

# **SECURE BANK TRANSACTIONS USING BLOCKCHAIN TECHNOLOGY**

*A Project Report submitted to*

**JNTUA, Ananthapuramu**

*In partial fulfilment of the requirements for the award of the degree of*

**Bachelor of Technology**

**(Computer Science & Engineering)**

By

**A.M. SAI CHARITH (19KB1A0514)**

**A.SARATH BABU (19KB1A0507)**

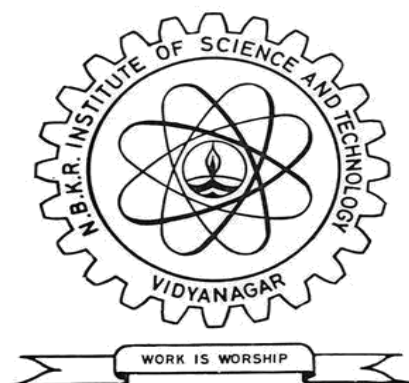
**K.YASWANTH (19KB1A0564)**

**V.LOKESH (20KB5A0501)**

Under the esteemed Guidance of

**Mr. B.SHOWRI RAYALU B.Tech.,M.Tech.,[Ph.D]**

**Assistant Professor, Dept. of CSE**



**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING**

**N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY**

**VIDYANAGAR – 524413, TIRUPATI DIST, AP**

**(AUTONOMOUS)**

**MAY-2023**



Website: [www.nbkrist.org](http://www.nbkrist.org)  
Email : [ist@nbkrist.org](mailto:ist@nbkrist.org)

Ph : 08624-228 247  
Fax: 08624-228 257

**N.B.K.R. INSTITUTE OF SCIENCE & TECHNOLOGY**  
**(Autonomous)**

(Approved by AICTE: Accredited by NBA: Affiliated to JNTUA, Ananthapuramu)

An ISO 9001-2000 Certified Institution

Vidyanagar-524413, Tirupati District, Andhra Pradesh, India

---

**BONAFIDE CERTIFICATE**

This is to certify that the project work entitled “**BANK TRANSACTIONS USING BLOCKCHAIN TECHNOLOGY**” is a bonafide work done by **Mr. A. SARATH BABU (19KB1A0507)**, **Mr. A. M. SAI CHARITH (19KB1A0514)**, **Mr. K. YASWANTH (19KB1A0564)**, **Mr. V. LOKESH (20KB5A0501)** in the department of **Computer Science & Engineering**, **N.B.K.R. Institute of Science & Technology, Vidyanagar** and is submitted to **JNTUA, Ananthapuramu** in the partial fulfilment for the award of B.Tech degree in **Computer Science & Engineering**. This work has been carried out under my supervision.

**Mr . B.SHOWRI RAYALU**

*Asst. Professor  
Department of CSE  
NBKRIST, Vidyanagar*

**Dr A. Raja Sekhar Reddy**

*Professor & Head  
Department of CSE  
NBKRIST, Vidyanagar*

Submitted for the Viva-Voce Examination held on\_\_\_\_\_

**Internal Examiner**

**External Examiner**

## ACKNOWLEDGEMENT

The satisfaction that accompanies the successful completion of a project would be incomplete without the people who made it possible of their consistent guidance and encouragement crowned our efforts with success.

We would like to express my profound sense of gratitude to our project guided **Mr. B.SHOWRI RAYALU Assistant Professor, Department of Computer Science & Engineering, N.B.K.R.I.S.T (Affiliated to JNTUA, Anantapuramu)**, Vidyannagar, for his masterful guidance and the constant encouragement throughout the project. My sincere appreciations for his suggestions and unmatched services without, which this work would have been an unfulfilled dream.

We convey my special thanks to **Dr. Y. VENKATARAMI REDDY, Chairman, N.B.K.R. Institute of Science and Technology**, for providing excellent infrastructure in our campus for the completion of the project.

We convey my special thanks to **Sri. N. RAM KUMAR**, Correspondent of **N.B.K.R. Institute of Science and Technology**, for providing excellent infrastructure in our campus for the completion of the project.

We are grateful to **Dr. V. VIJAYA KUMAR REDDY, Director, N.B.K.R. Institute of Science and Technology** for allowing me to avail all the facilities in the college.

We express my sincere gratitude to **Dr. A. RAJA SEKHAR REDDY, Professor& Head, Department of Computer Science & Engineering**, for providing exceptional facilities for successful completion of my project.

We would like to convey my heartfelt thanks to **Staff Members, Lab Technicians, and my friends**, who extend their cooperation in making this project as a successful one.

We would like to thank one and all who have helped me directly and indirectly to complete this project success.

## **ABSTRACT**

Blockchain is a distributed database or a decentralized ledger which is most commonly used to exchange digital currency and perform transactions securely. Every participant of the network has access to the ledger which will be updated by every new transaction. The Blockchain ledger is a collection of all transactions executed in the past. The Blockchain ledger is a continuously growing tamper-proof data structure containing blocks that hold batches of individual transactions. The completed blocks are added in chronological order. Blockchain via Bitcoin has had a massive impact on the world in the past decade and it's safe to say that this will continue, especially with many people working tirelessly to remove the various limitations that are prohibiting blockchains from becoming mainstream. One such limitation is the high processing and electrical costs that come from the Proof-of-Work consensus protocol. In this paper, we propose an alternative proof-by-approval protocol which is a more advanced form of the proof-of-reputation protocol, that offers better security and is a more decentralized approach than the former at the cost of being less performant and harder to setup.

## **TABLE OF CONTENTS**

<b>CHAPTER NO</b>	<b>CHAPTER NAME</b>	<b>PAGE NO</b>
ACKNOWLEDGEMENT		
ABSTRACT		
<b>CHAPTER 1:</b>	<b>INTRODUCTION</b>	<b>1-4</b>
	1.1 Introduction	1-2
	1.2 Motivation	3
	1.3 Aim & Objective	3
	1.4 Scheme of Chapter	4
<b>CHAPTER 2:</b>	<b>LITERATURE SURVEY</b>	<b>5-8</b>
	2.1 Introduction	5
	2.2 Literature Survey	5-6
	2.3 Existing System	7
	2.4 Proposed System	7-8
	2.5 Feasibility Analysis	8
<b>CHAPTER 3:</b>	<b>DESIGN ISSUES</b>	<b>9-14</b>
	3.1 System Design	9-11
	3.1.1 Description about Modules	10-11
	3.2 UML Diagrams	11-14
	3.2.1 Use-Case Diagram	12
	3.2.2 Activity Diagram	12-13
	3.2.3 Class Diagram	14
<b>CHAPTER 4:</b>	<b>IMPLEMENTATION DETAILS</b>	<b>15-37</b>
	4.1 Introduction	15
	4.2 Requirements	16-17
	4.2.1 Hardware Requirements	16
	4.2.2 Software Requirements	16-17

	4.3 Source Code	17-32
	4.4 Test Plan	33-35
	4.4.1 Test Procedure	34-34
	4.4.2 Test Cases	34-35
	4.5 Inputs/Outputs	35-37
<b>CHAPTER 5:</b>	<b>CONCLUSION &amp; FUTURE ENHANCEMENTS</b>	38
	5.1 Conclusion	38
	5.2 Future Enhancements	38
	<b>BIBLIOGRAPHY</b>	39-40
	-References	39
	-Text Books	40
	-Websites	40

## 1.1 INTRODUCTION

With nations moving toward digital transactions and the internet being as centralized as it is right now, many of our net-based activity often requires us to place a good deal of trust in various organizations. While this model of operation has served us well for many years, it does come with some flaws, the most obvious of which is that our data is usually at the mercy of the centralized authority to whom we conform with. Blockchain solved this issue by providing a decentralized peer-to-peer network that allows its users to carry out transactions and interactions without having to place trust in each other or a centralized authority, the users of the network only need to place trust in the globally accepted mechanism of the system, Blockchain stores the users' data in a distributed immutable ledger.

