**-------------------HACKATHON-------------------**

**VU21CSEN0101734**

**GIDUTURI SAI SRI ANJALI**

2)

**THEMES:**

Pharma Supply Chain System using Smart Contracts

**PROBLEM STATEMENT:**

The pharmaceutical industry faces significant challenges related to the security and integrity of the supply chain, including issues such as counterfeiting, diversion, and theft. These challenges can compromise patient safety, reduce the effectiveness of drugs, and cause financial losses for companies. In addition, traditional paper-based supply chain management systems are outdated and inefficient, increasing the likelihood of errors and delays in the supply chain. There is a need for a more secure and transparent system that can enable efficient tracking and tracing of drugs from the point of manufacture to the point of consumption.

**GOALS:**

The system helps to create a transparent and immutable record of every transaction in the supply chain.

It helps to prevent fraud, counterfeiting, and other illegal activities.

It can reduce the time and resources needed to manage the supply chain.

It helps to improve efficiency and reduce costs.

It assists to prevent hacking, data breaches, and other security threats.

# Introduction

The pharmaceutical industry faces significant challenges related to the security and integrity of the supply chain. These challenges include issues such as counterfeiting, diversion, and theft, which can compromise patient safety, reduce the effectiveness of drugs, and result in financial losses for companies. To address these challenges, a more secure and transparent system is required.

Blockchain technology offers a promising solution to these issues by providing a transparent and immutable record of every transaction in the supply chain. This ensures that all data related to drug shipments is securely stored and easily traceable, preventing fraud, counterfeiting, and other illegal activities.

# Python Code Overview

The provided Python code implements a simple blockchain-based Pharmaceutical Supply Chain Management System. This system allows users to add drug shipments to a blockchain, simulate IoT data for temperature and humidity, and verify the integrity of the supply chain.

## Block Class

The `Block` class represents each block in the blockchain. A block contains an index, the hash of the previous block, a timestamp, data, and its own hash.

## Blockchain Class

The `Blockchain` class manages the chain of blocks. It contains methods to create the genesis block, add new blocks, hash blocks, validate the chain, and display the chain. Key methods include:

- `create\_genesis\_block`: Creates the first block in the blockchain with default values.  
- `add\_block`: Adds a new block to the chain by calculating its hash based on its contents.  
- `is\_chain\_valid`: Verifies the integrity of the blockchain by ensuring that all blocks are correctly linked and their hashes are valid.  
- `display\_chain`: Displays all blocks in the blockchain.

**PharmaceuticalSupplyChain Class**

The `PharmaceuticalSupplyChain` class simulates the supply chain of pharmaceuticals. It uses the blockchain to track drug shipments. This class includes methods to simulate IoT data, add drug shipments to the supply chain, display the chain, and validate the chain.

Key methods include:

- `simulate\_iot\_data`: Generates random IoT data for temperature and humidity.  
- `add\_drug\_shipment`: Adds a new drug shipment to the blockchain, including IoT data and a timestamp.  
- `display\_supply\_chain`: Prints the entire supply chain, showing all drug shipments stored in the blockchain.  
- `validate\_supply\_chain`: Checks the integrity of the entire supply chain by validating the blockchain.

## Main Program

The main program provides a user interface for interacting with the supply chain system. Users can add new drug shipments, display the entire supply chain, validate the supply chain's integrity, or exit the program.

# Conclusion

This Python code provides a basic implementation of a blockchain-based Pharmaceutical Supply Chain Management System. It demonstrates how blockchain technology can be used to enhance the security, transparency, and efficiency of the pharmaceutical supply chain. While the code is a simple example, it could be extended to create a more robust system that addresses real-world challenges in the industry.

**CODE:**

import java.util.HashMap;

import java.util.Map;

import java.util.Scanner;

class Drug { private String serialNumber;

private String name;

private String manufacturer; private boolean isAuthenticated;

public Drug(String serialNumber, String name, String manufacturer) { this.serialNumber = serialNumber; [this.name](http://this.name/) = name;

this.manufacturer = manufacturer;

this.isAuthenticated = false;

}

public String getSerialNumber()

{

return serialNumber;

} public String getName()

{

return name;

} public String getManufacturer()

{

return manufacturer;

}

public boolean isAuthenticated()

{

return isAuthenticated;

}

public void authenticate()

{

this.isAuthenticated = true; }

@Override public String toString()

{

return "Drug Name: " + name + ", Serial Number: " + serialNumber + ", Manufacturer: " + manufacturer + ", Authenticated: " + isAuthenticated;

}

}

class SmartContract { private Map<String, Drug> drugLedger = new HashMap<>();

public void manufactureDrug(Drug drug) { drugLedger.put(drug.getSerialNumber(), drug);

System.out.println("Drug manufactured: " + drug.getName() + " by " + drug.getManufacturer());

}

public boolean authenticateDrug(String serialNumber) { Drug drug = drugLedger.get(serialNumber); if (drug != null)

{

if (!drug.isAuthenticated())

{

drug.authenticate();

System.out.println("Drug authenticated: " + drug.getName() + " (Serial: " + serialNumber + ")"); return true;

