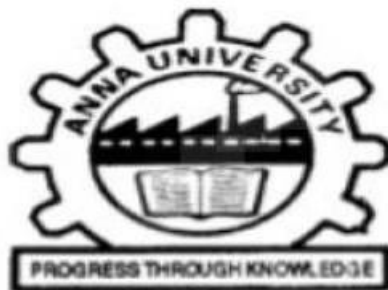


**THE TRUSTED SCHOLARSHIP PLATFORM USING  
SMART CONTRACT  
PROJECT REPORT**

Submitted by  
**DHINESH. V**  
**JABASELVI. A**

in partial fulfilment for the award of the degree  
of  
BACHELOR OF TECHNOLOGY  
IN  
INFORMATION TECHNOLOGY

UNIVERSITY COLLEGE OF ENGINEERING TINDIVANAM  
MELPAKKAM – 604 001



ANNA UNIVERSITY: CHENNAI 600025

JUNE -2022

**ANNA UNIVERSITY: CHENNAI 600025**

**BONAFIDE CERTIFICATE**

Certified that this project report “ **THE TRUSTED SCHOLARSHIP PLATFORM USING SMART CONTRACT A PROJECT REPORT** “is the bonafide work of “**DHINESH. V (422419205010), JABASELVI. A (422419205015)**”, who carried out the project work under my supervision.

**SIGNATURE**

Dr. S. MILTON GANESH, M.E., Ph.D.,

**HEAD OF THE DEPARTMENT**

Department of Information Technology,

University College of Engineering

Tindivanam,

Melpakkam - 604001.

**SIGNATURE**

Dr. R. SHYAMALA, M.E., Ph.D.,

**SUPERVISOR**

Department of Information Technology,

University College of Engineering

Tindivanam,

Melpakkam - 604001.

**Submitted for the University Examination held on \_\_\_\_\_**

**INTERNAL EXAMINAR**

**EXTERNAL EXAMINER**

## ACKNOWLEDGEMENT

We would like to thank our revered Dean, **Dr.P. Thamizhazhagan, M.E., Ph.D.**, for providing infrastructure facilities and wholehearted encouragement for completing our project successfully.

For mostly, we pay our grateful acknowledgement and extend our sincere gratitude to **Dr. S. MiltonGanesh,M.E., Ph.D.**, Assistant Professor and Head, Department of Information Technology, University College of Engineering Tindivanam, for extending the facilities of the department towards our project and for her unstinting support.

We express our sincere thanks to our supervisor and our class advisor, **Dr. R. Shyamala,M.E., Ph.D.**, Assistant Professor, Department of Information Technology, University College of Engineering Tindivanam, for guiding us for every phase of the Mini project. We appreciate her thoroughness, tolerance and ability to share her knowledge with us.

We thank her for being easily approachable and quite thoughtful. We owe her harnessing our potential and bringing out the best in us. Without her immense support through every step of the way, we could never have it to this extent. We thank all our teaching and non-teaching faculty members, our class advisor, and also our fellow friends helping us in providing valuable suggestions and timely ideas for the successful completion of the Mini project.

Last but not least, we extend our thanks to our family members, who have been a great source of inspiration and strength to us during the course of this Mini project work. We sincerely thank all of them.

## **ABSTRACT**

Disruptive technologies such as Cloud, IoT, Blockchain, and Data Analytics have raised new automation requirements in different Industries. Contract Management in Business-to-Business environment needs to be overhauled as well. Smart Contract Management System is Blockchain based innovative technology foreseen to automate future Business processes. Blockchain has brought in business process re-engineering by optimizing the business workflow activities, especially in multi-party arrangements. Smart Contract-mechanism to automate B2B processes are triggered by some events generated through IoT devices, data feeds or other applications.

This project will give a web-based service using which different organization can create smart contracts among themselves and will also be able to see the data of transaction stored on the blockchain. A user through a web interface will create a smart contract with another user. The Smart Contract will have some terms and clauses in it which will be necessary to be completed in order for some events to execute. If the terms of the contract are met then an event like transaction will be executed. These transactions will be bundled into a block and that block will be placed on a blockchain network. The contract stored in the blockchain has the property of immutability. It cannot be changed once it has been created. This makes smart contracts more secure than traditional paper contracts. Also, all the data generated by the smart contracts will be stored inside the blocks of the blockchain for security and tamper-proof property.

**DHINESH**

**JABASELVI**

# TABLE OF CONTENT

Chapter	Title	Page no
	Abstract	iv
<b>1</b>	<b>INTRODUCTION</b>	1
	1.1 Introduction	1
	1.2 Problem statement	1
	1.3 Scope	1
	1.4 Feasibility study	2
	1.4.1 Technical feasibility	2
	1.4.2 Economic feasibility	2
	1.4.3 Operation feasibility	3
<b>2</b>	<b>SYSTEM DESIGN</b>	4
	2.1 Literature review	4
	2.2 System architecture	6
	2.3 Data flow diagram	6
	2.4 UML diagram	10
	2.4.1 Use case diagram	11
	2.4.2 Class diagram	15
	2.4.3 Sequence diagram	17
	2.4.4 Activity diagram	20
	2.5 Key concepts of blockchain	24
<b>3</b>	<b>MODULE</b>	29
	3.2 Module description	29
	3.2.1 Registration module	29
	3.2.2 Authorization module	29
	3.2.3 Transaction module	29

	3.3 A smart contract management	30
	3.4 Characteristics	30
	3.5 Types	31
	3.6 Access smart contract	32
<b>4</b>	<b>CODING AND TESTING</b>	<b>33</b>
	4.1 Coding	33
	4.2 Developing methodologies	33
	4.3 System testing	33
	4.3.1 Unit testing	34
	4.3.2 Functional testing	34
	4.3.2.1 Performance testing	34
	4.3.2.2 Stress testing	35
	4.3.3 Integration testing	35
	4.3.4 Validation testing	35
	4.3.5 Output testing	35
	4.3.6 User acceptance testing	35
<b>5</b>	<b>CONCLUSION</b>	<b>36</b>
	5.1 Appendices	38
	5.1.1 Appendix: 1 Sample coding	47
	5.1.2 Appendix: 2 Screenshots	52

# **CHAPTER 1**

## **1.1 INTRODUCTION**

This chapter elucidates the problem statement and scope of the project. Section 1.1 explains the need of the system. The next section addresses to the scope of the problem.

## **1.2 PROBLEM STATEMENT**

Disruptive technologies such as Cloud, IoT, Blockchain, and Data Analytics have raised new automation requirements in different Industries. Traditional contract management systems are not able to keep up with these Disruptive technologies. So, these traditional systems are needed to be changed and modified. Contract Management in Business to Business environment needs to be overhauled as well.

## **1.3 SCOPE**

Smart Contract Management System is Blockchain based innovative technology foreseen to automate future Business processes. Blockchain has brought in business process re-engineering by optimizing the business workflow activities, especially in multi-party arrangements. Smart Contract-a mechanism to automate B2B processes are triggered by some events generated through IoT devices, data feeds or other applications. Smart contracts are auto-executing contracts which are deployed on the blockchain with the terms of the agreement between two or more organizations being directly written into lines of code. The code and the agreements contained within exist across a distributed, tamper-proof, decentralized blockchain network.

## **1.4 FEASIBILITY STUDY**

Feasibility study aims to objectively and rationally uncover the system, weakness of the existing system and strength of proposed system, opportunities and threats as presented by environment, there sources required to carry through and ultimately the proposes for process. The feasibility study is conducted to see whether a project can be further preceded or discontinuing the project. The feasibility analysis is useful to determine final product will benefit to the users and organizations. Three aspects of feasibility study are i) Technical Feasibility ii) Economic Feasibility iii) Operational Feasibility.

### **1.4.1 TECHNICAL FEASIBILITY**

This study is carried out to check the technical feasibility, that is, the technical feasibility requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The technical requirements are then compared to the technical capability of the organization. The project is considered technically feasible if the internal technical capability is sufficient to support the project requirements.

### **1.4.2 ECONOMIC 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 expenditure 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.



### 1.4.3 OPERATIONAL FEASIBILITY

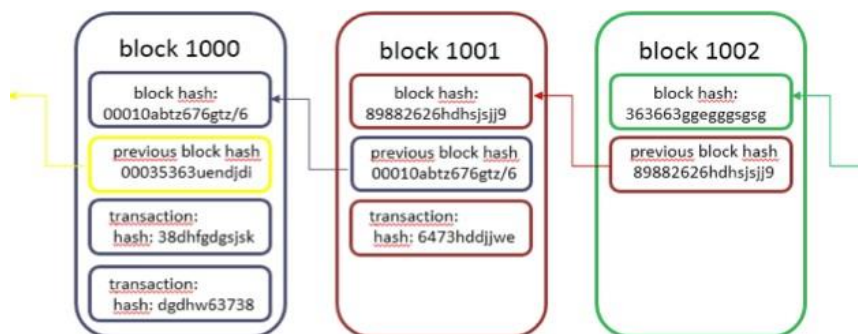
The aspect of study is to check the level of acceptance of system by the user. This includes the process of training the user to use the system efficiently.

## CHAPTER 2

### 2.1 LITERATURE REVIEW

#### What is Blockchain?

Blockchain is a list of blocks that are connected to each other. Traditionally, Blockchain uses the Linked List data structure to store all the blocks. Each block in the chain is preceded by and succeeded by another block in the chain. The first block of the blockchain is known as the Genesis Block. It doesn't have any transaction data just some rules that will be followed by the chain. Other than that the genesis block has a block number (index), hash value and a timestamp. When a block is added to the blockchain the index is incremented and the hash values are generated based on the data stored in the block. Each block has its own hash, the hash of the previous block and an automatic verification attribute. To confirm if the blockchain is valid or not, one can simply recompute the hashes and match with those stored in consecutive blocks. This is one of the basic features of blockchain that make them so popular and undisputed technology.

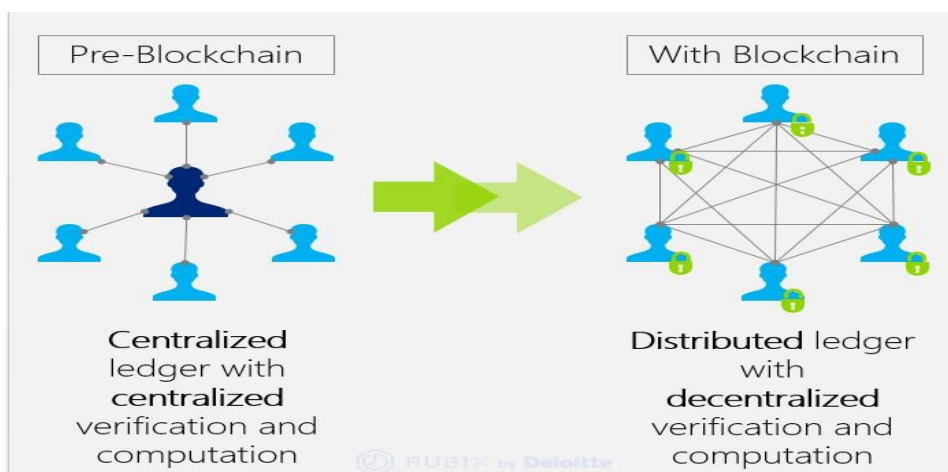


*Figure 1 A simple blockchain*

Blockchain is global just like the internet. Its architecture allows organizations to make their transactions secure without the intervention of any intermediary. One of the biggest features of Blockchain is the removal of the third-party which reduces the cost and time taken making the

process more reliable, cost-efficient and optimized.

In Blockchain addition of each block follows a consensus protocol. This consensus protocol ensures that all the nodes in the network have the same ledger and are valid blocks. The architecture allows the ledger to be created and shared among several parties. Thus, the ledger is maintained and controlled by participating nodes eliminating the central authorities. Every time there is a new transaction to be added to the ledger, the data is encrypted and its hash is generated. This hash is verified by a majority of nodes which are the participants of the network. When a consensus is reached, a new block is added to the blockchain. Once the transaction block is added to the blockchain, it is considered to be secure, trusted and immutable. The property of immutability refers to the property of blockchain which won't allow any kind of change in any block of the blockchain after it is created. Once the block is mined, it remains there forever. A new block is added through a complex process called mining. With the addition of a new block the difficulty of mining increases. This difficulty ensures that removing a block or changing the information inside the block in the blockchain and recreating all the hashes is not a simple task. Rather it requires a lot of computing power. By this process, trust is maintained amongst the nodes of the blockchain.



