

GOVERNMENT FUND ALLOCATION AND TRACKING SYSTEM USING BLOCKCHAIN TECHNOLOGY

Submitted in partial fulfillment of the requirements for the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KOTA SAI PREETHI (Reg. No.40111101)

SAMA SUBHADRA (Reg. No.40111114)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

CATEGORY -1 UNIVERSITY BY UGC

Accredited with Grade "A++" by NAAC | 12B Status by UGC | Approved by AICTE

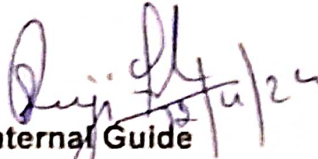
JEPPIAAR NAGAR, RAJIV GANDHI SALAI, CHENNAI - 600119

APRIL - 2024

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

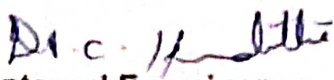
This is to certify that this Project Report is the bonafide work of KOTA SAI PREETHI (40111101) and SAMA SUBHADRA (40111114) who carried out the Project entitled "GOVERNMENT FUND ALLOCATION AND TRACKING SYSTEM USING BLOCKCHAIN TECHNOLOGY" under my supervision from November 2023 to April 2024.

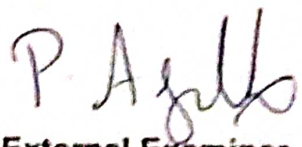

Internal Guide

Dr. L. SUJHELEN, M.E., Ph.D.,


HEAD OF THE DEPARTMENT
~~Head of the Department~~
SATHYABAMA
INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)
Dr. L. LAKSHMANAN, M.E., Ph.D.,
Jeppiaar Nagar, Rajiv Gandhi Salai,
Chennai - 600 119.

Submitted for Viva voce Examination held on 13/04/2024


Internal Examiner


External Examiner

DECLARATION

I, KOTA SAI PREETHI (40111104) hereby declare that the Project Report entitled "GOVERNMENT FUND ALLOCATION AND TRACKING SYSTEM USING BLOCKCHAIN TECHNOLOGY" done by me under the guidance of Dr. L. SUJHELEN, M.E., Ph.D., is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE: 13/04/2024.

PLACE: Chonnal

Sai preethi kota.

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **The Board of Management of Sathyabama Institute of Science and Technology** for their encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. SASIKALA M.E., Ph.D., Dean, School of Computing, Dr. L. LAKSHMANAN M.E., Ph.D., Head of The Department,** Department of Computer Science and Engineering for providing me necessary support during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Dr. L. SUJHELEN, M.E., Ph.D,** for her valuable guidance, suggestions, and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

State Governments need to cater to a huge number of responsibilities of a state. The working of state governments involves huge number of transactions towards various operations that need to be carried out throughout the state. This includes, repair and maintenance works, awarding contracts, paying of government employees, farmer schemes and so on. A major hurdle that the top government face is the low level corruption that is sometimes impossible to track which deprives the state progress. Tracking it is a very difficult task due to the current system. Here we propose a smart system to track funds allocated to the state government as they travel through the government process at each stage. We here make use of blockchain technology to secure the transactions at every stage while maintaining transparency in every transaction sealing every transaction with proofs as the funds move ahead. This allows to maintain crystal clear record with on demand right to transactional data on a need-to-know basis. The system makes use of encryption to secure transactional data using hashes to maintain a block of transactions in a chain manner which is maintained and verified by every node involved to verify the transaction and save the data in a transparent form within the government. The system allows for a full proof, secure and authentic fund allocation and fund tracking system to help form an incorruptible government process.

TABLE OF CONTENTS

Chapter No	TITLE	Page No.
	ABSTRACT	v
	LIST OF FIGURES	viii
1	INTRODUCTION	1
	1.1 Overview	1
	1.2 Purpose of government fund allocation using blockchain technology	1
	1.3 Problem statement	2
	1.4 Objective	3
2	LITERATURE SURVEY	4
3	REQUIREMENT ANALYSIS	6
	3.1 Drawbacks of existing system	6
	3.2 Proposed system	6
	3.3 Hardware requirements	7
	3.4 Software requirements	7
4	DESCRIPTION OF PROPOSED SYSTEM	9
	4.1 Proposed work	9
	4.2 Dataflow diagram	10
	4.3 UML Diagrams	11
	4.4 Usecase Diagram	12
	4.5 Activity Diagram	13
	4.6 Sequence diagram	14
5	SYSTEM IMPLEMENTATION	15
	5.1 Module Description	15
	5.2 Algorithms	15

5.3	User registration	15
5.4	Account management	16
5.5	Fund allocation	17
5.6	Verifier access	17
5.7	Real time monitoring	17
5.8	Security measures	18
6	RESULTS AND DISCUSSION	19
7	CONCLUSION	21
	REFERENCES	22
	APPENDIX	24
A	SOURCE CODE	24

LIST OF FIGURES

Figure No.	Figure Name	Page No.
4.1	Architecture diagram	7
4.2	Data flow diagram	9
4.3	Usecase diagram	16
4.4	Activity diagram	17
4.5	Sequence diagram	20
5.1	User registration page	21
5.2	Account management	22
5.3	Fund allocation	
6.1	Register & login	
6.2	Schemes	
6.3	Transactions	

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Blockchain is one of the technologies that have created a disruptive change in several industries. Currently, Blockchain is getting used in numerous places and there are more applications of Blockchain yet to be discovered and implemented. Blockchain is characterized by its decentralized nature, the integrity of the information stored within the chain, and its openness. Due to these characteristics, another area in which Blockchain can be used is to release funds for government projects. Governments have to cater to an enormous number of responsibilities of a state. The working of state governments involves numerous transactions towards various operations that require to be applied throughout the state. This includes new projects, repair, and maintenance work, awarding contracts, paying off government employees, farmer schemes, and so on. A serious hurdle that the highest government faces is the low-level corruption that's sometimes impossible to trace, which deprives the state of progress. Tracking it's a really difficult task because of the present system. Blockchain is touted for its capability to reinforce the trust and ease of information-based exchanges among people and associations. The innovation offers a guarantee when deliberately applied within the proper settings. Customarily, associations working their own, singular IT frameworks trying to group must pander to difficulties including compromise of data, recognizing a solitary wellspring of truth, and provoking establishment. Blockchain innovation tends to those difficulties by giving a specialized establishment that underpins the execution of shared business forms, such that no single substance controls the complete framework. Government incorporates a characteristic need to assemble, support, and ensure open trust in data and frameworks. In such kinds of situations, blockchain may help to boost this trust.

1.2 PURPOSE OF GOVERNMENT FUND ALLOCATION USING BLOCKCHAIN TECHNOLOGY

The use of blockchain technology for government fund allocation and tracking offers several benefits, including transparency, accountability, and efficiency by

leveraging the immutability, decentralization, and transparency of blockchain technology, governments can ensure secure and auditable tracking of funds from allocation to expenditure. This can help to reduce corruption, mismanagement, and fraud, as all transactions are recorded in a tamper-proof and auditable manner. One of the key features of blockchain-based government fund allocation tracking systems is the use of permissioned networks. Permissioned networks allow only authorized stakeholders, such as government agencies, auditors, and relevant stakeholders, to access and validate the data.

Another important feature of blockchain-based government fund allocation tracking systems is the use of smart contracts. Smart contracts are self-executing agreements with the terms of the contract directly written into code. They can be used to automate the allocation and disbursement of funds, ensuring that funds are released only when specific conditions are met. This can help to streamline processes, reduce manual intervention, and increase efficiency.

Real-time monitoring is another benefit of blockchain-based government fund allocation tracking systems. Blockchain technology enables real-time monitoring of government funds, allowing stakeholders to track the allocation and expenditure of funds in real-time. This can help to improve accountability, reduce the risk of fraud, and ensure that funds are being used efficiently and effectively.

In summary, the use of blockchain technology for government fund allocation and tracking offers several benefits, including transparency, accountability, and efficiency. By leveraging features like transparent fund allocation, permissioned networks, smart contract automation, real-time monitoring, and enhanced data security, blockchain addresses the limitations of traditional systems and provides a tamper-proof and auditable framework for managing public resources.

1.3 PROBLEM STATEMENT

Governments worldwide face challenges in effectively managing public funds due to issues such as lack of transparency, accountability, and inefficiencies in traditional fund allocation systems. The current systems are often prone to corruption, mismanagement, and fraud, leading to a loss of public trust and

hindering the progress of government projects. Tracking the flow of funds accurately and ensuring their proper utilization throughout various operations poses a significant challenge within the existing centralized frameworks. The inability to trace low-level corruption and discrepancies in fund allocation further complicates the task of ensuring the integrity of public finances.

In this context, there is a pressing need to enhance the transparency, accountability, and efficiency of government fund allocation processes. Leveraging blockchain technology to revolutionize the management of public funds presents a promising solution to address these challenges. By implementing a decentralized, tamper-proof ledger system, governments can establish a secure and auditable framework for tracking funds from allocation to expenditure. This shift towards blockchain-based fund allocation systems aims to mitigate corruption, streamline processes, and improve the overall management of public resources.

The problem statement revolves around the inefficiencies, lack of transparency, and accountability issues plaguing traditional government fund allocation systems, highlighting the urgent need for innovative solutions like blockchain technology to transform the way public funds are managed and allocated.

