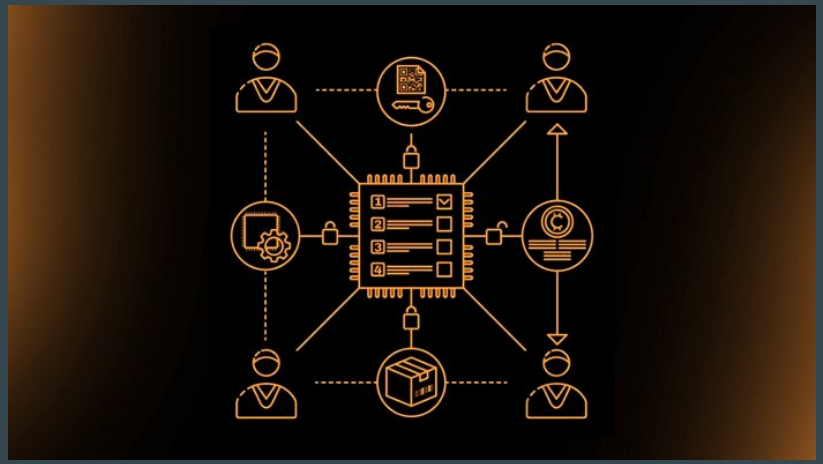
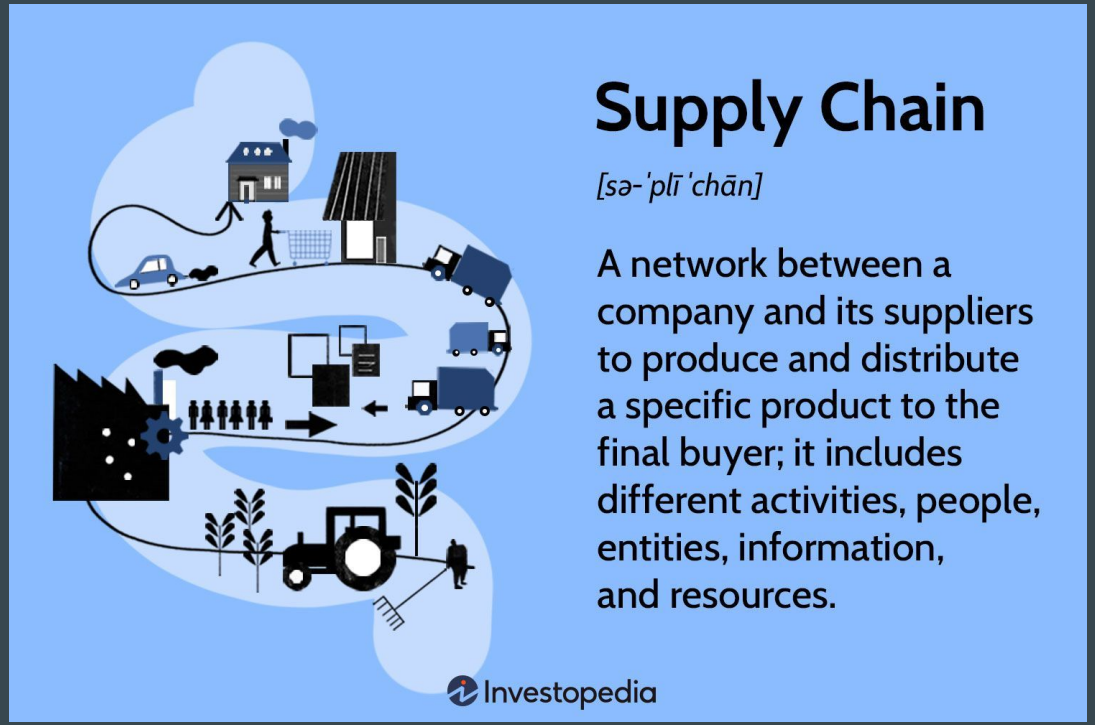