The mechanism of operation of the blockchain ensures that any data that is put in the ledger is permanently embedded into the ledger and cannot be changed. This is accomplished by making all the data in the ledger cryptographically linked to each other and by having the ledger distributed across a large peer-to-peer network where each node both validates and passes the ledger to fellow nodes, and where the network is able to quickly identify any illegitimate ledgers sent by an attacker node and quickly diffuse it, this procedure is done using a protocol called the consensus protocol.

There are many different consensus protocols developed for the Blockchain, the one that is most commonly used and accepted is the Proof-of-Work consensus protocol, this is also the protocol that is used by Bitcoin and Ethereum (although Ethereum is moving towards the Proof-of-Stake protocol). The protocol works by requiring anyone who wishes to add data to the ledger to solve a cryptographic puzzle to do so. The cryptographic puzzle, in turn, is designed to be very difficult to solve, meaning that anyone who wants to put a block on the blockchain would have invested in vast amount of resources to do so, thereby repelling any ideas of malicious activity. This process of solving a complex cryptographic puzzle to put data on the blockchain is referred to as mining.

Bitcoin was the first to pioneer the PoW protocol and has used it effectively for more than a decade now. But mining is an incredibly inefficient process which consumes an excessive amount of power. Studies have shown mining can consume as much as 75 TeraWatt Hours of power in a single year. For context, the country of Switzerland

consumes about 58 TeraWatt Hours of power in a year. It is for this specific reason that modern Blockchains struggle to see adoption at a large scale. To tackle this disadvantage, many alternate consensus protocols have been proposed including the proof-by-stake protocol. One other such example is the Proof-of-Reputation protocol.

In this protocol, the act of adding a block to the Blockchain can only be done by a verified group of users, these users are usually large companies or the like who have a strong reputation to maintain and would not risk losing this reputation by adding a fraudulent block to the blockchain. An example of a PoR blockchain is the GoChain. The PoR protocol works especially well in private or permissioned networks, for example, a company could deploy a PoR blockchain to maintain a record of public information amongst all of its employees without risk of being compromised while authorizing specific users to verify and add blocks to the Blockchain. Though this approach does start to dwindle a bit towards the centralized system that Blockchain is ultimately trying to work against, as any data has to be validated and thus passed to a centralized group of users who may then again be managed by a central before it can enter the blockchain, the primary rules of the Blockchain still haven't changed, once data has entered the chain, it cannot be changed by either the authority managing the permissioned users or the user who verified and put that block on the chain.

But centralized control over who has write privileges over the chain while useful in a private organization can be limiting in certain other cases, for example, what about a government office, which takes up the task of approving and managing various extremely important documents that concern a large number of citizens. If a citizen wishes to get a certain document from the office and the office wishes to make record of this, a Proof-of-Reputation based Chain could be used to accomplish this task but it also puts all authority in the hands of the employees of the Government office, which is not ideal as citizens should be allowed to have authority over the documents that pertain directly to them. To accomplish this, this paper presents a variant of the PoR protocol, called the Proof-by-Approval protocol that is designed to handle this task.



## **1.2 MOTIVATION**

The main motivation of the secure bank transactions using blockchain technology:

- The motivation for secure bank transactions using blockchain is to create a more secure efficient and cost-effective solutions for industrial banking. The traditional banking system is plagued by number of challenges, including slow transactions speed, high cost, lack of transparency, and inadequate security measures.
- By leveraging the power of blockchain technology, banks and other financial institutions can overcome these challenges and provide a more secure and efficient solution for their customers. With blockchain, transactions are processed much faster as there is no need for intermediaries to verify them. This not only saves time, but also reduces costs as intermediaries are eliminated.

## **1.3 AIM AND OBJECTIVE**

The Indian banking sector is now dealing with problems including increased operational expenses, growing susceptibility to fraudulent assaults on centralised infrastructure, and difficulties maintaining transparency. All of this is primarily due to the fact that the majority of banking transactions often involve time-consuming manual processing and documentation, expensive intermediaries, and delays caused by the need for multiple participants to validate the transactions at various points in time. As a result, there is almost never a real-time solution that is fraud proof. The objective of using blockchain for secure bank transactions is to offer a secure, open, and decentralized ledger for recording and validating financial transactions is the primary goal of adopting blockchain for secure bank transactions. The another objective of the paper is to create an application where one could use a decentralized network to make transactions between centralized systems.

## **1.4 SCHEME OF CHAPTER**

Most banking transactions may require intensive manual processing and documentation, involve costly intermediaries, and are time-consuming as these transactions need to be validated by various participants at various points in time causing delay thereby resulting in almost no fraud-proof real-time solution. Blockchain technology introduces a fundamentally different, decentralized structure into payment systems, with cryptography rather than a central clearing institution as to its very basis, and without intermediaries such as banks. The Indian banking industry today is struggling with issues such as rising costs of operations, increasing susceptibility to fraudulent attacks on centralized servers, and challenges in ensuring transparency.

## 2.1 INTRODUCTION

Blockchain technology is a core, underlying technology with promising application prospects in the banking industry. On one hand, the banking industry is facing the impact of interest rate liberalization and profit decline caused by the narrowing interest-rate spread.

## 2.2 LITERATURE SURVEY

<b>PAPER NAME AND AUTHOR</b>	<b>PUBLISHED /YEAR</b>	<b>PURPOSE</b>
Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto.[1]	2008	This paper introduced the concept of decentralized, peer-to-peer electronic payment system based on blockchain technology
Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform, Vitalik Buterin.[2]	2014	This paper proposed the Ethereum blockchain platform, which supports the development of smart contracts and decentralized applications.
Blockchain Technology: Principles and Applications, Marc Pilkington.[3]	2016	This paper provided an overview of the principles of blockchain technology and explored its potential applications in various fields, including finance, healthcare, and supply chain management.
Blockchain: A Graph Primer, Marc Barros and Ruben Cuevas.[4]	2016	This paper presented a graph-theoretical approach to understanding the structure and properties of blockchains
Blockchain application and outlook in the banking industry, Ye Guo and Chen Liang.[5]	2016	In this paper, Blockchains could revolutionize the underlying technology of the payment clearing and credit information systems in

		banks, thus upgrading and transforming them.
Blockchain and Distributed Ledger Technology: The Challenge of Scalability, Vigna and Casey.[6]	2017	This paper explored the scalability issues facing blockchain technology and proposed various solutions.
Blockchain Technology and the Financial Services Market, Krause et al.[7]	2017	The technology could remove trusted third parties, decrease costs and ultimately increase profits for various players within the industry.
Scalability and Performance of Blockchain Systems: A Comprehensive Survey, Li et al. [8]	2018	This paper surveyed the various approaches to improving the scalability and performance of blockchain systems.
Blockchain for Social Impact: Moving Beyond the Hype, Stanford Graduate School of Business.[9]	2019	This paper highlighted the potential of blockchain technology to address social and environmental challenges, and provided case studies of blockchain-based initiatives in areas such as renewable energy and humanitarian aid.
Blockchain Technology: A Review of the Security Challenges and Countermeasures, Hussain et al.[10]	2020	This paper reviewed the security challenges facing blockchain technology, such as consensus attacks and smart contract vulnerabilities, and proposed various countermeasures.

## **2.3 EXISTING SYSTEM**

The Existing system have used the Proof of Work (PoW) consensus algorithm. Proof of work (PoW) is a blockchain-based technology that enables participants to validate the legitimacy of digital transactions by asking them to solve a mathematical challenge. Despite the fact that PoW is frequently employed in cryptocurrencies like Bitcoin, it is not ideal for secure bank transactions due to a number of drawbacks.

Disadvantage:

- High computing power requirement: Participants in PoW must solve challenging mathematical puzzles, which consumes a lot of energy. For banks aiming to save expenses and improve sustainability, this results in excessive energy use.
- Slow transaction processing: The PoW consensus algorithm is made to be deliberately slow to guarantee the network's security. As a result, it may take a while for transactions to be processed, which is unacceptable for banks that aim to execute transactions quickly and effectively.
- Centralization of mining power: PoW mining requires specialized hardware and significant resources, which means that mining power tends to be concentrated in the hands of a few large players. This can lead to centralization of power and potential security risks.

