to the end of the string **Syntax**

```
fileObject.write(string);
```

Here, passed parameter is the content to be written into the opened file. **Example**

```
# Open a file
fo = open("foo.txt", "wb")
fo.write("Python is a great language.\nYeah its great!!\n");

# Close opened file
fo.close()
```

The above method would create *foo.txt* file and would write given content in that file and finally it would close that file. If you would open this file, it would have following content.

```
Python is a great language.
Yeah its great!!
```

### The *read()* Method

The *read()* method reads a string from an open file. It is important to note that Python strings can have binary data. apart from text data.

### Syntax

```
fileObject.read([count]);
```

Here, passed parameter is the number of bytes to be read from the opened file. This method starts reading from the beginning of the file and if *count* is missing, then it tries to read as much as possible, maybe until the end of file.

#### Example

Let's take a file *foo.txt*, which we created above.

```
# Open a file
fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str
# Close opened file
fo.close()
```

This produces the following result –

```
Read String is : Python is
```

#### File Positions

The `tell()` method tells you the current position within the file; in other words, the next read or write will occur at that many bytes from the beginning of the file.

32

The `seek(offset[, from])` method changes the current file position. The *offset* argument indicates the number of bytes to be moved. The *from* argument specifies the reference position from where the bytes are to be moved.

If *from* is set to 0, it means use the beginning of the file as the reference position and 1 means use the current position as the reference position and if it is set to 2 then the end of the file would be taken as the reference position.

#### Example

Let us take a file *foo.txt*, which we created above.

```
# Open a file
```

```

fo = open("foo.txt", "r+")
str = fo.read(10);
print "Read String is : ", str

# Check current position
position = fo.tell();
print "Current file position : ", position

# Reposition pointer at the beginning once again
position = fo.seek(0, 0);
str = fo.read(10);
print "Again read String is : ", str

# Close opened file
fo.close()

```

This produces the following result –

```

Read String is : Python is
Current file position : 10
Again read String is : Python is

```

## Renaming and Deleting Files

Python **os** module provides methods that help you perform file-processing operations, such as renaming and deleting files.

To use this module you need to import it first and then you can call any related functions.

### The `rename()` Method

The *rename()* method takes two arguments, the current filename and the new filename.

### Syntax

```
os.rename(current_file_name, new_file_name)
```

## Example

Following is the example to rename an existing file *test1.txt*:

```
import os

# Rename a file from test1.txt to test2.txt
os.rename( "test1.txt", "test2.txt" )
```

## The *remove()* Method

You can use the *remove()* method to delete files by supplying the name of the file to be deleted as the argument.

## Syntax

```
os.remove(file_name)
```

## Example

Following is the example to delete an existing file *test2.txt* –

```
#!/usr/bin/python
import os

# Delete file test2.txt
os.remove("test2.txt")
```

## Directories in Python

All files are contained within various directories, and Python has no problem handling these too. The **os** module has several methods that help you create, remove, and change directories.

## The *mkdir()* Method

You can use the *mkdir()* method of the **os** module to create directories in the current directory. You need to supply an argument to this method which contains the name of the directory to be created.

## Syntax

```
os.mkdir("newdir")
```

## Example

Following is the example to create a directory *test* in the current directory –

```
#!/usr/bin/python
import os

# Create a directory "test"
os.mkdir("test")
```

## The *chdir()* Method

You can use the *chdir()* method to change the current directory. The *chdir()* method takes an argument, which is the name of the directory that you want to make the current directory.

## Syntax

```
os.chdir("newdir")
```

## Example

Following is the example to go into `"/home/newdir"` directory –

```
#!/usr/bin/python
import os

# Changing a directory to "/home/newdir"
os.chdir("/home/newdir")
```

## The *getcwd()* Method

The *getcwd()* method displays the current working directory.

## Syntax

```
os.getcwd()
```

## Example

Following is the example to give current directory –

```
import os
```

```
# This would give location of the current directory
```

```
os.getcwd()
```

### The rmdir() Method

The `rmdir()` method deletes the directory, which is passed as an argument in the method.

Before removing a directory, all the contents in it should be removed.

Syntax:

```
os.rmdir('dirname')
```

### Example

Following is the example to remove `"/tmp/test"` directory. It is required to give fully qualified name of the directory, otherwise it would search for that directory in the current directory.

```
import os
# This would remove "/tmp/test" directory.
os.rmdir( "/tmp/test" )
```

### File & Directory Related Methods

There are three important sources, which provide a wide range of utility methods to handle and manipulate files & directories on Windows and Unix operating systems. They are as follows –

- File Object Methods: The *file* object provides functions to manipulate files.
- OS Object Methods: This provides methods to process files as well as directories.

Python provides two very important features to handle any unexpected error in your Python programs and to add debugging capabilities in them –

- **Exception Handling**: This would be covered in this tutorial. Here is a list standard Exceptions available in Python: Standard Exceptions.
- **Assertions**: This would be covered in Assertions in Python

## List of Standard Exceptions –

EXCEPTION NAME	DESCRIPTION
Exception	Base class for all exceptions
StopIteration	Raised when the next() method of an iterator does not point to any object.
SystemExit	Raised by the sys.exit() function.
StandardError	Base class for all built-in exceptions except StopIteration and SystemExit.
ArithmeticError	Base class for all errors that occur for numeric calculation.
OverflowError	Raised when a calculation exceeds maximum limit for a numeric type.
FloatingPointError	Raised when a floating point calculation fails.
ZeroDivisionError	Raised when division or modulo by zero takes place for all numeric types.
AssertionError	Raised in case of failure of the Assert statement.
AttributeError	Raised in case of failure of attribute reference or assignment.



EOFError	Raised when there is no input from either the <code>raw_input()</code> or <code>input()</code> function and the end of file is reached.
ImportError	Raised when an import statement fails.
KeyboardInterrupt	Raised when the user interrupts program execution, usually by pressing Ctrl+c.
LookupError	Base class for all lookup errors.
IndexError	Raised when an index is not found in a sequence.
KeyError	Raised when the specified key is not found in the dictionary.
NameError	Raised when an identifier is not found in the local or global namespace.
UnboundLocalError	Raised when trying to access a local variable in a function or method but no value has been assigned to it.
EnvironmentError	Base class for all exceptions that occur outside the Python environment.
IOError	Raised when an input/ output operation fails, such as the <code>print</code> statement or the <code>open()</code> function when trying to open a file that does not exist.
IOError	Raised for operating system-related errors.
SyntaxError	Raised when there is an error in Python syntax.
IndentationError	Raised when indentation is not specified properly.

SystemError	Raised when the interpreter finds an internal problem, but when this error is encountered the Python interpreter does not exit.
SystemExit	Raised when Python interpreter is quit by using the sys.exit() function. If not handled in the code, causes the interpreter to exit.
TypeError	Raised when an operation or function is attempted that is invalid for the specified data type.
ValueError	Raised when the built-in function for a data type has the valid type of arguments, but the arguments have invalid values specified.
RuntimeError	Raised when a generated error does not fall into any category.
NotImplementedError	Raised when an abstract method that needs to be implemented in an inherited class is not actually implemented.

### What is Exception?

An exception is an event, which occurs during the execution of a program that disrupts the normal flow of the program's instructions. In general, when a Python script encounters a situation that it cannot cope with, it raises an exception. An exception is a Python object that represents an error.

When a Python script raises an exception, it must either handle the exception immediately otherwise it terminates and quits.

### Handling an exception

If you have some suspicious code that may raise an exception, you can defend your program by placing the suspicious code in a **try:** block. After the **try:** block, include an **except:** statement, followed by a block of code which handles the problem as elegantly as possible.

The Python standard for database interfaces is the Python DB-API. Most Python database interfaces adhere to this standard.

You can choose the right database for your application. Python Database API supports a wide range of database servers such as –

- GadFly
- mSQL
- MySQL
- PostgreSQL
- Microsoft SQL Server 2000
- Informix
- Interbase
- Oracle
- Sybase

The DB API provides a minimal standard for working with databases using Python structures and syntax wherever possible. This API includes the following:

- Importing the API module.
- Acquiring a connection with the database.
- Issuing SQL statements and stored procedures.
- Closing the connection

## 9.2 SOURCE CODE

### MANAGE.PY

```
#!/usr/bin/env python

import os

import sys

if __name__ == '__main__':

    os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'StudentCredential.settings')

    try:

        from django.core.management import execute_from_command_line

    except ImportError as exc:

        raise ImportError(

            "Couldn't import Django. Are you sure it's installed and "

            "available on your PYTHONPATH environment variable? Did you "

            "forget to activate a virtual environment?"

        ) from exc

    execute_from_command_line(sys.argv)
```

### SETTINGS.PY

```
"""
```

Django settings for StudentCredential project.

Generated by 'django-admin startproject' using Django 2.1.7.

For more information on this file, see

<https://docs.djangoproject.com/en/2.1/topics/settings/>

For the full list of settings and their values, see

<https://docs.djangoproject.com/en/2.1/ref/settings/>

```
"""
```

```
import os
```

```
# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
```

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))
```

```
# Quick-start development settings - unsuitable for production
```

```
# See https://docs.djangoproject.com/en/2.1/howto/deployment/checklist/
```

```
# SECURITY WARNING: keep the secret key used in production secret!
```

```
SECRET_KEY = 'q0xb&&4w*p+0d3jj26yt142$5w-*psl@3y6te!b^u-jheft1='
```

```
# SECURITY WARNING: don't run with debug turned on in production!
```

```
DEBUG = True
```

```
ALLOWED_HOSTS = []
```

```
# Application definition
```

```
INSTALLED_APPS = [
```

```
    'django.contrib.admin',
```

```
'django.contrib.auth',
'django.contrib.contenttypes',
'django.contrib.sessions',
'django.contrib.messages',
'django.contrib.staticfiles',
'StudentCredentialApp'
]
```

```
MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]
```

```
ROOT_URLCONF = 'StudentCredential.urls'
```

```
TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [
            os.path.join('E:/takeoff/oct23/CredentialVerification/StudentCredentialApp',
'templates'),
```

```

    ],
    'APP_DIRS': True,
    'OPTIONS': {
        'context_processors': [
            'django.template.context_processors.debug',
            'django.template.context_processors.request',
            'django.contrib.auth.context_processors.auth',
            'django.contrib.messages.context_processors.messages',
        ],
    },
},
]

```

```
WSGI_APPLICATION = 'StudentCredential.wsgi.application'
```

```
# Database
```

```
# https://docs.djangoproject.com/en/2.1/ref/settings/#databases
```

```

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}

```

# Password validation

# <https://docs.djangoproject.com/en/2.1/ref/settings/#auth-password-validators>

```
AUTH_PASSWORD_VALIDATORS = [  
    {  
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',  
    },  
    {  
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',  
    },  
]
```

# Internationalization

# <https://docs.djangoproject.com/en/2.1/topics/i18n/>

LANGUAGE\_CODE = 'en-us'

TIME\_ZONE = 'UTC'



```
USE_I18N = True
```

```
USE_L10N = True
```

```
USE_TZ = True
```

```
# Static files (CSS, JavaScript, Images)
```

```
# https://docs.djangoproject.com/en/2.1/howto/static-files/
```

```
STATIC_URL = '/static/'
```

## **VIEWS.PY**

```
from django.shortcuts import render
```

```
from django.template import RequestContext
```

```
from django.contrib import messages
```

```
from django.http import HttpResponse
```

```
from django.conf import settings
```

```
import os
```

```
import json
```

```
from web3 import Web3, HTTPProvider
```

```
import hashlib
```

```
import os
```

```
from django.core.files.storage import FileSystemStorage
```

```
import pickle
```

global details, username, school\_name, company\_name

details=""

global contract

```
def readDetails(contract_type):
```

```
    global details
```

```
    details = ""
```

```
    print(contract_type+"=====")
```

```
    blockchain_address = 'http://127.0.0.1:9545' #Blockchain connection IP
```

```
    web3 = Web3(HTTPProvider(blockchain_address))
```

```
    web3.eth.defaultAccount = web3.eth.accounts[0]
```

```
    compiled_contract_path = 'Student.json' #student contract code
```

```
    deployed_contract_address = '0xd374Cb05bd6187D6cF905D7bBD85f2b704fBDD29'
```