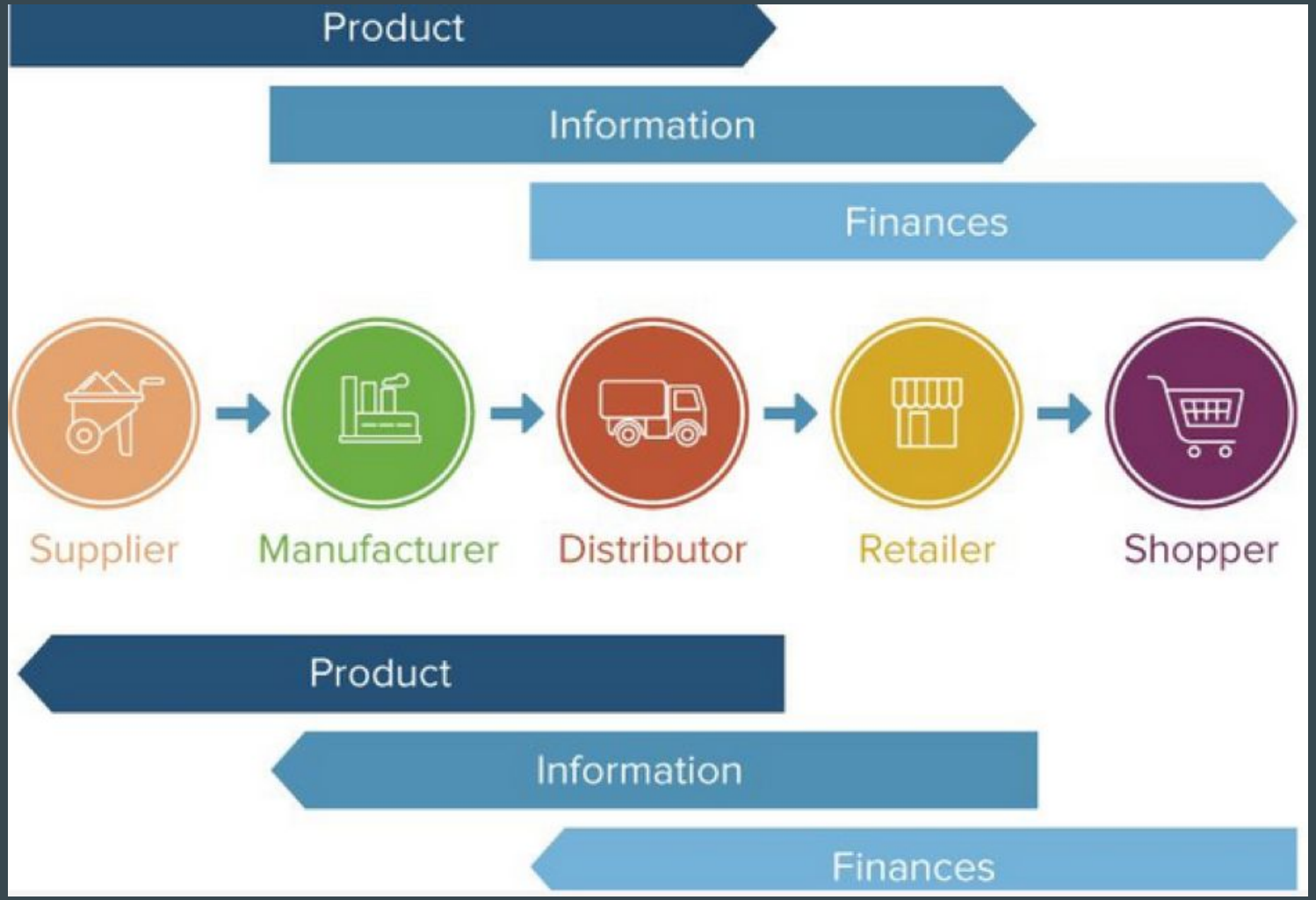
Smart Contract Project: Tackling the Supply Chain