*Figure 2 Comparing centralized and distributed ledger.*

## **Blockchain Platforms**

Since, after the introduction of Bitcoin many development teams have introduced their tools and frameworks for blockchain development. These platforms leverage the blockchain architecture to smaller organizations. Through these platforms, one can communicate with other participating nodes in the network and control the actual blockchain and consensus protocols without taking care of the details. Some of these platforms are discussed below.

### **Ethereum**

Ethereum is a platform that provides a general way to implement blockchain. Ethereum is mainly used for public blockchains in which any node can become part of the network. It also aims to provide a consolidated end-to-end system to develop distributed applications that may have previously been not developed. Also, ethereum gives services of smart contracts that are stored on the blockchain.

### **Hyperledger**

Hyperledger Fabric is also a framework for leveraging functionalities of a shared-distributed ledger and running smart contracts upon them. Hyperledger allows the development of both public and private blockchain. Smart contracts are digital and self-executing pieces of code that enforce themselves upon the fulfillment of any pre-defined conditions. Hyperledger is built upon modular architecture and contains several pluggable modules for developers of blockchain applications.

## **2.2 System Architecture**

Blockchain is a set of distributed nodes that transfer data among them. Each blockchain executes a piece of code called chain-code. Chain-code is another name of a smart contract. Blockchain also has a distributed ledger that contains all the records of transactions. Blockchain

## ARCHITECTURE:

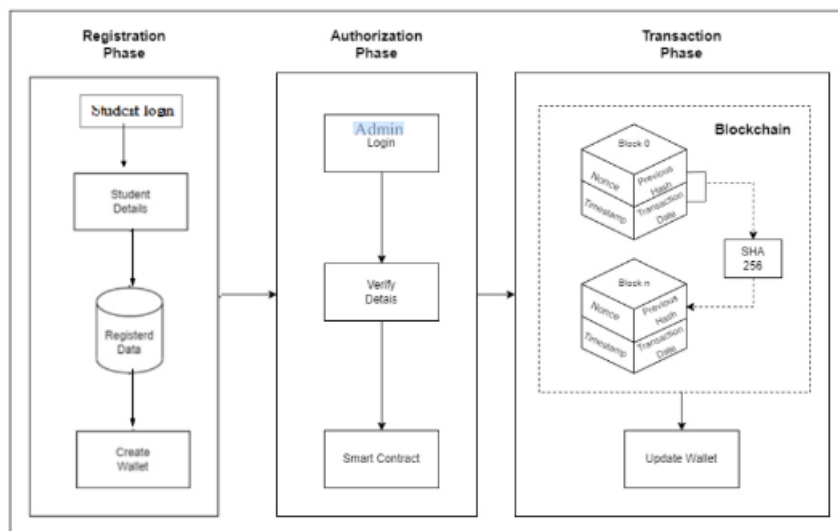
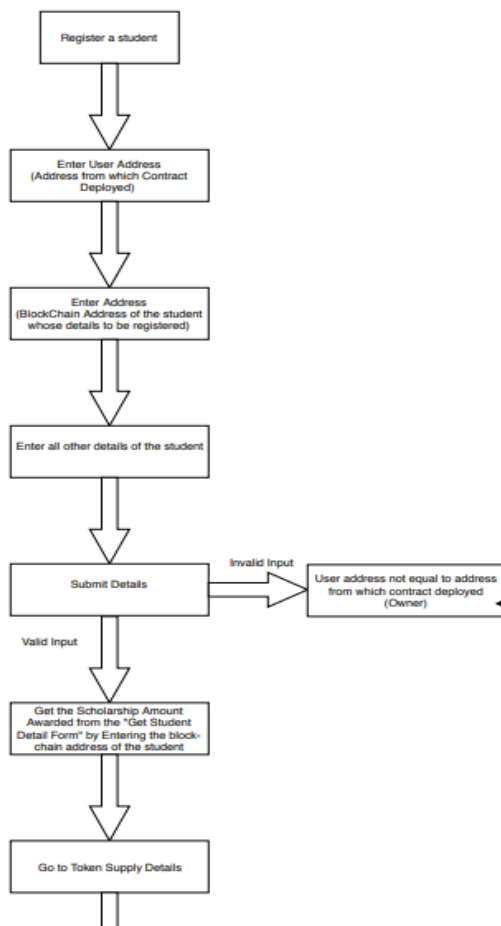
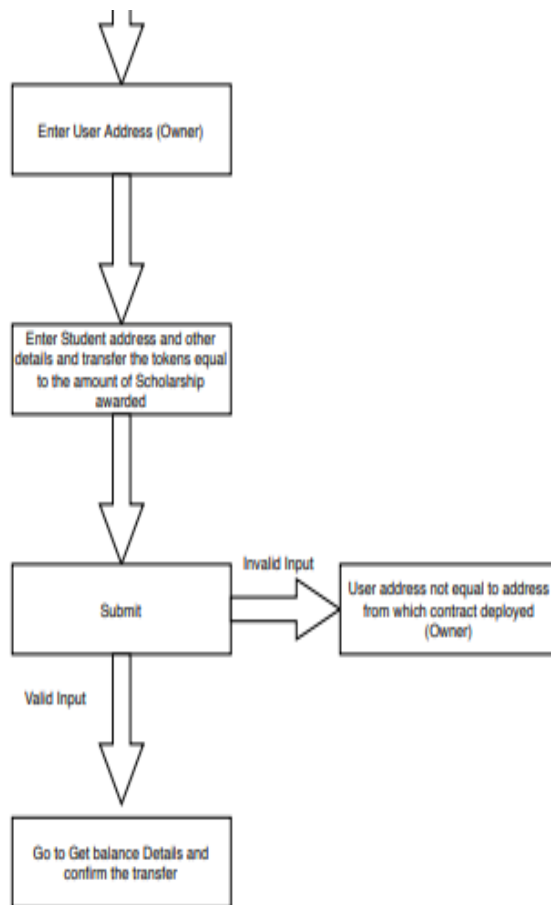


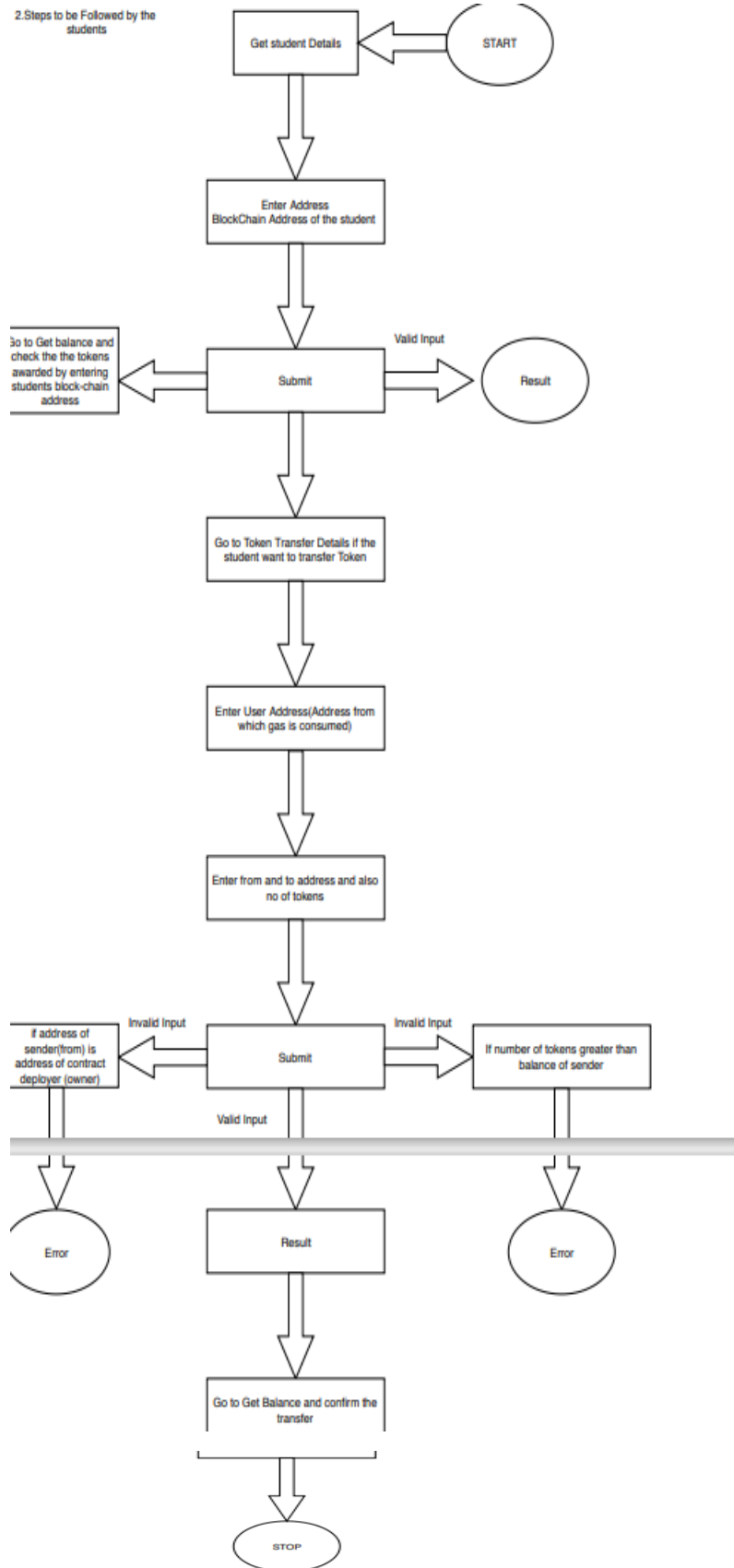
Figure 3 System architecture.

## 2.3 FLOWDIAGRAM

1. Steps to be Followed by the Owner (contract deployer)







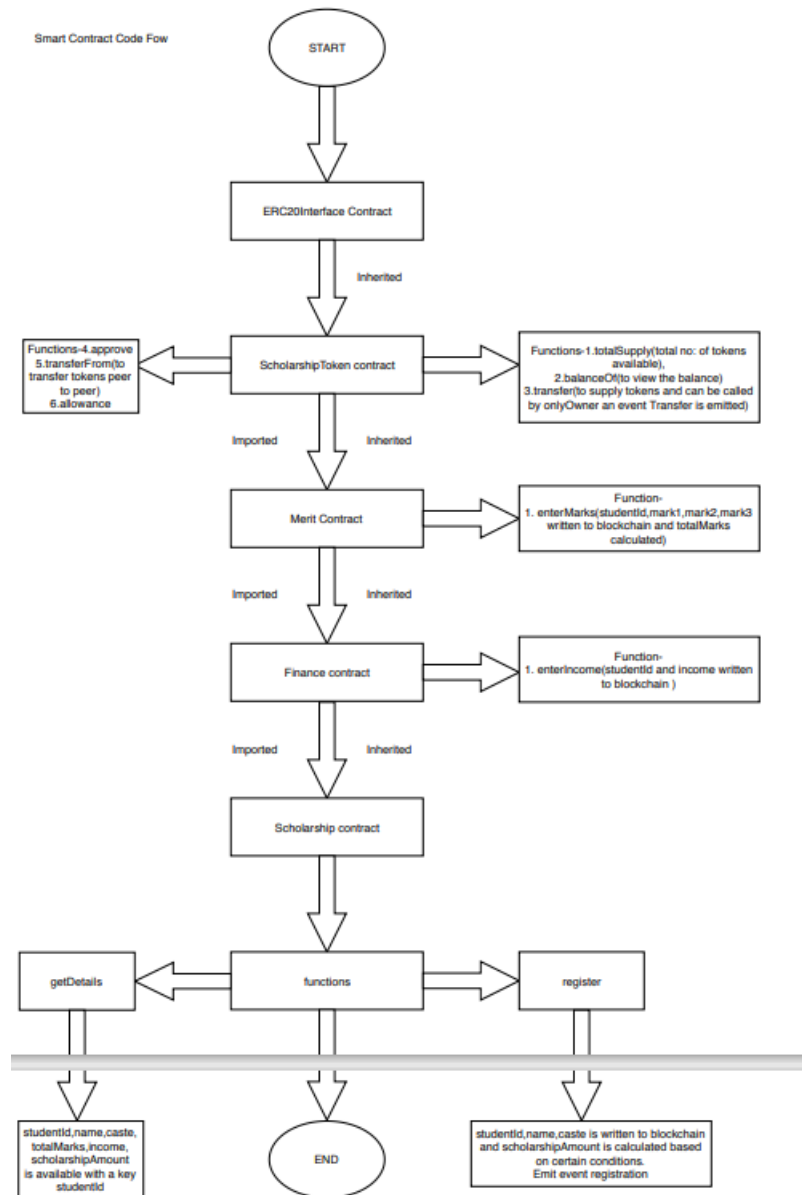


Figure 4 Data flow diagram

## 2.4 UML DIAGRAM

UML stands for unified Modelling Language. UML is a standardized general purpose modelling language in the field of Object-Oriented Software Engineering which is intended to provide a standard way to visualize the design of a system. The goal is for UML to become a common language for creating models of object-oriented computer software. UML is not a development method by itself; however, it was designed to be compatible with that leading object-