1.4 OBJECTIVE

Implementing blockchain technology for government fund allocation and tracking aims to revolutionize the management of public funds by enhancing transparency, accountability, and efficiency. By leveraging a decentralized, tamper-proof ledger system, governments can establish a secure and auditable framework for tracking funds from allocation to expenditure. This shift towards blockchain-based fund allocation systems addresses the challenges of corruption, mismanagement, and fraud that plague traditional systems. Through real-time monitoring, smart contract automation, and enhanced data security, blockchain technology offers a solution to streamline processes, improve accountability, and rebuild public trust in government institutions. By achieving these objectives, governments can establish a tamper-proof framework that promotes transparency, accountability, and efficiency in managing public resources, ultimately leading to a more effective and trustworthy allocation of public funds.

CHAPTER 2

LITERATURE SURVEY

1) GOVERNMENT SCHEMES , 2022

In this paper, the writer attempted to discover the ability of block chain technology to decrease embezzlement which can also additionally arise for the duration of the fund's switch of government schemes. Also discovered 8 appropriate traits of any virtual fund switch method which incorporates a sequence of layers. To make certain equity on this form of the virtual method the writer has proposed to apply block chain- based technology through the usage of mathematical model transaction traits to officially make certain equity at all levels. This system also can be audited through common human beings to trace the cash flow of any schemes, making the system absolutely obvious and fair.

2) SECURE TRANSACTIONAL DATA USING HASHES ,2020

This system makes use of encryption to secure transactional data using hashes to take care of a block of transactions in a very chain manner which is maintained and verified by every node involved to verify the transaction and save the information in a very transparent way. Using the system, a transparent, incorruptible government process can be formed by providing a completely secure, authentic, and reliable system of allocation and tracking of funds.

3) SECURE TRANSACTIONAL DATA USING HASHES, 2022

in this paper, the author considered block chain applications that allow maintaining crystal clear records with the on-demand right to transactional data on a need-to-know basis. the system makes use of encryption to secure transactional data using hashes to take care of a block of transactions during a chain manner which is maintained and verified by every node involved to verify the transaction and save the info in an exceedingly transparent form within the government. in order to form a completely incorruptible government process, the system provides a way to allocate funds securely and track them authentically.

4) GOVERNMENT ISSUES FUNDS, 2021

This paper discusses the use of block chain technology for tracking government funds. When the government issues funds for projects, often a large part of it is unutilized and is barely used for the actual project, one of the main causes being corruption. So, if block chains are introduced here, everyone can track all the amounts regarding when and where it is being utilized. This technology is utilized in the government sector and can have a major impact on the growth and economy of a country.

CHAPTER 3

REQUIREMENT ANALYSIS

3.1 DRAWBACKS OF EXISTING SYSTEMS

The drawbacks of existing systems, as highlighted in the provided sources, encompass various limitations and challenges that hinder the efficiency and effectiveness of current systems. One of the key drawbacks is the prevalence of errors and inconsistencies in the existing systems, which can make it challenging for human operators to maintain accuracy in data processing and management. This can lead to inaccuracies, delays, and inefficiencies, ultimately affecting the overall performance of the system. The drawbacks of existing systems also include limited accessibility, particularly for individuals who may not have easy access to networks or computers. This can restrict the participation of certain groups, such as candidates or veterans, in critical processes like voting or accessing essential information, leading to a lack of inclusivity and fairness.

Another major drawback is the lack of transparency in the existing systems, which can erode public trust and confidence in the system. Security vulnerabilities are also a significant concern, potentially exposing sensitive data to risks such as unauthorized access, data breaches, or manipulation. Inefficiencies in processes, limited innovation, and complexity in maintenance and updates are also common challenges in the current systems. Addressing these drawbacks is crucial for enhancing the performance, reliability, and security of systems across various domains. By understanding and mitigating these limitations, organizations and governments can strive towards more efficient, transparent, and secure systems that better serve their stakeholders and the public. This can lead to improved trust, accountability, and efficiency in managing critical processes and resources.

3.2 PROPOSED SYSTEM

The proposed system aims to address the challenges of public fund distribution by leveraging blockchain technology to create a transparent, secure, and efficient platform. The primary objective of the system is to streamline the process of public fund distribution by automating document management and aggregating donation information in one location. This approach will enable automatic reporting and

enhance transparency in the distribution process.

The proposed system also aims to maintain transparency and security at all levels by implementing a registration-based system for donations. This system will ensure that all donations are transparent and traceable, thereby reducing the risk of corruption and fraud. The system will also provide a consistent rate of return on investment, ensuring that the funds are used effectively and efficiently. Additionally, the system will use technology to maintain transparency in every business and reach organizations at every stage with information such as cash flow and balancing of each case. This approach will ensure that the distribution of public funds is free from corruption and that the funds are used for their intended purpose. Overall, the proposed system will provide a more efficient, transparent, and secure platform for managing public funds, ultimately benefiting the public and promoting accountability and trust in government institutions.

The project's major objective is to develop a platform that tracks government fund allocation using blockchain technology. The platform aims to accomplish the following goals: Making the work of government fund allocation with reporting documentation is more convenient. Due to the fact that the data on donations will be aggregated in one place, it will allow us to prepare reports automatically. by developing a blockchain-based system, charitable foundations can become more transparent. to maintain transparency and security at every stage. In order to create a government procedure that is incorruptible. to provide an immutable environment for funds tracking.

3.4 HARDWARE REQUIREMENTS

The hardware and software requirements for the proposed government fund allocation and tracking system play a crucial role in ensuring the system's efficiency, security, and performance.

System: The system should be equippd with a Pentium i3 Processor to handle the computational requirements efficiently.

Hard Disk: A 500 GB hard disk is necessary to store data securely and ensure sufficient storage capacity for transaction records.

Monitor: A 15" LED monitor provides a clear display for users interacting with the system.

Input Devices: Standard input devices like a keyboard and mouse are essential for user interaction.

RAM: A minimum of 2 GB RAM is required to support the system's operations and ensure smooth performance.

3.5 SOFTWARE REQUIREMENTS

Operating System: Windows 10 is the preferred operating system for compatibility and security features.

Coding Language: Java is the chosen coding language for developing the system, offering versatility and robustness in software development.

By meeting these hardware and software requirements, the proposed system can effectively utilize blockchain technology to ensure transparency, security, and accountability in government fund allocation and tracking processes. The specified hardware components and software environment are essential for the system to function optimally and provide a reliable platform for managing government funds securely and efficiently.

CHAPTER 4

DESCRIPTION OF PROPOSED SYSTEM

4.1 PROPOSED WORK

The proposed system for government fund allocation and tracking using blockchain technology aims to streamline the process of government fund allocation, making it more convenient and efficient. The system ensures transparency and security at every stage of the allocation process, reducing the risk of corruption and mismanagement of funds. By utilizing blockchain technology, the system creates an incorruptible government procedure that is transparent, secure, and accountable.

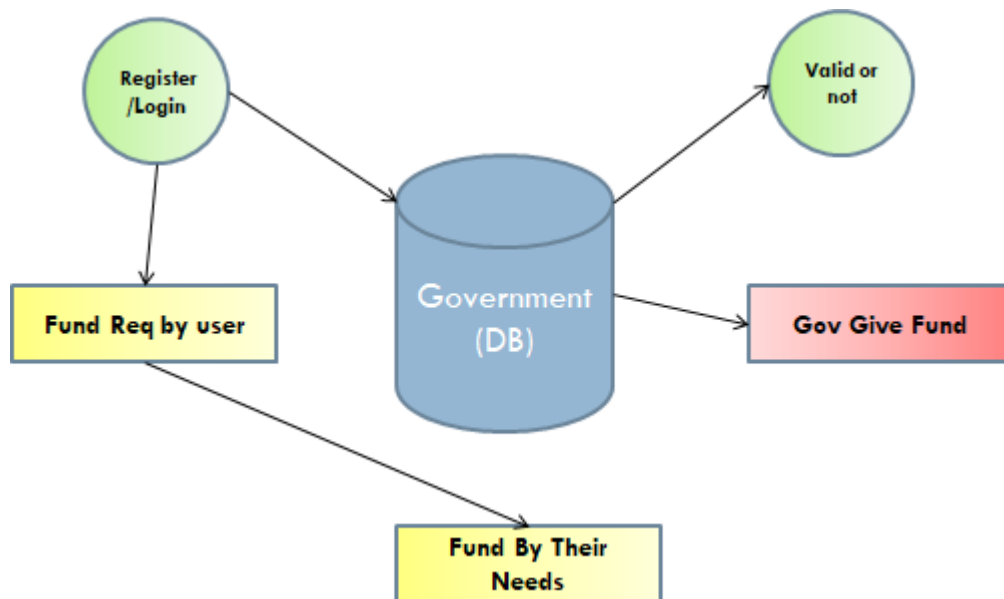
The system is designed to maintain an immutable environment for funds tracking, ensuring that all transactions are recorded and cannot be altered. This provides a reliable and authentic record of all fund allocation and utilization, making it easier to track and audit the use of government funds. The system also aims to optimize the utilization of funds, ensuring that they are used efficiently and effectively to benefit the intended recipients.

The system architecture includes several key components, including a user module, an admin (government) module, and a blockchain. The user module allows users to request funds and check their transaction history and wallet balance. The admin (government) module allows the government to provide the requested funds to users and track the allocation and utilization of funds. The blockchain component ensures the security and transparency of all transactions, providing a tamper-proof record of all fund allocation and utilization.

The system uses encryption to secure transactional data and maintain a block of transactions in a chain manner, which is maintained and verified by every node involved to verify the transaction and save the information in a transparent way. This ensures that all transactions are secure, authenticated, and transparent, reducing the risk of fraud and mismanagement.