## **2.4 PROPOSED SYSTEM**

The Proposed system have used the proof of approval (PoA) consensus algorithm. Proof of approval (PoA) is a consensus technique that offers a way to verify the legitimacy of digital transactions by necessitating the approval of the transactions by a set of trusted validators. For safe bank transactions, PoA offers a number of benefits over alternative consensus algorithms like proof-of-work (PoW).

Advantages:

- Energy-efficient: PoA is more ecologically and energy-friendly than PoW since it takes significantly less processing power. This is a crucial factor for institutions that want to save expenses and boost sustainability.
- Fast transaction processing: Without any challenging mathematical challenges to solve during the approval process, PoA transactions may be

handled rapidly. It is therefore perfect for banks who want to offer quick and effective transaction processing.

- **Trusted validators:** The PoA consensus technique uses trusted validators to lower attack risk and guarantee network integrity. The danger of centralization and associated security threats are decreased by the careful selection and oversight of the validators.
- **Flexible:** The PoA consensus process is adaptable and may be tailored to a bank's or financial institution's unique requirements. For banks wishing to deploy blockchain-based solutions that are customised to their particular needs, this makes it an appealing alternative.
- **Low cost:** PoA is a low-cost consensus mechanism since it doesn't need expensive mining equipment or a lot of computer power. This makes it a desirable option for banks seeking to save expenses and boost productivity.

## **2.5 FEASIBILITY ANALYSIS**

The usage of blockchain with proof of approval (PoA) for secure bank transactions is an attractive idea that has the potential to offer the financial sector a high level of security, transparency, and efficiency. This approach's feasibility is dependent on a number of elements, including as the regulatory environment, the scalability of the PoA consensus method, and the capacity for integration with current financial systems.

Ultimately, various considerations, including scalability, interaction with current systems, and regulatory compliance, influence whether integrating blockchain with PoA for safe bank transactions is feasible. Even if there are difficulties to be solved, this strategy has promise for the financial industry because of the possible advantages, such as improved security, transparency, and efficiency.

### **3.1 SYSTEM DESIGN**

#### **User Interface:**

This component provides a user-friendly interface for users to interact with the system. It includes screens for login, account creation, balance updates, and transaction history.

#### **Authentication:**

This component is responsible for authenticating users before allowing them to access the system. It ensures that only authorized users can perform transactions on the blockchain.

#### **Account Management:**

This component is responsible for managing user accounts, including account creation, balance updates, and transaction history.

#### **Transaction Processing:**

This component is responsible for processing transactions on the blockchain. It validates transactions and ensures that they are recorded on the blockchain ledger.

#### **Blockchain Ledger:**

This component is responsible for creating and maintaining the blockchain ledger. It ensures that the ledger is tamper-proof and secure.

#### **Encryption and Security:**

This component is responsible for encrypting user data and ensuring that it is secure. It also ensures that the system is protected against hacking and other security threats.

#### **Application Programming Interface (API):**

This component provides a communication interface between the user interface and the backend components of the system. It allows users to perform transactions and view their account information.

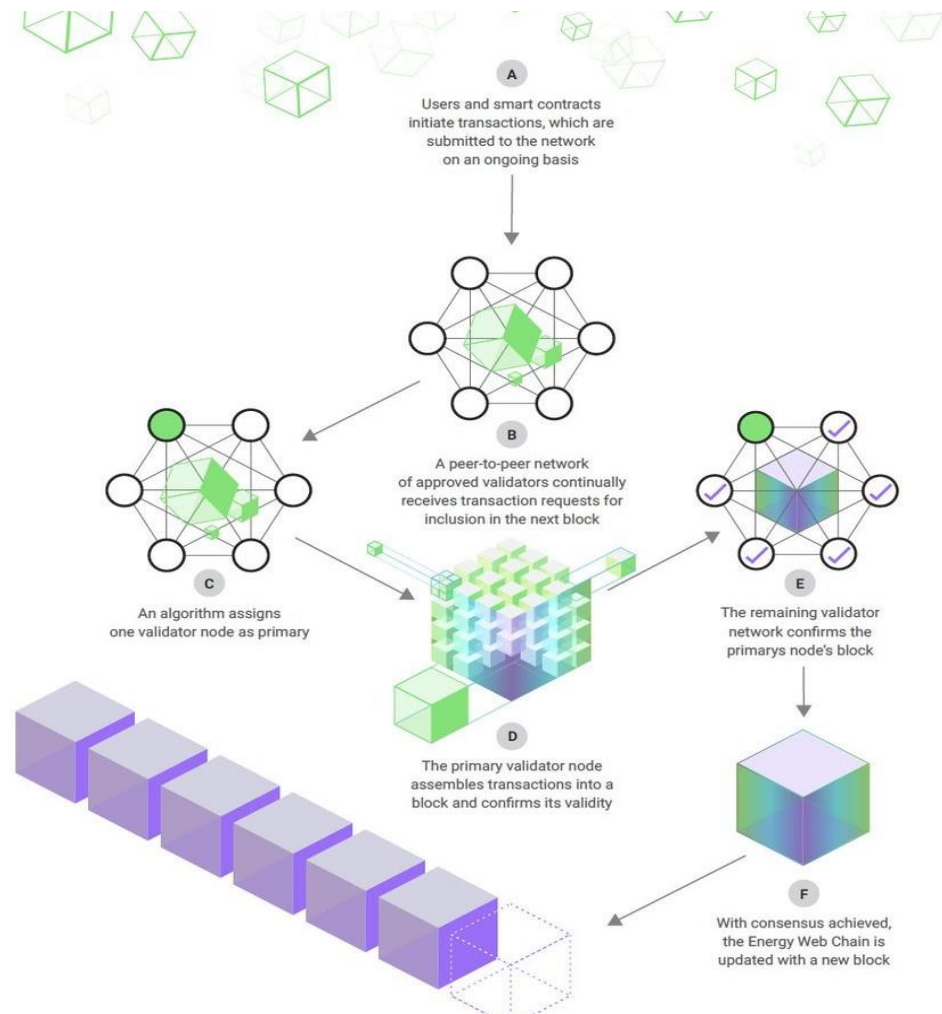


Fig 3.1 System Architecture (Referred from medium.com)

### 3.1.1. MODULES

A Module is a collection of source files and build settings that allow you to divide your project into discrete units of functionality. Your project can have one or many modules and one module may use another module as a dependency. Each module can be independently built, tested, and debugged.

Modules majorly reflects the design of the project and happens to be crucial when it comes to step-by-step and part-by-part execution of the program.

Typical blockchain modules for safe financial transactions include:

#### **Authentication Module:**

Prior to granting users access to the system, the authentication module is in charge of verifying their identities. It makes sure that only people who have been given permission may make transactions on the blockchain.



**Account Management Module:**

This module is in charge of handling user account management, which includes account setup, balance changes, and transaction history.

**Transaction Processing Module:**

The blockchain's transaction processing module is in charge of handling transactions. It guarantees that transactions are genuine and that the blockchain ledger has them recorded.

**Blockchain Ledger Module:**

The blockchain ledger is created and maintained by the module known as the blockchain ledger. It guarantees the security and tamper-resistance of the ledger.

**User Interface Module:**

This module is responsible for providing a user-friendly interface for users to interact with the system. It provides feedback to the user and makes it easy to navigate through the system.

**3.2 UML DIAGRAMS**

UML stands for “Unified Modelling Language”. UML is a great way to express our project details simply without any confusion to any person who may not know anything about your project. Moreover, it mainly contains the Diagrams which shows us the workflow, relationships, actions, etc. The Unified Modelling Language is a popular dialect for indicating, Visualization, Constructing and archiving the curios of programming framework, and for business demonstrating and different non-programming frameworks.