```
#hash address to access student contract
```

```
    with open(compiled_contract_path) as file:
```

```
        contract_json = json.load(file) # load contract info as JSON
```

```
        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions
```

```
    file.close()
```

```
    contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi) #now  
calling contract to access data
```

```
    if contract_type == 'schoolcompany':
```

```
        details = contract.functions.getUniversityCompany().call()
```

```
    if contract_type == 'enrollstudent':
```

```
        details = contract.functions.getStudent().call()
```

```
    if contract_type == 'credential':
```

```
        details = contract.functions.getCredential().call()
```

```

if contract_type == 'accessrequest':

    details = contract.functions.getAccess().call()

print(details)


def saveDataBlockchain(currentData, contract_type):

    global details

    global contract

    details = ""

    blockchain_address = 'http://127.0.0.1:9545'

    web3 = Web3(HTTPProvider(blockchain_address))

    web3.eth.defaultAccount = web3.eth.accounts[0]

    compiled_contract_path = 'Student.json' #student contract file

    deployed_contract_address    =    '0xd374Cb05bd6187D6cF905D7bBD85f2b704fBDD29'
#contract address

    with open(compiled_contract_path) as file:

        contract_json = json.load(file) # load contract info as JSON

        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions

    file.close()

    contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi)

    readDetails(contract_type)

    if contract_type == 'schoolcompany':

        details+=currentData

        msg = contract.functions.addUniversityCompany(details).transact()

        tx_receipt = web3.eth.waitForTransactionReceipt(msg)

    if contract_type == 'enrollstudent':

        details+=currentData

```

```

    msg = contract.functions.enrollStudent(details).transact()

    tx_receipt = web3.eth.waitForTransactionReceipt(msg)

if contract_type == 'credential':

    details+=currentData

    msg = contract.functions.setCredentialData(details).transact()

    tx_receipt = web3.eth.waitForTransactionReceipt(msg)

if contract_type == 'accessrequest':

    details+=currentData

    msg = contract.functions.setAccessRequest(details).transact()

    tx_receipt = web3.eth.waitForTransactionReceipt(msg)


def saveDataBlockchain1(currentData, contract_type):

    global details

    global contract

    details = ""

    blockchain_address = 'http://127.0.0.1:9545'

    web3 = Web3(HTTPProvider(blockchain_address))

    web3.eth.defaultAccount = web3.eth.accounts[0]

    compiled_contract_path = 'Student.json' #student contract file

    deployed_contract_address    =    '0xd374Cb05bd6187D6cF905D7bBD85f2b704fBDD29'
#contract address

    with open(compiled_contract_path) as file:

        contract_json = json.load(file) # load contract info as JSON

        contract_abi = contract_json['abi'] # fetch contract's abi - necessary to call its functions

    file.close()

    contract = web3.eth.contract(address=deployed_contract_address, abi=contract_abi)

```

```

if contract_type == 'credential':

    msg = contract.functions.setCredentialData(currentData).transact()

    tx_receipt = web3.eth.waitForTransactionReceipt(msg)

```

```

def StudentLoginAction(request):

    if request.method == 'POST':

        global details, username

        username = request.POST.get('username', False)

        readDetails('enrollstudent')

        arr = details.split("\n")

        status = "none"

        for i in range(len(arr)-1):

            array = arr[i].split("#")

            if array[1] == username:

                status = "success"

                break

        if status == "success":

            context= {'data': 'Welcome '+username}

            return render(request, "StudentScreen.html", context)

        else:

            context= {'data': 'Invalid username'}

            return render(request, 'StudentLogin.html', context)

def CompanyLoginAction(request):

    if request.method == 'POST':

```

```

global details, username, school_name, company_name

username = request.POST.get('username', False)

password = request.POST.get('password', False)

readDetails('schoolcompany')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

    array = arr[i].split("#")

    if array[0] == 'company' and array[4] == username and array[5] == password:

        status = "success"

        company_name = array[1]

        break

if status == "success":

    context= {'data':'Welcome '+username}

    return render(request, "CompanyScreen.html", context)

else:

    context= {'data':'Invalid username'}

    return render(request, 'CompanyLogin.html', context)

def UniversityLoginAction(request):

    if request.method == 'POST':

        global details, username, school_name, company_name

        username = request.POST.get('username', False)

        password = request.POST.get('password', False)

        readDetails('schoolcompany')

        arr = details.split("\n")

```

```

status = "none"

for i in range(len(arr)-1):

    array = arr[i].split("#")

    print(array)

    if array[0] == 'university' and array[4] == username and array[5] == password:

        status = "success"

        school_name = array[1]

        break

if status == "success":

    context= {'data':'Welcome '+username}

    return render(request, "UniversityScreen.html", context)

else:

    context= {'data':'Invalid username'}

    return render(request, 'UniversityLogin.html', context)


def UploadCertificate(request):

    if request.method == 'GET':

        return render(request, 'UploadCertificate.html', {})


def UpdateCertificate(request):

    if request.method == 'GET':

        return render(request, 'UpdateCertificate.html', {})


def index(request):

    if request.method == 'GET':

        return render(request, 'index.html', {})

```

```
def EnrollStudent(request):  
    if request.method == 'GET':  
        return render(request, 'EnrollStudent.html', {})  
  
def UniversityLogin(request):  
    if request.method == 'GET':  
        return render(request, 'UniversityLogin.html', {})  
  
def CompanyLogin(request):  
    if request.method == 'GET':  
        return render(request, 'CompanyLogin.html', {})  
  
def StudentLogin(request):  
    if request.method == 'GET':  
        return render(request, 'StudentLogin.html', {})  
  
def UniversitySignup(request):  
    if request.method == 'GET':  
        return render(request, 'UniversitySignup.html', {})  
  
def CompanySignup(request):  
    if request.method == 'GET':  
        return render(request, 'CompanySignup.html', {})  
  
def UniversitySignupAction(request):
```



```

if request.method == 'POST':

    global details

    school = request.POST.get('t1', False)

    address = request.POST.get('t2', False)

    contact = request.POST.get('t3', False)

    user = request.POST.get('t4', False)

    password = request.POST.get('t5', False)

    readDetails('schoolcompany')

    arr = details.split("\n")

    status = "none"

    for i in range(len(arr)-1):

        array = arr[i].split("#")

        if array[4] == user:

            status = user+" Username already exists"

            break

    if status == "none":

        data = "university#" + school + "#" + address + "#" + contact + "#" + user + "#" + password + "\n"

        saveDataBlockChain(data, "schoolcompany")

        context = {"data": "University signup task completed"}

        return render(request, 'UniversitySignup.html', context)

    else:

        context = {"data": status}

        return render(request, 'UniversitySignup.html', context)

def CompanySignupAction(request):

    if request.method == 'POST':

```

```

global details

company = request.POST.get('t1', False)

address = request.POST.get('t2', False)

contact = request.POST.get('t3', False)

user = request.POST.get('t4', False)

password = request.POST.get('t5', False)

readDetails('schoolcompany')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

    array = arr[i].split("#")

    if array[4] == user:

        status = user+" Username already exists"

        break

if status == "none":

    data = ""

    "company#" + company + "#" + address + "#" + contact + "#" + user + "#" + password + "\n"

    saveDataBlockChain(data, "schoolcompany")

    context = {"data": "Company signup task completed"}

    return render(request, 'CompanySignup.html', context)

else:

    context = {"data": status}

    return render(request, 'CompanySignup.html', context)

def ViewStudents(request):

    if request.method == 'GET':

```

```

global details, username, school_name

output = '<table border=1 align=center width=100%>'

font = '<font size="" color="white">'

arr = ['University Name','Student ID','Student Name','University Details','Course
Name','Joining Date']

output += "<tr>"

for i in range(len(arr)):

    output += "<th>"+font+arr[i]+"</th>"

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

    print(arr[i])

    array = arr[i].split("#")

    output += "<tr><td>"+font+array[0]+"</td>"

    output += "<td>"+font+array[1]+"</td>"

    output += "<td>"+font+array[2]+"</td>"

    output += "<td>"+font+array[3]+"</td>"

    output += "<td>"+font+array[4]+"</td>"

    output += "<td>"+font+array[5]+"</td>"

context= {'data':output}

return render(request, 'ViewStudents.html', context)

def EnrollStudentAction(request):

    if request.method == 'POST':

        global details, username, school_name

```

```

sid = request.POST.get('t1', False)

sname = request.POST.get('t2', False)

school_details = request.POST.get('t3', False)

course = request.POST.get('t4', False)

joining_date = request.POST.get('t5', False)

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

    array = arr[i].split("#")

    if array[0] == sid:

        status = sid+" Student ID already exists"

        break

if status == "none":

    details = ""

    data = school_name+"#"+sid+"#"+sname+"#"+school_details+"#"+course+"#"+joining_date+"\n"

    saveDataBlockChain(data,"enrollstudent")

    context = {"data":"New Student Enrollment Task Completed"}

    return render(request, 'EnrollStudent.html', context)

else:

    context = {"data":status}

    return render(request, 'EnrollStudent.html', context)

def UploadCertificateAction(request):

```

```

if request.method == 'POST':

    global details, username, school_name

    sid = request.POST.get('t1', False)

    certificate = request.POST.get('t2', False)

    issue_date = request.POST.get('t3', False)

    filename = request.FILES['t4'].name

    myfile = request.FILES['t4'].read()

    ""

    myfile = pickle.dumps(myfile)

    hashcode = api.add_pyobj(myfile)

    #readDetails('credential')

    ""

    readDetails('credential')

    credential = details.split("\n")

    certificate_id = len(credential)

    #if certificate_id > 1:

    #certificate_id = certificate_id + 1

    result = hashlib.sha256(myfile)

    hashcode = result.hexdigest()

    with open('StudentCredentialApp/static/certificates/'+str(certificate_id)+".png", "wb") as
file:

        file.write(myfile)

    file.close()

    data = school_name+"#" +str(certificate_id)+"#" +sid+"#" +certificate+"#" +issue_date+"#" +filename
+"#" +hashcode+"\n"

    saveDataBlockChain(data,"credential")

```

```

context = {"data": "Certificate saved with hashcode saving in Blockchain: "+hashcode}

return render(request, 'UploadCertificate.html', context)

```

```

def UpdateCertificateAction(request):

```

```

    if request.method == 'POST':

```

```

        global details, username, school_name

```

```

        cid = request.POST.get('certificate', False)

```

```

        sid = request.POST.get('t1', False)

```

```

        certificate = request.POST.get('t2', False)

```

```

        issue_date = request.POST.get('t3', False)

```

```

        filename = request.FILES['t4'].name

```

```

        myfile = request.FILES['t4'].read()

```

```

        result = hashlib.sha256(myfile)

```

```

        hashcode = result.hexdigest()

```

```

        if os.path.exists('StudentCredentialApp/static/certificates/'+cid+".png"):

```

```

            os.remove('StudentCredentialApp/static/certificates/'+cid+".png")

```

```

        with open('StudentCredentialApp/static/certificates/'+cid+".png", "wb") as file:

```

```

            file.write(myfile)

```

```

        file.close()

```

```

        """

```

```

        myfile = pickle.dumps(myfile)

```

```

        hashcode = api.add_pyobj(myfile)

```

```

        """

```

```

        readDetails('credential')

```

```

        data = ""

```

```

        credential = details.split("\n")

```

```

flag = False

output = "Given Certificate or Student Id doesn't match"

for i in range(len(credential)-1):

    array = credential[i].split("#")

    if array[1] != cid:

        data += credential[i]+"\\n"

    if array[1] == cid:

        flag = True

if flag == True:

    data += school_name+"#"+cid+"#"+sid+"#"+certificate+"#"+issue_date+"#"+filename+"#"+hashcode+"\\n"

    saveDataBlockchain1(data,"credential")

    output = "Modified certificate saved with hashcode saving in Blockchain: "+hashcode

    context = {"data":output}

    return render(request, 'UpdateCertificate.html', context)

```

```

def SendAccessRequest(request):

    if request.method == 'GET':

        global details, username, school_name

        output = '<table border=1 align=center width=100%>'

        font = '<font size="" color="white">'

        arr = ['School Name','Student ID','Student Name','School Details','Course Name','Joining Date','Send Access Request']

        output += "<tr>"

        for i in range(len(arr)):

```

```

        output += "<th>"+font+arr[i]+"</th>"