By Shane Rodricks and Ivan Orlovic



What is the supply chain?




Product
Information
Finances
Supplier
Manufacturer
Distributor
Retailer
Shopper
Product
Information
Finances

What Can Go Wrong?

1. Unreliable suppliers:
    a. Suppliers that do not deliver on time, do not meet quality standards, or are otherwise unreliable can cause significant disruptions to the supply chain.
2. Inefficient logistics:
    a. Poor logistics management can result in lost or damaged goods, missed delivery deadlines, and increased costs.
3. Lack of Visibility:
    a. A lack of visibility into the supply chain can make it difficult to quickly identify and address problems when they arise.
4. Disruption of Transportation:
    a. Disruptions to transportation, such as natural disasters, strikes, or political unrest, can cause significant disruptions to the supply chain.
5. Changing Market Conditions:
    a. Fluctuations in demand, changes in regulations, and shifts in consumer preferences can all cause problems in the supply chain.

Inspiration

Our Proposal:

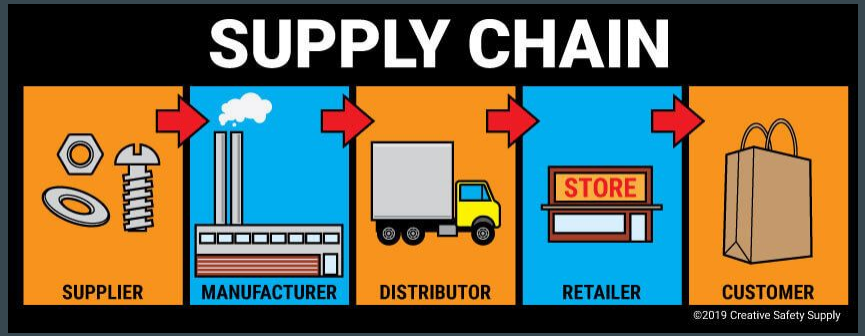
- An open-source blockchain-chain based supply chain management software system.

Objective:

- The objective of this project is to be able to track products/material as it moves through the supply chain.

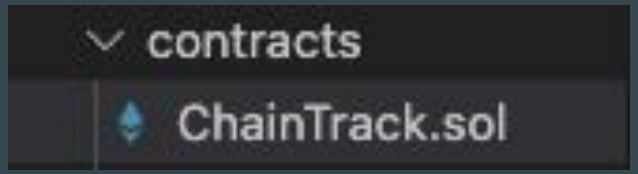
Diving Deeper

- The idea behind this is to be able to track material, through the use of blockchain verification, as it moves through the supply chain and to cut out the human error that may be involved in much of today's supply chain problems.
- We aimed to manage the transfer of the material from one source to another source.
- In addition to this, we provided a last visited destination and current destination of material within the supply chain.



Our Implementation:

- In this project, we designed an architecture that considers the following: manufacturers, distributors, retailers, and consumers.
- We map the materials that are being sent
  - Each Material gets a description: Counter, ID, description, LastRole, LastVisited, currentDestination



OUR CODE

CONTRACT

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract ChainTrack {

    // Define roles
    enum Role { Manufacturer, Distributor, Retailer, Consumer }

    // Define a Material
    struct Material {
        uint256 id;
        string description;
        Role lastRole;
        address lastVisited;
        address currentDestination;
    }

    // Mapping of material ID to Material
    mapping(uint256 => Material) public materials;

    // Mapping of user addresses to roles
    mapping(address => Role) public users;

    // Material counter
    uint256 public materialCounter;

    // Events
    event MaterialCreated(uint256 id, string description);
    event MaterialMoved(uint256 id, address indexed from, address indexed to, Role fromRole, Role toRole);

    // Modifier to check if a user has a specific role
    modifier onlyRole(Role role) {
        require(users[msg.sender] == role, "You don't have the required role");
        _;
    }

    // Assign a role to a user
    function setUserRole(address user, Role role) public {
        users[user] = role;
    }
```

```
    // Create a new material
    function createMaterial(string memory description) public onlyRole(Role.Manufacturer) {
        materialCounter++;
        materials[materialCounter] = Material({
            id: materialCounter,
            description: description,
            lastRole: Role.Manufacturer,
            lastVisited: msg.sender,
            currentDestination: msg.sender
        });
        emit MaterialCreated(materialCounter, description);
    }

    // Add this function to ChainTrack.sol
    function getMaterial(uint256 _id) public view returns (Material memory) {
        return materials[_id];
    }