### 3.2.1 USE CASE DIAGRAM

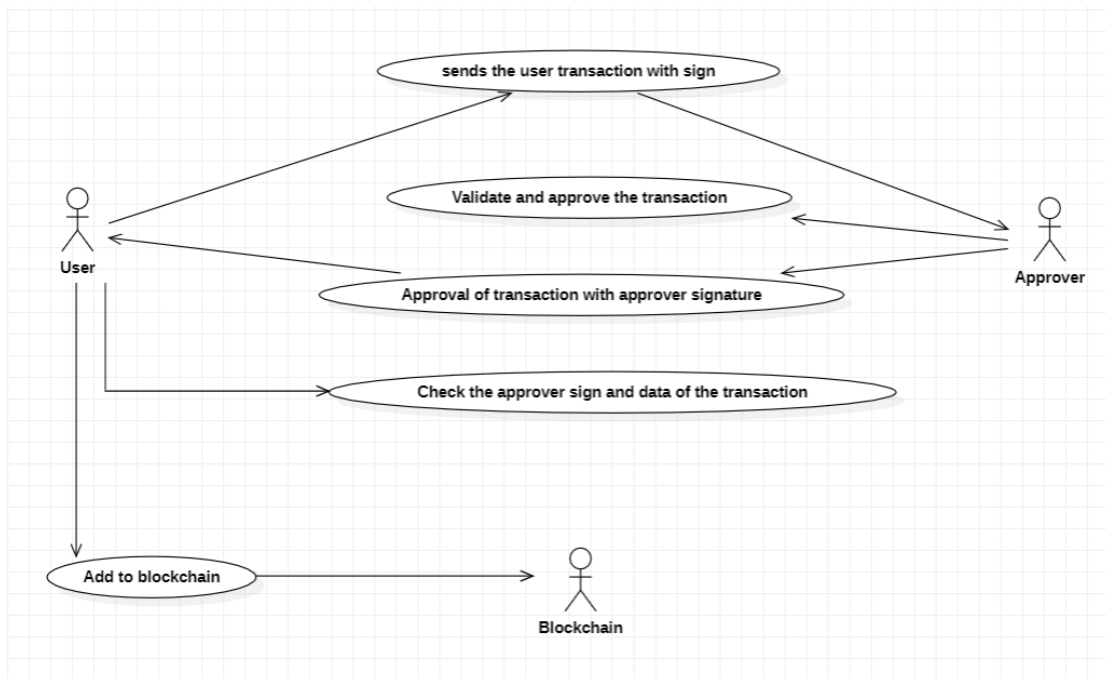


Fig 3.2.1 Use-Case Diagram

A use case diagram is a type of behavioural diagram created from a Use-case analysis. The purpose of use case is to present overview of the functionality provided by the system in terms of actors, their goals and any dependencies between those use cases.

### 3.2.2 ACTIVITY DIAGRAM

Activity diagrams are UML behaviour diagrams which show flow of control or object flow with emphasis on the sequence and conditions of the flow. Activity diagram is UML behaviour diagram which shows flow of control or object flow with emphasis on the sequence and conditions of the flow.

**Initial State:** It depicts the initial stage or beginning of the set of actions.

**Final State:** It is the stage where all the control flows and object flows end.

**Action Box:** It represents the set of actions that are to be performed.

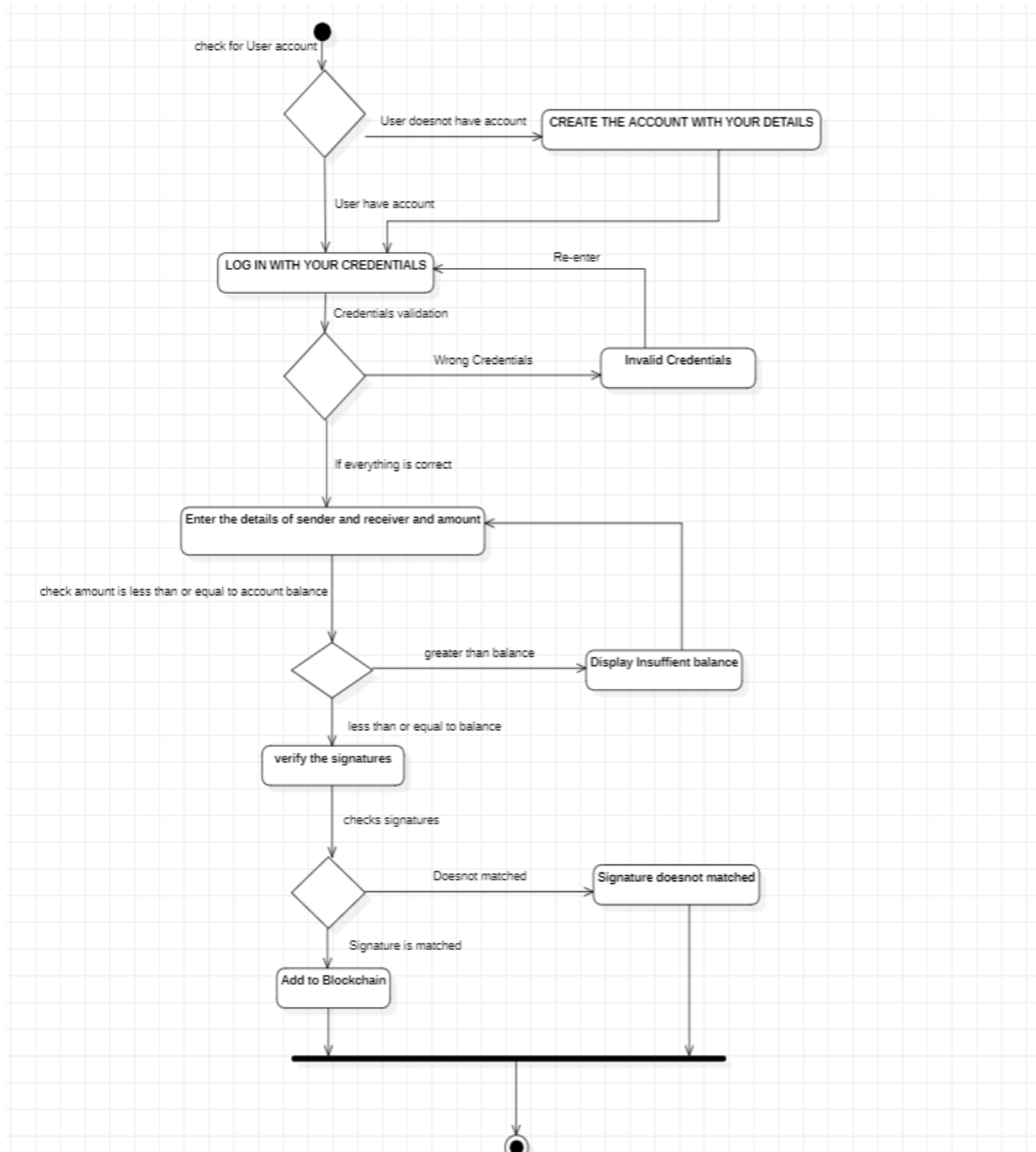


Fig 3.2.2 Activity Diagram

### 3.2.3 CLASS DIAGRAM

Class diagram is a static diagram. It represents the static view of an application. Class diagram is not only used for visualizing, describing, and documenting different aspects of a system but also for constructing executable code of the software application.

Class diagram describes the attributes and operations of a class and also the constraints imposed on the system. The class diagrams are widely used in the modelling of object-oriented systems because they are the only UML diagrams, which can be mapped directly with object-oriented languages.

Class diagram shows a collection of classes, interfaces, associations, collaborations, and constraints. It is also known as a structural diagram.

UML diagrams like activity diagram, sequence diagram can only give the sequence flow of the application, however class diagram is a bit different. It is the most popular UML diagram in the coder community.