Overall, the proposed system for government fund allocation and tracking using blockchain technology provides a robust and secure solution for managing government funds. By ensuring transparency, security, and accountability at every stage of the allocation process, the system helps to reduce corruption, optimize

fund utilization, and promote trust and confidence in the government's financial



management practices.

Fig: 4.1 Architecture Diagram

4.2 DATA FLOW DIAGRAM

1) The DFD is also called as bubble chart. It is a simple graphical formalism that can be used to represent a system in terms of input data to the system, various processing carried out on this data, and the output data is generated by this system.

2) The data flow diagram (DFD) is one of the most important modeling tools. It is used to model the system components. These components are the system process, the data used by the process, an external entity that interacts with the system and the information flows in the system.

3) DFD shows how the information moves through the system and how it is modified by a series of transformations. It is a graphical technique that depicts information flow and the transformations that are applied as data moves from input to output.

4) DFD is also known as bubble chart. A DFD may be used to represent a system at any level of abstraction. DFD may be partitioned into levels that represent increasing information flow and functional detail.

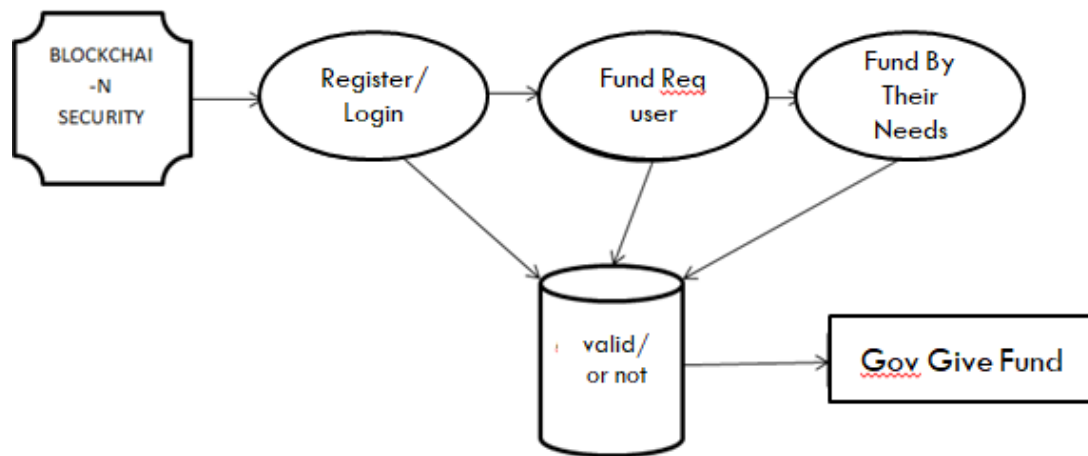


Fig: 4.2 Data Flow Diagram

4.3 UML DIAGRAMS

UML stands for Unified Modelling Language. UML is a standardized general-purpose modelling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified 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 non-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 is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software

4.4 USECASE DIAGRAM

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

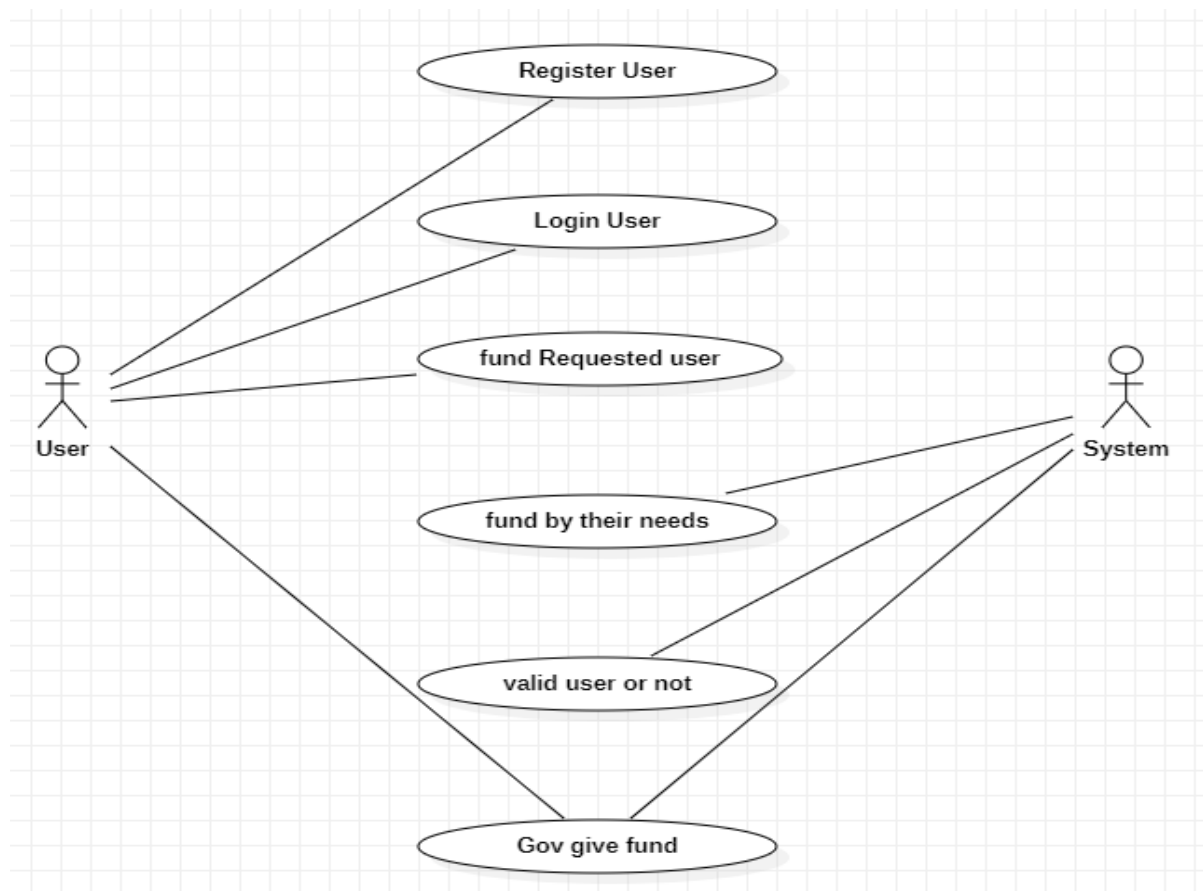


Fig: 4.3 Usecase diagram

4.5 ACTIVITY DIAGRAM

This activity diagram contains three activities that are Server, TPA, Client and. This diagram of control shows the flow between these activities.

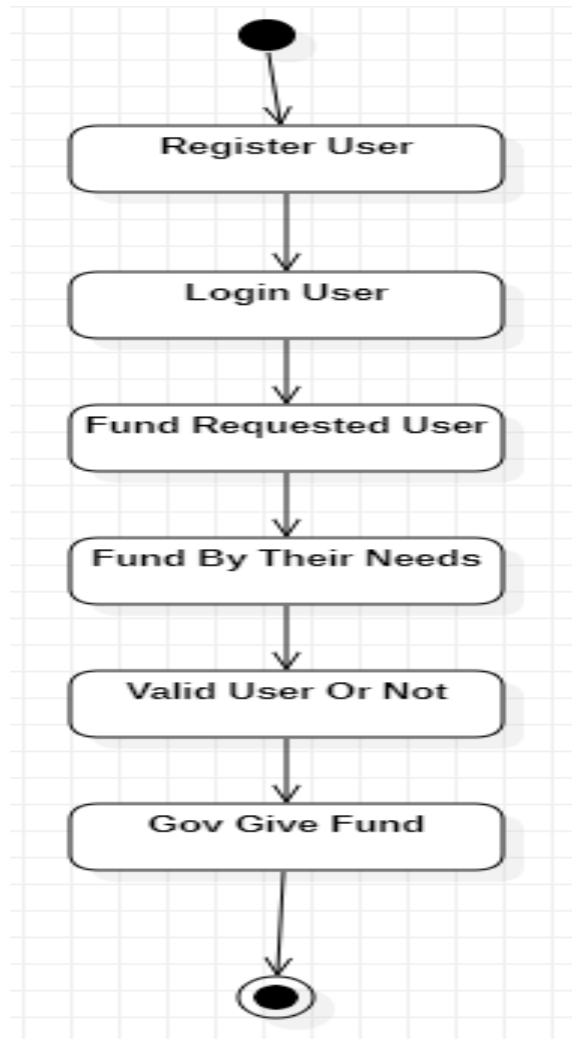


Fig: 4.4 Activity diagram

4.6 SEQUENCE DIAGRAM

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.

