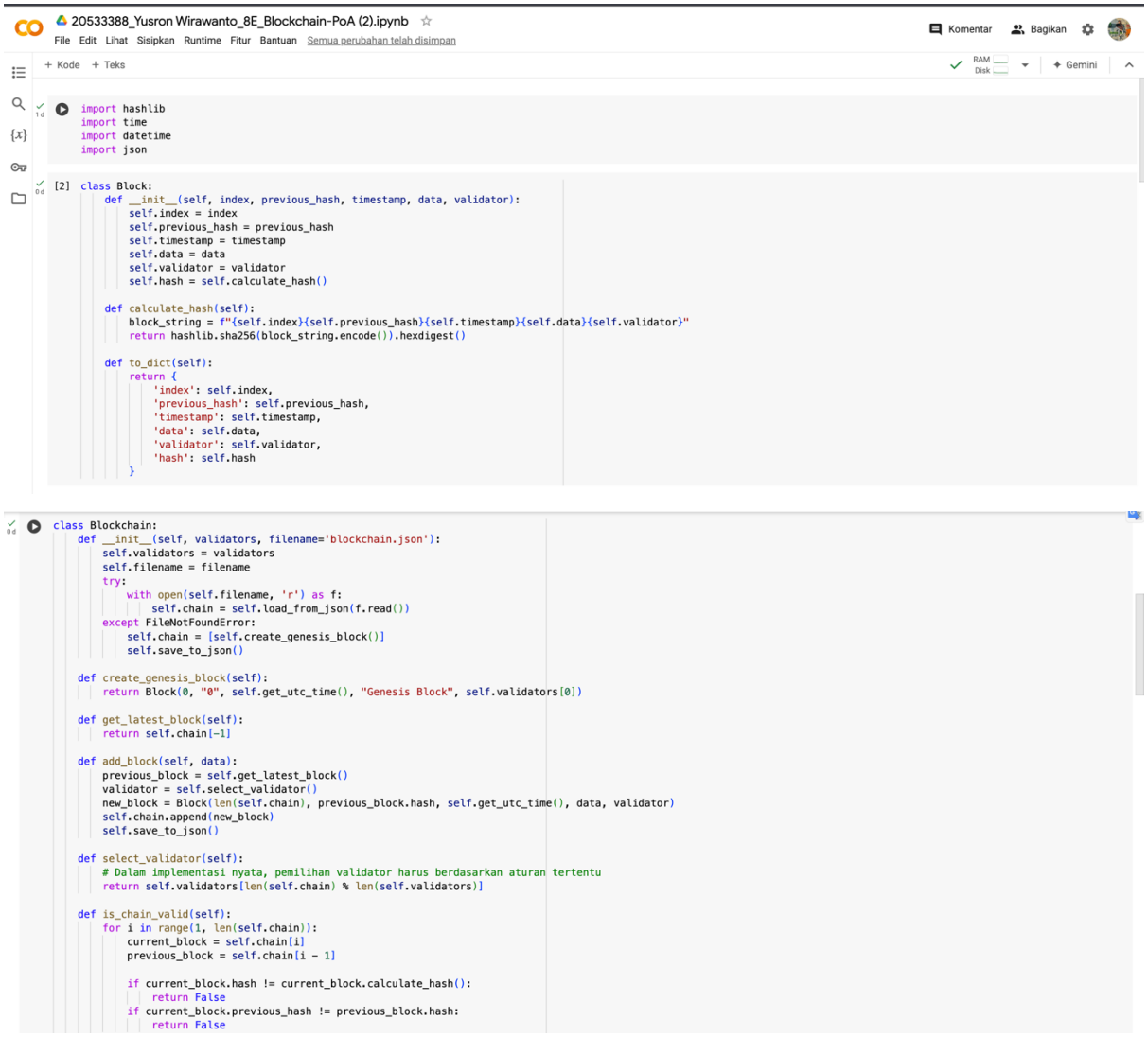
Nama : Yusron Wirawanto

NIM : 20533388

Kelas : TI 8E

BLOCKCHAIN PROOF OF AUTHORITY (POA)



```
import hashlib
import time
import datetime
import json
```

```
class Block:
    def __init__(self, index, previous_hash, timestamp, data, validator):
        self.index = index
        self.previous_hash = previous_hash
        self.timestamp = timestamp
        self.data = data
        self.validator = validator
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        block_string = f"{self.index}{self.previous_hash}{self.timestamp}{self.data}{self.validator}"
        return hashlib.sha256(block_string.encode()).hexdigest()

    def to_dict(self):
        return {
            'index': self.index,
            'previous_hash': self.previous_hash,
            'timestamp': self.timestamp,
            'data': self.data,
            'validator': self.validator,
            'hash': self.hash
        }
```

```
class Blockchain:
    def __init__(self, validators, filename='blockchain.json'):
        self.validators = validators
        self.filename = filename
        try:
            with open(self.filename, 'r') as f:
                self.chain = self.load_from_json(f.read())
        except FileNotFoundError:
            self.chain = [self.create_genesis_block()]
            self.save_to_json()

    def create_genesis_block(self):
        return Block(0, "0", self.get_utc_time(), "Genesis Block", self.validators[0])

    def get_latest_block(self):
        return self.chain[-1]

    def add_block(self, data):
        previous_block = self.get_latest_block()
        validator = self.select_validator()
        new_block = Block(len(self.chain), previous_block.hash, self.get_utc_time(), data, validator)
        self.chain.append(new_block)
        self.save_to_json()

    def select_validator(self):
        # Dalam implementasi nyata, pemilihan validator harus berdasarkan aturan tertentu
        return self.validators[len(self.chain) % len(self.validators)]

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

            if current_block.hash != current_block.calculate_hash():
                return False
            if current_block.previous_hash != previous_block.hash:
                return False
```

```
        return True

    def get_utc_time(self):
        return datetime.datetime.utcnow().timestamp()

    def display_chain(self):
        with open(self.filename, 'r') as f:
            blockchain_data = json.load(f)
            for block_data in blockchain_data:
                # Remove 'hash' from block_data as it's calculated internally
                block_data.pop('hash', None)
                block = Block(**block_data)  # Membuat objek Block dari data JSON
                print(f"Block {block.index}:")
                print(f"  Timestamp: {datetime.datetime.utcfromtimestamp(block.timestamp)}")
                print(f"  Data: {block.data}")
                print(f"  Validator: {block.validator}")
                print(f"  Hash: {block.hash}")
                print(f"  Previous Hash: {block.previous_hash}\n")

    def save_to_json(self):
        with open(self.filename, 'w') as f:
            json.dump([block.to_dict() for block in self.chain], f, indent=4)

    def load_from_json(self, json_string):
        data = json.loads(json_string)
        blocks = []
        for block_data in data:
            # Remove 'hash' from block_data as it's calculated internally
            block_data.pop('hash', None)
            block = Block(**block_data)  # Membuat objek Block dari data JSON