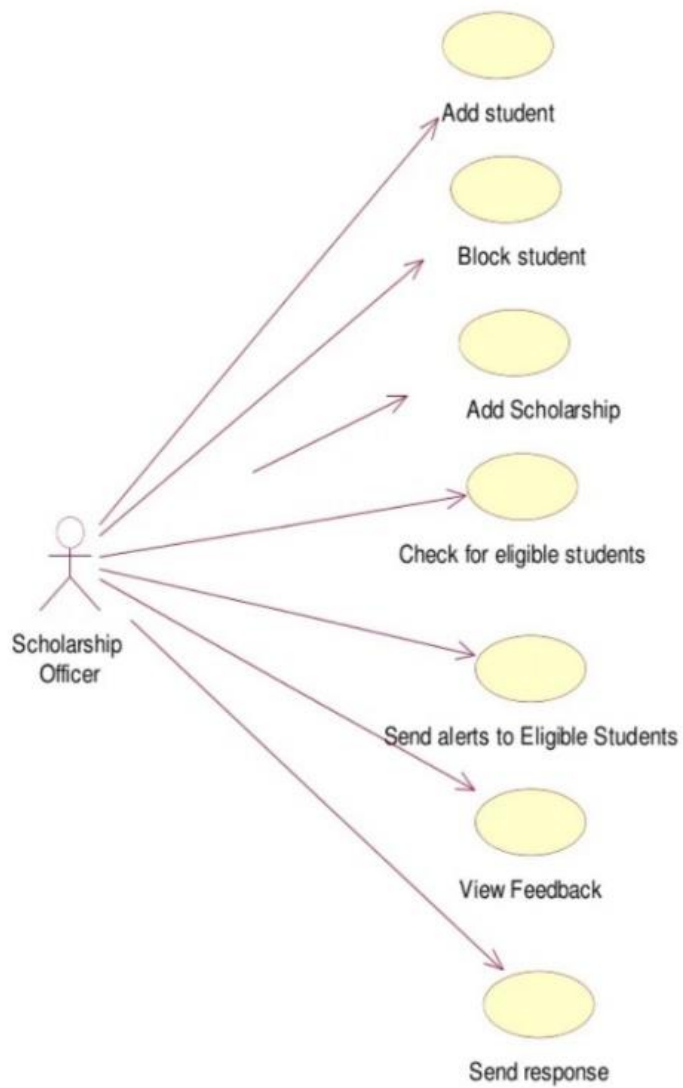
oriented software development method. The Unified Modelling Language is a standard language for Specifying, Visualization, constructing and documenting the artifacts of software system, as well as for business modelling and other no software systems. The UML represents a collection of best engineering practices that have proven successful in the modelling of large and complex systems. The UML very important part of developing objects-oriented software and the software development process. The UML uses mostly graphical notation to express the design of software projects.

## **TYPES OF UML DIAGRAMS**

UML defines nine types of Diagrams. Class(package), Use Case, Sequence, Collaboration, State Chart, Activity, Component and Deployment.

### **2.4.1 USE CASE DIAGRAM**

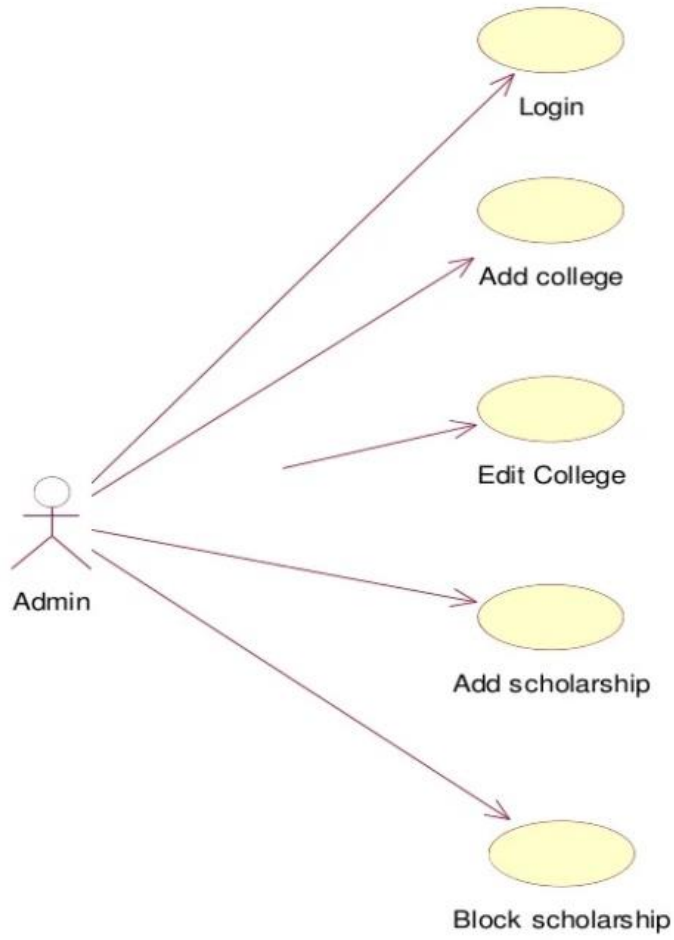
A use case diagram is a graph of actors, a set of use cases enclosed by a system boundary, communication (participation) associations between the actor and users and generalization among use cases. The use case model defines the outside (actors) and inside (use case) of the system's behavior.



*Figure 5 Use case diagram-1*

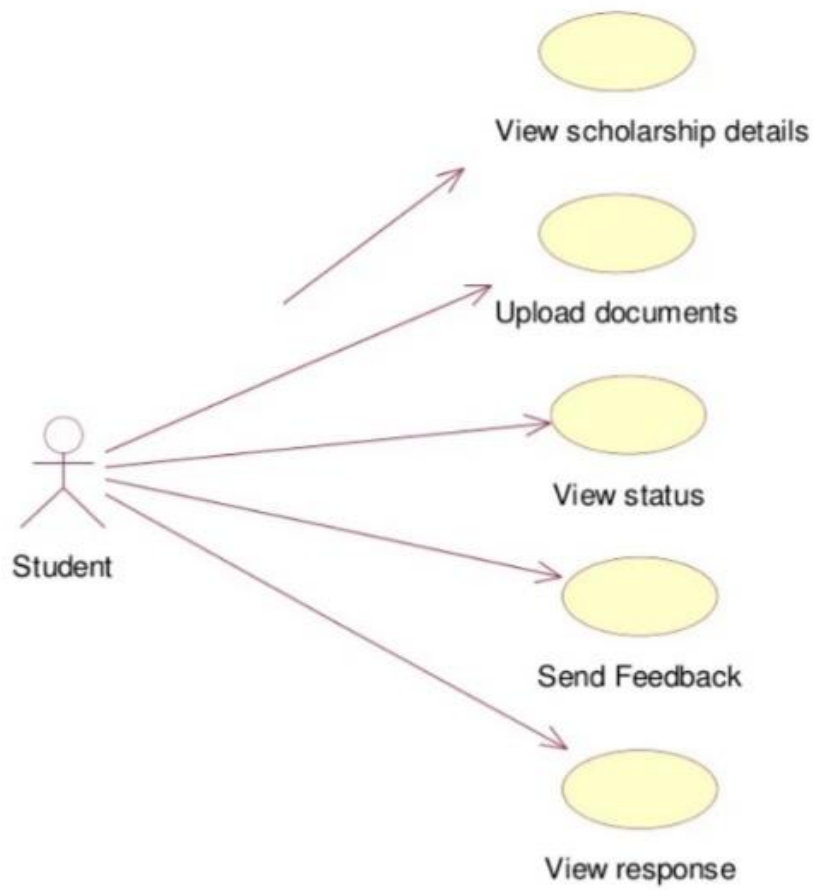
Student:

**Admin:**



*Figure 6 Use case diagram-2*

**Scholarship Officer:**



*Figure 7 Use case diagram-3*

**Class diagram:**

## 2.4.2 CLASS DIAGRAM

UML Class diagram shows the static structure of the model. The class diagram is a collection of static modelling elements, such as classes and their relationships, connected as a graph to each other and to their contents. Public visibility allows all other classes to view the marked information. Protected visibility allows child classes to access information they inherited from a parent class. Private visibility hides information from anything outside the class partition.

