Blockchain Technology Lab

Lab – 4

Aim : Implement PoW Consensus Mechanism on your own Blockchain

Code:

```
import hashlib
import json
import os


class Student:
    def __init__(self, name, rollnumber, branch):
        self.name = name
        self.rollnumber = rollnumber
        self.branch = branch

    def to_dict(self):
        return {
            "name": self.name,
            "rollnumber": self.rollnumber,
            "branch": self.branch
        }

    @classmethod
    def from_dict(cls, data):
        return cls(data["name"], data["rollnumber"], data["branch"])


class Block:
    def __init__(self, block_id, nonce, student, previous_hash):
        self.block_id = block_id
        self.nonce = nonce
        self.data = student.to_dict()
        self.previous_hash = previous_hash
        self.block_hash = self.calculate_hash()

    def calculate_hash(self):
        block_dict = {
            "block_id": self.block_id,
            "nonce": self.nonce,
            "data": self.data,
            "previous_hash": self.previous_hash
        }
        block_string = json.dumps(block_dict, sort_keys=True).encode()
        return hashlib.sha256(block_string).hexdigest()

    def to_dict(self):
        return {
            "block_id": self.block_id,
```

```
            "nonce": self.nonce,
            "data": self.data,
            "previous_hash": self.previous_hash,
            "block_hash": self.block_hash
        }

    @classmethod
    def from_dict(cls, block_data):
        student_data = Student.from_dict(block_data["data"])
        block = cls(
            block_id=block_data["block_id"],
            nonce=block_data["nonce"],
            student=student_data,
            previous_hash=block_data["previous_hash"]
        )
        block.block_hash = block_data["block_hash"]
        return block


class Blockchain:
    def __init__(self, difficulty, filename="blockchain.json"):
        self.chain = []
        self.difficulty = difficulty
        self.filename = filename
        if os.path.exists(self.filename):
            self.load_chain()
        else:
            self.create_genesis_block()

    def create_genesis_block(self):
        genesis_block = Block(
            block_id=1, nonce=0, student=Student("Genesis", "0000", "Genesis"),
previous_hash='0')
        self.chain.append(genesis_block)
        self.save_chain()

    def add_block(self, student):
        previous_block = self.chain[-1]
        nonce = self.proof_of_work(student, previous_block.block_hash)
        new_block = Block(len(self.chain) + 1, nonce,
                          student, previous_block.block_hash)
        self.chain.append(new_block)
        self.save_chain()
        return new_block

    def proof_of_work(self, student, previous_hash):
        nonce = 0
        while True:
            block_candidate = Block(
                len(self.chain) + 1, nonce, student, previous_hash)
            block_hash = block_candidate.calculate_hash()
            if block_hash[:self.difficulty] == '0' * self.difficulty:
```

```
                return nonce
            nonce += 1

    def validate_chain(self):
        for i in range(1, len(self.chain)):
            current_block = self.chain[i]
            previous_block = self.chain[i - 1]

            # Recalculate the current block's hash and compare
            if current_block.block_hash != current_block.calculate_hash():
                print(f"Invalid hash at block {current_block.block_id}")
                return False

            # Check if the previous hash matches
            if current_block.previous_hash != previous_block.block_hash:
                print(
                    f"Invalid previous hash at block {current_block.block_id}")
                return False

        return True

    def save_chain(self):
        with open(self.filename, 'w') as file:
            json.dump([block.to_dict()
                       for block in self.chain], file, indent=4)
        print(f"Blockchain saved to {self.filename}.")

    def load_chain(self):
        with open(self.filename, 'r') as file:
            chain_data = json.load(file)
            self.chain = [Block.from_dict(block_data)
                          for block_data in chain_data]
        print(f"Blockchain loaded from {self.filename}.")

    def display_chain(self):
        for block in self.chain:
            print(f"Block ID: {block.block_id}")
            print(f"Nonce: {block.nonce}")
            print(f"Data: {block.data}")
            print(f"Previous Hash: {block.previous_hash}")
            print(f"Block Hash: {block.block_hash}\n")


def menu():
    blockchain = Blockchain(difficulty=4)

    while True:
        print("\nBlockchain Menu")
        print("1. Add a block")
        print("2. Display blockchain")
        print("3. Validate blockchain")
        print("4. Exit")
```

```
        choice = input("Enter your choice (1-4): ")

        if choice == '1':
            name = input("Enter student's name: ")
            rollnumber = input("Enter student's roll number: ")
            branch = input("Enter student's branch: ")
            student = Student(name, rollnumber, branch)
            blockchain.add_block(student)
            print("Block added successfully!\n")
        elif choice == '2':
            blockchain.display_chain()
        elif choice == '3':
            if blockchain.validate_chain():
                print("Blockchain is valid.")
            else:
                print("Blockchain is invalid.")
        elif choice == '4':
            print("Exiting...")
            break
        else:
            print("Invalid choice. Please try again.")


if __name__ == "__main__":
    menu()
```

Output:

View chain

```
Blockchain Menu
1. Add a block
2. Display blockchain
3. Validate blockchain
4. Exit
Enter your choice (1-4): 2
Block ID: 1
Nonce: 0
Data: {'name': 'Genesis', 'rollnumber': '0000', 'branch': 'Genesis'}
Previous Hash: 0
Block Hash: 9f04d274200855de615f645370ce593db4c8656c8bb9b837160f6a66908c46cc
```

Add new block

```
Blockchain Menu
1. Add a block
2. Display blockchain
3. Validate blockchain
4. Exit
Enter your choice (1-4): 1
Enter student's name: Ansh Raiyani
Enter student's roll number: 21BCP391
Enter student's branch: CSE
Blockchain saved to blockchain.json.
Block added successfully!
```

```
[      You, 3 weeks ago • .
    {
        "block_id": 1,
        "nonce": 0,
        "data": {
            "name": "Genesis",
            "rollnumber": "0000",
            "branch": "Genesis"
        },
        "previous_hash": "0",
        "block_hash": "9f04d274200855de615f645370ce593db4c8656c8bb9b837160f6a66908c46cc"
    },
    {
        "block_id": 2,
        "nonce": 134523,
        "data": {
            "name": "Ansh",
            "rollnumber": "391",
            "branch": "CSE"
        },
        "previous_hash": "9f04d274200855de615f645370ce593db4c8656c8bb9b837160f6a66908c46cc",
        "block_hash": "00002cbfb1229393d177c338ebd03a2a4e36b553cd00da6edb1e3f776e975d18"
    }
]
```


Validating via proof of work

```
Blockchain Menu
1. Add a block
2. Display blockchain
3. Validate blockchain
4. Exit
Enter your choice (1-4): 3
Blockchain is valid.
```