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

    array = arr[i].split("#")

    output += "<tr><td>"+font+array[0]+"</td>"

    output += "<td>"+font+array[1]+"</td>"

    output += "<td>"+font+array[2]+"</td>"

    output += "<td>"+font+array[3]+"</td>"

    output += "<td>"+font+array[4]+"</td>"

    output += "<td>"+font+array[5]+"</td>"

    output+='<td><a          href=\'SendRequest?t1=\'+array[1]+'\'><font          size=3
color=white>Click Here</font></a></td></tr>'

    context= {'data':output}

    return render(request, 'SendAccessRequest.html', context)

def SendRequest(request):

    if request.method == 'GET':

        global details, username, company_name

        sid = request.GET.get('t1', False)

        data = sid+"#"+company_name+"#Pending\n"

        saveDataBlockChain(data,"accessrequest")

        context = {"data":"Access Request sent to student: "+sid}

        return render(request, 'SendAccessRequest.html', context)

```



```

def checkAccess(credential_arr, sid):

    status = False

    for i in range(len(credential_arr)-1):

        arr = credential_arr[i].split("#")

        if arr[0] == sid and arr[1] == company_name and arr[2] == "Accepted":

            status = True

            break

    return status


def AccessCertificate(request):

    if request.method == 'GET':

        global details, username, company_name

        output = '<table border=1 align=center width=100%>'

        font = '<font size="" color="white">'

        arr = ['University Name','Certificate ID', 'Student ID','Certificate Details','Issue Date','File
Name','Hashcode','Certificate']

        output += "<tr>"

        for i in range(len(arr)):

            output += "<th>"+font+arr[i]+"</th>"

        readDetails('accessrequest')

        credential_arr = details.split("\n")

        readDetails('credential')

        access_certificate = details.split("\n")

        for i in range(len(access_certificate)-1):

            array = access_certificate[i].split("#")

            status = checkAccess(credential_arr, array[2])

```

```

if status == True:

    output += "<tr><td>" + font + array[0] + "</td>"

    output += "<td>" + font + array[1] + "</td>"

    output += "<td>" + font + array[2] + "</td>"

    output += "<td>" + font + array[3] + "</td>"

    output += "<td>" + font + array[4] + "</td>"

    output += "<td>" + font + array[5] + "</td>"

    output += "<td>" + font + array[6] + "</td>"

    ""

    content = api.get_pyobj(array[5])

    content = pickle.loads(content)

    ""

    output += '<td><img      src=static/certificates/'+array[1]+''.png      width=400
height=400></img></td>'

    context= {'data':output}

    return render(request, 'AccessCertificate.html', context)


def ViewDetails(request):

    if request.method == 'GET':

        global details, username

        output = '<table border=1 align=center width=100%>'

        font = '<font size="" color="white">'

        arr = ['University  Name','Student  ID','Student  Name','School  Details','Course
Name','Joining Date']

        output += "<tr>"

        for i in range(len(arr)):

            output += "<th>" + font + arr[i] + "</th>"

```

```

readDetails('enrollstudent')

arr = details.split("\n")

status = "none"

for i in range(len(arr)-1):

    array = arr[i].split("#")

    if array[1] == username:

        output += "<tr><td>" + font + array[0] + "</td>"

        output += "<td>" + font + array[1] + "</td>"

        output += "<td>" + font + array[2] + "</td>"

        output += "<td>" + font + array[3] + "</td>"

        output += "<td>" + font + array[4] + "</td>"

        output += "<td>" + font + array[5] + "</td>"

context= {'data':output}

return render(request, 'ViewDetails.html', context)

```

```

def AccessOwnCertificate(request):

    if request.method == 'GET':

        global details, username

        output = '<table border=1 align=center width=100%>'

        font = '<font size="" color="white">'

        arr = ['University Name','Certificate ID', 'Student ID','Certificate Details','Issue Date','File
Name','Hashcode','Certificate']

        output += "<tr>"

        for i in range(len(arr)):

            output += "<th>" + font + arr[i] + "</th>"

```

```

readDetails('credential')

access_certificate = details.split("\n")

for i in range(len(access_certificate)-1):

    array = access_certificate[i].split("#")

    if array[2] == username:

        output += "<tr><td>" + font + array[0] + "</td>"

        output += "<td>" + font + array[1] + "</td>"

        output += "<td>" + font + array[2] + "</td>"

        output += "<td>" + font + array[3] + "</td>"

        output += "<td>" + font + array[4] + "</td>"

        output += "<td>" + font + array[5] + "</td>"

        output += "<td>" + font + array[6] + "</td>"

        ""

        content = api.get_pyobj(array[5])

        content = pickle.loads(content)

        if os.path.exists('StudentCredentialApp/static/certificates/'+array[4]):

            os.remove('StudentCredentialApp/static/certificates/'+array[4])

        with open('StudentCredentialApp/static/certificates/'+array[4], "wb") as file:

            file.write(content)

        file.close()

        ""

        output += '<td><img      src=static/certificates/'+array[1]+''.png      width=400
height=400></img></td>'

        context= {'data':output}

    return render(request, 'AccessOwnCertificate.html', context)

```

```

def GrantAccess(request):

    if request.method == 'GET':

        global details, username

        output = '<table border=1 align=center width=100%>'

        font = '<font size="" color="white">'

        arr = ['Student ID','Company Name','Access Status','Grant Access']

        output += "<tr>"

        for i in range(len(arr)):

            output += "<th>" + font + arr[i] + "</th>"

        readDetails('accessrequest')

        arr = details.split("\n")

        status = "none"

        for i in range(len(arr)-1):

            array = arr[i].split("#")

            if array[0] == username:

                output += "<tr><td>" + font + array[0] + "</td>"

                output += "<td>" + font + array[1] + "</td>"

                output += "<td>" + font + array[2] + "</td>"

                output += '<td><a'
href='\GrantAccessAction?t1='+array[0]+'&t2='+array[1]+'\'><font size=3 color=white>Click
Here</font></a></td></tr>'

                context= {'data':output}

            return render(request, 'GrantAccess.html', context)

def GrantAccessAction(request):

```

```

if request.method == 'GET':

    global details, username, company_name

    sid = request.GET.get('t1', False)

    company = request.GET.get('t2', False)

    data = ""

    readDetails('accessrequest')

    arr = details.split("\n")

    status = "none"

    for i in range(len(arr)-1):

        array = arr[i].split("#")

        if array[0] != sid and array[1] != company:

            data+=array[0]+"#"+array[1]+"#"+array[2]+"\\n"

    data+=sid+"#"+company+"#Accepted\\n"

    saveDataBlockChain(data,"accessrequest")