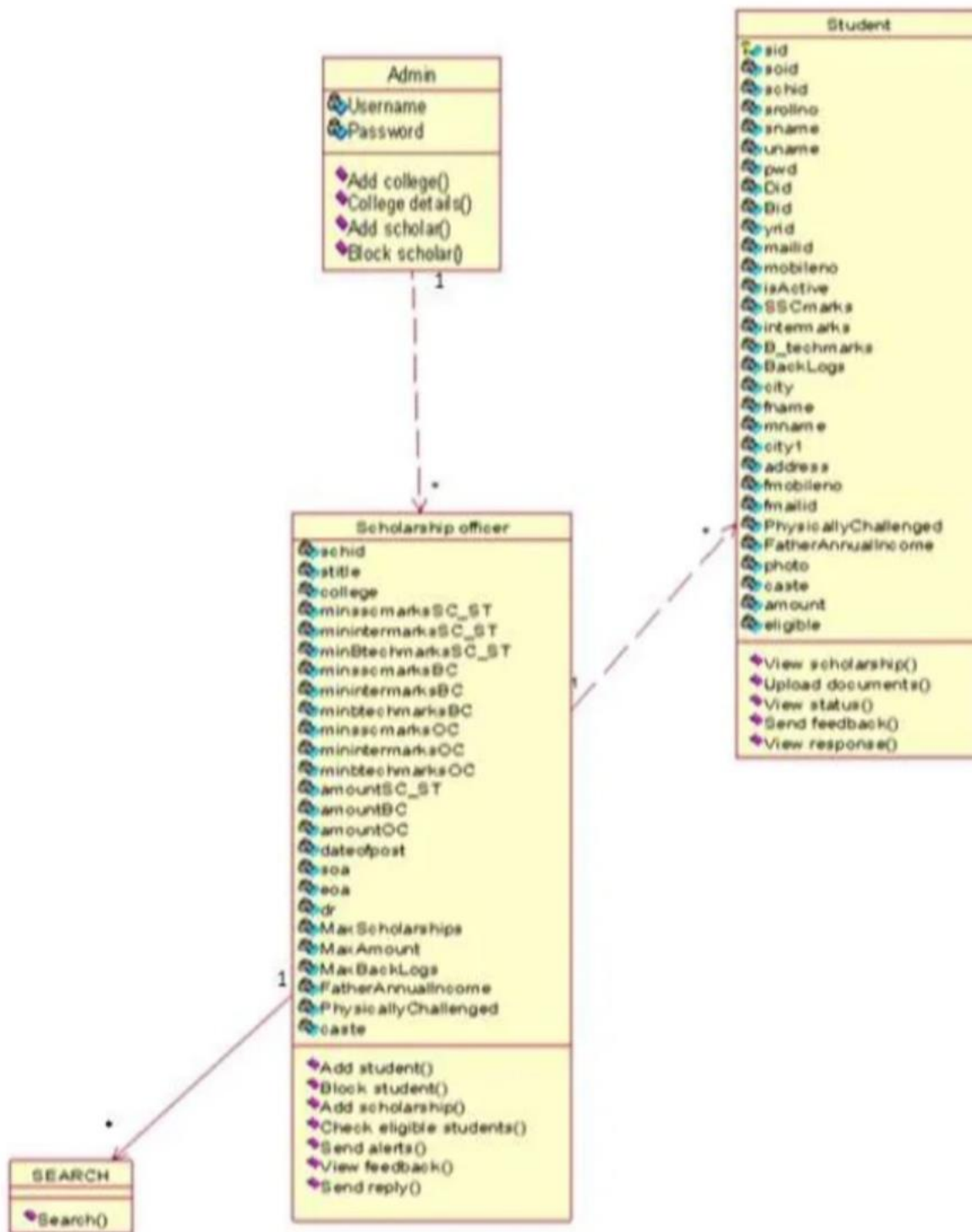


Figure 8 Class diagram

### 2.4.3 SEQUENCE DIAGRAM

Sequence diagram are an easy and intuitive way of describing the behaviour of a system by viewing the interaction between the system and its environment. A Sequence diagram shows an interaction arranged in a time sequence. A sequence diagram has two dimensions: vertical dimension represents time; the horizontal Dimension represents different objects. The vertical line is called is the object’s life line.

Admin:

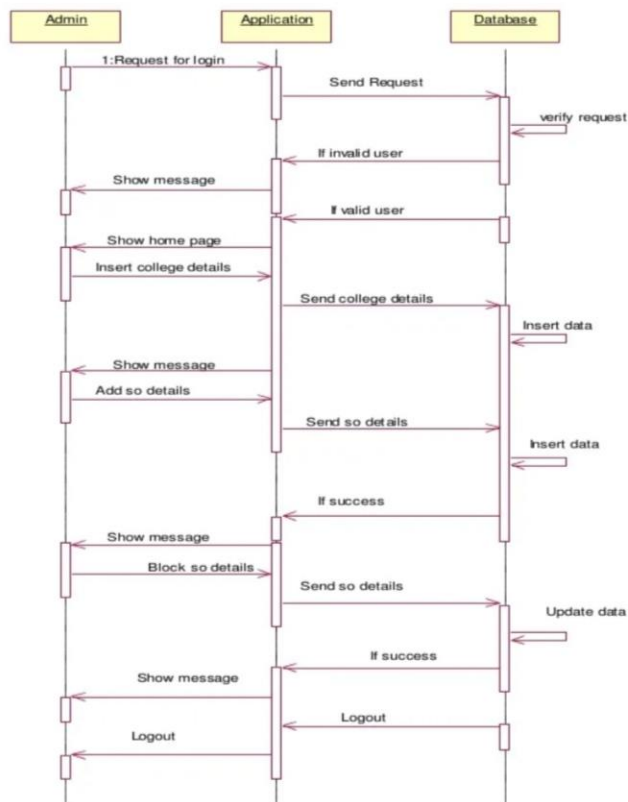


Figure 9 Sequence diagram-1

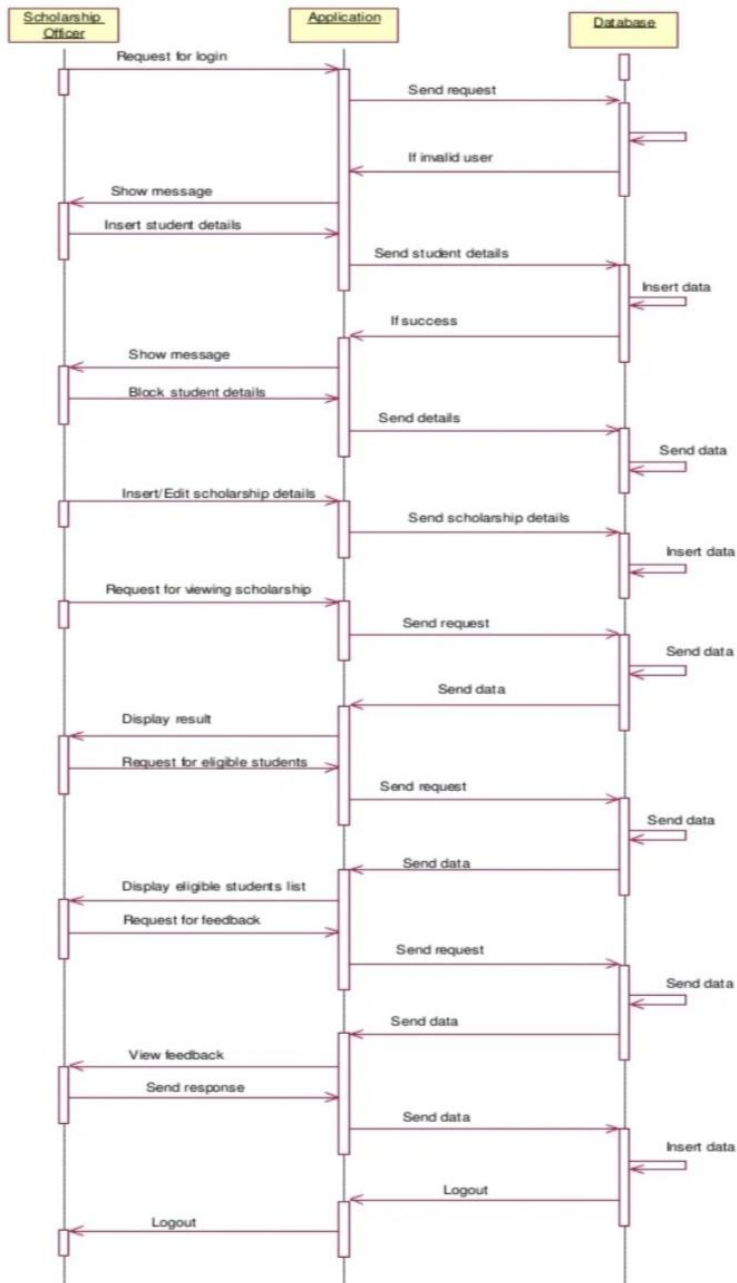


Figure 10 Sequence diagram-2

Student:



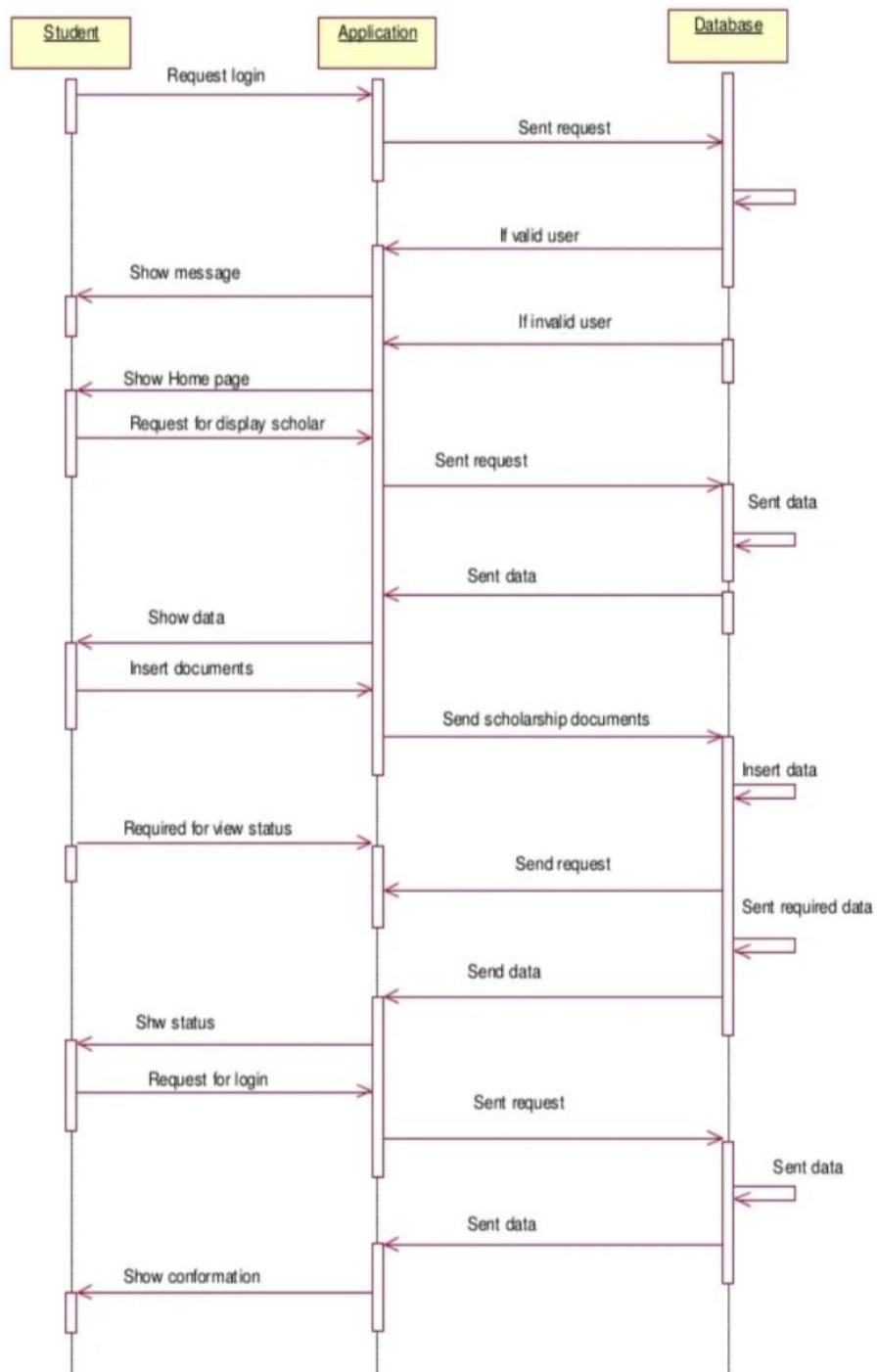


Figure 11 Sequence diagram-3

#### 2.4.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the UML activity diagrams can be used to describe the business and operational step by-step workflows of components in a system. An activity diagram shows the overall flow of control. An activity is shown as a rounded box containing the name of the operation