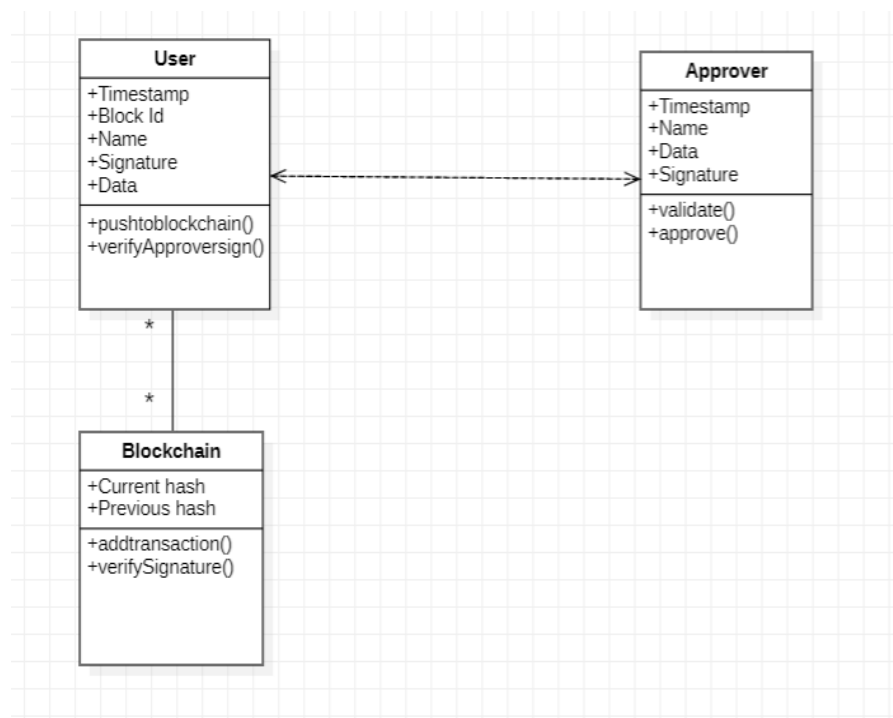


Fig 3.2.3 Class Diagram

## 4.1 INTRODUCTION

### (A) Algorithm Specification:

#### **Proof of Approval:**

The working process of proof of approval can be described in three steps:

- A User creates a block he/she wishes to add to the blockchain, signs it and sends it to an Approver via the P2P Network (The process of sending it can be done by randomly choosing an approver or in this case by having the user specifically choose an approver).
- The Approver receives the block from the user, validates the block by checking its hash etc, checking the signature to ensure that the user is also a valid user and finally approves the information inside of the block after which the approver then signs the block (which acts as an approval) and sends it back to the user.
- The User receives the block from the approver, checks to make sure that the block has not been altered, checks the signature of the approver to ensure the approver is genuine and finally pushes the block to the blockchain.

In this case, the blocks are passed between the user and approver through the Peer-to-Peer network, hashing is can be done using any standardized hashing algorithm (though we recommend SHA256 just because it is the most commonly used today but if you wish for a stronger hashing algorithm such as SHA512, it can be used as well, though keep in mind that it will be slower) and signing can done using something like RSA or ECDSA.

## 4.2 REQUIREMENTS

### 4.2.1 HARDWARE REQUIREMENTS

The hardware requirements may serve as the basis for a contract for the implementation of the system and should therefore be a complete and consistent specification of the whole system. They are used by software engineers as the starting point for the system design. What the system does and not how it should be implemented.

H/W REQUIREMENTS	SPECIFICATION
System	Intel core i5 processor and newer
Hard Disk	1 Terabytes or more
RAM	8 Gigabytes or more

### 4.2.2 SOFTWARE REQUIREMENTS

The software requirements document is the specification of the system. It should include both a definition and a specification of requirements. It is a set of what the system should do rather than how it should do it. The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team's progress throughout the development activity.

S/W REQUIREMENTS	SPECIFICATION
IDE	Visual Studio Code
Operating System	Any (Linux, Win 10/8/7XP, Mac)
Language	Node.js

## **LIBRARIES IMPORTED:**

### **Express:**

Express is a popular unopinionated web framework, written in JavaScript and hosted within the Node.js runtime environment. This module explains some of the key benefits of the framework, how to set up your development environment and how to perform common web development and deployment tasks.

### **Openssl:**

openssl-nodejs is a package which gives you a possibility to run every OpenSSL command in Node.js in a handy way. Moreover, parameters like -in, -keyin, -config and etc can be replaced by a raw data.

### **Body-parser:**

Body-parser is the Node.js body parsing middleware. It is responsible for parsing the incoming request bodies in a middleware.

### **sha256:**

sha256 is the Node.js module which gives access to perform the process of Secure Hashing.

### **Crypto:**

The crypto module provides cryptographic functionality that includes a set of wrappers for OpenSSL's hash, HMAC, cipher, decipher, sign, and verify functions.

## **4.3 SOURCE CODE:**

### **Index.js**

```
const express = require('express');
const port1=process.env.PORT||5000;
const app = express();
const bodyParser = require('body-parser');
const blockchain = require('./blockchain');
const sha256 = require('sha256');
const crypto = require('crypto');
const handlebars = require('handlebars');
let data=[];
let name="";
```

```

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
let users
= {"admin": '8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c923adc
6c92', "charith": '8d969eef6ecad3c29a3a629280e686cf0c3f5d5a86aff3ca12020c
923adc6c92'};
let balance= {"admin": 10000, "charith": 10000};
const source = `
<!DOCTYPE html>
<html>
<head>
  <title>BlockChain Ledger</title>
  <style>
    body {
      background-color: #f2f2f2;
      font-family: Arial, sans-serif;
    }
    h1 {
      text-align: center;
      font-size: 3em;
      color: #4285f4;
      margin-top: 50px;
    }
    .container {
      max-width: 800px;
      margin: 50px auto;
      padding: 20px;
      background-color: #fff;
      box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.3);
      border-radius: 10px;
      text-align: center;
    }
    table {
      width: 100%;
      border-collapse: collapse;
      margin-top: 20px;
    }
    th, td {
      padding: 10px;
      text-align: left;
      border-bottom: 1px solid #ddd;
    }
    th {
      background-color: #4285f4;
      color: #fff;
    }
    a.button {

```



```

        display: inline-block;
        background-color: #4285f4;
        color: #fff;
        padding: 10px 20px;
        font-size: 1.2em;
        border: none;
        border-radius: 5px;
        text-decoration: none;
        margin-top: 20px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
    a.button:hover {
        background-color: #3367d6;
    }
</style>
</head>
<body>
    <div class="container">
        <h1>BlockChain Ledger</h1>
        <table>
            <thead>
                <tr>
                    <th>Sender's Name</th>
                    <th>Receiver's Name</th>
                    <th>Amount</th>
                </tr>
            </thead>
            <tbody>
                {{#each space}}
                <tr>
                    <td>{{this.sender}}</td>
                    <td>{{this.receiver}}</td>
                    <td>{{this.amount}}</td>
                </tr>
                {{/each}}
            </tbody>
        </table>
        <a href="/transactions" class="button">Go To Another Transaction</a>
        <a href="/login" class="button">Logout</a>
    </div>
</body>
</html>
`;
const source1=`
<!DOCTYPE html>
<html>

```

```

<head>
  <title>Secure Bank Transactions</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <style>
    /* General Styles */
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
    }
    a {
      text-decoration: none;
    }
    h1 {
      text-align: center;
      margin-top: 50px;
      color: #4c96af;      }
    form {
      max-width: 500px;
      margin: 0 auto;
      padding: 20px;
      background-color: #fff;
      border-radius: 10px;
      box-shadow: 0px 0px 10px #aaa;
    }
    label {
      display: block;
      margin-bottom: 10px;
      font-weight: bold;
      color: #333;
    }
    input[type="text"] {
      width: 100%;
      padding: 12px;
      margin-bottom: 20px;
      box-sizing: border-box;
      border: 1px solid #ccc;
      border-radius: 5px;
      font-size: 16px;
    }
    input[type="submit"] {
      width: 100%;

```

```

        background-color: #4c96af;
        color: white;
        padding: 14px 20px;
        margin-bottom: 20px;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        font-size: 16px;
        transition: background-color 0.3s ease;
    }
    input[type="submit"]:hover {
        background-color: #45a049;
    }
    .logout-btn {
        display: block;
        margin-top: 20px;
        text-align: center;
    }
    .logout-btn a {
        display: inline-block;
        background-color: #4c96af;
        color: white;
        padding: 10px 20px;
        font-size: 1.2em;
        border: none;
        border-radius: 5px;
        text-decoration: none;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
    .logout-btn a:hover {
        background-color: #45a049;
    }

    /* Responsive Styles */
    @media (max-width: 768px) {
        form {
            max-width: 90%;
        }
    }
}

</style>
<meta name='viewport' content='width=device-width, initial-
scale=1'>
<script src='https://kit.fontawesome.com/a076d05399.js'
crossorigin='anonymous'></script>
</head>
<body>