            blocks.append(block)
        return blocks
```

```
[4] validators = ["Yusron", "Yusuf", "Yohan", "Yara", "Yuni", "Yudith", "Yanto", "Yasmine", "Yasmin", "Yosua"]
    blockchain = Blockchain(validators)
```

```
[5] blockchain.add_block("Data untuk blok 1")
    blockchain.add_block("Data untuk blok 2")
    blockchain.add_block("Data untuk blok 3")
    blockchain.add_block("Data untuk blok 4")
    blockchain.add_block("Data untuk blok 5")
    blockchain.add_block("Data untuk blok 6")
    blockchain.add_block("Data untuk blok 7")
    blockchain.add_block("Data untuk blok 8")
    blockchain.add_block("Data untuk blok 9")
    blockchain.add_block("Data untuk blok 10")
```

```
[6] blockchain.display_chain()
```

```
[6] Block 0:
      Timestamp: 2024-06-28 12:26:12.000909
      Data: Genesis Block
      Validator: Yusron
      Hash: a969a617529ac583602eccc2ff52059e8cee99771e8eb8b84cd8e7a5562e863b
      Previous Hash: 0

    Block 1:
      Timestamp: 2024-06-28 12:26:12.958566
      Data: Data untuk blok 1
      Validator: Yusuf
      Hash: f05f5c9aad6c6ce0563df0aed8b13efd1da58460b8427a08e6897af67b05eea1
      Previous Hash: a969a617529ac583602eccc2ff52059e8cee99771e8eb8b84cd8e7a5562e863b

    Block 2:
      Timestamp: 2024-06-28 12:26:12.959180
      Data: Data untuk blok 2
      Validator: Yohan
      Hash: 01b1d36d23b6997ce6ed588d7bc9038de56bba11cea0704106bcc5df95e13543
      Previous Hash: f05f5c9aad6c6ce0563df0aed8b13efd1da58460b8427a08e6897af67b05eea1

    Block 3:
      Timestamp: 2024-06-28 12:26:12.960193
      Data: Data untuk blok 3
      Validator: Yara
      Hash: c80acf90f91562260e9bfd9c6a3d0f4877932758571 70f3039909710312e134
      Previous Hash: 01b1d36d23b6997ce6ed588d7bc9038de56bba11cea0704106bcc5df95e13543

    Block 4:
      Timestamp: 2024-06-28 12:26:12.960579
      Data: Data untuk blok 4
      Validator: Yuni
      Hash: 32787f2f1996e5a5a39d51c4db3b042891795e4884bd968b45df32e32e9bc5e2
      Previous Hash: c80acf90f91562260e9bfd9c6a3d0f4877932758571 70f3039909710312e134

    Block 5:
      Timestamp: 2024-06-28 12:26:12.963144
      Data: Data untuk blok 5
      Validator: Yudith
      Hash: d835ac55f4e1fc4473b5b3781767e787ed6034eac701f9c91e12cd650fc2d610
      Previous Hash: 32787f2f1996e5a5a39d51c4db3b042891795e4884bd968b45df32e32e9bc5e2
```

```
Block 6:
  Timestamp: 2024-06-28 12:26:12.963601
  Data: Data untuk blok 6
  Validator: Yanto
  Hash: 20e17643dba283af25ac7c95a34d5ef350ee5fa4aa0fdf1792a51fe5b21d46f5
  Previous Hash: d835ac55f4e1fc4473b5b3781767e787ed6034eac701f9c91e12cd650fc2d610

Block 7:
  Timestamp: 2024-06-28 12:26:12.964650
  Data: Data untuk blok 7
  Validator: Yasmine
  Hash: 75a3e010128bba3d4afb1051d4d02b273e3f50602d6d0f72b32e09cf3b258e8f
  Previous Hash: 20e17643dba283af25ac7c95a34d5ef350ee5fa4aa0fdf1792a51fe5b21d46f5

Block 8:
  Timestamp: 2024-06-28 12:26:12.965157
  Data: Data untuk blok 8
  Validator: Yasmin
  Hash: c3d99f4188d964c91cb5d0bf84d6a3ca6f85818d0b58dec4384f6eba87727226
  Previous Hash: 75a3e010128bba3d4afb1051d4d02b273e3f50602d6d0f72b32e09cf3b258e8f

Block 9:
  Timestamp: 2024-06-28 12:26:12.965600
  Data: Data untuk blok 9
  Validator: Yosua
  Hash: e55d7069a53837f3160eaaee531c4fb90a12a3e9808f3b5778402d5d510582ed
  Previous Hash: c3d99f4188d964c91cb5d0bf84d6a3ca6f85818d0b58dec4384f6eba87727226

Block 10:
  Timestamp: 2024-06-28 12:26:12.966071
  Data: Data untuk blok 10
  Validator: Yusron
  Hash: c9ef39a86a89cbdc6d4a29d3c0716d49c958c1a7cf609f06b9957d84b32eb5bd
  Previous Hash: e55d7069a53837f3160eaaee531c4fb90a12a3e9808f3b5778402d5d510582ed
```

