TEXTILE SUPPLY CHAIN

Overview

The Textile Supply Chain project is a decentralized application (dApp) that enables users to issue, update, and delete digital products on the Ethereum blockchain. This ensures transparency and security in tracking textile products using smart contracts.

1. Smart Contract Structure

1.1 Features

- **Adding a Product**: Manufacturers can register a textile product with details like name, origin, and material. Each product receives a unique ID stored on the blockchain.
- **Tracking Progress**: Products move through four stages:
    - In production, manufacturing starts.
    - Quality Check – Inspected for quality.
    - In Transit – Being transported.
    - Delivered – Reached its final destination.
- **Updating Product Details**: Owners can modify product details and status, ensuring up-to-date records.
- **Deleting a Product**: If a product is not yet delivered, the owner can remove it from the system.

1.2 Security Measures

- **Access Control**: Only authorized manufacturers can create, update, and delete products.
- **Reentrancy Protection**: Uses `ReentrancyGuard` to prevent reentrancy attacks.
- **State Validation**: Ensures product status follows a logical sequence (e.g., cannot revert from Delivered to In Transit).
- **Ownership Restriction**: Only product owners can modify or delete their products.

1.3 Contract Overview

Imports

- `ERC721` – Creates NFTs representing textile products.
- `Ownable` – Restricts certain functions to the contract owner.

- **`ReentrancyGuard –`** Prevents reentrancy attacks.
- **`Counters`** – Generates unique product IDs automatically.

Structs

- **DigitalProduct** – Stores product details such as name, origin, material, production date, and current status.

Functions

- **issueProduct**: Creates a new textile product with a unique ID.
- **updateProduct**: Allows the owner to update product details and change status.
- **deleteProduct**: Removes a product from the blockchain if it is not yet delivered.

Modifiers

- **onlyOwnerOf**: Ensures only the product owner can update or delete it.

1.4 Smart Contract Workflow

Step 1: Compile the Contract

- Navigate to the "Solidity Compiler" tab.
- Set the compiler version to 0.8.20.
- Click "Compile TextileSupplyChain.sol".
- Verify that "Compilation successful" appears.

Step 2: Deploy to Sepolia Testnet

- Go to "Deploy & Run Transactions".
- Set "Environment" to "Injected Provider - MetaMask".
- Connect MetaMask wallet and switch to Sepolia testnet.
- Click "Deploy" next to `TextileSupplyChain`.
- Confirm and pay the deployment fee in MetaMask.
- Copy the contract address: **`0x30979ac99E0D2beEfCA20edb4591B56caA9AbAb2`**.

Step 3: Testing the Contract

- **Issue a Product**: Call **`issueProduct`** with product details.
- **Update a Product**: Call **`updateProduct`** with new details.
- **Delete a Product**: Call **`deleteProduct`** for removal.

---

2. Frontend Structure

2.1 Overview

The frontend is built using **React + ethers.js** and interacts with the deployed smart contract on the Ethereum Sepolia testnet. It enables users to create, update, and delete digital products via a user-friendly interface.

2.2 Features

- **MetaMask Integration**: Users can connect their Ethereum wallet.
- **Issue a New Product**: Users can enter product details and create a new product.
- **Update an Existing Product**: Users can modify product details and update status.
- **Delete a Product**: Users can remove products from the blockchain.

2.3 Security Measures

- **Wallet Connection Check**: Ensures only authenticated users can interact.
- **Gas Limit Optimization**: Limits gas usage to **`500000`** for updates.
- **State Validation**: Prevents unauthorized updates to delivered products.
- **Error Handling**: Displays errors if incorrect inputs are provided.

2.4 Smart Contract Communication

Imports

- `ethers.js`: Facilitates blockchain interaction.
- `React`: Manages state and UI rendering.

State Management

- **User Account Information**: Stores connected MetaMask account.
- **Contract Instance**: Holds smart contract instance.
- **Product Details**: Maintains product data (name, origin, material, etc.).

Functions

- **connectWallet**: Connects MetaMask wallet.
- **issueProduct**: Calls the smart contract to create a new product.
- **updateProduct**: Calls the smart contract to modify an existing product.
- **deleteProduct**: Calls the smart contract to remove a product.

useEffect Hook

- Ensures MetaMask is connected when the user opens the dApp.

2.5 Frontend Workflow

Step 1: Setting Up the Development Environment

Prerequisites

- Install **VS Code**: https://code.visualstudio.com/
- Install **Node.js**: https://nodejs.org/

Verify installations:

node -v # Check Node.js version
npm -v # Check npm version

- git --version # Check Git version

Step 2: Installing & Running the dApp

Installing Dependencies

- npm create vite@latest textile-supply-chain --template react
- cd textile-supply-chain
- npm install react vite ethers openzeppelin
- npm install

Running the Application

npm run dev

- Open in browser: *http://localhost:5172*
- Connect MetaMask and switch to Sepolia Testnet.

Clon the Repository:

git clone: *https://github.com/vinaykumar0103/Textile_Supply_Chain.git*

cd Textile_Supply_Chain

Installing Dependencies

npm install

Running the Application

npm run dev

- Open in browser: *http://localhost:5172*
- Connect MetaMask and switch to Sepolia Testnet.

Step 3: Testing in Browser

Connecting MetaMask

- Open MetaMask and connect the wallet.

- Switch to Sepolia Ethereum Testnet.

Testing dApp Functions

- **Issue a Product**: Enter details and click **Issue Product**.
- **Update a Product**: Enter updated details and click **Update Product**.
- **Delete a Product**: Enter Product ID and click **Delete Product**.

---

3. Summary

The **Textile Supply Chain dApp** is a blockchain-based solution designed to enhance transparency, security, and efficiency in managing textile products. This architecture covers both **smart contract** and **frontend** implementations, ensuring seamless tracking and verification of products on the Ethereum blockchain. The **smart contract** governs product issuance, updates, and deletion with robust security measures, while the **frontend** provides an intuitive interface for user interactions via **React and ethers.js**. This structured approach ensures reliable, real-time supply chain management with on-chain integrity.

GitHub Repository: Textile Supply Chain

Vinay Kumar