    // Transfer a material to the next role in the supply chain
    function transferMaterial(uint256 id, address to) public {
        require(materials[id].currentDestination == msg.sender, "Material not at your location");

        // Ensure the next role in the supply chain is valid
        Role currentRole = users[msg.sender];
        Role nextRole = users[to];
        require(
            (currentRole == Role.Manufacturer && nextRole == Role.Distributor) ||
            (currentRole == Role.Distributor && nextRole == Role.Retailer) ||
            (currentRole == Role.Retailer && nextRole == Role.Consumer),
            "Invalid role transfer"
        );

        // Update material details
        materials[id].lastRole = currentRole;
        materials[id].lastVisited = msg.sender;
        materials[id].currentDestination = to;

        emit MaterialMoved(id, msg.sender, to, currentRole, nextRole);
    }
}
```

TEST

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

//import "remix_tests.sol"; // This import is required for Remix testing
import "../contracts/ChainTrack.sol"; // Adjust the path to your ChainTrack.sol file

contract ChainTrackTest {
    ChainTrack chainTrack;

    // Deploy a new ChainTrack contract before each test
    function beforeEach() public {
        chainTrack = new ChainTrack();
    }

    // Test creating a new material
    function testCreateMaterial() public {
        chainTrack.setUserRole(address(this), ChainTrack.Role.Manufacturer);
        chainTrack.createMaterial("Material 1");
        ChainTrack.Material memory material = chainTrack.materials(1);

        require(material.id == 1, "Material ID should be 1");
        require(material.description == "Material 1", "Material description should be 'Material 1'");
        require(uint(material.lastRole) == 0, "Material lastRole should be Manufacturer");
        require(material.lastVisited == address(this), "Material lastVisited should be the test contract address");
        require(material.currentDestination == address(this), "Material currentDestination should be the test contract address");

    }

    // function getMaterial(uint256 _id) public view returns (Material memory) {
    //     return materials[_id];
    // }

    // Test transferring a material to the next role
    function testTransferMaterial() public {
        chainTrack.setUserRole(address(this), ChainTrack.Role.Manufacturer);
        chainTrack.createMaterial("Material 1");
        chainTrack.setUserRole(address(0x1), ChainTrack.Role.Distributor);

        chainTrack.transferMaterial(1, address(0x1));
        ChainTrack.Material memory material = chainTrack.getMaterial(1);

        assert.equal(material.lastRole, 0, "Material lastRole should be Manufacturer");
        assert.equal(material.lastVisited, address(this), "Material lastVisited should be the test contract address");
        assert.equal(material.currentDestination, address(0x1), "Material currentDestination should be address(0x1)");
    }
}
```

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.4;

import "forge-std/Test.sol";
import "contracts/CountContract.sol";

contract ContractTest is Test {
    CountContract countContract;
    function setUp() public {
        countContract = new CountContract(10);
    }

    function testIncrement() public {
        countContract.increment();
        assertEq(countContract.count(), 11);
    }

    function testDecrement() public {
        countContract.decrement();
        assertEq(countContract.count(), 9);
    }