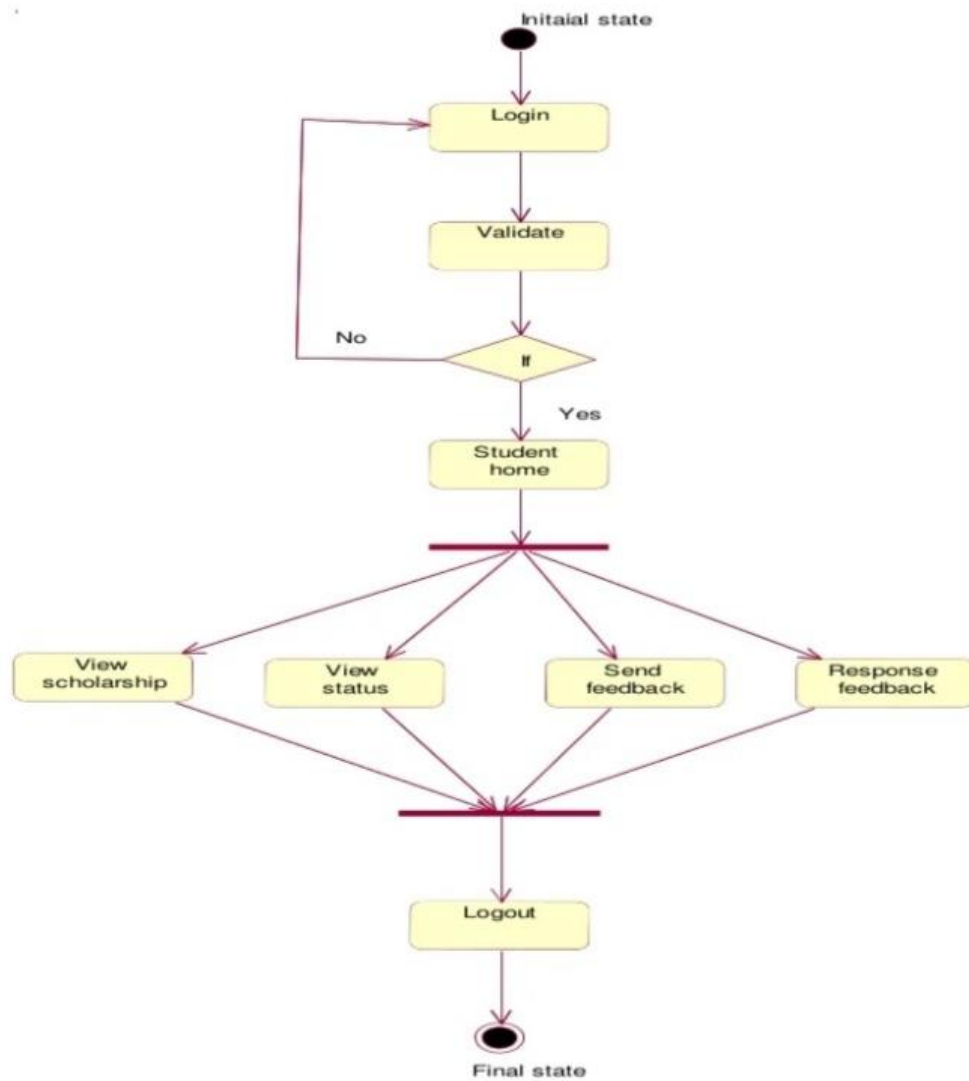


Figure 12 Activity diagram-1

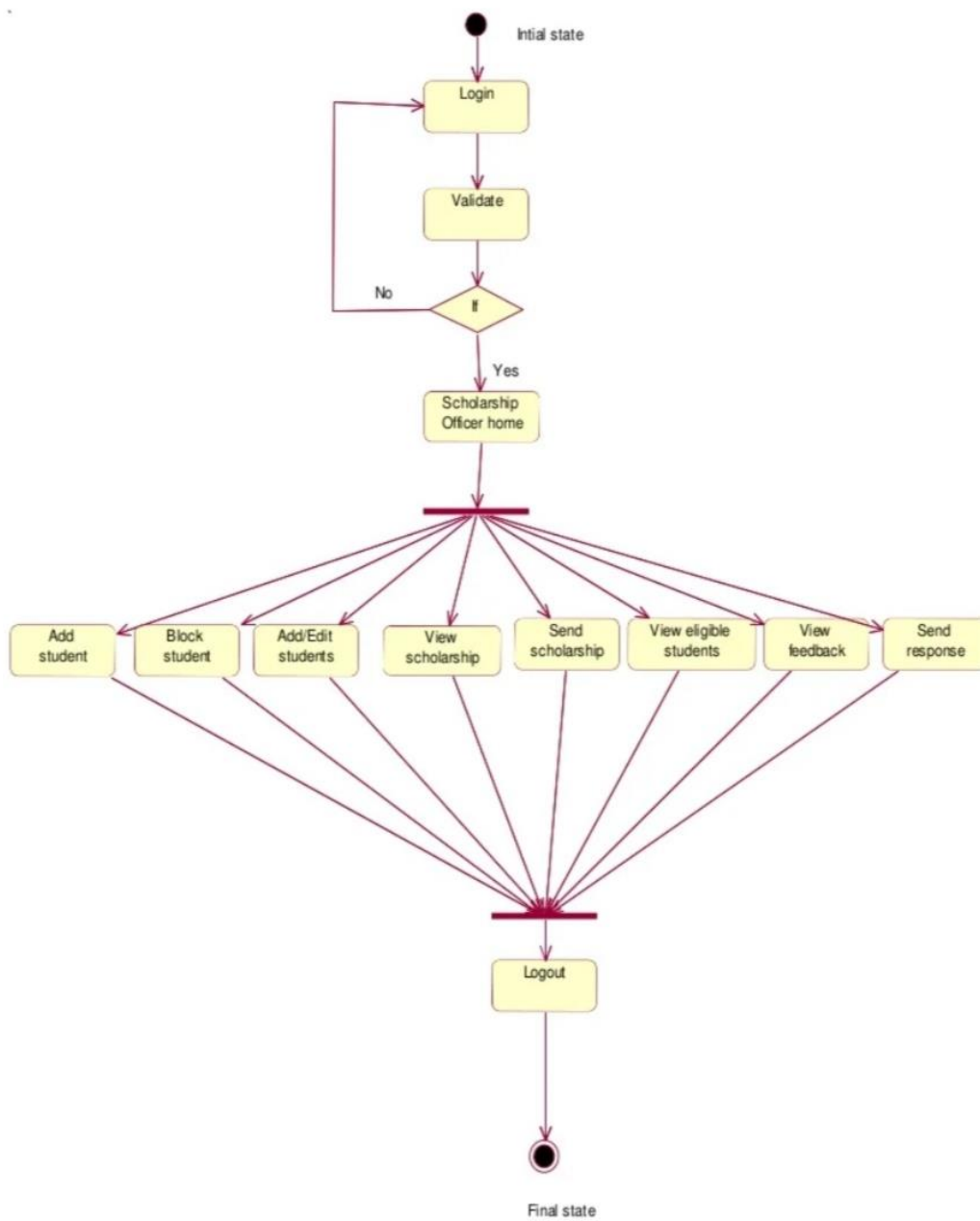


Figure 13 Activity diagram-2

Student:

## Admin:

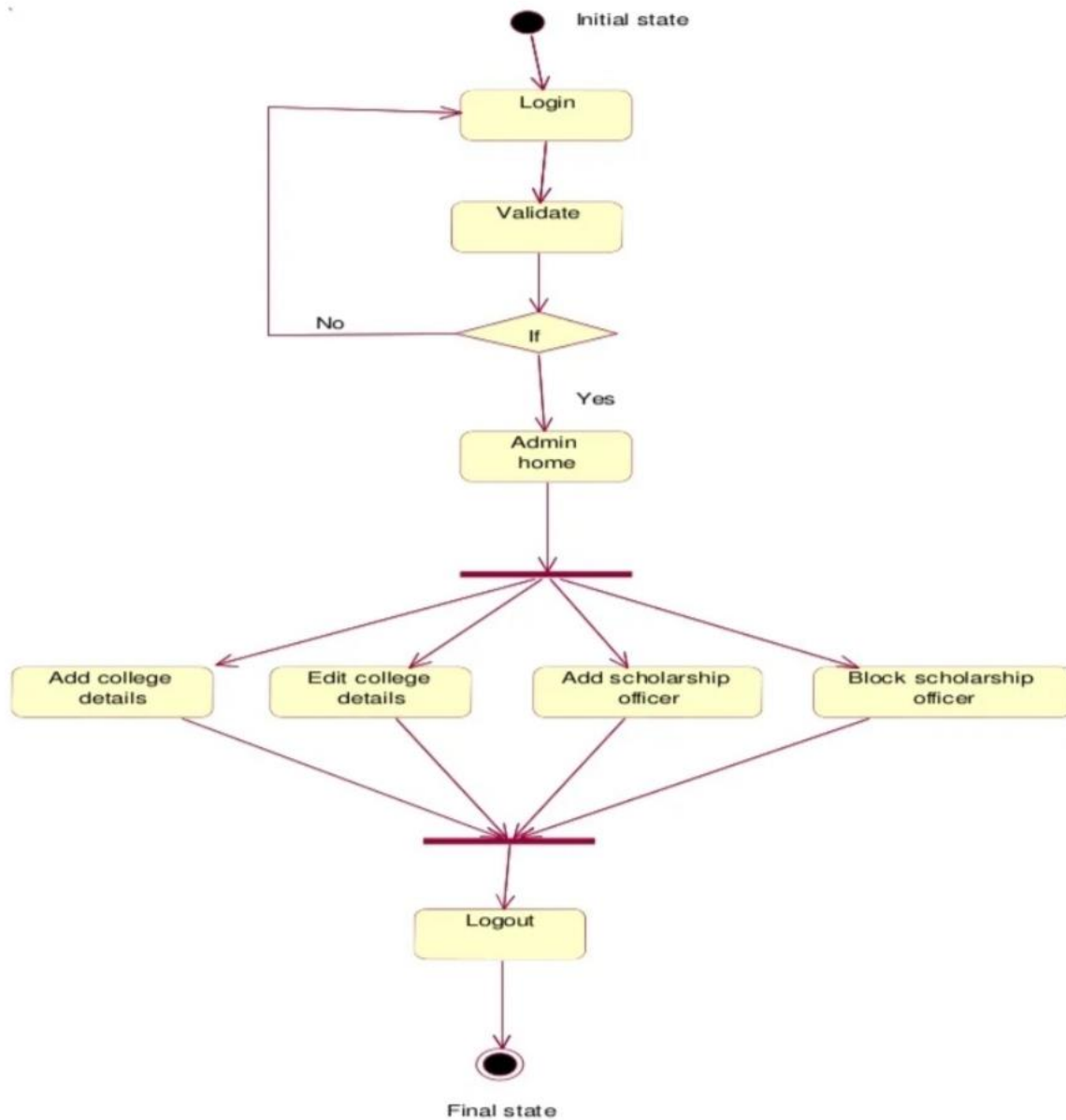


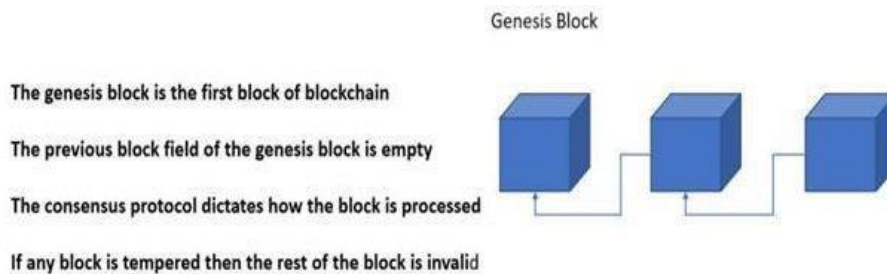
Figure 14 Activity diagram-3

## Scholarship Officer:

## 2.5 Key Concepts of Blockchain

### A Block in Blockchain

One part of Blockchain is individual block that holds the data part of the Blockchain. This data could include any type of data like the record of the transaction, a document or any kind of business-related data. The next part of the block is the hash part which holds the hash of the block. The third part of the block is the hash of the previous block. This assures the authenticity of previous blocks and makes sure that the current block is connected to the previous chain. Now, if anyone tampers with any block of the chain the hash of blocks will change and the intrusion will be detected. Genesis Block is the first block of the chain. It doesn't have any data in it.



*Figure 15 Blockchain*

### Genesis Block

The creation of blockchain requires Genesis Block and also some rules to obey in order to add blocks to the chain. This is known as the consensus protocol. This tells us how to process each block in the blockchain. If any block is tampered with then all blocks become invalid. The only way to invalidate a whole chain is to tamper with a block and recalculate all the later blocks.

## **Attack (51% attack)**

The Blockchain is vulnerable to a famous attack called which is 51% attack. In this attack if an attacker invalidate more than 50% of the Blockchain nodes, then he will have successfully gained control of the network.

## **Consensus Algorithm**

To tackle the problem of 51% attack there is something called proof of work. In proof of work, you calculate the hash of the block according to some rules. The hash should have some specific number of zeros which is called difficulty or it should start with at least 'x' number of characters or numbers. This makes the process of generating the hash more difficult and requires more CPU-time because you have to try different key combinations until you reach an agreeable block and then you broadcast that key with the block so that the other nodes in a network can validate it.