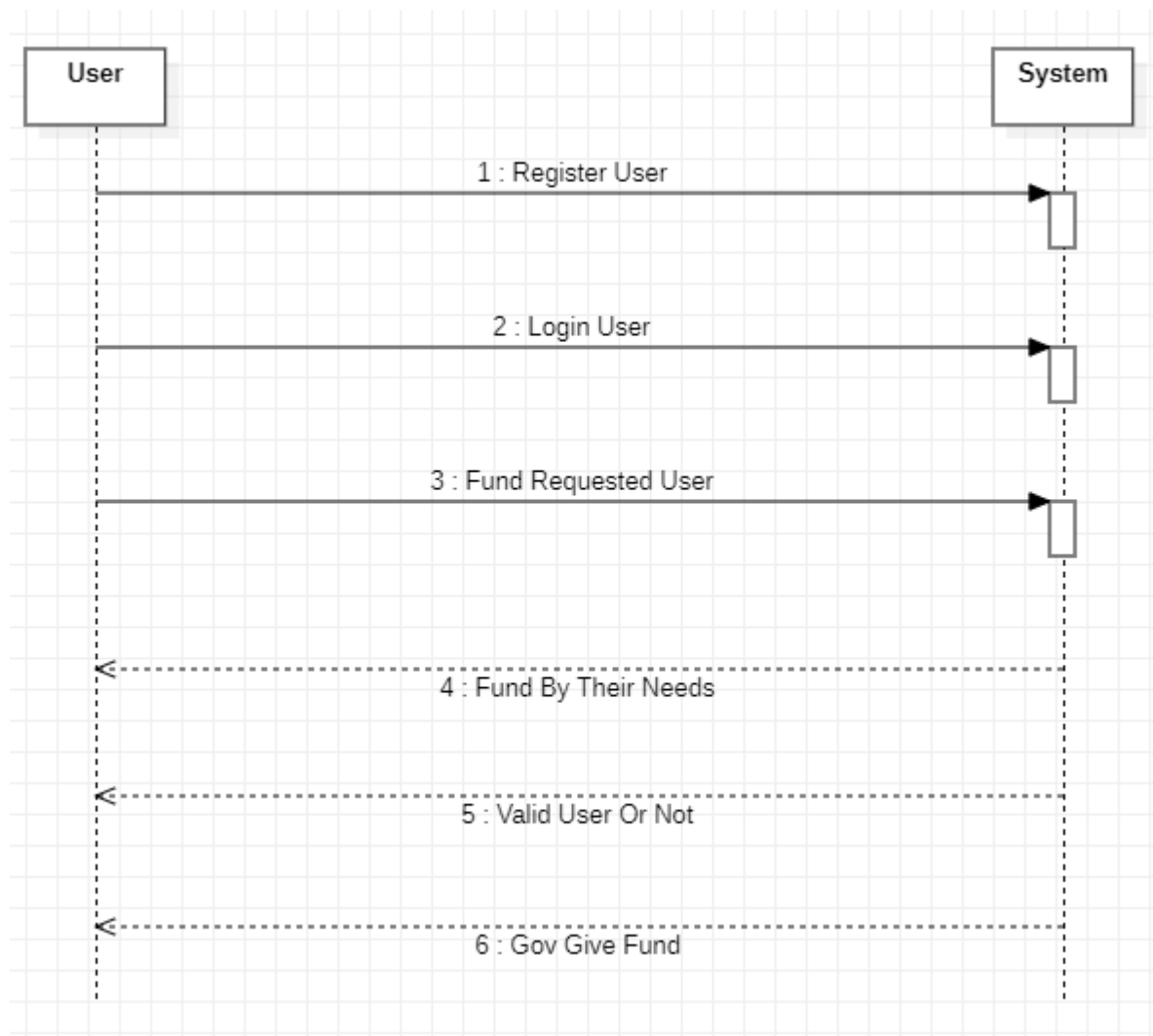


Fig: 4.5 Sequence diagram

CHAPTER 5

SYSTEM IMPLEMENTATION

To set up a government fund allocation and tracking system using blockchain technology, the following steps can be taken to create a user-friendly interface and enable verifiers to access funds securely.

5.1 MODULE DESCRIPTION

- Module 1 - Government: - Government will give the fund which is requested by the user.
- Module 2 – Authority (TPA):- This will authorize or verify the user that it is a valid user as well as valid request or not.
- Module 3 - User (Customer):- User will request for the fund according to their needs.

5.2 ALGORITHMS

- AES with SHA Key: For encrypting metadata file we are using AES encryption algorithm. The encryption process generates a symmetric key called around a key.

5.3 User Registration

- Create a user-friendly registration process that allows users to create an account using their government-issued identification.
- Implement a Know Your Customer (KYC) process to verify the user's identity and ensure compliance with anti-money laundering regulations.
- Generate a unique private key and public key for each user upon registration.

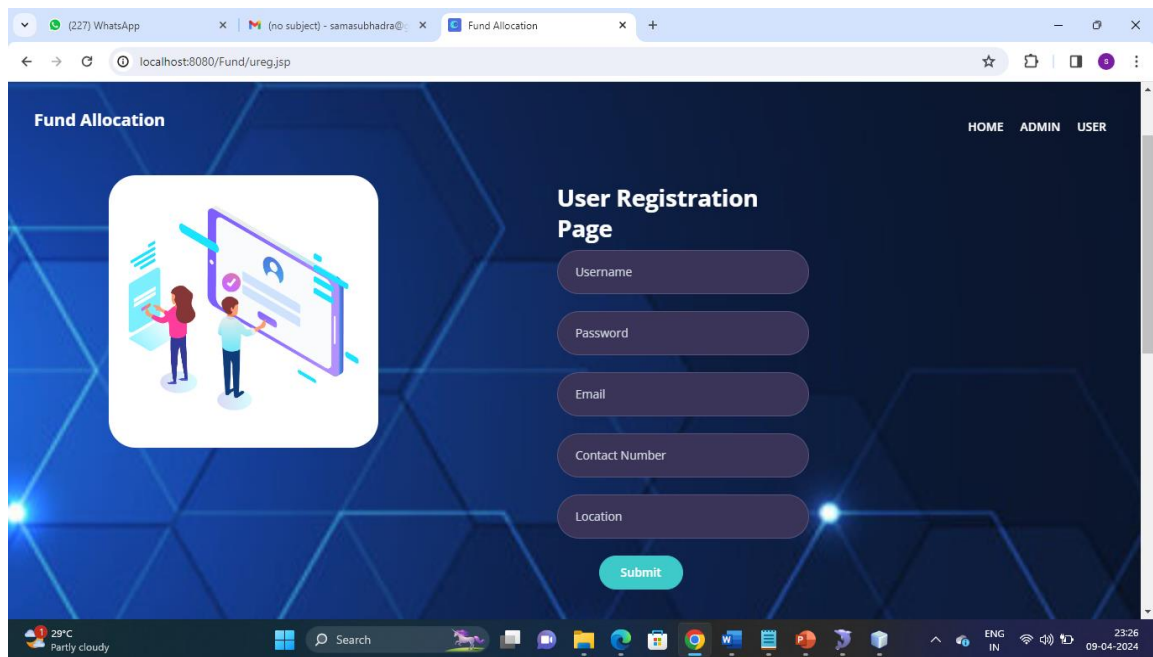


Fig: 5.1 User registration page

5.4 Account Management

- Implement a secure account management system that allows users to view their transaction history, monitor their fund allocation, and request fund transfers.
- Enable users to set up two-factor authentication and biometric verification for added security.