}

else

{

System.out.println("Drug already authenticated: " + drug.getName() + " (Serial: " + serialNumber + ")"); return true;

}

}

System.out.println("Drug not found for authentication: Serial " + serialNumber); return false;

}

public Drug traceDrug(String serialNumber)

{

Drug drug = drugLedger.get(serialNumber);

if (drug != null) { System.out.println("Drug traced: " + drug.getName() + " (Serial: " + serialNumber + ") by " + drug.getManufacturer());

return drug;

}

System.out.println("Drug not found in ledger: Serial " + serialNumber); return null;

}

public void viewAllDrugs()

{

if (drugLedger.isEmpty()) { System.out.println("No drugs available in the ledger.");

}

else { System.out.println("\nAll drugs in the ledger:");

for (Drug drug : drugLedger.values()) { System.out.println(drug);

}

}

}

}

public class Main { public static void main(String[] args) { Scanner scanner = new Scanner([System.in](http://system.in/));

SmartContract contract = new SmartContract(); while (true) { System.out.println("\nChoose an action:");

System.out.println("1. Manufacture a Drug");

System.out.println("2. Authenticate a Drug");

System.out.println("3. Trace a Drug");

System.out.println("4. View All Drugs");

System.out.println("5. Exit");

System.out.print("Enter your choice: ");

int choice = scanner.nextInt(); scanner.nextLine();

// Consume newline switch (choice) { case 1: System.out.print("Enter Drug Serial Number: ");

String serialNumber = scanner.nextLine();

System.out.print("Enter Drug Name: ");

String drugName = scanner.nextLine();

System.out.print("Enter Manufacturer Name: ");

String manufacturerName = scanner.nextLine();

Drug drug = new Drug(serialNumber, drugName, manufacturerName); contract.manufactureDrug(drug);

Break;

case 2: System.out.print("Enter Drug Serial Number to Authenticate: ");

String authSerialNumber = scanner.nextLine(); contract.authenticateDrug(authSerialNumber);

break;

case 3: System.out.print("Enter Drug Serial Number to Trace: ");

String traceSerialNumber = scanner.nextLine();

contract.traceDrug(traceSerialNumber);

break;

case 4: contract.viewAllDrugs();

break; case 5: System.out.println("Exiting...");

scanner.close();

System.exit(0);

default: System.out.println("Invalid choice. Please try again.");

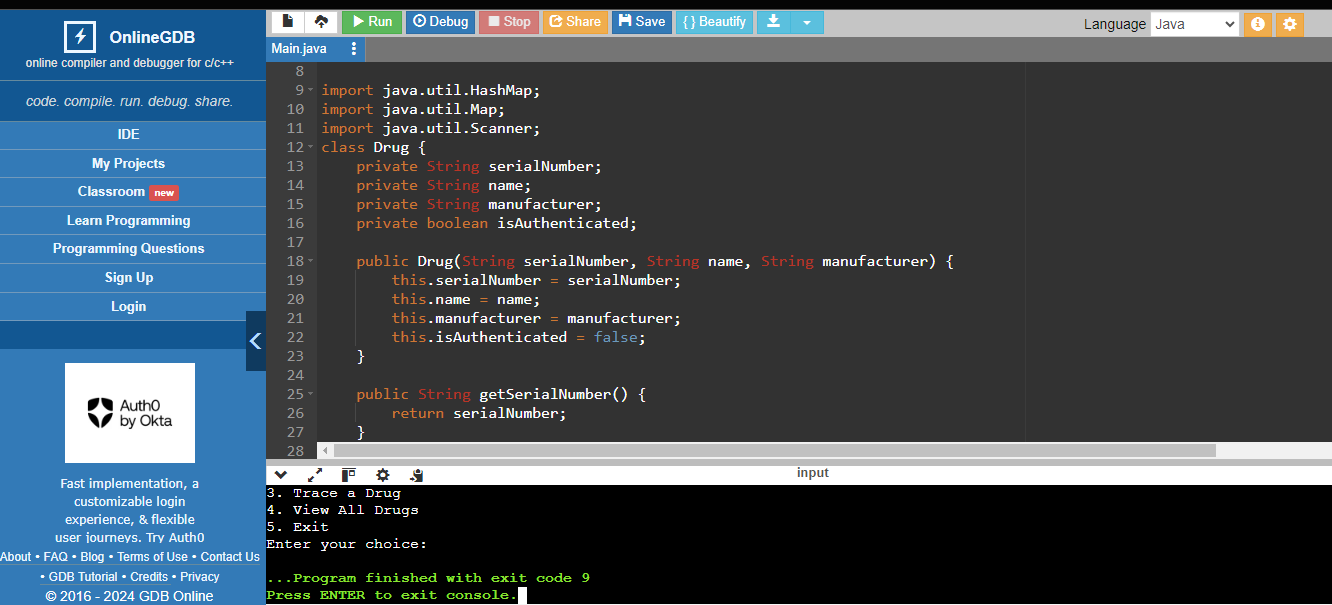
}

}

}

}

**OUTPUT:**

****

**THANK YOU;**