## **Mining**

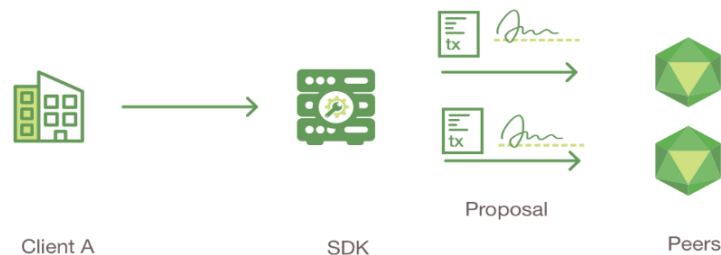
There are two types of nodes in the blockchain i.e. miners and full nodes. Miners are the particular nodes that validate the transaction and contribute to the processing of the blockchain. These nodes are called the mining nodes. The concept of mining in the blockchain is where the nodes receive the block, by using a random key calculates the hash of the block until the difficulty is reached. Once the hash is calculated then we can say that we have successfully mined the block. The block is then broadcast the network. As a reward, the miner gets some currency for its contribution to the network.

## **Smart Contracts**

Smart contracts are one of the biggest features that Blockchain gives. A smart contract is an auto-executable code that executes when certain pre-defined conditions are met. The idea behind the smart contracts is the contractual governance of transactions between two or more participants. Smart contracts are stored distributed and are not held by one single central authority. Also, smart contracts enable users to control ownership by offering controlled data exposure.

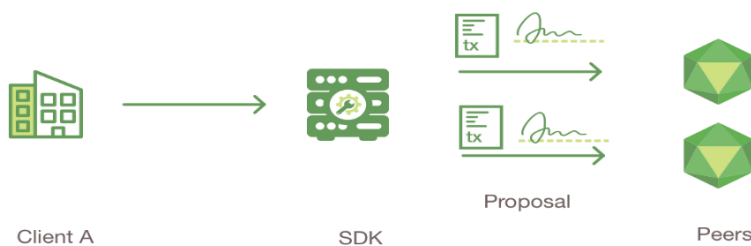
## Transaction Workflow

The basic transactional workflow follows six steps generally that are explained below:



*Figure 16 Client initiates transaction*

The application is installed at the client and is used to send a request to initiate a transaction. Initiating a transaction means to invoke a code that is already written as a function in the chain-code. The client may query already present data in the blockchain or may add some data to the chain. This request is sent to multiple peers for the purpose of verification. The application installed at the client generates a transactional proposal and sends it to the endorsing peers.



*Figure 17 Transaction verification*

On receiving the transaction proposal, the endorsing peers verify the structure of the transaction. It must follow the syntactical rules and make sure that the required parameters are available. Secondly, it makes sure that this isn't a replay of an already submitted request. After that, it verifies the signature of the issuer. This signature authenticates the proposer on the network as well as on the channel it's going to perform the transaction on.





*Figure 18 Inspection of the proposal response*

The response of the endorsing peers is then sent back to the client. Now it's the client's turn to verify the signing of the endorsing peers to keep out any suspicious signatures. It compares the responses of proposals from both the peers to make sure they are the same. If the transaction was querying a data, then this proposal will not be sent to the ordering service, but if it's updating the ledger, then it'll be sent to the ordering service to order all the transactions it receives.



*Figure 19 Transaction of respective channel*

Now the client packs all the responses into a transaction and forwards it to the Ordering service. The transaction package will contain all the read/write sets that are key-value pairs of data that are being read or written by the transactions in process.

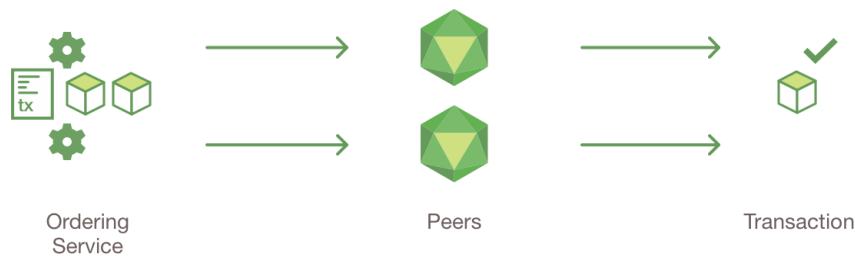


Figure 20 Broadcast transactions

After that, the transactions are sent to all the peers of the network in order to finalize and commit the requests made by the client initially. It is sent to all the peers to make sure the blockchain resides at different places and is free from non-repudiation. These transaction blocks are marked as valid/invalid after multiple inspections. One of the most important checks is to ensure that the data is not contaminated during the process of initialization and verification. Finally, the distributed ledger of the blockchain on all nodes is updated. Every peer writes the data to its blockchain and also updates the state database.

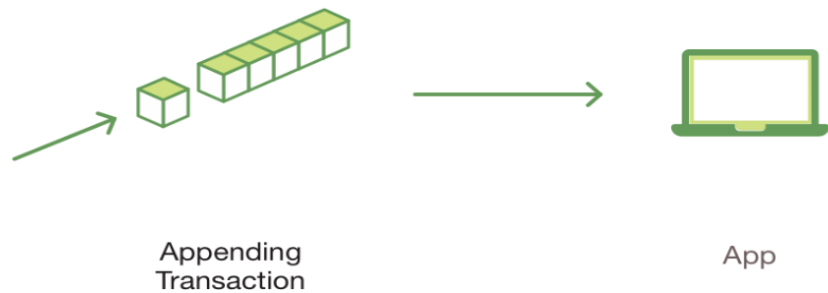


Figure 21 Ledger is updated

## **CHAPTER 3**

### **3.1 MODULE**

There are three modules used in this project.

- Registration module
- Authorization module
- Transaction module

### **3.2 MODULE DESCRIPTION**

#### **3.2.1 Registration module**

Student details need to registered. Data is collected and stored in database. Then wallet was created.

#### **3.2.2 Authorization module**

Admin need to login and verify the details of the students. Then give permission to proceed.

#### **3.2.3 Transaction module**

Smart contract automatically makes a transaction based on contract. Scholarship provider will transact amount to student. Transaction will be stored in blockchain using SHA256.

### 3.1 SMARTCON: A SMART CONTRACT MANAGEMENT SYSTEM

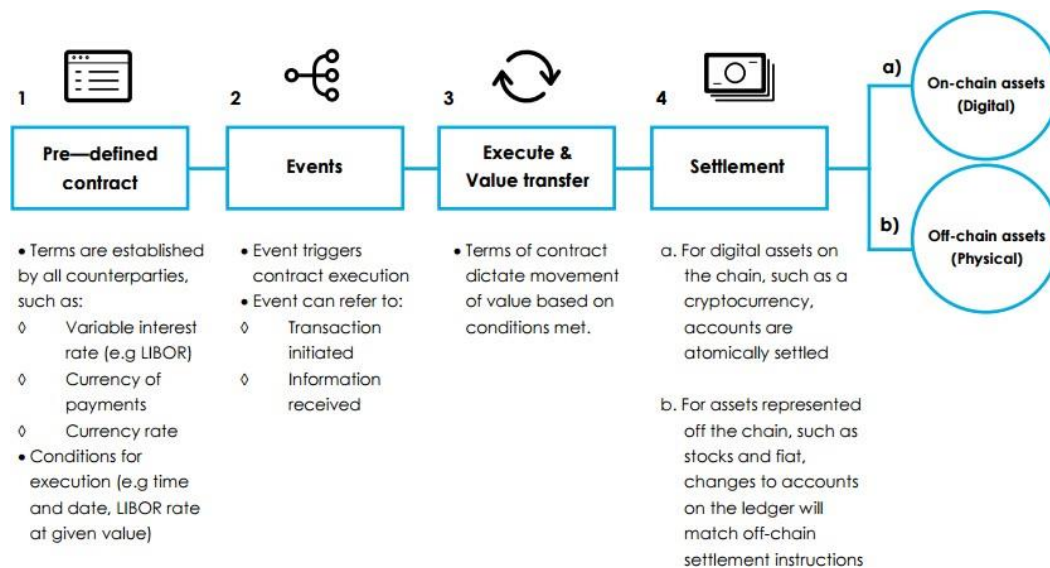
Smart Contract Management System is a blockchain based technology foreseen to automate cross-organizational business processes. Disruptive technologies such as Cloud, IoT, Blockchain, and Data Analytics have raised new automation requirements in different Industries. Contract Management in B2B environment needs to be overhauled as well. Smart Contract Management System is Blockchain based innovative technology foreseen to automate future Business processes. Blockchain has brought in business process re-engineering by optimizing the business workflow activities, especially in multi-party arrangements. Smart Contract is a mechanism to automate B2B processes are triggered by some events generated through IoT devices, data feeds or other applications. Previously, paper contracts are being used for this purpose and third parties were required to ensure trust among the related organizations which are not really an efficient way of doing business as such processes could take a lot of time and there is a lot of paperwork involved.

The code and the agreements contained within exist across a distributed, tamper-proof, decentralized blockchain network.

#### **Characteristics of Smart Contracts**

Smart contracts can track performance in real time and can bring huge cost savings. Agreement and controlling happen on the fly. A smart contract needs some information sources, which feed the smart contract with some essential information.

- Smart Contracts are
- Self-executing
- Self-verifying
- Tamper Resistant Smart Contracts can
- Turn legal commitments into an automated process.
- Ensures a greater level of security.
- Eliminate reliance on trusted 3rd party intermediaries.
- Lower transaction costs.



*Figure 22 Smart Contract Working*

## Types of Smart Contracts

We have identified three types of smart contract that can cover almost all the domains of the real world.

One to one: This is the smart contract that exist between one person and the other person or between one device and the other device. Only one of these person/device can execute the smart contract.

Example.

Suppose there is a smart contract between two organizations if one organizations fill full all of the demands that are mentioned in the smartcontract (meets all of the conditions) then some transaction will take place between the creator and executer of smart contract.

One to many: It is the type of smart contract that exist between the person who has created that and the all other person on the network and anyone can execute that smart contract.

Example.

Suppose there is smart contract that says if you provide me hundred dollars I will provide you the key of some specific software.

Many to many: This is the type of smart contract in which there are more than two participants are involved like there can more than two organizations involved in that smart contract and for smart contract to be executed all of the conditions must be full filled that written in the smart contract.

We have implemented the first two types of smart contracts as a demo. Users can create a smart contract with one party. Or a smart contract is created as a public contract with which any user of the network can interact. According to the use case, one of these three types of smart contracts will be used.

### **How to Access Smart Contract?**

Each smart contract will be stored on the Block Chain. We have created two Block Chains one is used for the storage of the smart contracts and the other is used for the storage of the transactions. Each of the smart contracts will be stored in the one Block.

There is a class S-Block for Smart Contract storage and that class has all the attributes that are used to specify the smart contract. There is an attribute S-file which will store the smart contract file that file will be executed when a specific smart contract is executed.

There are two modes of execution of smart contracts.

Self-execution: In self-execution mode, we need a continuous input feed from the one device or two devices. That input feed will be checked against all the conditions in the smart contract and based on the smart contract will be executed.

Access mode: In Access mode, the smart contract will provide its input field and description on the web and the user can execute the smart contract by accessing the smart contract.

## **CHAPTER 4**

### **CODING AND SYSTEM TESTING**

#### **4.1 CODING**

Once the design aspect of the system finalizes the system enters into the coding and testing phase. The coding phase brings the actual system into action by converting the design of the system into the code in a given programming language. Therefore, a good coding style has to be taken whenever changes are required it should be easily screwed into the system.

#### **4.2 DEVELOPING METHODOLOGIES**

The test process is initiated by developing a compressive plan to test the general functionality and special features on a variety of platform combinations. Strict quality control procedures are used. The process verifies that the application meets the requirements specified in the system requirements document and is bug free. The following are the considerations used to develop the framework from developing the testing methodologies.