    context = {"data":"Access Request granted to company: "+company}

    return render(request, 'StudentScreen.html', context)

```

## **CHAPTER-10**

### **RESULTS/DISCUSSION**

#### **10.1 SYSTEM TESTING**

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.

#### **TYPES OF TESTS**

##### **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.

##### **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 satisfactory, 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.

## **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.

## **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.

## **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.

## **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.

### **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

### 10.1.1 TEST CASES:

#### Test case1:

Test case for Login form:

FUNCTION:	LOGIN
EXPECTED RESULTS:	Should Validate the user and check his existence in database
ACTUAL RESULTS:	Validate the user and checking the user against the database
LOW PRIORITY	No
HIGH PRIORITY	Yes

#### Test case2:

Test case for User Registration form:

FUNCTION	SUPPLIER
EXPECTED RESULTS:	Should check if all the fields are filled by the user and saving the user to database.
ACTUAL RESULTS:	Checking whether all the fields are filled by user or not through validations and saving user.
LOW PRIORITY	No
HIGH PRIORITY	Yes

### Test case3:

Test case for Change Password:

When the old password does not match with the new password , then this results in displaying an error message as “ OLD PASSWORD DOES NOT MATCH WITH THE NEW PASSWORD”.

### Test case 4:

Test case for Forget Password:

When a user forgets his password he is asked to enter Login name, ZIP code, Mobile number. If these are matched with the already stored ones then user will get his Original password.

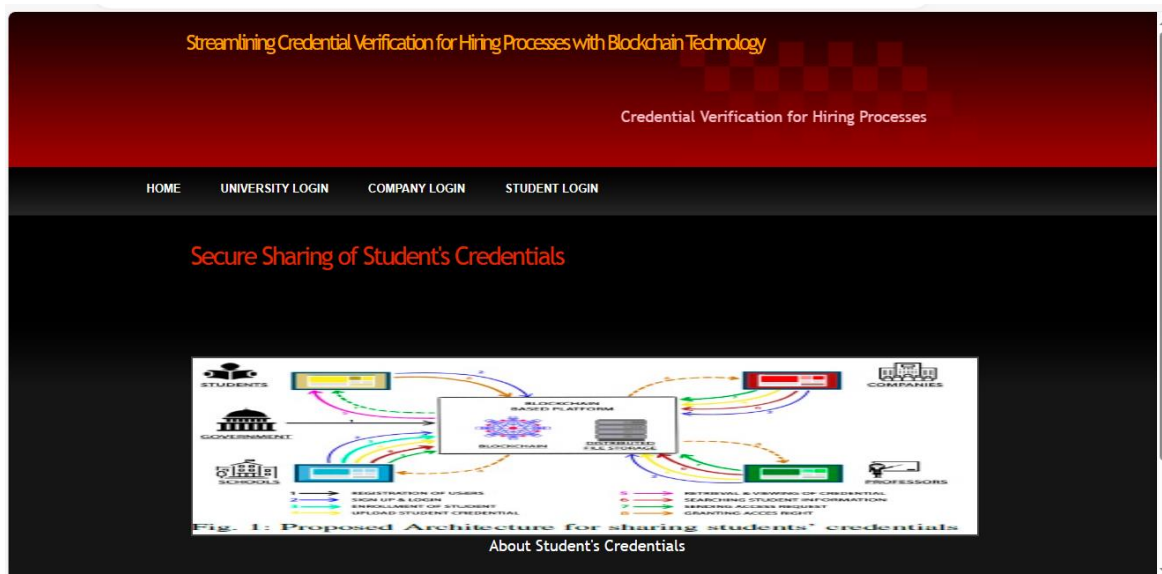
Module	Functionality	Test Case	Expected Results	Actual Results	Result	Priori
--------	---------------	-----------	------------------	----------------	--------	--------

						ty
User	Login Usecase	1. Navigate To Www.Sample.Co m  2. Click On Submit Button Without Entering Username and Password	A Validation Should Be As Below “Please Enter Valid Username & Password”	A Validation Has Been Populated As Expected	Pass	High
		1. aNavigate To Www.Sample.Co m  2. Click On Submit Button With Out Filling Password And With Valid Username	A Validation Should Be As Below “Please Enter Valid Password Or Password Field Can Not Be Empty “	A Validation Is Shown As Expected	Pass	High
		1. NNavigate To Www.Sample.Co m  2. Enter Both Username And Password Wrong And Hit Enter	A Validation Shown As Below “The Username Entered Is Wrong”	A Validation Is Shown As Expected	Pass	High

		1. Navigate To Www.Sample.Co m	Validate Username And Password In DataBase And Once If They Correct Then Show The Main Page	Main Page/Pass Home Page Has Been Displayed	High
		2. Enter Validate Username And Password And Click On Submit			

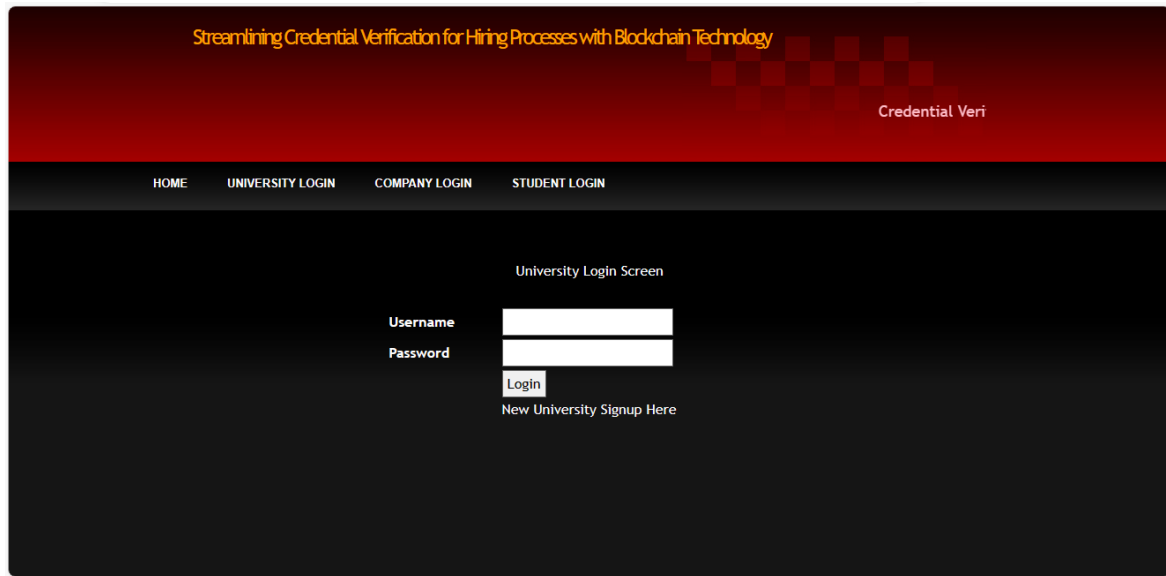
## 10.2 SCREENSHOTS

FIG 1:



In above screen click on 'University Login' link to get below page

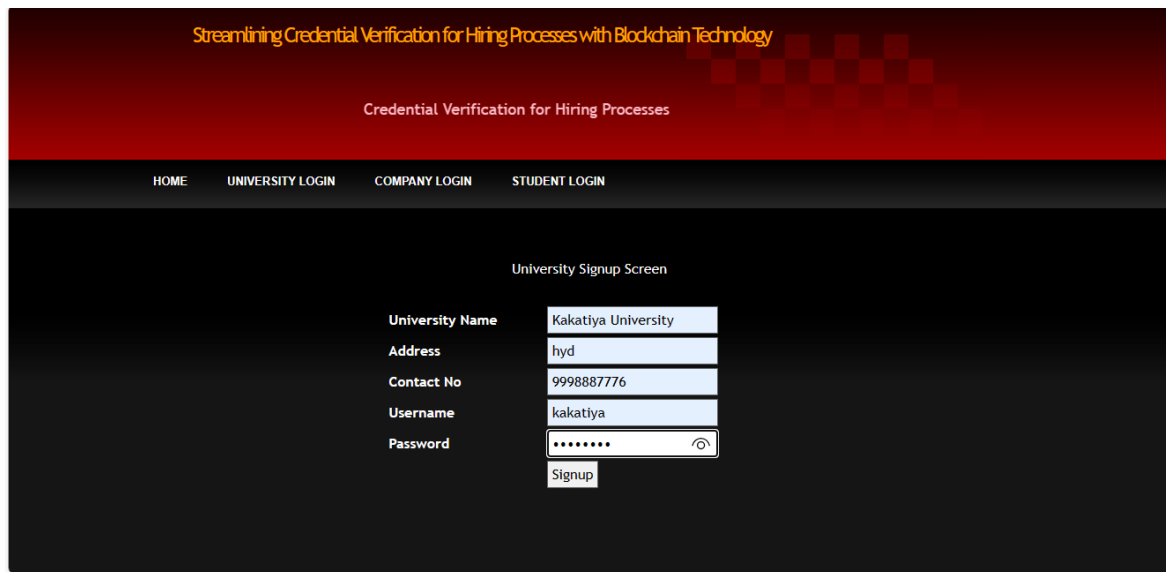
