**Blockchain-DPOS-Land-Management System:**

**Group-31:**

Ravindra Rao – 2020B4A42239H

Tangirala Ashish Raj – 2020A3PS1757H

Saurjyesh Hota – 2020B2AA2462H

Paryetri Banerjee – 2020B1A82001H

**Project Description:**

This project aims to develop a system for supply chain management with the following characteristics:

To register new distributors to the system to buy products from the manufacturer.

Register new clients to the system to buy products from the distributor if they can sell that product.

Delegated Proof of Stake (DPOS) consensus algorithm is incorporated to improve the security of the blockchain.

We are implementing the Merkle tree to calculate the root hash of all the transactions inside a block.

Users can view the transaction history that is related to a product.

Technologies Used:

* JavaScript
* Node.js

Packages Used:

* Prompt-sync
* qrcode-terminal
* crypto-js/sha256

**Working of Every Module:**

**blockchain.js file:**

Blockchain (): This function initializes an empty array called blockchain, containing all the blocks in sequential order. It will also initialize an empty array called newTransaction, including all transactions chronologically. It will also create the genesis block.

addBlock ()**:** It will have parameters of transaction data, hash of the previous block, timestamp, and soup of the current block. It will create a new block with all its data and add it to the array "blockchain."

getLastBlock (): This will return the last block in the array "blockchain."

createNewTrans (): This will take property ID, seller ID, and buyer ID as parameters. It will create a new structure called 'newTransactions' with the seller, buyer, and property ID. This transaction will be added to the array 'newTransaction.'

calculateHash (): This will take the previous block's hash, Merkel root, and data as parameters. This will use the sha256 algorithm to calculate the block's hash using all the parameters and current time.

printBlockchain (): Prints all the transactions in the blockchain mined till now.

DPOS: Delegated Proof of Stake is a blockchain consensus mechanism where network users vote and elect delegates to validate the next block. Like a traditional proof-of-stake mechanism, DPoS uses a collateral staking system. However, it also uses a specific democratic process to address POS's limitations.

startVote (): Initializes all the validators votes to zero before calling the voteValidators function.

addValidator (): adds clients or distributors to the validators array during registration.

voteValidators (): All the validators vote to delegate the responsibility of validating a transaction before adding it to the blockchain.

selectValidators (): The top 1/3 validators are selected to validate the transactions.

isChainValid (): This function compares each block's hashes with its hash and the prevHash stored in league with the previous block's hash to validate the blockchain.

validateBlock (): It considers the majority of 2/3 of people delegated to validate the blocks; if true, the block is added to the blockchain.

merkelRoot (): It takes several transactions as a parameter. It implements the Merkel tree to calculate the Merkel root.

**registration.js file:**

Registration (): It initializes the clients and distributors array, which will contain the IDs of all the registered users. It will also initialize the minimum deposit required to register as a client or distributor.

addClient (): This function will check if the client ID requested by the user is available in the client array, and if it is not present, it will register the user by adding his ID to the client array.

addDistributor (): This function will check if the distributor ID requested by the user is available in the client array, and if it is not present, it will register the user by adding his ID to the distributor array.

**transaction.js ():**

This file initializes the product array containing their IDs and a price array for each product that the manufacturer has. It also has an array allTransactions to store all the transactions before they are validated and mined into the blockchain.

**test.js:**

It will include all the files in the project and give them object names.

It will create instances of all the js files, which will then be used to access all the functions from these files.

This file creates a menu that displays all the options to use the project's features, like registering distributors and clients, buying, selling, mining the block, displaying the QR Code, and settling the disputes in case any exist. This file is to be run to use this project.

To run this file, use the command “node test.js” in the terminal.