#### **4.3 SYSTEM TESTING**

Software testing is a critical element of software quality assurance and represents the ultimate reviews of specification, design and coding. Testing is vital to the success of the system. Errors can be injected at any stage during development. System testing makes logical assumptions that if all the parts of the system are incorrect, it will handle successfully. During testing, the program to be tested is executed with set of data and output of the program for the test data is evaluated to determine if the programs are performing as expected. A series of testing are performed for the proposed system before the system is ready for user acceptance testing.

The testing are

- Unit testing
- Functional testing
- Integration testing
- Validation testing
- Output testing
- User acceptance testing
- White box testing
- Black box testing

### 4.3.1 Unit Testing

In this different test modules are tested against the specification of the modules. Unit testing was done for the verification of the code produced during the coding phase to test the internal logic or modules. It refers to the verification of the single program module in installed environment.

### 4.3.2 Functional Testing

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals. Functional is spotlighted at 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 output must be exercised. Systems/Procedures: Interfacing systems or procedures must be invoked. Two types of testing in Functional test:

- Performance testing
- Structure testing

#### 4.3.2.1 PERFORMANCE TESTING

It determines the amount of execution time spent in various parts of the unit, program throughput and response time and device utilization by the program unit.



#### 4.3.2.2 STRESS TESTING

It is the test designed to intentionally break the system as unit into many small parts or units.

#### 4.3.3 INTEGRATION TESTING

In this project modules are integrated properly, the emphasis being and testing interfaces between modules. Internal interfaces and external interfaces are tested as each module is incorporated into the structure. This test is designed to uncover errors associated with local or global data structures. It is also designed to verify performance levels established when software design is conducted. Thus, all these modules are combined, verified and the information about the item is properly carried on to the next module and then it is checked.

#### 4.3.4 VALIDATION TESTING

At the culmination of integral testing, software is completely assembled as a package, interfacing errors have been uncovered and corrected, and a final series of software tests validation may begin. Validation can be defined in many ways but a simple definition is that validation succeeds when software function in a manner that can be reasonably expected by the customer.

#### 4.3.5 OUTPUT TESTING

After performing the validation, next step is the output testing of the proposed system. Since no system could be useful if it does not produce the output in the specified formats. The output generator or displayed by the system under consideration is tested for user acceptance by constantly keeping touch with perspective system and user at the time of developing and making changes whenever required.

#### 4.3.6 USER ACCEPTANCE TESTING

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. This is done regarding to the following points.

- Input Screen design
- Output Screen design
- Format of the report and other output

## **CHAPTER 5**

### **5.1 Conclusion**

This chapter concludes all the work that we have done in this project

#### **Achievement of Objective**

As it is clear from the above discussion that our project was research project. In this project their main tasks were.

Integration of smart contract with blockchain.

Shifting linear architecture to blockchain to tree-based architecture that will reduce the chain validation process.

Parallel mining to increase speed of the process of mining.

We proposed architecture for integration of smart contract with blockchain. In the architecture contracts are stored on a blockchain after creating hash of the contract file and they are retrieved from that blockchain. And when smart contract is executed its transactions will be stored on transaction blockchain and a ledger of transactions will be updated on each node of the network

We shifted architecture from linear to tree because of performance issues. In linear blockchain validation will occur in a linear way but in tree architecture blocks will be in parent to child relationship so chain validation speed will increase. In parallel mining we worked on two approaches progressive and interval approach to increase the speed of mining.

Smart contract integration architecture practical implementation was demonstration using unity simulation in which if collision between truck and sensor occurs then contract executed and actual transaction took place.

## **Future Recommendations**

The Smart Contract Management System has accomplished basic architectural design requirements but there is always room for improvement. To make the system more useful and enhanced, some future recommendations are listed below:

Ability to write contracts in browser rather than creating javascript class and storing it locally.

Implementing other modes of contract like one to many and many to many contracts.

The user interface of the system can be improved.

Create custom more user-friendly language to write smart contracts in.

Using different and more efficient consensus algorithms.

This project can be extended by using actual IOT device that are used in industry 4.0.

## Appendix

### Backend Server Code

#### Ledger of Transactions

```
app.get('/Ledger', function (req, res){ var chain = SupariCoin.chain;
var c=[];
for(var i=0;i<chain.length;i++){ c.push(chain[i].transactions);
}
res.render('Ledger', {c});
});
```

#### Making Transactions

```
app.post('/transaction', function (req, res) { const transaction = req.body; blockIndex=
SupariCoin.addTransactionToPendingTransactions(transaction); res.json(
{
message: `Transaction will be added to block with index:
${blockIndex}`
}
);});
app.get('/MakeTransaction',function (req, res){ var node = SupariCoin.nodeUrl;
res.render('MakeTransaction',{qs:req.query,node:node})
});
app.post('/MakeTransaction', urlencodedParser , function (req,res) {
const transaction = SupariCoin.makeNewTransaction(req.body.amount,
req.body.sender, req.body.recipient
);
SupariCoin.addTransactionToPendingTransactions(transaction);const requests = [];
SupariCoin.networkNodes.forEach(networkNode => {
const requestOptions = {
```

```

uri: networkNode + '/transaction',method: 'POST',
body: transaction,json: true
};
requests.push(reqPromise(requestOptions));
});
Promise.all(requests).then(data => { res.render('MakeTransaction',{msg:"Transaction
Broadcasted Successfully"})
}).catch(err => err); });

```

### **Creating and Invoking Smart Contract**

```

app.get('/contract', function (req, res){ res.render('contract',{qs:req.query});
});
app.post('/invoke', urlencodedParser, function (req, res){ sender = req.body.sender;
recipient = req.body.recipient; amount = req.body.amount; console.log(sender,recipient,amount)
const newContract = new Contract('1',Date.now(),amount,sender, recipient);
const transaction = newContract.makeContract();console.log(newContract.makeContract());
if (transaction !== false) { SupariCoin.addTransactionToPendingTransactions(transaction);
const requests = []; SupariCoin.networkNodes.forEach(networkNode => {
const requestOptions = {
uri: networkNode + '/transaction',method: 'POST',
body: transaction,json: true
};
requests.push(reqPromise(requestOptions));
});
Promise.all(requests)
.then(data => { res.render('Success',{msg:"Transaction Broadcasted
Successfully"})

```

```

}).catch(err => err);
}
else {
res.render('Failed',{msg: "The values don't add up"})    });app.post('/contract',
urlencodedParser, function (req, res){
sender = req.get('host'); sender = "http://" + sender;console.log(sender);
const recipient = req.body.recipient; console.log(recipient);
res.render('invoke',{sender:sender,recipient:recipient});
});

```

## Mining

```

app.get('/mine', function (req, res) {
const latestBlock = SupariCoin.getLatestBlock();const prevBlockHash = latestBlock.hash;
const currentBlockData = {
transactions: SupariCoin.pendingTransactions,
index: latestBlock.index + 1
}
const nonce = SupariCoin.proofOfWork(prevBlockHash,currentBlockData);
const blockHash=SupariCoin.hashBlock(prevBlockHash,currentBlockD ata, nonce);
console.log(blockHash);const newBlock=
SupariCoin.creatNewBlock(nonce,prevBlockHash, blockHash) const requests = [];
SupariCoin.networkNodes.forEach(networkNode => {
const requestOptions = {
uri: networkNode + '/add-block',method: 'POST',
body: { newBlock: newBlock },json: true
};
requests.push(reqPromise(requestOptions));
});

```

```
res.redirect('Ledger');  
});
```

0

### **Broadcasting and Adding block**

```
app.post('/add-block', function (req, res) {  
  const block = req.body.newBlock;  
  const latestBlock = SupariCoin.getLatestBlock();  
  
  if ((latestBlock.hash === block.prevBlockHash) && (block.index === latestBlock.index + 1)) {  
    SupariCoin.chain.push(block); SupariCoin.pendingTransactions = [];  
    res.json(  
      {  
        message: 'Add new Block successfully!',newBlock: block  
      }    );  
  } else {  
    res.json(  
      {  
        message: 'Cannot add new Block!',newBlock: block  
      } ); });  
app.get('/register-and-broadcast-node',function(req,res){res.render('blockchain',{qs: req.query})  
})  
app.post('/register-and-broadcast-node',urlencodedParser, function (req, res) {  
  var nodeUrl = req.body; nodeUrl = nodeUrl['url'];  
  if (SupariCoin.networkNodes.indexOf(nodeUrl) == -1) {  
    SupariCoin.networkNodes.push(nodeUrl);  
  }  
}
```

```

const registerNodes = []; SupariCoin.networkNodes.forEach(networkNode => {
const requestOptions = {
uri: networkNode + '/register-node',method: 'POST',
body: { nodeId: nodeId },json: true
};
registerNodes.push(reqPromise(requestOptions));
});
Promise.all(registerNodes)
.then(data => {
const bulkRegisterOptions = {
uri: nodeId + '/register-bulk-nodes',method: 'POST',
body: { networkNodes: [...SupariCoin.networkNodes,SupariCoin.nodeUrl] },
json: true
};
return reqPromise(bulkRegisterOptions);
}).then(data => { res.redirect('blockchain');
}).catch(err => err);}); app.post('/register-node', function (req, res) {
const nodeId = req.body.nodeId;
if ((SupariCoin.networkNodes.indexOf(nodeId) == -1)&& (SupariCoin.nodeUrl !== nodeId))
{ SupariCoin.networkNodes.push(nodeId);
res.json( {
message: 'A node registers successfully!'
} );}
else {
res.json( {
message: 'This node cannot register!'});}
});
app.post('/register-bulk-nodes', function (req, res) {const networkNodes =

```



```
req.body.networkNodes; networkNodes.forEach(nodeUrl => {
if ((SupariCoin.networkNodes.indexOf(nodeUrl) == -1)&& (SupariCoin.nodeUrl !== nodeUrl))
{ SupariCoin.networkNodes.push(nodeUrl);} }); });
```

## Consensus

```
app.get('/consensus', function (req, res) {const requests = [];
SupariCoin.networkNodes.forEach(nodeUrl => {const requestOptions = {
uri: nodeUrl + '/blockchain',method: 'GET',
json: true    }; requests.push(reqPromise(requestOptions));
});
Promise.all(requests)
.then(blockchains => {
const currentChainLength = SupariCoin.chain.length;let maxChainLength =
currentChainLength;
let longestChain = null;
let pendingTransactions = null; blockchains.forEach(blockchain => {
if (blockchain.chain.length > maxChainLength) {maxChainLength = blockchain.chain.length;
longestChain = blockchain.chain; pendingTransactions =
blockchain.pendingTransactions;} });if (!longestChain ||
(longestChain &&
!SupariCoin.isChainValid(longestChain))) {res.json({
message: 'Current chain cannot be replaced!',chain: SupariCoin.chain
});} else if (longestChain && SupariCoin.isChainValid(longestChain)) {

SupariCoin.chain = longestChain; SupariCoin.pendingTransactions =
pendingTransactions;
res.json({
message: 'Chain is updated!',chain: SupariCoin.chain
```

```
});})).catch(err => err);});
```

### **Getting Record of Specific Node**

```
app.get('/address/:address', function (req, res) { var address = req.params.address;
address = 'http://' + address;
const data = SupariCoin.findTransactionsByAddress(address);res.render('record',
{data:data.transactions,address: address});
});
app.post('/address/search', urlencodedParser, function (req,res) { var address =
req.body.address;
res.redirect('/address/' + address);});
```