**FIG 2:**



The screenshot shows a web application interface with a dark red header and a black navigation bar. The header contains the text "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" and "Credential Veri". The navigation bar has links for "HOME", "UNIVERSITY LOGIN", "COMPANY LOGIN", and "STUDENT LOGIN". The main content area is black and displays the "University Login Screen". It includes a "Username" field, a "Password" field, a "Login" button, and a link that says "New University Signup Here".

In above screen click on 'New University Signup Here' link to get below signup page

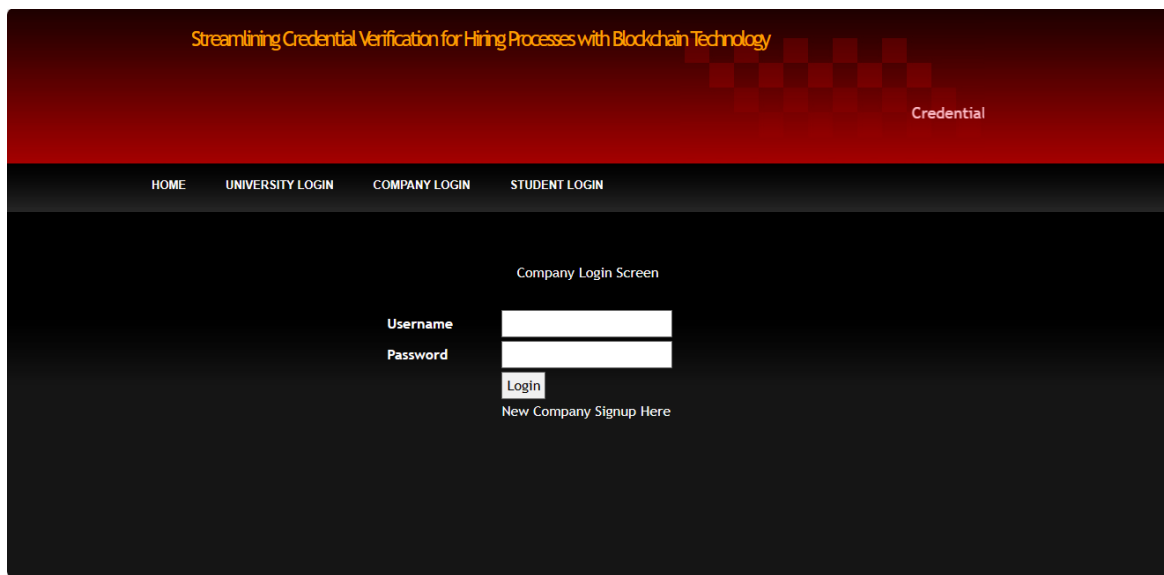
**FIG 3:**



The screenshot shows the "University Signup Screen" of the same web application. It features the same header and navigation bar as FIG 2. The main content area is black and displays the "University Signup Screen". It includes fields for "University Name", "Address", "Contact No", "Username", and "Password". The "University Name" field contains "Kakatiya University", "Address" contains "hyd", "Contact No" contains "9998887776", and "Username" contains "kakatiya". The "Password" field is masked with dots and has a toggle icon. A "Signup" button is located below the password field.

In above screen adding university signup details and then press button to save details in Blockchain and get below page

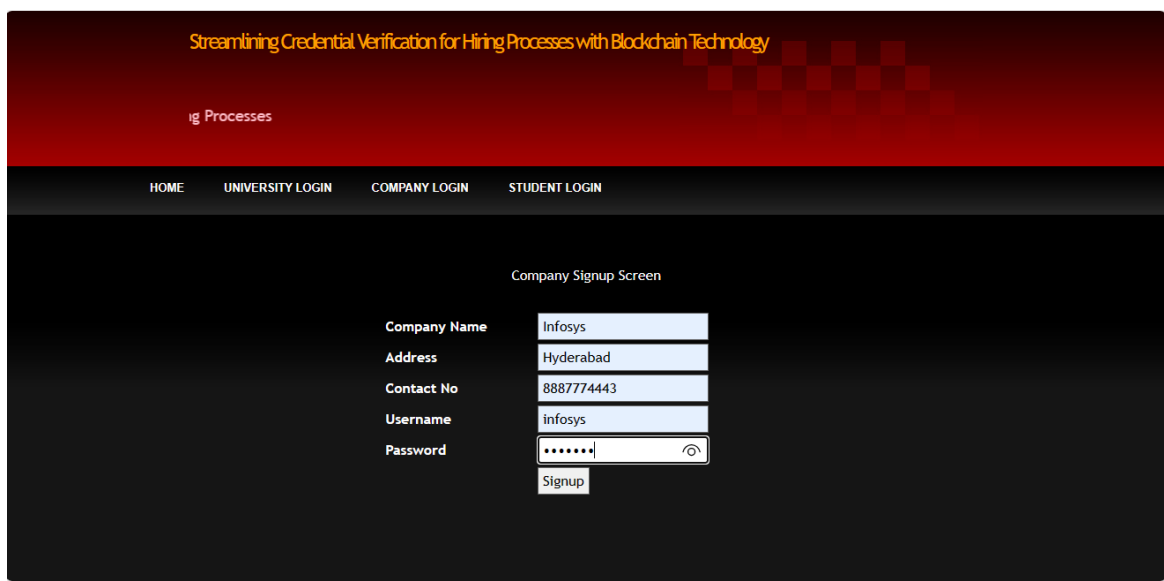
**FIG 4:**



The screenshot shows a web application interface with a dark red header and a black navigation bar. The header contains the text "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" and a "Credential" link. The navigation bar has links for "HOME", "UNIVERSITY LOGIN", "COMPANY LOGIN", and "STUDENT LOGIN". The main content area is titled "Company Login Screen" and contains a login form with fields for "Username" and "Password", a "Login" button, and a link for "New Company Signup Here".

In above screen click on 'New Company Signup Here' link to get below company signup screen

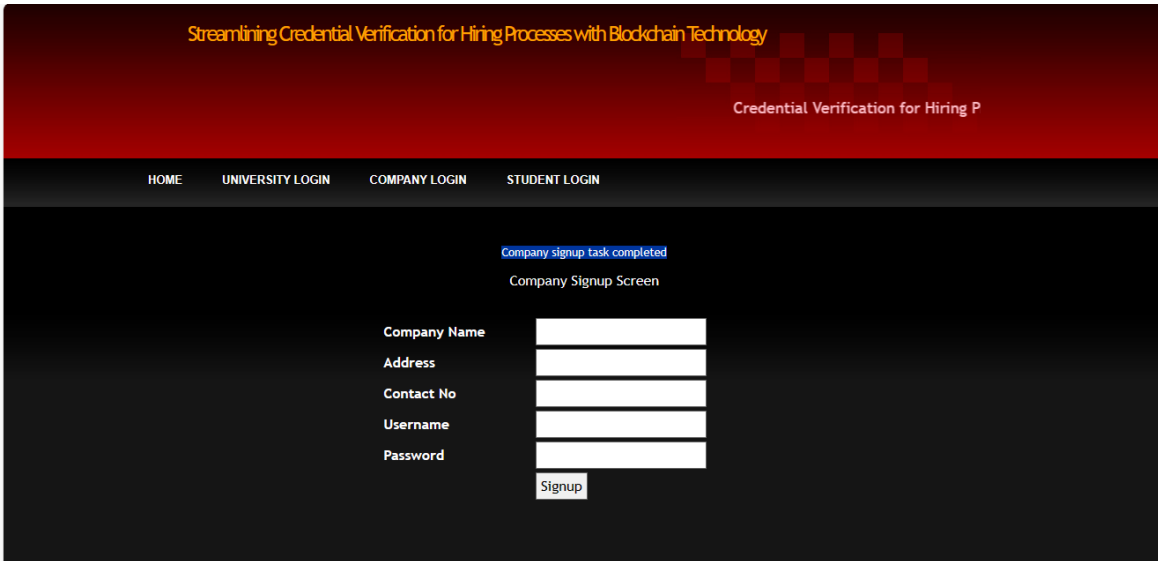
**FIG 5:**



The screenshot shows a web application interface with a dark red header and a black navigation bar. The header contains the text "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" and a "ig Processes" link. The navigation bar has links for "HOME", "UNIVERSITY LOGIN", "COMPANY LOGIN", and "STUDENT LOGIN". The main content area is titled "Company Signup Screen" and contains a signup form with fields for "Company Name", "Address", "Contact No", "Username", and "Password", a "Signup" button, and a toggle for password visibility.

In above screen adding company details and then press button to save company details in Ethereum and get below page

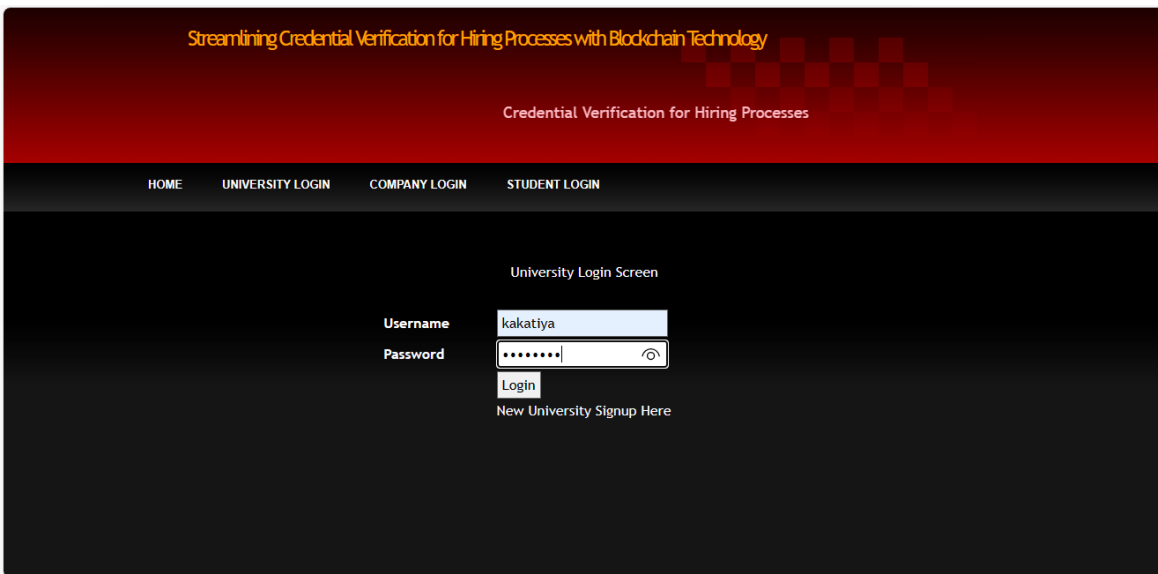
**FIG 6:**



The screenshot shows a web application interface with a dark red header. The header contains the text "Streamlining Credential Verification for Hiring Processes with Blockchain Technology" in yellow and "Credential Verification for Hiring P" in white. Below the header is a navigation bar with links: HOME, UNIVERSITY LOGIN, COMPANY LOGIN, and STUDENT LOGIN. The main content area is dark blue and displays a message "Company signup task completed" in blue. Below this message is the title "Company Signup Screen". The form includes fields for Company Name, Address, Contact No, Username, and Password, each with a corresponding input box. A "Signup" button is located at the bottom of the form.

In above Screen Company signup completed and now click on 'University Login' link to get below page

**FIG 7:**

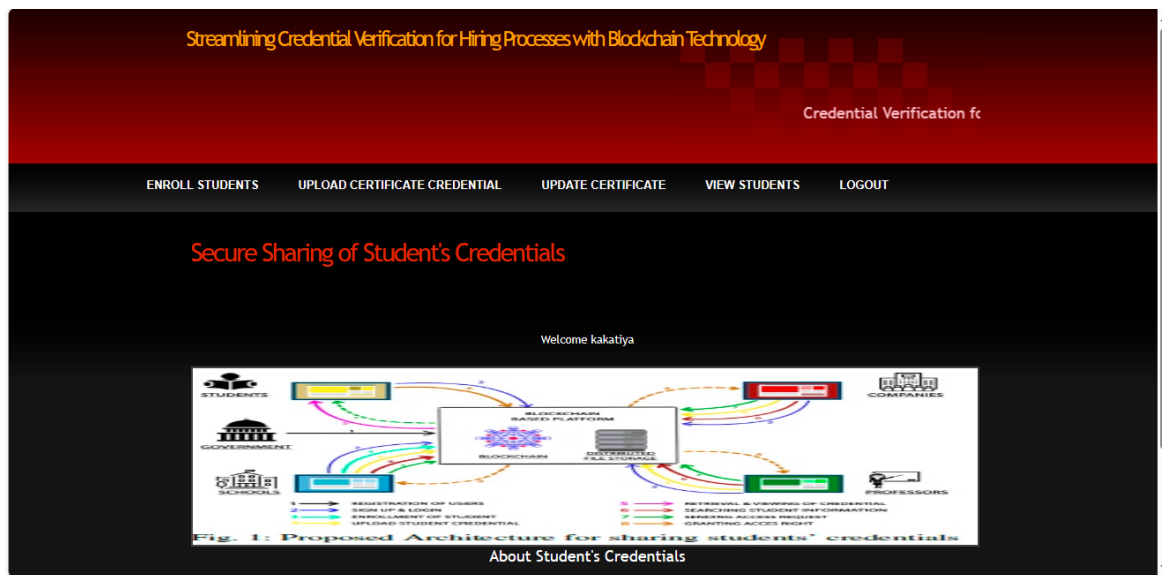