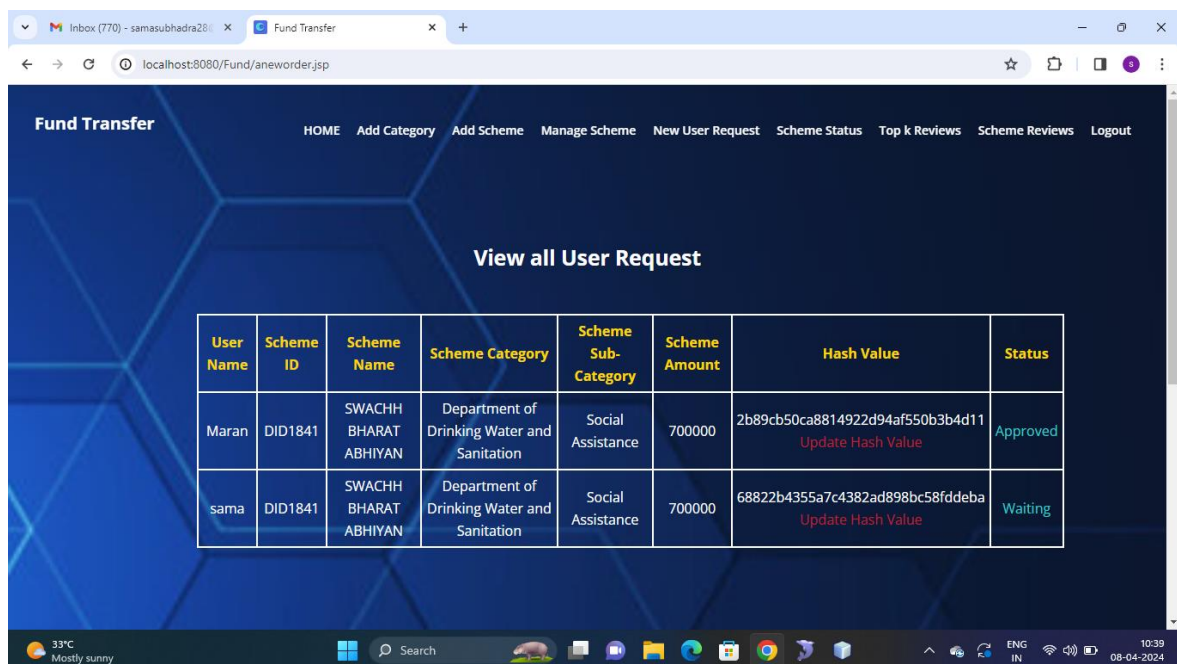


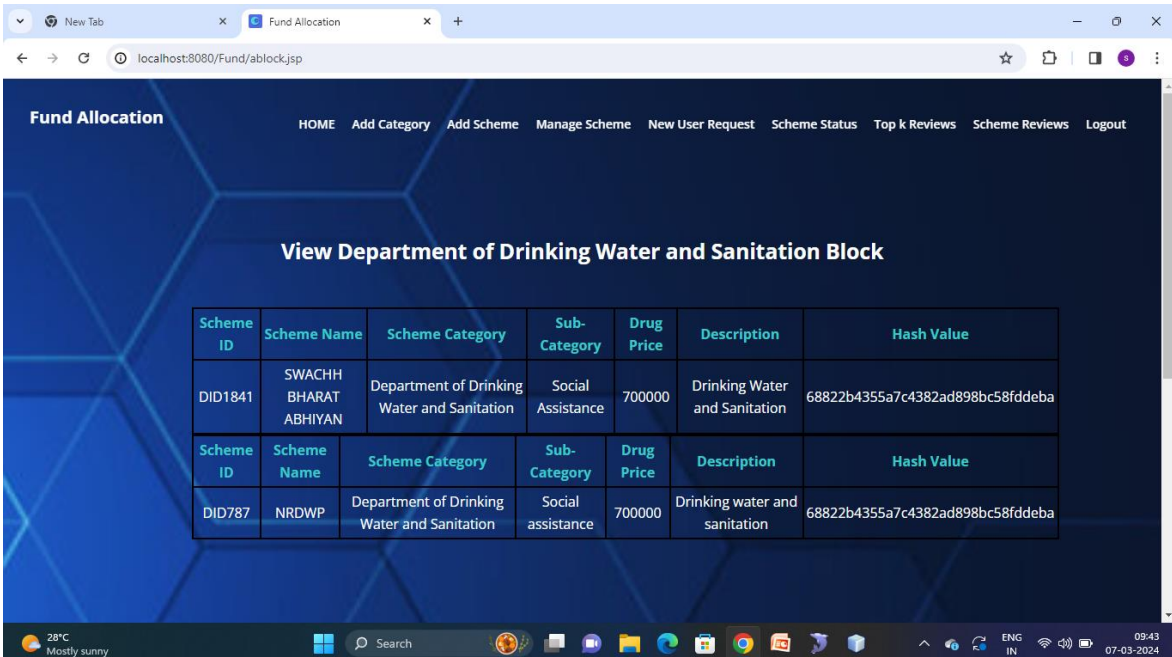
Fig: 5.2 Account management

5.5 Fund Allocation

- Implement a smart contract system that automatically allocates funds to users based on predefined rules and regulations.
- Utilize blockchain technology to ensure the transparency and immutability of fund allocation transactions.
- Implement a proof-of-work or proof-of-stake consensus algorithm to validate transactions and prevent double-spending.

5.6 Verifier Access

- Implement a secure verification process that allows authorized verifiers to access funds and perform audits.
- Utilize multi-signature technology to ensure that fund transfers require the approval of multiple verifiers.
- Implement a secure communication protocol between users and verifiers to ensure the confidentiality and integrity of fund transfer requests.



The screenshot shows a web browser window with the URL `localhost:8080/Fund/ablock.jsp`. The application has a navigation bar with links: HOME, Add Category, Add Scheme, Manage Scheme, New User Request, Scheme Status, Top k Reviews, Scheme Reviews, and Logout. The main content area is titled "View Department of Drinking Water and Sanitation Block" and displays a table with fund allocation details.

Scheme ID	Scheme Name	Scheme Category	Sub-Category	Drug Price	Description	Hash Value
DID1841	SWACHH BHARAT ABHIYAN	Department of Drinking Water and Sanitation	Social Assistance	700000	Drinking Water and Sanitation	68822b4355a7c4382ad898bc58fddeba
DID787	NRDWP	Department of Drinking Water and Sanitation	Social assistance	700000	Drinking water and sanitation	68822b4355a7c4382ad898bc58fddeba

Fig: 5.3 Fund allocation

5.7 Real-Time Monitoring

- Implement a real-time monitoring system that allows users and verifiers to track the status of fund transfers and monitor transaction histories.

- Utilize blockchain technology to ensure the transparency and immutability of transaction data.
- Implement a notification system that alerts users and verifiers of any suspicious activity or security breaches.

5.8 Security Measures

- Implement advanced security measures such as encryption, decryption, and hashing algorithms to secure user data and transaction information.
- Utilize a decentralized network architecture to prevent single points of failure and ensure high availability.
- Implement a disaster recovery and business continuity plan to ensure the system can recover from any unexpected events or security breaches.

CHAPTER 6

RESULTS AND DISCUSSION

The home page will have all the available schemes listed. A normal user can visit the site and check for schemes and available and allocated funds is shown in Fig 6.1.



Fig: 6.1 Register and login

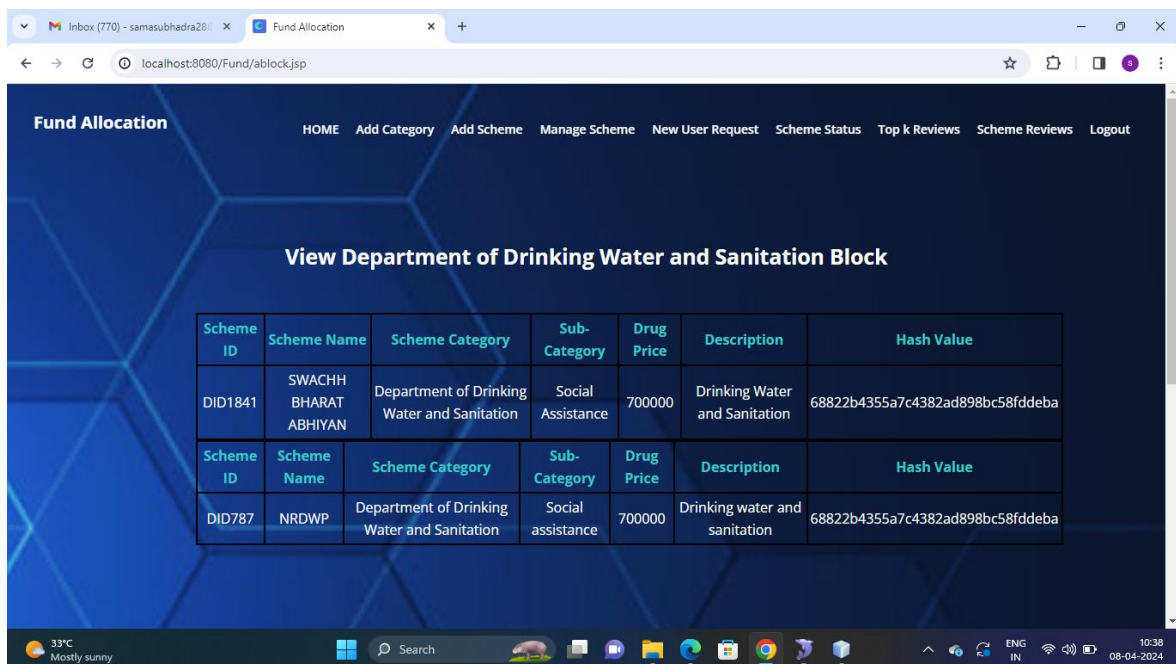
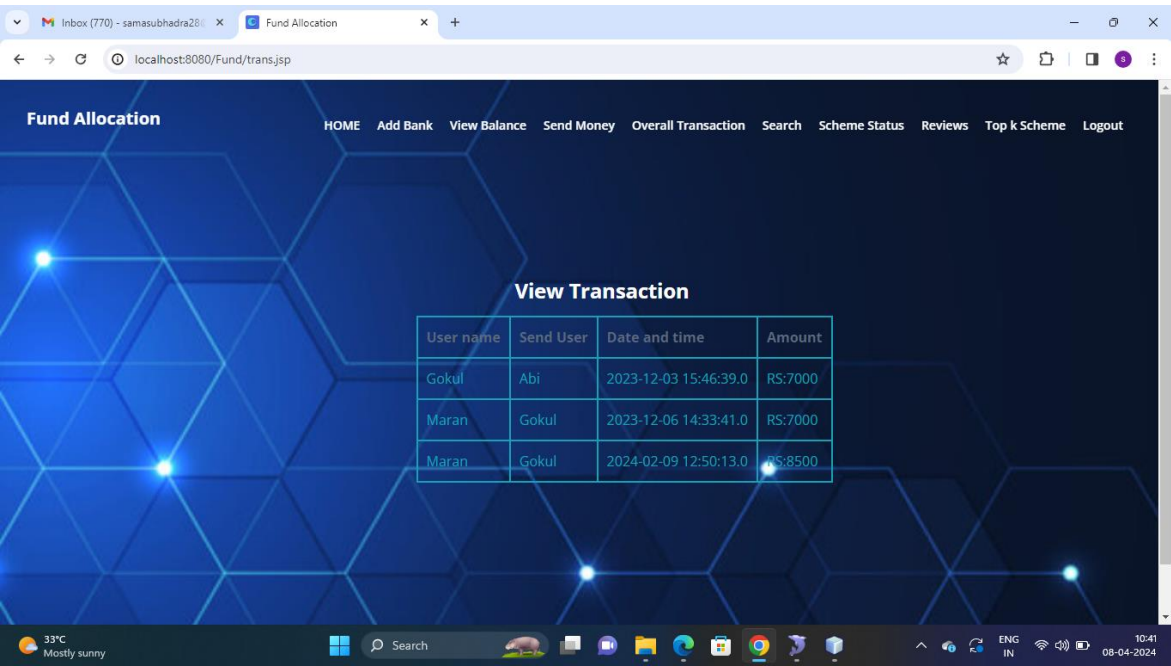


Fig: 6.2 Schemes

A separate blockchain is available for each fund, where a user can check for all transactions. That transactions are publicly available as shown in Fig 6.2.