### **Blockchain Class Code**

#### **Block Class**

```
class Block {
constructor(index, timestamp, nonce, prevBlockHash, hash,transactions) {
this.index = index; this.timestamp = timestamp; this.transactions = transactions;this.nonce =
nonce;
this.hash = hash;
this.prevBlockHash = prevBlockHash; }}
```

```
Blockchain Class class Blockchain { constructor() {
this.chain = []; this.pendingTransactions = [];this.nodeUrl = nodeUrl; this.networkNodes = [];
this.creatNewBlock(100, '0', 'Genesis block'); }creatNewBlock(nonce, prevBlockHash, hash) {
const newBlock = new Block(this.chain.length + 1, Date.now(),
nonce, prevBlockHash,hash,
```

```

this.pendingTransactions); this.pendingTransactions = [];this.chain.push(newBlock);
return newBlock; }getLatestBlock() {
return this.chain[this.chain.length - 1];} makeNewTransaction(amount, sender, recipient) {
const transaction = { amount: amount, sender: sender, recipient: recipient,
id: uuid().split('-').join("")}return transaction;}
addTransactionToPendingTransactions(transaction) {
this.pendingTransactions.push(transaction);
return this.getLatestBlock().index + 1;} hashBlock(prevBlockHash, currentBlock, nonce) {
const data = prevBlockHash + JSON.stringify(currentBlock)
+ nonce;
const hash = sha256(data);return hash;}
proofOfWork(prevBlockHash, currentBlockData) {let nonce = 0;
let hash = this.hashBlock(prevBlockHash, currentBlockData,nonce);
while (hash.substring(0, 2) !== '00') {nonce++;
hash = this.hashBlock(prevBlockHash, currentBlockData,
nonce);
};
return nonce;
}
isChainValid(blockchain) {
const genesisBlock = blockchain[0];if ((genesisBlock.nonce !== 100) ||
(genesisBlock.hash !== 'Genesis block') || (genesisBlock.prevBlockHash !== '0') ||
(genesisBlock.transactions.length !== 0)) {return false;
}
for (let i = 1; i < blockchain.length; i++) { const currentBlock = blockchain[i]; const
previousBlock = blockchain[i - 1];const currentBlockData = {
transactions: currentBlock.transactions,index: currentBlock.index }
const blockHash = this.hashBlock(previousBlock.hash,currentBlockData, currentBlock.nonce);

```

```

if (blockHash.substring(0, 2) !== '00') {return false; }
if (currentBlock.prevBlockHash !== previousBlock.hash)
{
return false; } }return true; }
findTransactionsByAddress(address) {let transactions = []; this.chain.forEach(block => {
block.transactions.forEach(transaction => {
if (transaction.sender === address || transaction.recipient
=== address) {
transactions.push(transaction); }));});let balance = 0; transactions.forEach(transaction => {
if (transaction.sender === address) {balance -= +transaction.amount;
} else if (transaction.recipient === address) {balance += +transaction.amount;});
return {
transactions: transactions,balance: balance}} }

```

#### Contract Class Code

```

class SmartContract{
constructor(index, timestamp, amount, sender, recipient){this.index = index;
this.timestamp = timestamp;this.sender = sender;
this.recipient = recipient;this.amount = amount
}
makeContract(){
var sender = this.sender;
var recipient = this.recipient;var amount = this.amount;
if (amount > 1500) {var transaction = {
amount: amount, sender: sender, recipient: recipient,
id: uuid().split('-').join("") }return transaction;
} else {
return false;} } }

```

## Output screenshot

The screenshot displays a web browser window with the address bar showing 'localhost:3000/home'. The page title is 'Students Scholarship Scheme (3S)'. The content is organized into two columns of forms.

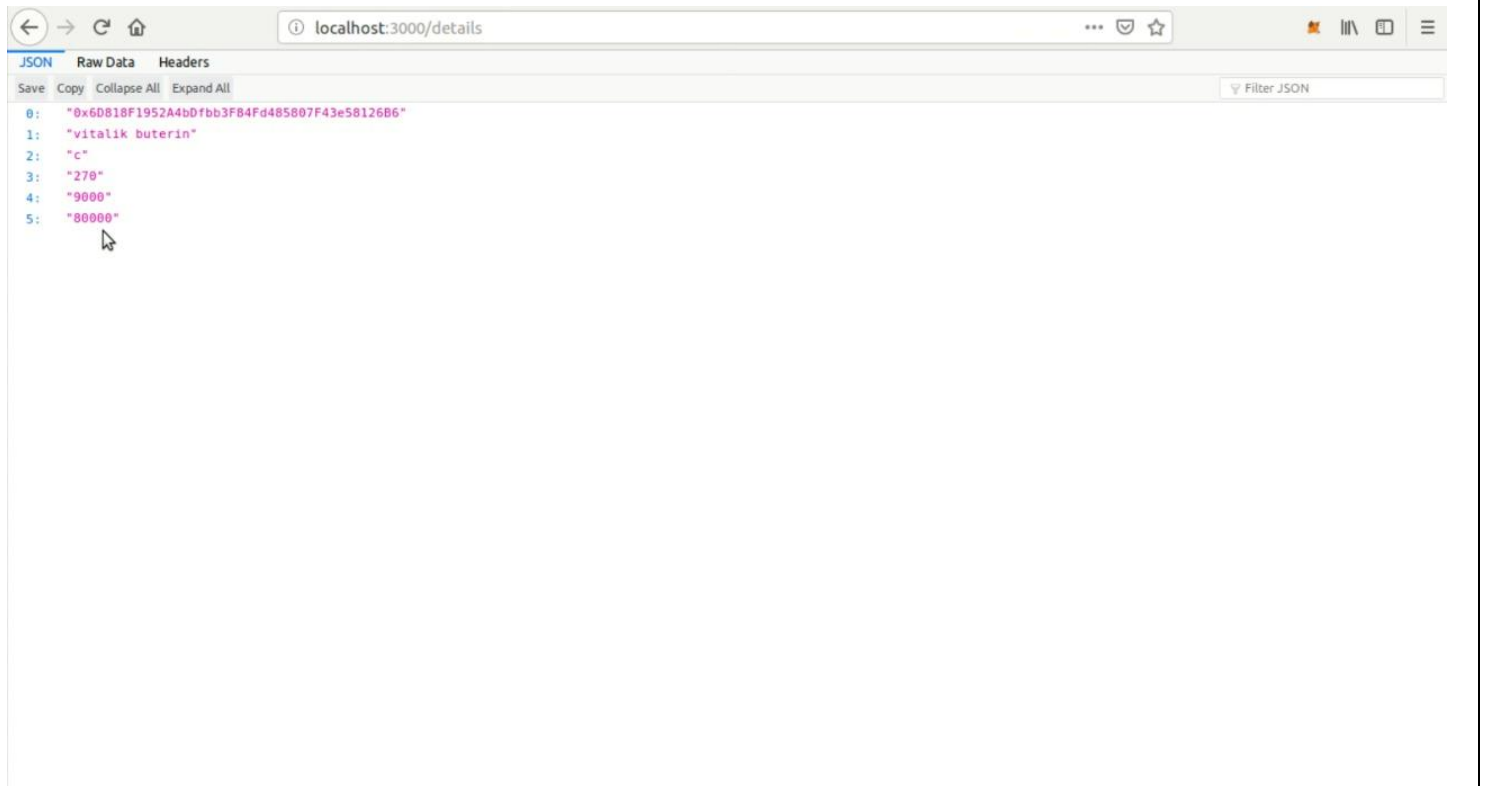
**Left Column:**

- Enter Register Details**
  - Enter User Ethereum Address:
  - Enter Student Ethereum Address:
  - Enter Name:
  - Enter Caste:
- Enter Income Details**
  - Enter Income:
- Enter Marks Details**
  - Enter Mark1:
  - Enter Mark2:

**Right Column:**

- Enter Token Transfer Details**
  - Enter User Ethereum Address:
  - Enter From Ethereum Address:
  - Enter To Ethereum Address:
  - Enter No of Tokens:
  -
- Get Student Details**
  - Enter Student Ethereum Address:
  -
- Get Balance Details**

[illegible]





Private Network



## New Account

Create **Import** Connect

Imported accounts will not be associated with your originally created MetaMask account seedphrase. Learn more about imported accounts [here](#)

Select Type

Private Key

Paste your private key string here:


.....






CANCEL

IMPORT





 Private Network 






Account 2

0x61Fa...035F





99.9354 ETH

\$15,046.28 USD

DEPOSIT

SEND

History

You have no transactions

## REFERENCES

- "Applications of smart-contracts and smart-property utilizing blockchains" Erik Hillbom Tobias Tillström
- "Blockchain Technology and Smart Contracts Privacy-preserving Tools", Jonatan H. Bergquist, 2017
- "Design and Implementation of a Smart Contract Application" Florian Schüpfer, 2017
- "Enabling a decentralized organization through smart contracts and tokens on the Ethereum blockchain" Hung Huy Tran, 2018
- "Trust in Smart Contracts is a Process, As Well" Firas Al Khalil, Tom Butler, Leona O'Brien, and Marcello Ceci
- "A Solution for the Problems of Translation and Transparency in SmartContracts" Firas Al Khalil Marcello Ceci Leona O'Brien & Tom Butler
- T. Economist, "The Great Chain of Being Sure about Things," *Econ. Accedido desde <http://www.Econ.com>*, 2015.
- D. Z. Morris, "Leaderless, Blockchain-Based Venture Capital Fund Raises \$100 Million, And Counting," *Fortune (magazine)*, pp. 5–23, 2016.
- N. Popper, "A venture fund with plenty of virtual capital, but no capitalist," *New York Times*, vol. 21, 2016.
- S. Underwood and Sarah, "Blockchain beyond bitcoin," *Commun. ACM*, vol. 59, no. 11, pp. 15–17, Oct. 2016.
- Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An overview of blockchain technology: Architecture, consensus, and future trends," in *Big Data (BigData Congress), 2017 IEEE International Congress on*, 2017, pp. 557–564.

## References

- A. Kosba, A. Miller, E. Shi, Z. Wen, and C. Papamanthou, “Hawk: TheBlockchain Model of Cryptography and Privacy-Preserving Smart Contracts,” in *2016 IEEE Symposium on Security and Privacy (SP)*, 2016, pp. 839–858.
- C. Cachin, “Blockchain, Cryptography, and Consensus,” *Electron. Proc. Theor. Comput. Sci.*, vol. 261, pp. 1–1, 2017.
- M. Iansiti and K. R. Lakhani, “The truth about blockchain,” *Harv. Bus.Rev.*, vol. 95, no. 1, pp. 118–127, 2017.
- L. S. Sankar, M. Sindhu, and M. Sethumadhavan, “Survey of consensusprotocols on blockchain applications,” in *2017 4th International Conference on Advanced Computing and Communication Systems (ICACCS)*, 2017, pp. 1–5.
- M. Pilkington, “11 Blockchain technology: principles and applications,” *Res. Handb. Digit. Transform.*, p. 225, 2016.
- V. L. Lemieux, “Trusting records: is Blockchain technology the answer?,” *Rec. Manag. J.*, vol. 26, no. 2, pp. 110–139, Jul. 2016.
- S. Nakamoto, “Bitcoin: A Peer-to-Peer Electronic Cash SyNakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System. Consulted, 1–9. doi:10.1007/s10838-008-9062-0stem,” *J. Gen. Philos. Sci.*, vol. 39,no. 1, pp. 53–67, 2008.
- S. Tikhomirov, “Ethereum: State of Knowledge and ResearchPerspectives,” Springer, Cham, 2018, pp. 206–221.
- G. Wood, “Ethereum: A secure decentralised generalised transactionledger,” *Ethereum Proj. Yellow Pap.*, vol. 151, pp. 1–32, 2014.
- S. Omohundro, “Cryptocurrencies, smart contracts, and artificialintelligence,” *AI Matters*, vol. 1, no. 2, pp. 19–21, Dec. 2014.
- M. Valenta and P. Sandner, “Comparison of Ethereum, HyperledgerFabric and Corda,” FSBC Working Paper, 2017.

## References

- E. Androulaki *et al.*, “Hyperledger fabric: a distributed operating system for permissioned blockchains,” 2018, pp. 1–15.
- A. S. Tanenbaum, “Distributed operating systems anno 1992. Whathave we learned so far?,” *Distrib. Syst. Eng.*, vol. 1, no. 1, p. 3, 1993.
- "IOTA The Tangle", Serguei Popov, 2018
- "Smart Contracts: 12 Use Cases for Business & Beyond A Technology, Legal & Regulatory Introduction" — Foreword by Nick Szabo, 2016
- "Smart contracts: Building blocks for digital markets", Nick Szabo
- “Smart Contracts: A Primer”, Matthew N. O. Sadiku, Kelechi Eze, Sarhan M. Musa.