    function testSetCount() public {
        countContract.setCount(20);
        assertEq(countContract.count(), 20);
    }
}
```

REMIX

Deployed Contracts

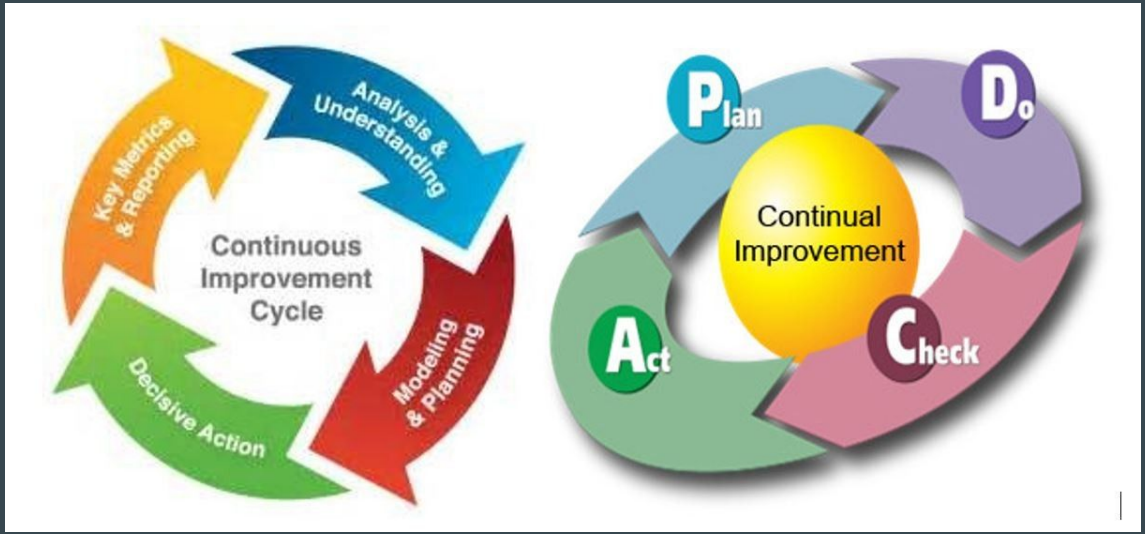
CHAINTRACK AT 0XD91...39138

Balance: 0 ETH

createMaterial	string description
setUserRole	address user, uint8 role
transferMateri	uint256 id, address to
materialCount	
materials	uint256
users	address

In the Future

- We were not able to connect to the Goerli Etherscan (yet)
- This provided us with no gas and thus failed us on some tests that we ran



FINDINGS

```
⊗ (base) ivanorlovic@Ivans-MacBook-Pro-2 contracts % npm run test

  > smartcontractframework@1.0.0 test
  > forge test -vvv

  [⠊] Compiling...
  [⠊] Compiling 29 files with 0.8.10
  [⠊] Solc 0.8.10 finished in 2.51s
  Compiler run successful

  Running 2 tests for test/ChainTrack.t.sol:ChainTrackTest
  [FAIL. Reason: EvmError: Revert] testCreateMaterial() (gas: 5089)
  Traces:
    [5089] ChainTrackTest::testCreateMaterial()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: EvmError: Revert] testTransferMaterial() (gas: 5111)
  Traces:
    [5111] ChainTrackTest::testTransferMaterial()
      └─ ← "EvmError: Revert"

  Test result: FAILED. 0 passed; 2 failed; finished in 4.12ms

  Running 3 tests for test/CountContract.t.sol:ContractTest
  [PASS] testDecrement() (gas: 11177)
  [PASS] testIncrement() (gas: 11230)
  [PASS] testSetCount() (gas: 11237)
  Test result: ok. 3 passed; 0 failed; finished in 4.15ms

  Running 13 tests for test/Treasury.t.sol:TreasuryTest
  [FAIL. Reason: EvmError: Revert] testGetBalance() (gas: 8323)
  Traces:
    [8323] TreasuryTest::testGetBalance()
      ├─ [0] VM::deal(0x0000000000000000000000000000000000000001, 800000000000000000)
      │   └─ ← ()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: EvmError: Revert] testInitializeOwner() (gas: 5187)
  Traces:
    [5187] TreasuryTest::testInitializeOwner()
      ├─ [0] 0x0000000000000000000000000000000000000000::owner() [staticcall]
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: Call reverted as expected, but without data] testInvalidDepositAmount() (gas: 8778)
  Traces:
    [8778] TreasuryTest::testInvalidDepositAmount()
      ├─ [0] VM::deal(0x0000000000000000000000000000000000000001, 500000000000000000)
      │   └─ ← ()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      ├─ [0] VM::expectRevert(Treasury: Deposit amount should be greater than zero)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: Call reverted as expected, but without data] testNotOwnerTransferOwnership() (gas: 8386)
  Traces:
    [8386] TreasuryTest::testNotOwnerTransferOwnership()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      ├─ [0] VM::expectRevert(0x4f776e61626c653a2063616c6c6572206973206e6f7420746865206f776e6572)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: Call reverted as expected, but without data] testNotOwnerWithdraw() (gas: 8379)
  Traces:
    [8379] TreasuryTest::testNotOwnerWithdraw()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      ├─ [0] VM::expectRevert(0x4f776e61626c653a2063616c6c6572206973206e6f7420746865206f776e6572)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: Call reverted as expected, but without data] testNotOwnerWithdrawAll() (gas: 8347)
  Traces:
    [8347] TreasuryTest::testNotOwnerWithdrawAll()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      ├─ [0] VM::expectRevert(0x4f776e61626c653a2063616c6c6572206973206e6f7420746865206f776e6572)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: EvmError: Revert] testOwnerTransferOwnership() (gas: 10046)
  Traces:
    [10046] TreasuryTest::testOwnerTransferOwnership()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000007)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: EvmError: Revert] testSuccessfulDeposit() (gas: 8323)
  Traces:
    [8323] TreasuryTest::testSuccessfulDeposit()
      ├─ [0] VM::deal(0x0000000000000000000000000000000000000001, 500000000000000000)
      │   └─ ← ()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: EvmError: Revert] testSuccessfulWithdraw() (gas: 8344)
  Traces:
    [8344] TreasuryTest::testSuccessfulWithdraw()
      ├─ [0] VM::deal(0x0000000000000000000000000000000000000001, 500000000000000000)
      │   └─ ← ()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      └─ ← "EvmError: Revert"