Blockchain di atas menunjukkan implementasi Proof of Authority (PoA) dengan 11 blok, dimulai dari blok Genesis hingga blok 10. Dalam sistem PoA ini, validator yang ditunjuk bertanggung jawab untuk memvalidasi dan menambahkan blok baru. Setiap blok memiliki struktur yang konsisten: timestamp, data, validator, hash, dan previous hash. Blok Genesis memiliki previous hash 0, sedangkan blok-blok berikutnya menggunakan hash blok sebelumnya, membentuk rantai yang tak terputus. Data pada setiap blok bervariasi, dengan blok Genesis berisi "Genesis Block" dan blok lainnya berisi "Data untuk blok X". Validator berbeda-beda untuk setiap blok, termasuk Yusron, Yusuf, Yohan, Yara, Yuni, Yudith, Yanto, Yasmine, Yasmin, dan Yosua, dengan Yusron muncul dua kali (di blok Genesis dan blok 10).

Blockchain PoA ini mendemonstrasikan beberapa karakteristik kunci. Pertama, kehadiran validator yang ditunjuk menunjukkan bahwa ini adalah jaringan yang diotorisasi, di mana identitas validator diketahui dan dipercaya. Kedua, kecepatan pembuatan blok dengan selisih waktu dalam milidetik menunjukkan efisiensi tinggi dalam konsensus dan pembuatan blok, yang merupakan keunggulan PoA. Ketiga, rotasi validator antar blok menunjukkan distribusi tanggung jawab dan menghindari sentralisasi kekuasaan. Keempat, integritas data dijamin melalui penggunaan hash dan linking antar blok. Meskipun blockchain ini tampaknya dibuat untuk tujuan demonstrasi atau pengujian, ia secara efektif mengilustrasikan prinsip-prinsip utama blockchain PoA: kecepatan, efisiensi, dan keandalan dengan validator yang dikenal dan dipercaya.