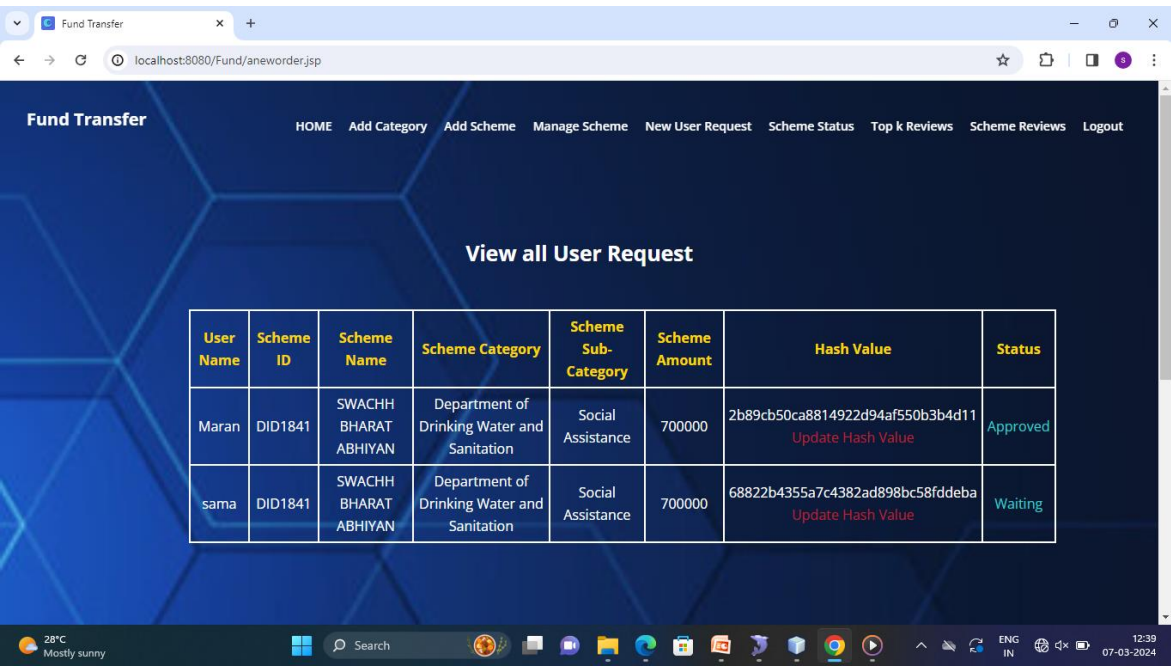


The screenshot shows a web browser at localhost:8080/Fund/trans.jsp. The page has a navigation bar with links: HOME, Add Bank, View Balance, Send Money, Overall Transaction, Search, Scheme Status, Reviews, Top k Scheme, and Logout. The main content area is titled 'View Transaction' and contains a table with transaction details.

User name	Send User	Date and time	Amount
Gokul	Abi	2023-12-03 15:46:39.0	RS:7000
Maran	Gokul	2023-12-06 14:33:41.0	RS:7000
Maran	Gokul	2024-02-09 12:50:13.0	RS:8500

Fig: 6.3 Transactions

Users can check all the transactions within a particular block, all the information about that transaction is available here like username, amount, date, and time of transaction shown in Fig 6.3.



The screenshot shows a web browser at localhost:8080/Fund/anotherworder.jsp. The page has a navigation bar with links: HOME, Add Category, Add Scheme, Manage Scheme, New User Request, Scheme Status, Top k Reviews, Scheme Reviews, and Logout. The main content area is titled 'View all User Request' and contains a table with user request details.

User Name	Scheme ID	Scheme Name	Scheme Category	Scheme Sub-Category	Scheme Amount	Hash Value	Status
Maran	DID1841	SWACHH BHARAT ABHIYAN	Department of Drinking Water and Sanitation	Social Assistance	700000	2b89cb50ca8814922d94af550b3b4d11 Update Hash Value	Approved
sama	DID1841	SWACHH BHARAT ABHIYAN	Department of Drinking Water and Sanitation	Social Assistance	700000	68822b4355a7c4382ad898bc58fddeba Update Hash Value	Waiting

Fig: 6.4 User Authentication

CHAPTER 7

CONCLUSION

In this full-proof, secure government fund allocation and tracking system, the allotted funds are tracked at each level until it reaches the beneficiaries. This proposed framework is added to assist the authorities to lessen corruption and offer transparency in all transactions because of the functions of blockchain-like immutability, proof of work, and security. It offers the right governance and transparency. It will maintain track of all transactions made. As blockchain technology is used the transactions as soon as made cannot be changed and if there's any try of tempering, we can get to recognize approximately that easily. There might be no requirement for the outsider and the exchanges might be regulated all the extra sturdily and transparently. In addition to preventing human errors and delays, it will help eliminate human errors. This framework will make the general public authority framework activities appreciably extra stable and productive. We can nevertheless upload customary schemes from everywhere in the world for fundraising to take it to the subsequent stage for big price range required the humans in need.

REFERENCES

- [1] D. A. Wijaya, "Extending asset management system functionality in bitcoin platform," 2016 International Conference on Computer, Control, Informatics and its Applications (IC3INA), Tangerang, 2016, pp. 97-101, doi: 10.1109/IC3INA.2016.7863031.
- [2] G. Hurlburt, "Could Blockchain Outlive Bitcoin ?," in IT Professional, vol. 18, no. 2, pages 12-16, Mar.- Apr. 2016, i-doi: 10.1109 / MITP.2016.2
- [3] K. Saito and H. Yamada, "What's So Different about Blockchain? — Blockchain is a Probabilistic State Machine," 2016 IEEE 36th International Conference on Distributed Computing Systems Workshops (ICDCSW), Nara, 2016, pp. 168-175, doi: 10.1109/ICDCSW.2016.28.
- [4] M. Moser, R. Bohme noD. Breuker, "An investigation into fraudulent tools in the Bitcoin ecosystem," 2013 APWG at Crime Researchers Summit, San Francisco, CA, 2013, pages 1-14, doi: 10.1109 / CRS. 2013.6805780.
- [5] Mohanta, Bhabendu & Jena, Debasish & Panda, Soumyashree & Sobhanayak, Srichandan. (2019). Blockchain Technology: A Survey on Applications and Security Privacy Challenges. 8. 100107. 10.1016/j.iot.2019.100107.
- [6] P. Asuquo et al., "Security and privacy in location-based services for vehicular and mobile communications: An overview, challenges, and countermeasures," IEEE Internet Things J., vol. 5, no. 6, pp. 4778–4802, Dec. 2018.
- [7] Q. D. Vo and P. De, "A survey of fingerprint-based outdoor localization," IEEE Common. Surveys Tuts., vol. 18, no. 1, pp. 491–506, 1st Quart., 2016.
- [8] R.Gupta and U. P. Rao, "An exploration to location–based service and its privacy preserving techniques: A survey," Wireless Pers. Common., vol. 96, no. 2, pp. 1973- 2007, 2017.

- [9] S. Gambs, C. Onete, and J. M. Robert, "Prover anonymous and deniable distance bounding authentication," in Proc. Asia CCS, 2014, pp. 501–506.
- [10] S. Mauw, Z. Smith, J. Toro-Pozo, and R. Trujillo-Rasua, "Distance bounding protocols: Verification without time and location," in Proc. IEEE Symp. Secure. Privacy (SP), San Francisco, CA, USA, May 2018, pp. 549–566.
- [11] S. Vaudenay, "Private and secure public-key distance bounding: Application to NFC payment," in Proc. Financial Cryptor., in Lecture Notes in Computer Science. Berlin, Germany: Springer, 2015, pp. 207–216.
- [12] T. Chothia, J. D. Ruiter, and B. Smyth, "Modelling and analysis of a hierarchy of distance bounding attacks," in Proc. 27th USENIX Secure. Symp. (USENIX Security), Apr. 2018.
- [13] Y. Li, L. Zhou, H. Zhu, and L. Sun, "Privacy-preserving location proof for securing large-scale database-driven cognitive radio networks," IEEE Internet Things J., vol. 3, no. 4, pp. 563–571, Aug. 2016.
- [14] Y. Zheng, M. Li, W. Lou, and Y. T. Hou, "Location based handshake and private proximity test with location tags," IEEE Trans. Depend. Sec. Compute., vol. 14, no. 4 pp. 406–419, Jul./Aug. 2017.
- [15] Z. Gao, H. Zhu, Y. Liu, M. Li, and Z. Cao, "Location privacy in database-driven cognitive radio networks: Attacks and countermeasures," in Proc. IEEE INFOCOM, Apr. 2013, pp. 2751–2759.

APPENDIX

SOURCE CODE

Transaction.java

```
package block;

public class Transaction {

    private String sourceName;

    private String destinationName;

    private Long sum;

    public String getSourceName() {

        return sourceName;

    }

    public void setSourceName(String sourceName) {

        this.sourceName = sourceName;

    }

    public String getDestinationName() {

        return destinationName;

    }

    public void setDestinationName(String destinationName) {
```

```
        this.destinationName = destinationName;
    }
}
```

```
public Long getSum() {
    return sum;
}
```

```
public void setSum(Long sum) {
    this.sum = sum;
}
```

```
public Transaction(String sourceName, String destinationName, Long sum)
{
    this.sourceName = sourceName;
    this.destinationName = destinationName;
    this.sum = sum;
}
```

@Override

```
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
```

```
    Transaction that = (Transaction) o;
```

```

        if (sourceName != null ? !sourceName.equals(that.sourceName) :
that.sourceName != null) return false;

        if (destinationName != null ?
!destinationName.equals(that.destinationName) : that.destinationName !=
null)

            return false;

        return sum != null ? sum.equals(that.sum) : that.sum == null;
    }

```

@Override

```

public int hashCode() {

    int result = sourceName != null ? sourceName.hashCode() : 0;

    result = 31 * result + (destinationName != null ?
destinationName.hashCode() : 0);

    result = 31 * result + (sum != null ? sum.hashCode() : 0);

    return result;
}

}

```

SHA256.java

```

package block;

```

```

import java.nio.charset.StandardCharsets;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

```

```

public class SHA256 {

    public static String generateHash(String value) {

        String hash = null;

        try {

            MessageDigest md = MessageDigest.getInstance("SHA-
256");

            byte[] bytes =
md.digest(value.getBytes(StandardCharsets.UTF_8));

            StringBuilder sb = new StringBuilder();

            for (int i = 0; i < bytes.length; i++) {

                sb.append(Integer.toString((bytes[i] & 0xff) + 0x100,
16).substring(1));

            }

            hash = sb.toString();

        } catch (NoSuchAlgorithmException e) {

            e.printStackTrace();

        }

        return hash;

    }

}

```

TripleDES.java

```

/*

```

* To change this license header, choose License Headers in Project Properties.