  [FAIL. Reason: EvmError: Revert] testSuccessfulWithdrawAll() (gas: 8301)
  Traces:
    [8301] TreasuryTest::testSuccessfulWithdrawAll()
      ├─ [0] VM::deal(0x0000000000000000000000000000000000000001, 800000000000000000)
      │   └─ ← ()
      ├─ [0] VM::prank(0x0000000000000000000000000000000000000001)
      │   └─ ← ()
      └─ ← "EvmError: Revert"
```

```
[FAIL. Reason: Call reverted as expected, but without data] testWithdrawAllNoBalance() (gas: 10521)
Traces:
  [10521] TreasuryTest::testWithdrawAllNoBalance()
    ├─ [0] VM::prank(0x0000000000000000000000000000000000000007)
    │   └─ ← ()
    ├─ [0] VM::expectRevert(0x54726561737572793a204e6f2062616c616e636520746f207769746864726177)
    │   └─ ← ()
    └─ ← "EvmError: Revert"

[FAIL. Reason: Call reverted as expected, but without data] testWithdrawInvalidAmount() (gas: 10463)
Traces:
  [10463] TreasuryTest::testWithdrawInvalidAmount()
    ├─ [0] VM::prank(0x0000000000000000000000000000000000000007)
    │   └─ ← ()
    ├─ [0] VM::expectRevert(Treasury: Not enough balance to withdraw)
    │   └─ ← ()
    └─ ← "EvmError: Revert"

[FAIL. Reason: Call reverted as expected, but without data] testWithdrawReceiverZeroAddress() (gas: 10485)
Traces:
  [10485] TreasuryTest::testWithdrawReceiverZeroAddress()
    ├─ [0] VM::prank(0x0000000000000000000000000000000000000007)
    │   └─ ← ()
    ├─ [0] VM::expectRevert(Treasury: receiver is zero address)
    │   └─ ← ()
    └─ ← "EvmError: Revert"

Test result: FAILED. 0 passed; 13 failed; finished in 4.21ms

Running 14 tests for test/TokenPayable.t.sol:TokenPayableTest
[PASS] testGetTokenBalance() (gas: 57628)
[PASS] testInitializeOwner() (gas: 9821)
[PASS] testNotOwnerTransferOwnership() (gas: 16243)
[PASS] testNotOwnerWithdraw() (gas: 11167)
[PASS] testNotOwnerWithdrawAllToken() (gas: 10917)
[PASS] testOwnerTransferOwnership() (gas: 18369)
[PASS] testSuccessfulDeposit() (gas: 58475)
[PASS] testSuccessfulWithdraw() (gas: 107568)
[PASS] testSuccessfulWithdrawAllTokens() (gas: 107462)
[PASS] testWithdrawAllNoBalanceTokens() (gas: 20713)
[PASS] testWithdrawAllTokensUnapprovedAmount() (gas: 65817)
[PASS] testWithdrawInvalidAmount() (gas: 62728)
[PASS] testWithdrawReceiverZeroAddress() (gas: 13316)
[PASS] testWithdrawUnapprovedAmount() (gas: 65979)
Test result: ok. 14 passed; 0 failed; finished in 3.78ms