The screenshot shows the same web application interface as FIG 6, but with the "UNIVERSITY LOGIN" link selected in the navigation bar. The main content area is dark blue and displays the title "University Login Screen". The form includes fields for Username and Password, each with a corresponding input box. The Username field contains the text "kakatiya". The Password field contains a series of dots and a toggle icon. A "Login" button is located at the bottom of the form. Below the login button is a link "New University Signup Here".

In above screen university is login and after login will get below page



**FIG 8:**



In above screen university can click on 'Enroll Student' link to add student details in Ethereum

**FIG 9:**

Streamlining Credential Verification for Hiring Processes with Blockchain Technology

Credential Verification for Hiring Processes

ENROLL STUDENTS    UPLOAD CERTIFICATE CREDENTIAL    UPDATE CERTIFICATE    VIEW STUDENTS    LOGOUT

Enroll Student Screen

Student ID: 1

Student Name: Kumar

School Details: Kakatiya University

Course Name: MBA

Joining Date: 6-Oct-2020

Enroll

In above screen university is adding student details and then click on 'Enroll' button to save student details in Ethereum

**FIG 10:**

The screenshot shows a web application interface with a red header and a dark sidebar. The main content area is titled 'Enroll Student Screen' and contains a form for entering student details. The form fields are: Student ID, Student Name, School Details, Course Name, and Joining Date. Below the form is an 'Enroll' button. A notification message at the top of the form area reads 'New Student Enrollment Task Completed'.

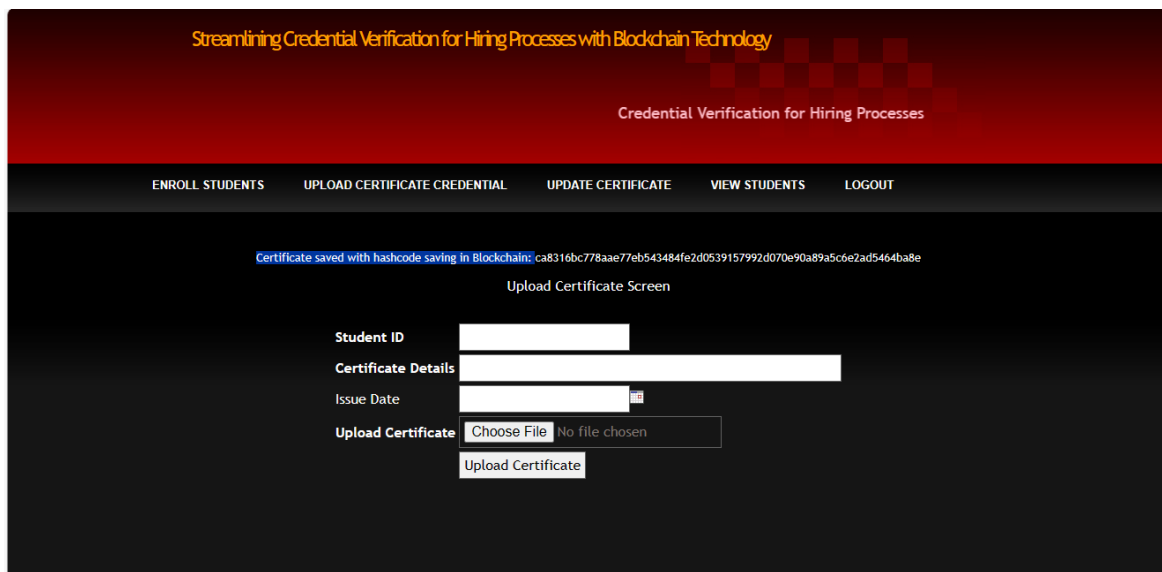
In above screen student details added and university can click on 'Upload Certificate Credential' link to upload student certificates in Blockchain

**FIG 11:**

The screenshot shows the 'Upload Certificate Screen' of the same web application. The form fields are: Student ID (1), Certificate Details (MBA Certificate), Issue Date (28-Oct-2022), and an 'Upload Certificate' button. A file explorer window is open over the form, showing a folder named 'certificate\_templates' containing five certificate images labeled 1 through 5. The file explorer window also shows the 'File name' field with '1' entered and the 'Open' button.

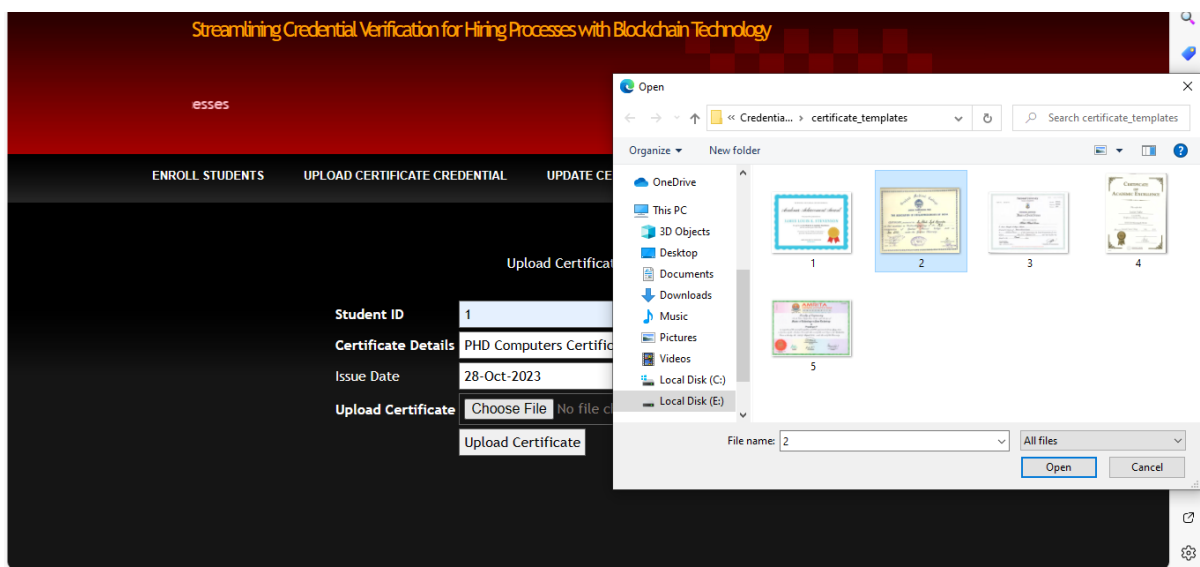
In above Screen University is uploading certificate for student ID 1 and then press button to save certificate in Ethereum

**FIG 12:**



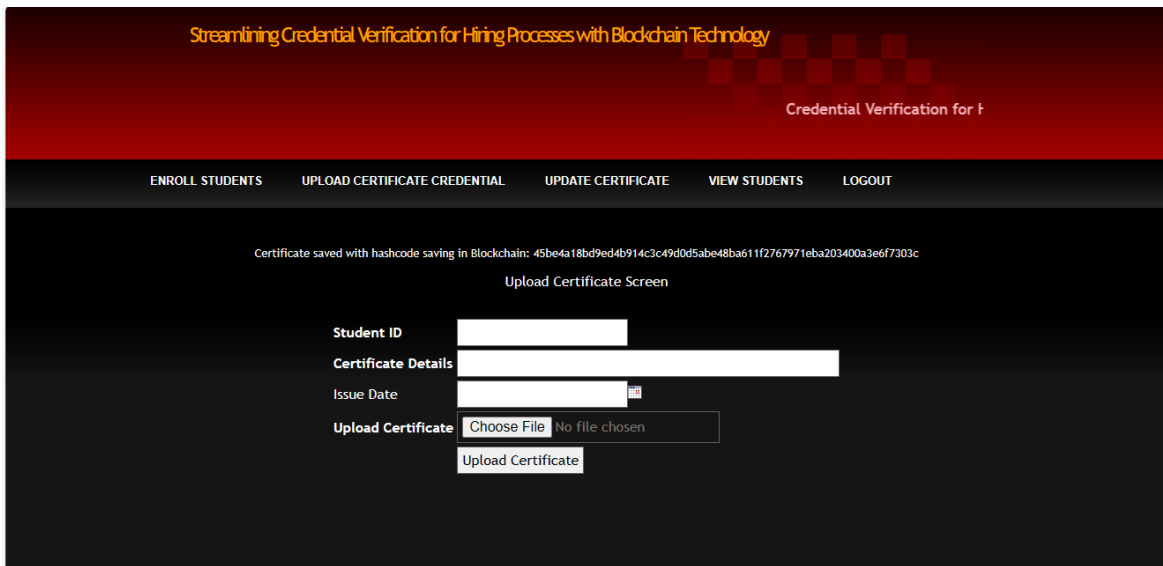
In above screen in blue colour text we can see certificate saved in Blockchain with displaying hashcode and under same student we can upload multiple certificates of different courses

**FIG 13:**



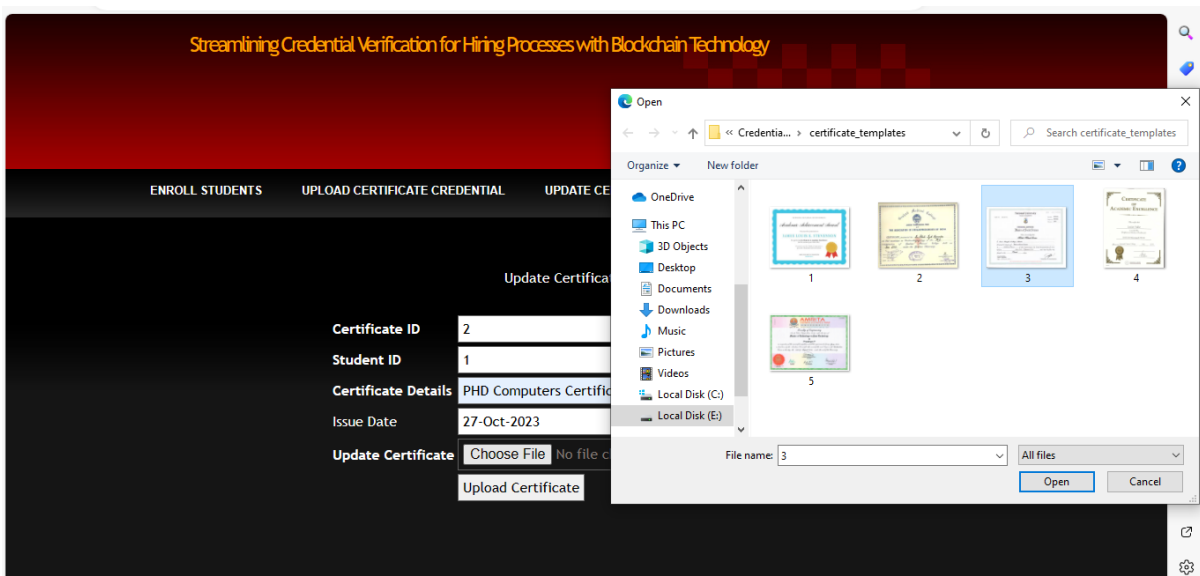
In above uploading another PHD certificate for same student and then press button to get below page

**FIG 14:**



In above screen certificate details saved and now in above screen click on ‘Update Certificate’ link to upload certificate for same student if changes required in certificate

**FIG 15:**

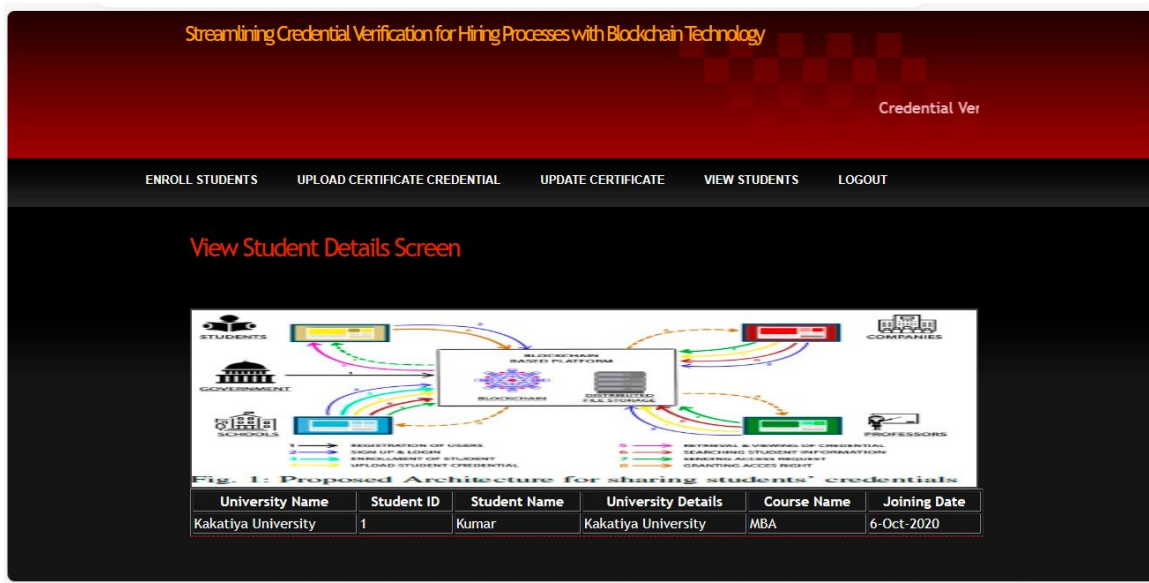


In above screen entering certificate and student id and then uploading modified certificate to make changes and get below screen

**FIG 16:**

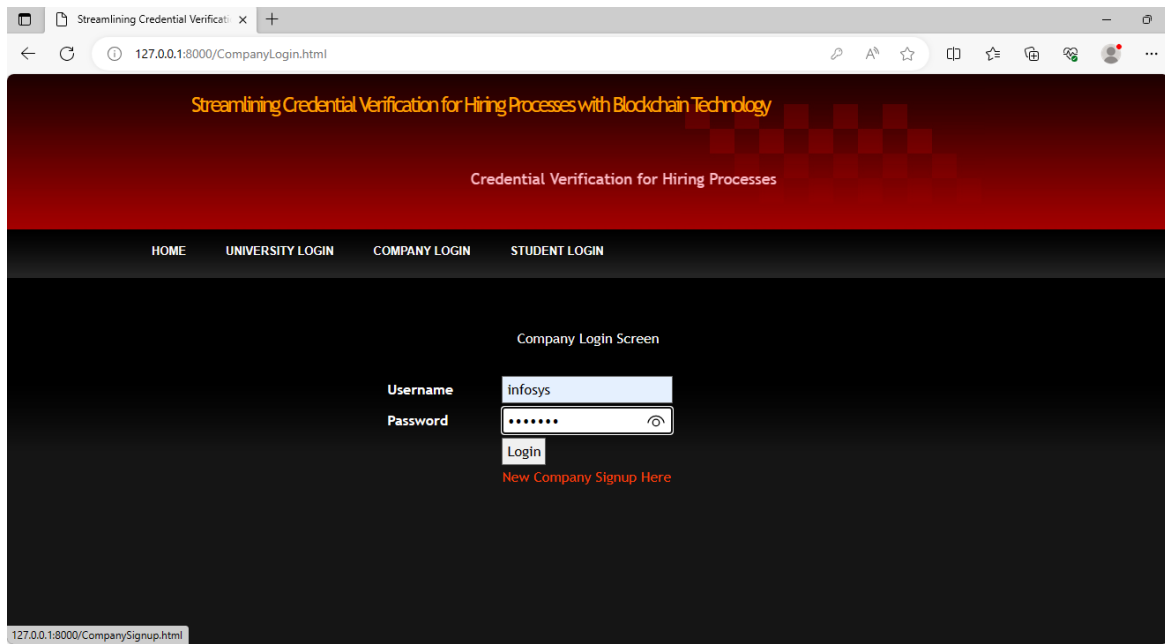