```

```

    <h1>Secure Bank Transactions</h1>
    <br><br>
    <form action="/transactions" method="post">
        <label for="sender">Sender username:</label>
        <input type="text" readonly id="sender" name="sender"
value="{{this.name}}">
        <br>
        <label for="receiver">Receiver username:</label>
        <input type="text" id="receiver" name="receiver">
        <label for="amount">Amount:</label>
        <input type="text" id="amount" name="amount">
        <input type="submit" value="Submit Transaction">
        <div class="logout-btn">
            <a href="/login">Logout</a>
        </div>
        <div class="logout-btn">
            <a href="/bal">Know your Balance</a>
        </div>
    </form>
</body>
</html>`;
const source2=`
<!DOCTYPE html>
<html>
<head>
    <title>Insufficient Balance</title>
    <style>
        body {
            background-color: #f2f2f2;
            font-family: Arial, sans-serif;
        }
        h1 {
            text-align: center;
            font-size: 2em;
            color: #4285f4;
            margin-top: 50px;
        }
        .container {
            max-width: 800px;
            margin: 50px auto;
            padding: 20px;
            background-color: #fff;
            box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.3);
            border-radius: 10px;
            text-align: center;
        }
        button {
            background-color: #4285f4;

```

```

        color: #fff;
        padding: 10px 20px;
        font-size: 1.2em;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
    button:hover {
        background-color: #3367d6;
    }
</style>
</head>
<body>
    <div class="container"><h1>
    Total Balance:</h1>
    <h1>{{this.b}}</h1>
    <form action="/transactions">
        <button >Back</button>
    </form>
    </div>
</body>
</html>
`;
const template = handlebars.compile(source);
const template1=handlebars.compile(source1);
const template3=handlebars.compile(source2);
// middleware function to authenticate user
function authenticateUser(req, res, next) {
    // check if user is logged in
    if (false) {
        res.status(401).send('Unauthorized access. Please login to
continue.');
```

```

    } else {
        next();
    }
}
}

// route to display the login page
app.get('/login', (req, res) => {
    res.sendFile(__dirname + '/login.html');
});
app.get('/register', (req, res) => {
    res.sendFile(__dirname + '/register.html');
    /* <center><i class='fab fa-amazon-pay' style='font-
size:50px'></i></center><br>
        <center><i class="fas fa-user-circle" style="font-
size:60px;color:#4c96af;"></i></center>
```

```

        <center>
    */
});

app.post("/register", (req, res) => {
    const username = req.body.username;
    const password = req.body.password;
    const hashedPassword = sha256(password);
    users[username] = hashedPassword;
    balance[username] = 10000;
    console.log(users);
    console.log(balance);
    res.redirect('/login');
})

// route to handle login request
app.post('/login', (req, res) => {
    const username = req.body.username;
    const password = req.body.password;
    // hash the password
    const hashedPassword = sha256(password);
    // check if the credentials match with the ones in the database
    if (username in users && hashedPassword == users[username]) {
        // create a session for the user
        // req.session.ip = true;
        name = username;
        const html = template1({ name });
        // Send the HTML as the response
        res.send(html);
    } else {
        res.redirect('/error');
    }
});

app.get('/error', (req, res) => {
    res.sendFile(__dirname + '/error.html');
})

// route to display the transactions page
app.get('/transactions', authenticateUser, (req, res) => {
    const html = template1({ name });
    // Send the HTML as the response
    res.send(html);
});

app.get('/bal', (req, res) => {
    const b = balance[name];
    const html = template3({ b });
    res.send(html);
});

```

```

    });
    // route to handle transaction request
    app.post('/transactions', authenticateUser, (req, res) => {
        // retrieve transaction details from the request body
        const sender = req.body.sender;
        const receiver = req.body.receiver;
        const amount = req.body.amount;
        if(receiver in users){
            // create a new transaction object
            const transaction = {
                sender,
                receiver,
                amount
            };
            // create a hash for the transaction
            if(amount<=balance[sender]){
                console.log(balance)
                balance[sender]=balance[sender]-amount;
                balance[receiver]=parseInt(balance[receiver])+parseInt(amount);
                console.log(balance);
                const transactionHash = sha256(JSON.stringify(transaction));
                const { privateKey, publicKey } =
crypto.generateKeyPairSync('ec', {
                    namedCurve: 'sect239k1'
                });

                // Create
                const sign = crypto.createSign('SHA256');
                sign.write(transactionHash);
                sign.end();
                const signature = sign.sign(privateKey, 'hex');
                data=blockchain.addTransaction(transaction,
signature,publicKey);
                let space=[];
                for(let i=0;i<data.length;i++){
                    if(!space.includes(data[i])) {
                        space.push(data[i]);
                    }
                }
                const html = template({ space });
                // Send the HTML as the response
                res.send(html);
            }
        }
        else{
            res.sendFile(__dirname+"/balanceerror.html");
        }
    }
}

```

```

else{
    res.sendFile(__dirname + '/invalid.html');
}
});
app.listen(port1, () => {
    console.log('Server running on port 5000');
});

```

## Index.html

```

<!DOCTYPE html>
<html>
<head>
    <title>Secure Bank using Blockchain</title>
    <style>
        body {
            background-color: #f2f2f2;
            font-family: Arial, sans-serif;
        }
        h1 {
            text-align: center;
            font-size: 3em;
            color: #4285f4;
            margin-top: 50px;
        }
        form {
            margin: 50px auto;
            width: 80%;
            max-width: 600px;
            background-color: #fff;
            padding: 20px;
            border-radius: 10px;
            box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.3);
        }
        label {
            display: block;
            font-size: 1.2em;
            color: #555;
            margin-bottom: 10px;
        }
        input[type="text"],
        input[type="username"],
        input[type="password"] {
            width: 100%;
            padding: 10px;
            font-size: 1.2em;

```



```

        border: none;
        border-radius: 5px;
        margin-bottom: 20px;
        box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.1);
    }
    input[type="submit"] {
        background-color: #4285f4;
        color: #fff;
        padding: 10px 20px;
        font-size: 1.2em;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
    input[type="submit"]:hover {
        background-color: #3367d6;
    }

    a.button {

        display: inline-block;
        background-color: #4285f4;
        color: #fff;
        padding: 10px 20px;
        font-size: 1.2em;
        border: none;
        border-radius: 5px;
        text-decoration: none;
        margin-top: 20px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
    a.button:hover {
        background-color: #3367d6;
    }
</style>
</head>
<body>
    <h1>Secure Bank using Blockchain</h1>
    <form action="/login" method="post">
        <label for="username">Username:</label>
        <input type="text" id="username" name="username" required>
        <br>
        <label for="password">Password:</label>
        <input type="password" id="password" name="password" required>

        <input type="submit" value="Log In">

```

```

</form>
<a href="/register" class="button">Sign Up</a>
</body>
</html>

```

## Register.html

```

<!DOCTYPE html>
<html>
<head>
  <title>Registration Form</title>
  <style>
    body {
      font-family: Arial, sans-serif;
      background-color: #f2f2f2;
    }

    .container {
      background-color: #fff;
      border-radius: 5px;
      padding: 20px;
      margin: 50px auto;
      width: 500px;
      box-shadow: 0px 0px 10px rgba(0,0,0,0.5);
    }

    input[type=text], input[type=password] {
      width: 100%;
      padding: 12px 20px;
      margin: 8px 0;
      display: inline-block;
      border: 1px solid #ccc;
      border-radius: 4px;
      box-sizing: border-box;
    }

    button {
      background-color: #4CAF50;
      color: white;
      padding: 14px 20px;
      margin: 8px 0;
      border: none;
      border-radius: 4px;
      cursor: pointer;
      width: 100%;
    }

    button:hover {

