Mini Task 1: Build & Explain a Simple Blockchain

Theoretical Part

1. Blockchain Basics

Q: What is blockchain, in simple, everyday words?
A: Imagine a notebook that you and your friends pass around. Each of you writes down transactions—like who paid whom—on a new page. Once a page is full, you lock it with a special signature and pass it on. No one can change a locked page without redoing everyone's signatures afterward, which nobody wants to do. That's blockchain: a digital notebook shared around the world, where each 'page' (or block) is permanently sealed before moving on.

Real-Life Uses:

• *Tracking Groceries:* Ever wonder where your salad greens came from? Blockchain can show every stop from farm to store. • *Protecting Your ID:* Think of it like a digital passport that only you control, making it super tough for identity thieves.

2. Block Anatomy

Q: Draw and explain a block's main parts.
A: Here's a visual you can doodle:

```
+----------------------------+
|         Block              |
+----------------------------+
| Data: Your transactions    |
| Timestamp: When it happened|
| Nonce: Number we tweak      |
| Previous Hash: Last page ID|
| Merkle Root: Overall receipt|
+----------------------------+
```

The Merkle Root is like a master checksum: if even one transaction on a page changes, this root changes, so you instantly know something's fishy. No need to reread every line—just check the summary.

3. Consensus Mechanisms

Proof of Work (PoW): Think of it as a brain teaser contest—computers race to solve a puzzle. The winner gets to add the next page and earns tokens. But running this contest nonstop uses a ton of electricity.

Proof of Stake (PoS): Instead of who works hardest, it's like who has the most stake in a game. The more coins you hold, the more likely you'll be picked to add the page—using far less energy.

Delegated PoS (DPoS): Picture a student council election. Coin holders vote for a few trusted delegates to write the next page. Fast, democratic, but you need to trust your delegates.

Practical Part: Code Simulations

1. Block Simulation in Code

We'll build a tiny chain of three blocks so you can see how they link together and why tampering one breaks the rest.

```
import hashlib
import time
```

```
class Block:
    def __init__(self, index, data, previous_hash):
        self.index = index
        self.timestamp = time.time()
        self.data = data
        self.previous_hash = previous_hash
        self.nonce = 0
        self.hash = self.calculate_hash()

    def calculate_hash(self):
        return hashlib.sha256(f"{self.index}{self.timestamp}{self.data}{self.previous_hash}{self.nonc

# Build chain
chain = [Block(0, 'Genesis', '0')]
chain.append(Block(1, 'Second', chain[0].hash))
chain.append(Block(2, 'Third', chain[1].hash))
for b in chain:
    print(f"Block {b.index}: Hash={b.hash}, Prev={b.previous_hash}")
```

2. Nonce Mining Simulation

Let’s pretend we need a hash that starts with '0000'. We’ll keep tweaking the nonce until we hit the target.

```
class BlockMiner(Block):
    def mine(self, difficulty):
        target = '0'*difficulty
        while not self.hash.startswith(target):
            self.nonce += 1
            self.hash = self.calculate_hash()
        return self.nonce

start = time.time()
b = BlockMiner(0, 'MineMe', '0')
attempts = b.mine(4)
print(f"Nonce={b.nonce}, Attempts={attempts}, Time={time.time()-start:.2f}s")
```

3. Consensus Mechanisms Demo

We’ll play out PoW, PoS, and DPoS with mock players to see who gets to 'write the page'.

```
# PoW
powers = {'A': random.randint(1,100), 'B': random.randint(1,100)}
winner_pow = max(powers, key=powers.get)
print(f"PoW Winner: {winner_pow}")

# PoS
stakes = {'X': random.randint(1,100), 'Y': random.randint(1,100)}
winner_pos = max(stakes, key=stakes.get)
print(f"PoS Winner: {winner_pos}")

# DPoS
votes = {'D1': 3, 'D2': 2}
winner_dpos = max(votes, key=votes.get)
print(f"DPoS Winner: {winner_dpos}")
```