* To change this template file, choose Tools | Templates

* and open the template in the editor.

*/

package block;

/**

*

* @author windows

*/

import java.io.DataInputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.security.InvalidKeyException;

import java.security.NoSuchAlgorithmException;

import java.security.Provider;

import java.security.Security;

import java.security.spec.InvalidKeySpecException;

```

import javax.crypto.BadPaddingException;

import javax.crypto.Cipher;

import javax.crypto.CipherOutputStream;

import javax.crypto.IllegalBlockSizeException;

import javax.crypto.KeyGenerator;

import javax.crypto.NoSuchPaddingException;

import javax.crypto.SecretKey;

import javax.crypto.SecretKeyFactory;

import javax.crypto.spec.DESedeKeySpec;


/**
 * This class defines methods for encrypting and decrypting using the Triple
DES
 * algorithm and for generating, reading and writing Triple DES keys. It also
 * defines a main() method that allows these methods to be used from the
command
 * line.
 */

public class TripleDES {

/**
 * The program. The first argument must be -e, -d, or -g to encrypt,
 * decrypt, or generate a key. The second argument is the name of a file
 * from which the key is read or to which it is written for -g. The -e and
 * -d arguments cause the program to read from standard input and encrypt or
 * decrypt to standard output.

```

```

*/

public static void main(String[] args) {

    try {

        // Check to see whether there is a provider that can do TripleDES
        // encryption. If not, explicitly install the SunJCE provider.

        try {

            Cipher c = Cipher.getInstance("DESede");

        } catch (Exception e) {

            // An exception here probably means the JCE provider hasn't
            // been permanently installed on this system by listing it
            // in the $JAVA_HOME/jre/lib/security/java.security file.
            // Therefore, we have to install the JCE provider explicitly.

            System.err.println("Installing SunJCE provider.");

            Provider sunjce = new com.sun.crypto.provider.SunJCE();

            Security.addProvider(sunjce);

        }

        // This is where we'll read the key from or write it to

        File keyfile = new File(args[1]);

        // Now check the first arg to see what we're going to do

        if (args[0].equals("-g")) { // Generate a key

            System.out.print("Generating key. This may take some time...");

            System.out.flush();

```

```

        SecretKey key = generateKey();

        writeKey(key, keyfile);

        System.out.println("done.");

        System.out.println("Secret key written to " + args[1]
            + ". Protect that file carefully!");
    } else if (args[0].equals("-e")) { // Encrypt stdin to stdout

        SecretKey key = readKey(keyfile);

        encrypt(key, System.in, System.out);
    } else if (args[0].equals("-d")) { // Decrypt stdin to stdout

        SecretKey key = readKey(keyfile);

        decrypt(key, System.in, System.out);
    }
} catch (Exception e) {

    System.err.println(e);

    System.err.println("Usage: java " + TripleDES.class.getName()
        + " -d|-e|-g <keyfile>");
}
}

/** Generate a secret TripleDES encryption/decryption key */
public static SecretKey generateKey() throws NoSuchAlgorithmException {

    // Get a key generator for Triple DES (a.k.a DESede)

    KeyGenerator keygen = KeyGenerator.getInstance("DESede");

    // Use it to generate a key

```

```

    return keygen.generateKey();
}

/** Save the specified TripleDES SecretKey to the specified file */
public static void writeKey(SecretKey key, File f) throws IOException,
    NoSuchAlgorithmException, InvalidKeySpecException {
    // Convert the secret key to an array of bytes like this
    SecretKeyFactory keyfactory = SecretKeyFactory.getInstance("DESede");
    DESedeKeySpec keyspec = (DESedeKeySpec) keyfactory.getKeySpec(key,
        DESedeKeySpec.class);
    byte[] rawkey = keyspec.getKey();

    // Write the raw key to the file
    FileOutputStream out = new FileOutputStream(f);
    out.write(rawkey);
    out.close();
}

/** Read a TripleDES secret key from the specified file */
public static SecretKey readKey(File f) throws IOException,
    NoSuchAlgorithmException, InvalidKeyException,
    InvalidKeySpecException {
    // Read the raw bytes from the keyfile
    DataInputStream in = new DataInputStream(new FileInputStream(f));

```

```

byte[] rawkey = new byte[(int) f.length()];

in.readFully(rawkey);

in.close();


// Convert the raw bytes to a secret key like this
DESedeKeySpec keyspec = new DESedeKeySpec(rawkey);
SecretKeyFactory keyfactory = SecretKeyFactory.getInstance("DESede");
SecretKey key = keyfactory.generateSecret(keyspec);

return key;
}


/**
 * Use the specified TripleDES key to encrypt bytes from the input stream
 * and write them to the output stream. This method uses CipherOutputStream
 * to perform the encryption and write bytes at the same time.
 */
public static void encrypt(SecretKey key, InputStream in, OutputStream out)
    throws NoSuchAlgorithmException, InvalidKeyException,
        NoSuchPaddingException, IOException {
    // Create and initialize the encryption engine
    Cipher cipher = Cipher.getInstance("DESede");
    cipher.init(Cipher.ENCRYPT_MODE, key);

    // Create a special output stream to do the work for us

```

```

CipherOutputStream cos = new CipherOutputStream(out, cipher);

// Read from the input and write to the encrypting output stream

byte[] buffer = new byte[2048];

int bytesRead;

while ((bytesRead = in.read(buffer)) != -1) {

    cos.write(buffer, 0, bytesRead);

}

cos.close();


// For extra security, don't leave any plaintext hanging around memory.

java.util.Arrays.fill(buffer, (byte) 0);

}


/**
 * Use the specified TripleDES key to decrypt bytes ready from the input
 * stream and write them to the output stream. This method uses Cipher
 * directly to show how it can be done without CipherInputStream and
 * CipherOutputStream.
 */

public static void decrypt(SecretKey key, InputStream in, OutputStream out)

    throws NoSuchAlgorithmException, InvalidKeyException, IOException,

    IllegalBlockSizeException, NoSuchPaddingException,

    BadPaddingException {

```

```

// Create and initialize the decryption engine
Cipher cipher = Cipher.getInstance("DESede");
cipher.init(Cipher.DECRYPT_MODE, key);

// Read bytes, decrypt, and write them out.
byte[] buffer = new byte[2048];
int bytesRead;
while ((bytesRead = in.read(buffer)) != -1) {
    out.write(cipher.update(buffer, 0, bytesRead));
}

// Write out the final bunch of decrypted bytes
out.write(cipher.doFinal());
out.flush();
}
}

```

Block.java

```

package block;

import java.util.List;

import public class Block {

```

```
private int previousHash;

private List<Transaction> transactions;


public Block(int previousHash, List<Transaction> transactions) {

    this.previousHash = previousHash;

    this.transactions = transactions;

}


public int getPreviousHash() {

    return previousHash;

}


public void setPreviousHash(int previousHash) {

    this.previousHash = previousHash;

}


public List<Transaction> getTransactions() {

    return transactions;

}


public void setTransactions(List<Transaction> transactions) {

    this.transactions = transactions;

}
```

@Override

```
public boolean equals(Object o) {  
    if (this == o) return true;  
    if (o == null || getClass() != o.getClass()) return false;  
  
    Block block = (Block) o;  
  
    if (previousHash != block.previousHash) return false;  
    return transactions != null ? transactions.equals(block.transactions) :  
    block.transactions == null;  
}
```

@Override

```
public int hashCode() {  
    int result = previousHash;  
    result = 31 * result + (transactions != null ? transactions.hashCode() : 0);  
    return result;  
}  
}
```

CipherHelper.java

```
package algo;  
  
import javax.crypto.Cipher;  
import javax.crypto.SecretKey;  
import javax.crypto.spec.SecretKeySpec;
```

```

public class CipherHelper {

    // Algorithm used

    private final static String ALGORITHM = "DES";

    /**
     * Encrypt data
     * @param secretKey - a secret key used for encryption
     * @param data - data to encrypt
     * @return Encrypted data
     * @throws Exception
     */

    public String cipher(String secret, String data) throws Exception {

        // Key has to be of length 8

        if (secretKey == null || secretKey.length() != 8)

            throw new Exception("Invalid key length - 8 bytes key needed!");

        SecretKey key = new SecretKeySpec(secretKey.getBytes(),
ALGORITHM);

        Cipher cipher = Cipher.getInstance(ALGORITHM);

        cipher.init(Cipher.ENCRYPT_MODE, key);

        return toHex(cipher.doFinal(data.getBytes()));

    }

