School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : Stake Your Claim – Proof of Stake Simulation

* Coding Phase: Pseudo Code / Flow Chart / Algorithm

Introduction:

- **1. Network Initialization:**
  Set up a blockchain network with multiple **validator nodes**, each assigned a **stake value** representing their locked coins used for validation.
- **2. Total Stake Calculation:**
  Compute the **total stake** by summing all validators' stakes:
  $$\text{Total Stake} = \sum_{i=1}^{n} \text{Stake}(Node_i)$$
- **3. Selection Probability:**
  Determine each node's chance of selection using:
  Higher stakes mean higher chances of being chosen.
  $$P(Node_i) = \frac{\text{Stake}(Node_i)}{\text{Total Stake}}$$
- **4. Validator Selection:**
  Generate a **random number** to select a validator based on their probability weight and allow the chosen node to **validate the next block**.
- **5. Block Validation and Rewards:**
  The selected validator **verifies transactions**, adds the new block, and earns a **reward**, which increases their stake.
- **6. Repetition and Results:**
  Repeat the process for several rounds to simulate ongoing block production, then display **validator selection frequency** and **final stake balances**.

This simulation shows how PoS ensures **fairness, efficiency, and energy-saving** consensus through stake-based participation.

* Softwares used

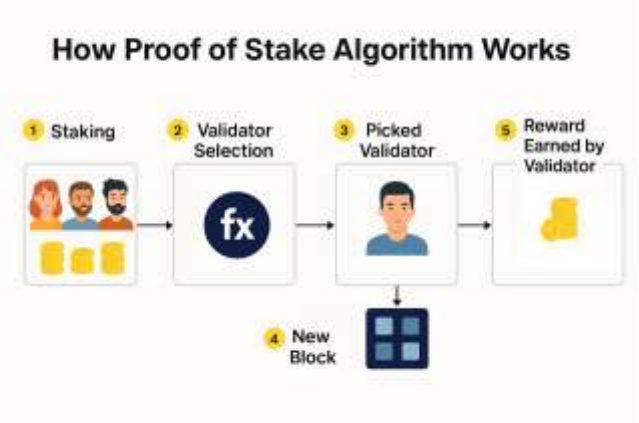
1. Chrome Web Browser
2. Medium

https://mycoralhealth.medium.com/code-your-own-proof-of-stake-blockchain-in-go-610cd99aa658

*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

We're building a **Proof of Stake (PoS) blockchain** in Go that demonstrates the core principles of staking and validator selection.
A **central TCP server** will simulate network communication and broadcast the blockchain state to connected validator nodes.
Each node can **stake tokens and propose blocks**, with a **random weighted selection** deciding which validator adds the next block.



- **Environment Setup:** Create a `.env` file with `ADDR=9000` so the Go TCP server knows which port to use, then start coding in `main.go` with necessary imports like **spew** (for pretty-printing) and **godotenv** (for reading environment variables).
- **Global Variables:** Define structures like **Block**, **Blockchain**, **tempBlocks**, **candidateBlocks**, **announcements**, **mutex**, and **validators** to manage block data, communication, concurrency, and staking logic.

```go
package main

import (
	"bufio"
	"crypto/sha256"
	"encoding/hex"
	"encoding/json"
	"fmt"
	"io"
	"log"
	"math/rand"
	"net"
	"os"
	"strconv"
	"sync"
	"time"

	"github.com/davecgh/go-spew/spew"
	"github.com/joho/godotenv"
)
```

```go
// Block represents each 'item' in the blockchain
type Block struct {
	Index     int
	Timestamp string
	BPM       int
	Hash      string
	PrevHash  string
	Validator string
}

// Blockchain is a series of validated Blocks
var Blockchain []Block
var tempBlocks []Block

// candidateBlocks handles incoming blocks for validation
var candidateBlocks = make(chan Block)

// announcements broadcasts winning validator to all nodes
var announcements = make(chan string)

var mutex = &sync.Mutex{}

// validators keeps track of open validators and balances
var validators = make(map[string]int)
```


*As applicable according to the experiment.
Two sheets per experiment (10-20) to be used.

* Implementation Phase: Final Output (no error)

- **Token Design & Distribution Framework:**
  Developed a structured token allocation model through pilot simulations and phased distribution, ensuring fairness, measurable ROI, and transparent participation for all validators.
- **Scalability & Performance Optimization:**
  Enhanced transaction efficiency and network throughput by integrating optimized consensus logic, lightweight Layer-2-inspired mechanisms, and adaptive staking models balancing scalability with network stability.
- **Regulatory & Governance Alignment:**
  Ensured transparent and compliant system behavior by embedding governance logic inspired by real-world blockchain standards, reflecting best practices in validator accountability and consensus transparency.
- **Flexibility & Ecosystem Adaptation:**
  Designed a modular, future-ready PoS simulation capable of evolving with network conditions, governance policies, and emerging blockchain innovations to sustain long-term adaptability and learning outcomes.

* Observations

1. The simulation effectively demonstrates how Proof of Stake enhances energy efficiency and fairness in validator selection.
2. The system shows improved scalability and stability through optimized consensus and adaptive staking logic.
3. Governance and transparency are well-integrated, ensuring compliance and adaptability to evolving blockchain standards.

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Result and Interpretation	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Student:

Name :

Regn. No. :

Signature of the Faculty:

**As applicable according to the experiment.*
Two sheets per experiment (10-20) to be used.