## **Mini Task 1: Build & Explain a Simple Blockchain**

## **Goal:**

Understand blockchain fundamentals, block structure, and consensus mechanisms by simulating a mini blockchain and explaining how it works — both technically and conceptually.

### **📌 Task Instructions:**

#### **Theoretical Part:**

1. **Blockchain Basics**
   * Define blockchain in your own words (100–150 words).
   * List 2 real-life use cases (e.g., supply chain, digital identity).
2. **Block Anatomy**
   * Draw a block showing: data, previous hash, timestamp, nonce, and Merkle root.
   * Briefly explain with an example how the Merkle root helps verify data integrity.
3. **Consensus Conceptualization**
   * Explain in brief (4–5 sentences each):  
     + What is Proof of Work and why does it require energy?
     + What is Proof of Stake and how does it differ?
     + What is Delegated Proof of Stake and how are validators selected?

## **Practical Part (Code-Based Tasks)**

### **1. Block Simulation in Code**

**Objective:** Build a basic blockchain with 3 linked blocks using code.

#### **Task:**

* Create a Block class with:  
  + index, timestamp, data, previousHash, hash, and nonce
* Implement a simple hash generator using SHA-256
* Link 3 blocks by chaining their previousHash
* Display all blocks with their hashes

#### **Challenge:**

* Change the data of Block 1 and recalculate its hash.
* Observe how all following blocks become invalid unless hashes are recomputed.

*Goal:* Understand how tampering one block affects the entire chain.

### **2. Nonce Mining Simulation**

**Objective:** Simulate Proof-of-Work by mining a block that satisfies a difficulty condition.

#### **Task:**

* Modify your Block class to include a mineBlock(difficulty) function
* Set difficulty (e.g., hash must start with "0000")
* In mineBlock(), repeatedly increment the nonce until the hash meets the difficulty condition

#### **Output:**

* Print how many nonce attempts were needed
* Measure time taken using a timer

*Goal:* Experience how computational effort increases with difficulty

### **3. Consensus Mechanism Simulation**

**Objective:** Simulate and compare PoW, PoS, and DPoS logic in code.

#### **Task:**

* Create mock objects for 3 validators:  
  + miner = {power: random value} for PoW
  + staker = {stake: random value} for PoS
  + voters = [3 mock accounts voting] for DPoS

#### **Simulate:**

* For PoW: Select validator with highest power
* For PoS: Select validator with highest stake
* For DPoS: Randomly choose a delegate based on most votes

#### **Output:**

* Print selected validator and consensus method used
* Include a console.log explanation of the selection logic

*Goal:* Compare decision-making in various consensus mechanisms

### **Submission Instructions:**

* Submit a GitHub repo or folder with:  
  + blockchain\_simulation.js or .py (3 linked blocks)
  + mining\_simulation.js or .py (nonce task)
  + consensus\_demo.js or .py (PoW, PoS, DPoS logic)
* Include brief comments and console logs explaining your output