```

```

/**
 * Decrypt data
 * @param secretKey - a secret key used for decryption
 * @param data - data to decrypt
 * @return Decrypted data
 * @throws Exception
 */

public String decipher(String secret, String data) throws Exception {

    // Key has to be of length 8

    if (secretKey == null || secretKey.length() != 8)

        throw new Exception("Invalid key length - 8 bytes key needed!");

    SecretKey key = new SecretKeySpec(secretKey.getBytes(),
ALGORITHM);

    Cipher cipher = Cipher.getInstance(ALGORITHM);

    cipher.init(Cipher.DECRYPT_MODE, key);

    System.out.println(cipher.doFinal(toByte(data)));

    return new String(cipher.doFinal(toByte(data)));

}

// Helper methods

private static byte[] toByte(String hexString) {

    int len = hexString.length()/2;

```

```

byte[] result = new byte[len];

for (int i = 0; i < len; i++)

    result[i] = Integer.valueOf(hexString.substring(2*i, 2*i+2),
16).byteValue());

return result;
}

public static String toHex(byte[] stringBytes) {

    StringBuffer result = new StringBuffer(2*stringBytes.length);

    for (int i = 0; i < stringBytes.length; i++) {

result.append(HEX.charAt((stringBytes[i]>>4)&0x0f)).append(HEX.charAt(stringBytes[i]&0x0f));

    }

    return result.toString();

}

String secretKey="12345678";

private final static String HEX = "0123456789ABCDEF";

public static void main(String[] args) {

    try {

```

```

        String secretKey = "01234567";

        String data="test";

    } catch (Exception e) {

        e.printStackTrace();

    }

}

}

```

UploadFile.java

```

/*
 * To change this license header, choose License Headers in Project
Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.File;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import static java.lang.System.out;

import java.security.MessageDigest;

import java.security.NoSuchAlgorithmException;

import java.sql.Connection;

```

```
import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.sql.Statement;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.List;

import java.util.logging.Level;

import java.util.logging.Logger;

import java.util.regex.Matcher;

import java.util.regex.Pattern;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.annotation.WebServlet;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.FileItemFactory;

import org.apache.commons.fileupload.FileUploadException;
```

```

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;

/**
 *
 * @author ADMIN
 */
@WebServlet(urlPatterns = {"/UploadFile"})
public class UploadFile extends HttpServlet {

    /**
     * Processes requests for both HTTP <code>GET</code> and
     <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    protected void processRequest(HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException, NoSuchAlgorithmException,
        ClassNotFoundException {

        response.setContentType("text/html;charset=UTF-8");

```

```

try {

    PrintWriter out = response.getWriter();

    HttpSession session=request.getSession();

    ServletContext sc=request.getSession().getServletContext();

    List<String> ms = new ArrayList<String>();

    String finalimage = "";

    boolean isMultipart = ServletFileUpload.isMultipartContent(
        request);

    List<String> m = new ArrayList<String>();

    File savedFile;

    if (!isMultipart) {

        System.out.println("File Not Uploaded");
    }

    else {

        FileItemFactory factory = new DiskFileItemFactory();

        ServletFileUpload upload = new ServletFileUpload(factory);

        List items = null;

        try {

            items = upload.parseRequest(request);

```

```

    }

    catch (FileUploadException e)

    {

        e.printStackTrace();

    }

    Iterator itr = items.iterator();

    while (itr.hasNext()) {

        List<String> al = new ArrayList<String>();

        String sss = "";

        FileItem item = (FileItem) itr.next();

        String value = "";

        String a[];

        if (item.isFormField()) {

            String name = item.getFieldName();

            //System.out.println("name: "+name+"\n");

            value = item.getString();

            // System.out.println("value: "+value);

            al.add(value);

            for (int i = 0; i < al.size(); i++) {

                sss += al.get(i);

                System.out.println("is this image" + sss);

```

```

    }

    a = sss.split("-");
    for (int i = 0; i < a.length; i++) {

        String am = a[i];

        System.out.println("aaaaaaaaaaaaaaaa" + a[i]);

        m.add(a[i]);
    }
} else {

    String itemName = item.getName();

    String reg = "[.*]";

    String replacingtext = "";

    Pattern pattern = Pattern.compile(reg);

    Matcher matcher = pattern.matcher(itemName);

    StringBuffer buffer = new StringBuffer();

    while (matcher.find())
    {

```

```

        matcher.appendReplacement(buffer, replacingtext);
    }

    int IndexOf = itemName.indexOf(".");
    int Indexf = itemName.indexOf(".");
    String domainName = itemName.substring(IndexOf);

    finalimage = buffer.toString() + domainName;

    System.out.println("Final Image===" + finalimage);

    ms.add(finalimage);

    ms.get(0);

    String fn=ms.get(0);

    System.out.println("trying to put all in store");

    savedFile = new File(sc.getRealPath("profile")+"\\ "+finalimage);

    System.out.println(savedFile);

    System.out.println("Completed");

    try
    {
        item.write(savedFile);
    } catch (Exception e) {
        e.printStackTrace();
    }

}

```

```

    }
}

System.out.println("Connection Started");

Class.forName("com.mysql.jdbc.Driver");

Connection
con=DriverManager.getConnection("jdbc:mysql://localhost/block","root","root");

System.out.println("Connection execute");

Statement st=null,st1=null;

ResultSet rs=null;

int id1=0;

String prevdata="";

st = con.createStatement();

ResultSet rs1 = st.executeQuery("select * from blockchain");

System.out.println(rs1);

while(rs1.next()) {

id1=rs1.getInt(1)+1;

prevdata=rs1.getString(2);

System.out.println(prevdata);

}

java.util.Date date = new java.util.Date() ;

SimpleDateFormat dateFormat = new
SimpleDateFormat("yyyy-MM-dd HH:mm") ;

dateFormat.format(date);

System.out.println(dateFormat.format(date));

```

```

String data1 = prevdata+dateFormat.format(date);

StringBuffer sb=new StringBuffer();

MessageDigest md =
MessageDigest.getInstance("MD5");

md.update(String.valueOf(data1).getBytes());

byte byteData[] = md.digest();

//convert the byte to hex format method 1

for (int i = 0; i < byteData.length; i++) {
    sb.append(Integer.toString((byteData[i] & 0xff)
+ 0x100, 16).substring(1));
}

File file = new File("C://blockchain//block"+id1+".txt");

System.out.println(file);

FileWriter writer = new FileWriter(file);

writer.write(sb.toString());

writer.close();

System.out.println("Hashkey:"+sb.toString());

Statement st2 = con.createStatement();

```

```
int add1=st2.executeUpdate("insert into blockchain  
values(""+id1+"",""+sb.toString()+"");
```

```
String query = "update upload set hashkey = ""+sb.toString()+" ";  
System.out.println(query);
```

```
System.out.println(" create PrepareStatment");
```

```
PreparedStatement ps = null;
```

```
//1st step
```

```
System.out.println("1st Step");
```

```
ps =con.prepareStatement("insert into  
upload(did,cate,sub,dname,dprice,des,image)values(""+m.get(0)+"",""+m.get(1)  
+"",""+m.get(2)+"",""+m.get(3)+"",""+m.get(4)+"",""+m.get(5)+"",""+ms.get(0)+"");
```

```
System.out.println(ps);
```

```
ps.executeUpdate();
```

```
System.out.println(ps);
```

```
//2nd Step
```

```
//3rd Step
```

```
PreparedStatement ps1 = null;
```

```

ps1 =con.prepareStatement("update upload set hashkey='"+sb.toString()+" ");

    System.out.println(ps1);

ps1.executeUpdate();

    System.out.println(1);

        out.println("<script>"

                +"alert('Added Sucessfully')"

                +"</script>");

        RequestDispatcher rd=request.getRequestDispatcher("apro.jsp");

        rd.include(request, response);

    } catch (SQLException ex) {

        out.println("<script>"

                +"alert('Fail')"

                +"</script>");

        RequestDispatcher rd=request.getRequestDispatcher("apro.jsp");

        rd.include(request, response);

    }

}

```

// <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">

/**

* Handles the HTTP <code>GET</code> method.

```

*

* @param request servlet request

* @param response servlet response

* @throws ServletException if a servlet-specific error occurs

* @throws IOException if an I/O error occurs

*/

@Override

protected void doGet(HttpServletRequest request, HttpServletResponse
response)

    throws ServletException, IOException {

    try {

        processRequest(request, response);

    } catch (NoSuchAlgorithmException ex) {

        Logger.getLogger(UploadFile.class.getName()).log(Level.SEVERE,
null, ex);

    } catch (ClassNotFoundException ex) {

        Logger.getLogger(UploadFile.class.getName()).log(Level.SEVERE,
null, ex);

    }

}

}

/**

* Handles the HTTP <code>POST</code> method.

*

* @param request servlet request

```

```

* @param response servlet response

* @throws ServletException if a servlet-specific error occurs

* @throws IOException if an I/O error occurs

*/

@Override

protected void doPost(HttpServletRequest request, HttpServletResponse
response)

    throws ServletException, IOException {

    try {

        processRequest(request, response);

    } catch (NoSuchAlgorithmException ex) {

        Logger.getLogger(UploadFile.class.getName()).log(Level.SEVERE,
null, ex);

    } catch (ClassNotFoundException ex) {

        Logger.getLogger(UploadFile.class.getName()).log(Level.SEVERE,
null, ex);

    }

}

}

/**

* Returns a short description of the servlet.

*

* @return a String containing servlet description

*/

@Override

```

```
public String getServletInfo() {  
    return "Short description";  
} // </editor-fold>  
  
}
```