Failing tests:
Encountered 2 failing tests in test/ChainTrack.t.sol:ChainTrackTest
[FAIL. Reason: EvmError: Revert] testCreateMaterial() (gas: 5089)
[FAIL. Reason: EvmError: Revert] testTransferMaterial() (gas: 5111)
```

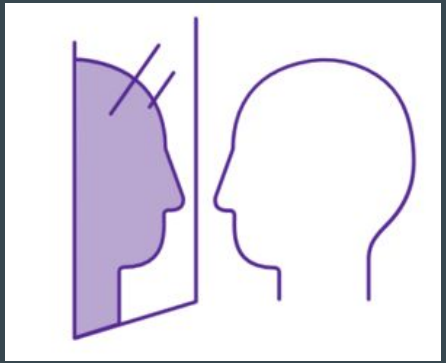
```
Failing tests:
Encountered 2 failing tests in test/ChainTrack.t.sol:ChainTrackTest
[FAIL. Reason: EvmError: Revert] testCreateMaterial() (gas: 5089)
[FAIL. Reason: EvmError: Revert] testTransferMaterial() (gas: 5111)

Encountered 13 failing tests in test/Treasury.t.sol:TreasuryTest
[FAIL. Reason: EvmError: Revert] testGetBalance() (gas: 8323)
[FAIL. Reason: EvmError: Revert] testInitializeOwner() (gas: 5187)
[FAIL. Reason: Call reverted as expected, but without data] testInvalidDepositAmount() (gas: 8778)
[FAIL. Reason: Call reverted as expected, but without data] testNotOwnerTransferOwnership() (gas: 8386)
[FAIL. Reason: Call reverted as expected, but without data] testNotOwnerWithdraw() (gas: 8379)
[FAIL. Reason: Call reverted as expected, but without data] testNotOwnerWithdrawAll() (gas: 8347)
[FAIL. Reason: EvmError: Revert] testOwnerTransferOwnership() (gas: 10046)
[FAIL. Reason: EvmError: Revert] testSuccessfulDeposit() (gas: 8323)
[FAIL. Reason: EvmError: Revert] testSuccessfulWithdraw() (gas: 8344)
[FAIL. Reason: EvmError: Revert] testSuccessfulWithdrawAll() (gas: 8301)
[FAIL. Reason: Call reverted as expected, but without data] testWithdrawAllNoBalance() (gas: 10521)
[FAIL. Reason: Call reverted as expected, but without data] testWithdrawInvalidAmount() (gas: 10463)
[FAIL. Reason: Call reverted as expected, but without data] testWithdrawReceiverZeroAddress() (gas: 10485)

Encountered a total of 15 failing tests, 17 tests succeeded
(base) ivanorlovic@Ivans-MacBook-Pro-2 contracts %
```

Real World Implications:

1. Improved accuracy: Automated systems reduce the risk of human error, ensuring that data is accurate and consistent throughout the supply chain.
2. Increased efficiency: Automated processes can reduce lead times, minimize waste, and increase productivity, leading to improved efficiency and lower costs.
3. Better visibility: Automated systems provide real-time visibility into the supply chain, making it easier to identify and address issues in a timely manner.
4. Enhanced collaboration: Automated systems facilitate communication and collaboration between suppliers, manufacturers, and distributors, helping to build strong relationships and improve overall performance.
5. Improved decision making: Automated systems provide access to data and analytics, allowing companies to make informed decisions and continuously improve their supply chain processes.

Reflection:

- Due to many errors we encountered at the beginning of the project, we found it difficult to get off the ground running.
- We found that after much frustration it was crucial to use all the resources available to us in order to work on the project
- For example, asking CHAT-GPT to write tests or explain error messages to us was very crucial in working on the project



THANK YOU