```

```

        background-color: #45a049;
    }
    input[type="submit"]:hover {
        background-color: #3367d6;
    }
    input[type="submit"] {
        background-color: #4285f4;
        color: #fff;
        padding: 10px 20px;
        font-size: 1.2em;
        border: none;
        border-radius: 5px;
        cursor: pointer;
        transition: background-color 0.3s ease;
    }
</style>
</head>
<body>
    <div class="container">
        <h2>Registration Form</h2>

        <form action="/register" onsubmit="return valid()" method=post>
            <label for="username">Username:</label>
            <input type="text" id="username" name="username"><br><br>
            <label for="password">Password:</label>
            <input type="password" id="password" name="password"><br><br>
            <label for="repassword">Re-enter Password:</label>
            <input type="password" id="repassword"
name="repassword"><br><br>
            <input type="submit" value="Log In">
        </form>
        <script>

            function valid(){
                let users ={"admin":12345}
                var username=document.getElementById("username").value;
                var password=document.getElementById("password").value;
                var repassword=document.getElementById("repassword").value;
                if (password !== repassword) {
                    alert("Passwords do not match!");
                    return false;
                }
                else if(username in users){
                    alert("Existing User name . Please try again.")
                    return false;
                }
                users[username]=password;

```

```

        alert("Registration Successful!");
        return true;
    }

```

```

</script>

```

```

</body>

```

```

</html>

```

## Blockchain.js

```

const verify=require('crypto');
const sha256=require('sha256');
const fs = require('fs');
const handlebars = require('handlebars');
const blockchain = [];
const data=[];
function addTransaction(transaction,signature,publicKey) {
    // verify the signature
    if (verifySignature(transaction, signature,publicKey)) {
        // add the transaction to the blockchain
        blockchain.push(transaction);
        blockchain.forEach(function(elem){
            data.push(elem);
        })
        return data;
    }
    else {
        console.log('Invalid signature. Transaction not added to the
        blockchain.');
```

```

    }

```

```

}

```

```

function verifySignature(transaction, signature,publicKey) {
    // create a hash for the transaction
    const transactionHash = sha256(JSON.stringify(transaction));

```

```

    // verify the signature using the public key
    const vf = verify.createVerify('SHA256');
    vf.update(transactionHash);
    return vf.verify(publicKey, signature, 'hex');
```

```

}

```

```

module.exports = {

```

```

    addTransaction,
    verifySignature

```

```

};

```

## Balanceerror.html

```
<!DOCTYPE html>
<html>
<head>
  <title>Insufficient Balance</title>
  <style>
    body {
      background-color: #f2f2f2;
      font-family: Arial, sans-serif;
    }
    h1 {
      text-align: center;
      font-size: 2em;
      color: #4285f4;
      margin-top: 50px;
    }
    .container {
      max-width: 800px;
      margin: 50px auto;
      padding: 20px;
      background-color: #fff;
      box-shadow: 0px 2px 10px rgba(0, 0, 0, 0.3);
      border-radius: 10px;
      text-align: center;
    }
    button {
      background-color: #4285f4;
      color: #fff;
      padding: 10px 20px;
      font-size: 1.2em;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      transition: background-color 0.3s ease;
    }
    button:hover {
      background-color: #3367d6;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Insufficient Balance</h1>
    <form action="/transactions">
      <button >Back</button>
    </form>
  </div> </body> </html>
```

## Invalid.html

```
<html>
<head>
  <style>
    body{
      padding: 0;
      margin: 0;
      background-color: #f2f2f2;
      font-family: Arial, sans-serif;
      font-size: 16px;
      line-height: 1.5;
      color: #333;
    }
    h1{
      font-size: 48px;
      font-weight: bold;
      color: #ff5722;
      margin-top: 100px;
      margin-bottom: 50px;
      text-align: center;
      text-transform: uppercase;
    }
    input[type="submit"]{
      display: block;
      margin: 0 auto;
      padding: 10px 20px;
      background-color: #4caf50;
      color: #fff;
      font-size: 18px;
      font-weight: bold;
      border: none;
      border-radius: 5px;
      cursor: pointer;
      transition: all 0.2s ease;
    }
    input[type="submit"]:hover{
      background-color: #3e8e41;
    }
  </style>
</head>
<body>
  <h1>Invalid Receiver</h1>
  <center>
    <form action="/transactions" >
      <input type="submit" value="Back">
    </form>
  </center> </body> </html>
```

#### **4.4 TEST PLAN:**

##### **Functionality Testing:**

- Verify that the user can access the system and log in.
- Verify the user's capability of creating a new account.
- Verify the user's ability to make deposits and withdrawals from their account.
- Verify whether the individual can transfer money to another account.
- Verify whether the transaction was added to the blockchain ledger.

##### **Performance Testing:**

- Verify to determine whether or not transactions are handled fast and effectively.
- Verify to determine if the system can manage a lot of transactions.

##### **Security Testing:**

- Verify the encryption and security of the user's personal and financial information.
- Verify the security and tamper-resistance of the blockchain ledger.
- Determine whether the system is secure against hacking and other security risks.

##### **Compatibility Testing:**

- Check the system's compatibility with various operating systems and browsers.
- Check to see if the system works with other gadgets, including smartphones and tablets.

##### **Usability Testing:**

- Check to see if the system is simple to navigate and utilise.
- Check to see if the user interface is simple and easy to use.
- Check to see if the system gives the user enough feedback.

#### **4.4.1 TEST PROCEDURE**

##### **Functionality Testing:**

- Login to the system using valid credentials.
- Create a new account with valid information.
- Deposit and withdraw money from the account.
- Transfer funds to another account.
- Verify that the transaction is recorded on the blockchain ledger.

##### **Performance Testing:**

- Create multiple transactions and verify that they are processed quickly and efficiently.
- Create a large number of transactions and verify that the system can handle them without slowing down.

##### **Security Testing:**

- Verify that the user's personal and financial information is encrypted and secure.
- Attempt to tamper with the blockchain ledger and verify that it is tamper-proof.
- Attempt to hack into the system and verify that it is protected against hacking and other security threats.

##### **Compatibility Testing:**

- Access the system using different browsers and operating systems.
- Access the system using different devices, such as mobile phones and tablets.

##### **Usability Testing:**

- Navigate through the system and verify that it is easy to use.
- Verify that the user interface is intuitive and user-friendly.
- Verify that the system provides adequate feedback to the user.

#### **4.4.2 TEST CASES**

##### **Login Test Cases:**

- Verify that the user can successfully login to the system with valid credentials.



- Verify that the user is unable to login with invalid credentials.

#### **Account Creation Test Cases:**

- Verify that the user can successfully create a new account.
- Verify that the user is unable to create a new account with incomplete or incorrect information.

#### **Deposit and Withdrawal Test Cases:**

- Verify that the user can successfully deposit and withdraw money from their account.
- Verify that the user is unable to withdraw more money than they have in their account.

#### **Transfer Funds Test Cases:**

- Verify that the user can successfully transfer funds to another account.
- Verify that the user is unable to transfer more money than they have in their account.

#### **Blockchain Ledger Test Cases:**

- Verify that all transactions are recorded on the blockchain ledger.

## **4.5 INPUTS AND OUTPUTS**

### **Login Page**

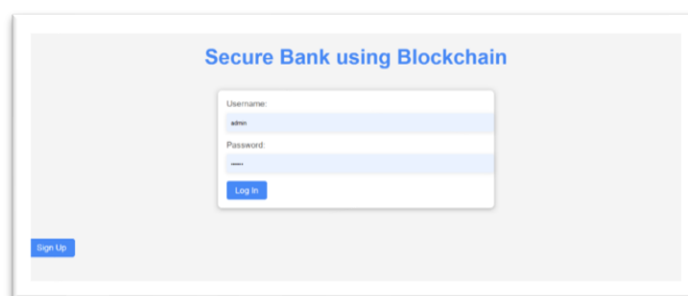


Fig 4.5.1 Login page

Once login credentials have been submitted, a transaction page will be displayed if they are valid; otherwise, an error page will be displayed.