In above screen modified certificate saved in Blockchain and now click on ‘View Students’ link to view all student details

**FIG 17:**



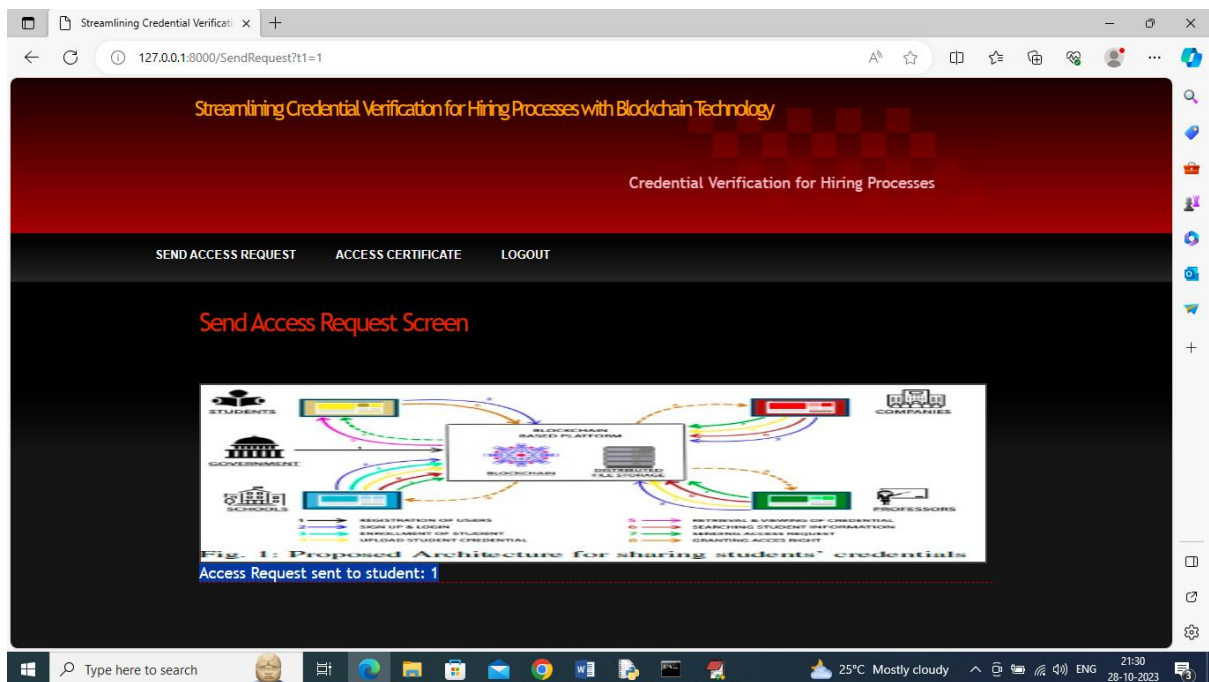
In above screen in tabular format we can see student details and now click on ‘Logout’ link and then login as company

**FIG 18:**



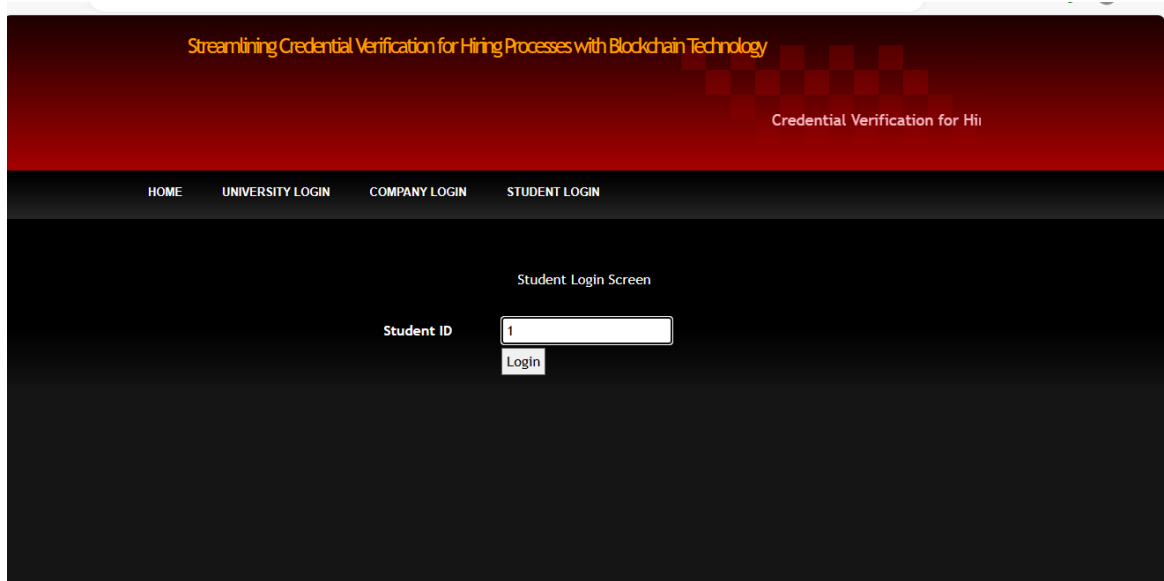
In above screen company is login and after login will get below page

**FIG 19**



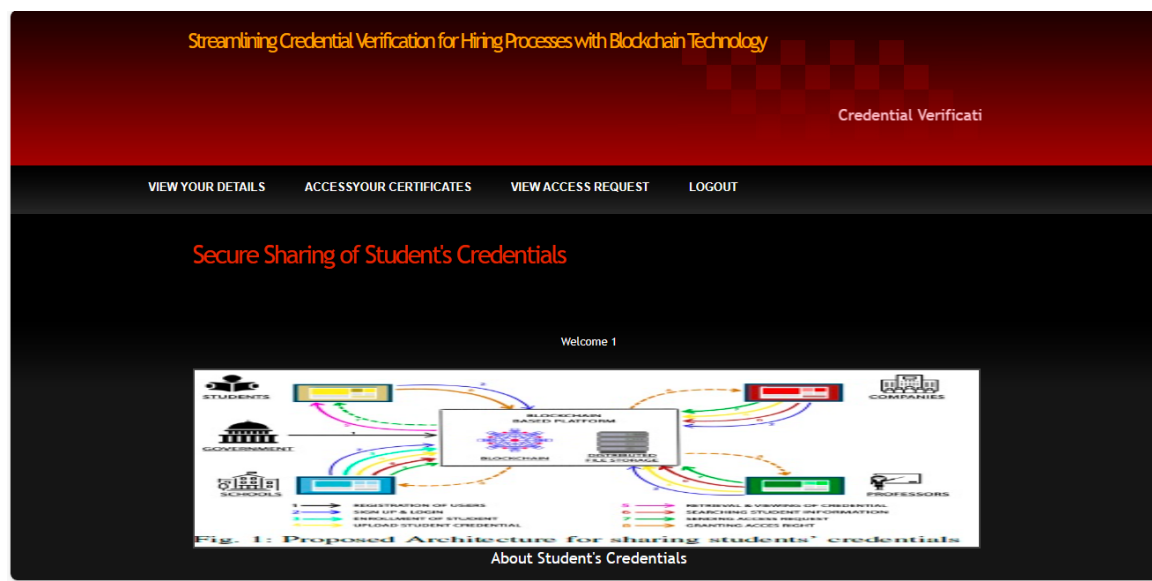
In above screen company can view all student and their course details and then click on 'Click Here' link to send access request to that student

**FIG 20:**



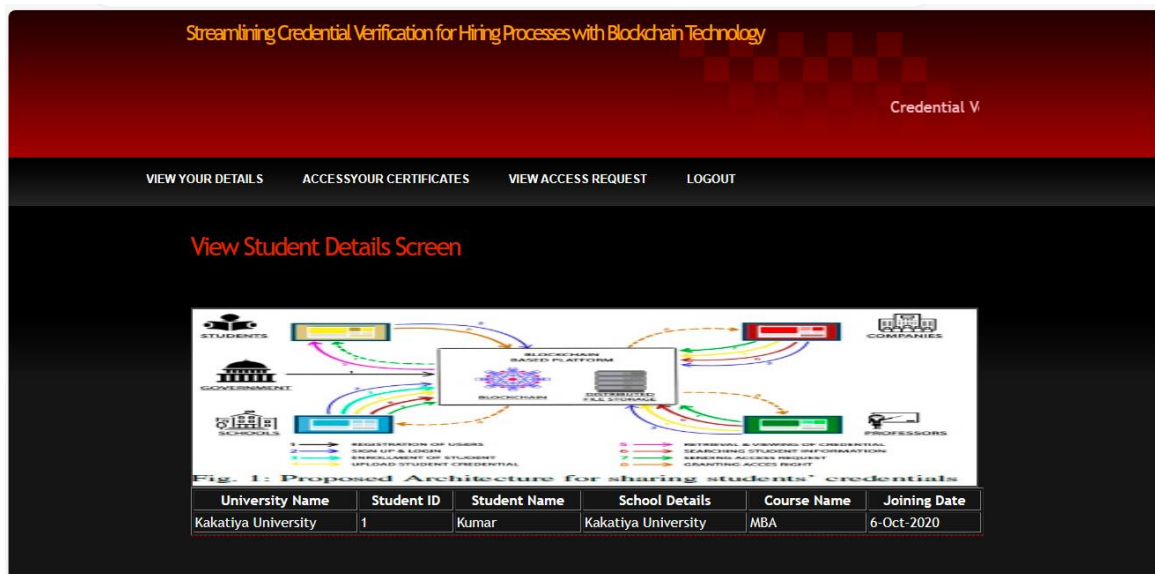
In above screen student is login by using his ID and then press button to get below page

**FIG 21:**



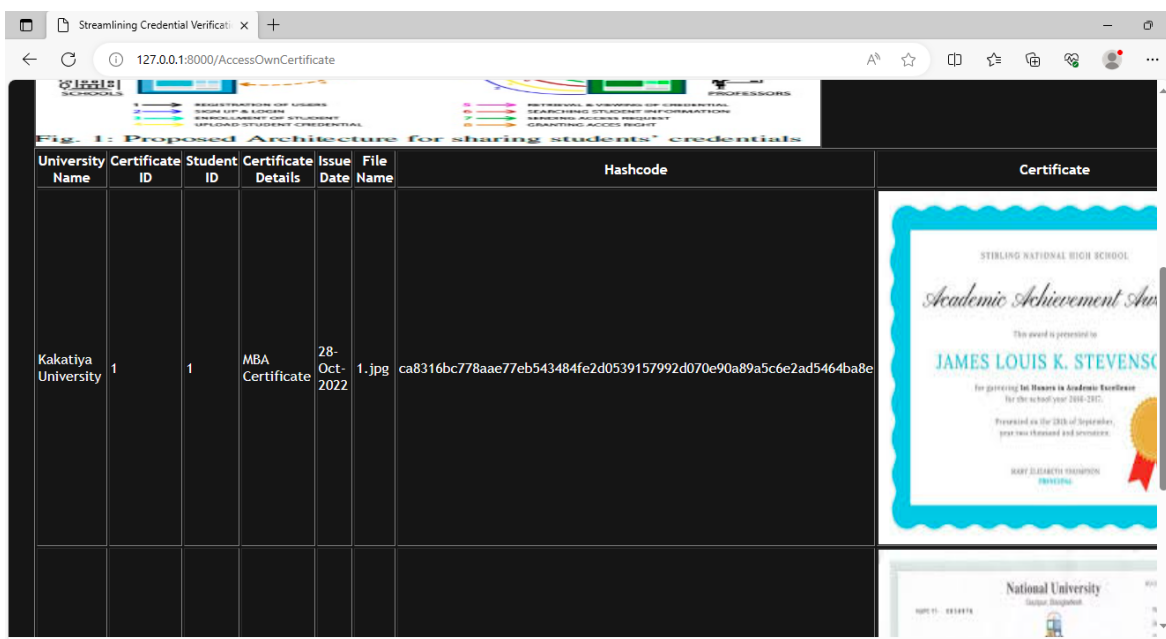
In above screen student can click on 'View Your Details' link to view their course details

**FIG 22:**

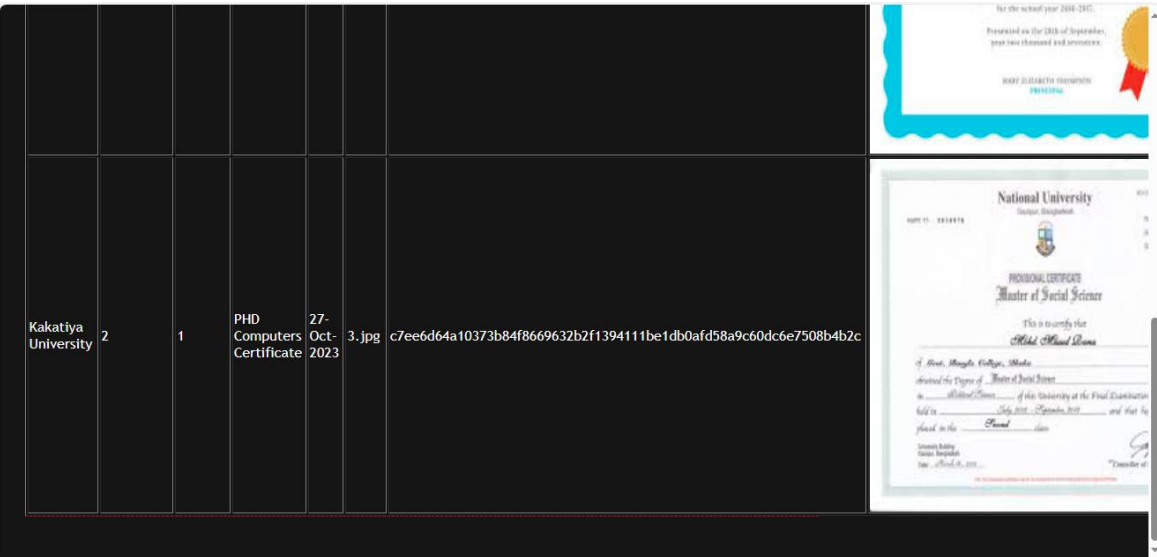


In above screen student can view his details and now click on 'Access Your Certificate' link to view their certificates like below screen

**FIG 23:**

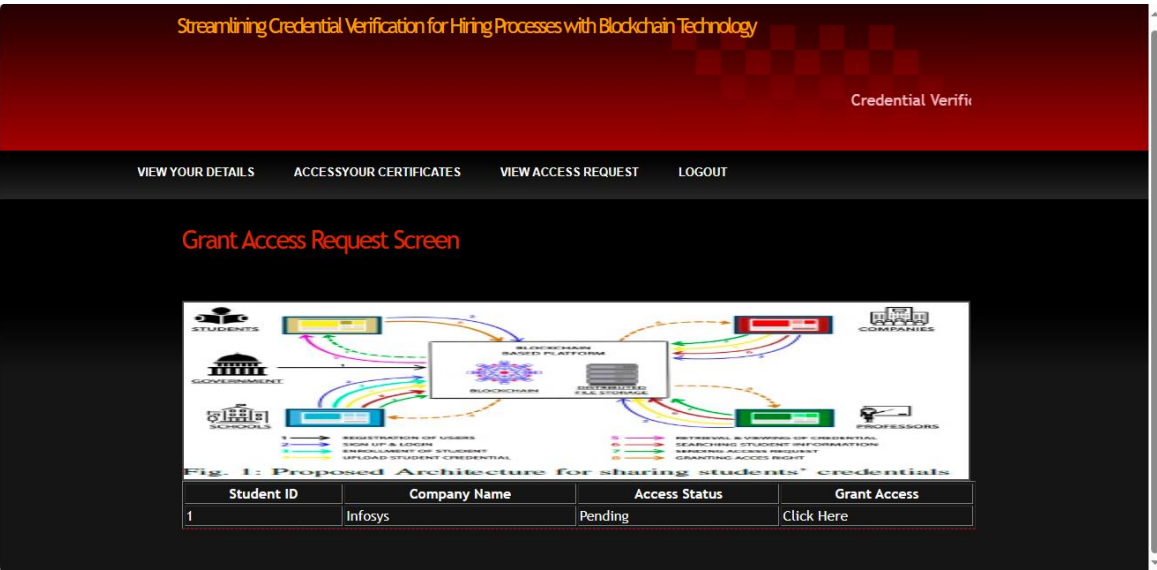






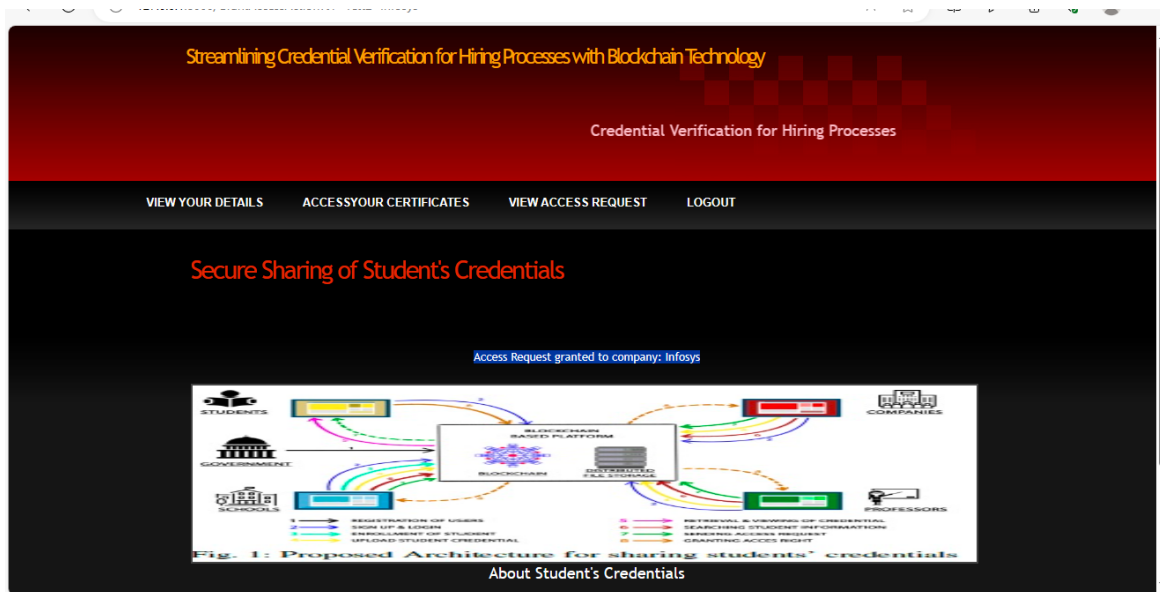
In above screen student can view all his certificates and now click on ‘View Access Request’ link to view request from companies

FIG 24:



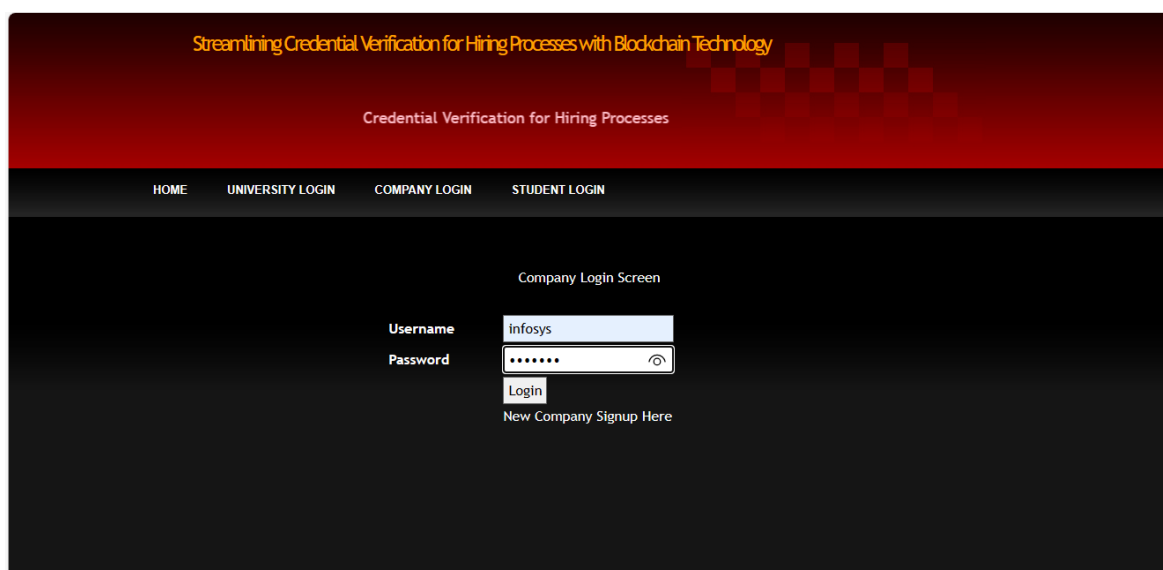
In above screen student can view company details and then click on ‘Click Here’ to approve request and get below page

FIG 25:



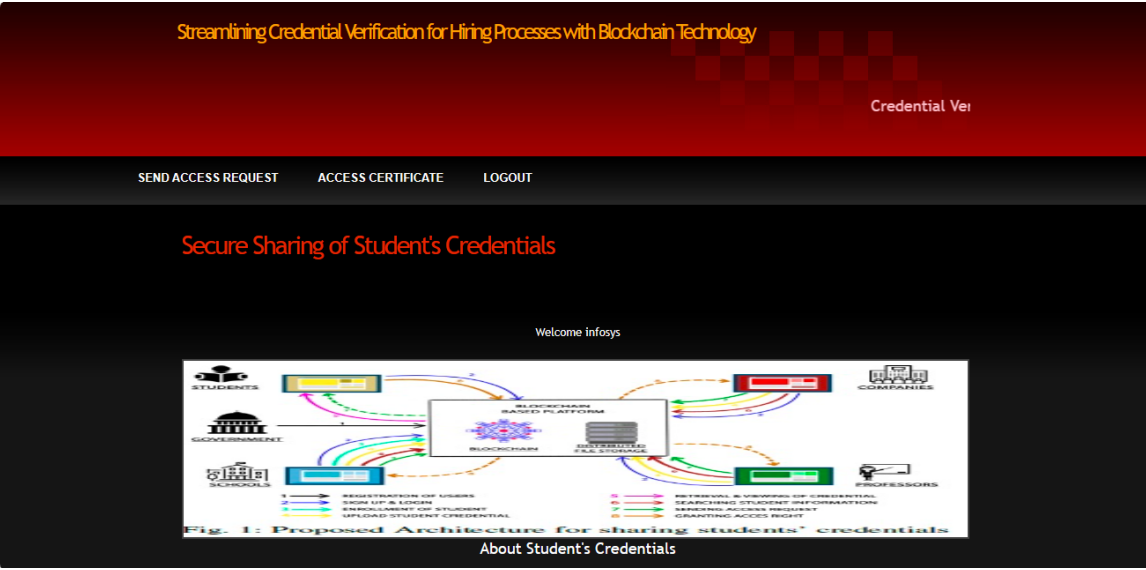
In above screen in blue colour text we can see Access Request Granted for given company and now logout and login as company so company can access granted certificates

**FIG 26:**



In above screen company is login and after login will get below page

**FIG 27:**



In above screen company can click on ‘Access Certificate’ link to view access granted certificates

FIG 28:

Certificate ID	Student ID	Certificate Details	Issue Date	File Name	Hashcode	Certificate
1	1	MBA Certificate	28-Oct-2022	1.jpg	ca8316bc778aae77eb543484fe2d0539157992d070e90a89a5c6e2ad5464ba8e	

In above screen company can view all certificates of all students

Similarly by following above screens you can upload any number of student credentials to Blockchain

# **CHAPTER-11**

## **CONCLUSION**

### **11.1 CONCLUSION**

In conclusion, the adoption of blockchain technology in credential verification heralds a new era of reliability and security in hiring processes. This project addresses the inadequacies of existing systems by introducing a decentralized, tamper-proof framework that not only enhances the trustworthiness of credentials but also streamlines the entire verification process. By empowering universities, companies, and students with secure access to authenticated records, the proposed system ensures a seamless, transparent, and efficient credential verification process, ultimately contributing to a more trustworthy and resilient hiring ecosystem.