### Error page:

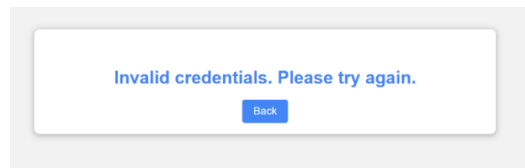


Fig 4.5.2 Error page

### Transaction Page:

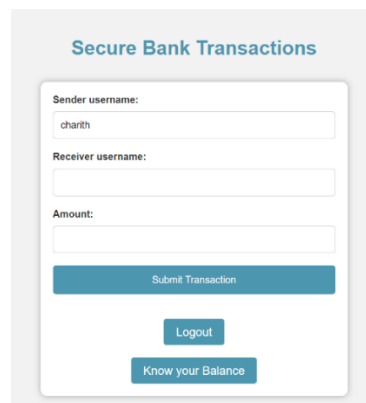
A screenshot of a transaction page titled "Secure Bank Transactions" in a blue font. The page contains a white form with three input fields: "Sender username:" (containing "charith"), "Receiver username:", and "Amount:". Below these fields is a blue button labeled "Submit Transaction". At the bottom of the form are two more buttons: "Logout" and "Know your Balance", both in blue with white text.

Fig 4.5.3 Transaction page

Enter the transaction details now, then click the button to submit the transaction. If the entered account balance is less than or equal to the balance of the account holder, the blockchain ledger page is rendered; otherwise, the message "Insufficient Balance" is displayed. If you need to know the balance, click on the know your balance. Then balance is displayed.

### Balance page:

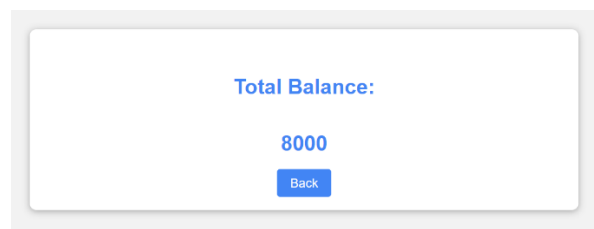
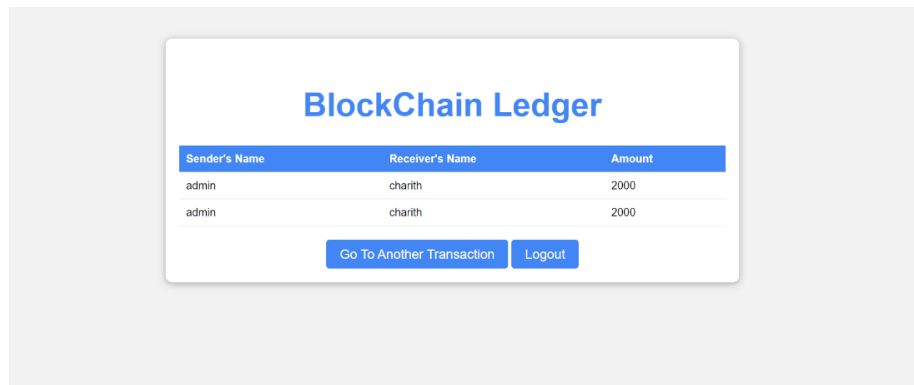


Fig 4.5.4 Balance page

### BlockChain page:



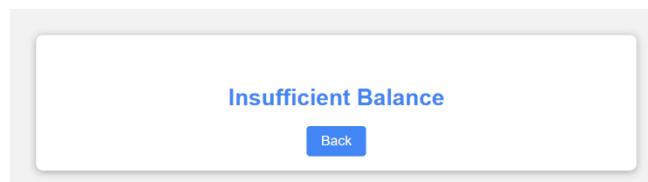
The image shows a web page titled "BlockChain Ledger". It features a table with three columns: "Sender's Name", "Receiver's Name", and "Amount". The table contains two rows of data. Below the table, there are two buttons: "Go To Another Transaction" and "Logout".

Sender's Name	Receiver's Name	Amount
admin	charith	2000
admin	charith	2000

[Go To Another Transaction](#) [Logout](#)

Fig 4.5.5 Blockchain page

### Insufficient balance page:



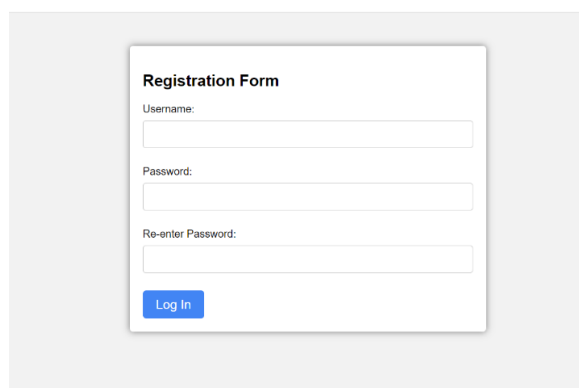
The image shows a web page titled "Insufficient Balance". It features a single button labeled "Back".

[Back](#)

Fig 4.5.5 Error page

If the user doesnot have account ,then user can register using register page.

### Register page:



The image shows a web page titled "Registration Form". It features three input fields: "Username:", "Password:", and "Re-enter Password:". Below the input fields, there is a button labeled "Log In".

**Registration Form**

Username:

Password:

Re-enter Password:

[Log In](#)

Fig 4.5.6 Register page

## **5.1 CONCLUSION**

Blockchain has proven to be an extremely ground-breaking technology but its implementation mainstream has been hindered due to various limitations. In this paper, we seek to present a protocol that would address one of the major limitations of Blockchain, the high resource utilization, cost and maintenance of the predominant consensus protocol being used. Our protocol looks to allow for Blockchain's continued integration into everyday life with realistic results. In this paper, we show how Blockchain can be used alongside traditional banking systems to enhance their security and improve the transparency of their transactions.

## **5.2 FUTURE ENHANCEMENTS**

Future enhancements that could be made to secure bank transactions using blockchain technology are increasing of Scalability, Interoperability, Privacy, User experience and usage of Smart Contracts. As the technology continues to evolve, we can expect to see many more innovations and improvements in this space.

## BIBLIOGRAPHY

### REFERENCES:

- [1] S.Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System". Cryptography Mailing list at <https://metzdowd.com>. Oct 2008.
- [2] Androulaki, Elli, Artem Barger, Vita Bortnikov, Christian Cachin, Konstantinos Christidis, Angelo De Caro, David Enyeart et al. "Hyperledger fabric: a distributed operating system for permissioned blockchains." In Proceedings of the thirteenth EuroSys conference, pp. 1-15. 2018.
- [3] P. Hooda, "Proof of Work (PoW) Consensus - GeeksforGeeks", GeeksforGeeks, 2019. [Online]. Available: <https://www.geeksforgeeks.org/proof-of-work-pow-consensus/>.
- [4] P. Hooda, "practical Byzantine Fault Tolerance,", GeeksforGeeks, 2019. [Online]. Available: <https://www.geeksforgeeks.org/practical-byzantine-fault-tolerancepbft/>.
- [5] P. Javeri, "Smart Contracts and Blockchains", Medium, 2019. [Online]. Available: <https://medium.com/@prashunjaveri/smart-contracts-and-blockchains-c24538418bf6>.
- [6] W. Kenton, "Proof of Burn (Cryptocurrency) Definition", Investopedia, 2019. [Online]. Available: <https://www.investopedia.com/terms/p/proof-burn-cryptocurrency.asp>.
- [7] P. Hooda, "Proof of Stake (PoS) in Blockchain,", GeeksforGeeks, 2019. [Online]. Available: <https://www.geeksforgeeks.org/proof-of-stake-pos-in-blockchain/>.
- [8] T. Marler, "Hyperledger Fabric: Transaction," Medium, 2018. [Online]. Available: Hyperledger Fabric Medium
- [9] P. B, "Architecting a Hyperledger Solution - Things to keep in mind," Hackernoon, 2018. [Online]. Available: Hyperledger Fabric Hackernoon
- [10] J. H, "Proposed System - Google Docs 1,", Scribd, 2019 [Online]. Available: Proposed System Ethereum Official webpage. Available: <http://www.ethdocs.org/en/latest/introduction/what-is-ethereum.html> (2016).

**TEXTBOOKS:**

- A Reputation-Based Consensus Protocol for Peer-to-Peer Network.
- Buterin, Vitalik. "What is Ethereum?."

**WEBSITES:**

- <https://stackoverflow.com/>
- <https://youtube.com/>