### **11.2 FUTURE SCOPE**

The future scope of streamlining credential verification for hiring processes with blockchain technology holds significant promise for transforming the recruitment landscape. As blockchain continues to mature and gain widespread adoption, its application in credential verification is poised to revolutionize how organizations verify the authenticity of candidates' qualifications. One key aspect of this future scope lies in the enhanced efficiency and accuracy that blockchain offers, enabling seamless verification of credentials in real-time while reducing administrative burdens and costs associated with traditional methods. Additionally, blockchain's immutable ledger ensures data integrity and transparency, fostering greater trust between employers and candidates. Looking ahead, advancements in decentralized identity management and self-sovereign identity solutions powered by blockchain will further streamline the verification process, empowering individuals to control and share their credentials securely. Moreover, as regulatory frameworks evolve to accommodate blockchain-based verification systems, we can anticipate increased acceptance and adoption across industries. Overall, the future scope of streamlining credential verification with blockchain technology not only promises to enhance hiring processes but also to contribute to a more efficient, transparent, and inclusive recruitment ecosystem.

## CHAPTER-12

### REFERENCES

1. Protecting the Public in Action, Nov. 2022, [online] Available: <https://pubmed.ncbi.nlm.nih.gov/29758142/>.
2. S. D. Barnett, "Growing pains of credentialing research: Discussions from the institute of medicine workshop", *J. Continuing Educ. Nursing*, vol. 46, no. 2, pp. 5355, 2015.
3. M. H. Baumann, S. Q. Simpson, M. Stahl, S. Raoof, D. D. Marciniuk and D. D. Gutterman, "First do no harm: Less training  $\neq$  quality care", *Amer. J. Crit. Care*, vol. 21, no. 4, 2012.
4. S. Chen, T. Luo, W. Liu and J. Song, "A framework for managing access of large-scale distributed resources in a collaborative platform", *Data Sci. J.*, vol. 7, Dec. 2008.
5. G. Legotlo and A. Mutezo, "Understanding the types of fraud in claims to south African medical schemes", *South Afr. Med. J.*, vol. 108, no. 4, pp. 299, Mar. 2018.
6. M. Hölbl, M. Kompara, A. Kamišalić and L. Nemec Zlatolas, "A systematic review of the use of blockchain in healthcare", *Symmetry*, vol. 10, no. 10, pp. 470, Oct. 2018.
7. T. McGhin, K.-K. R. Choo, C. Z. Liu and D. He, "Blockchain in healthcare applications: Research challenges and opportunities", *J. Netw. Comput. Appl.*, vol. 135, pp. 62-75, Jun. 2019.
8. D. J. Skiba, "The potential of blockchain in education and health care", *Nursing Educ. Perspect.*, vol. 38, no. 4, pp. 220-221, Aug. 2017.
9. Q. Mamun, "Blockchain technology in the future of healthcare", *Smart Health*, vol. 23, Mar. 2022.
10. M. N. Kamel Boulos, J. T. Wilson and K. A. Clauson, "Geospatial blockchain: Promises challenges and scenarios in health and healthcare", *Int. J. Health Geographics*, vol. 17, no. 1, pp. 25, Dec. 2018.
11. A. Hevner and S. Chatterjee, "Design science research in information systems" in *Design Research in Information Systems: Theory and Practice*, New York, NY, USA: Springer, pp. 9-22, 2010.
12. S. Smith, Top 10 Challenges in Healthcare Credentialing, Tollanis Solutions Inc, Sep. 2022, [online] Available: <https://tollanis.com/credentialing/top-challenges-healthcare-credentialing/>.

13.R. Arenas and P. Fernandez, "CredenceLedger: A permissioned blockchain for verifiable academic credentials", *Proc. IEEE Int. Conf. Eng. Technol. Innov. (ICE/ITMC)*, pp. 1-6, Jun. 2018.

14.L. Zhou, L. Wang and Y. Sun, "MIStore: A blockchain-based medical insurance storage system", *J. Med. Syst.*, vol. 42, no. 8, pp. 117, Aug. 2018.