

**Course title: MCS7013** [**Collaborative Research Project 1**](https://lawrencetech.instructure.com/courses/15635)

[**Collaborative Research Project 1**](https://lawrencetech.instructure.com/courses/15635)**:   
 Supply Chain Management with Block chain Technology**

**Student Name: Mahesh Vemasani**

**Sunil Kumar Kodi  
 Akhil kanamuri**

**Overview:**

Creating a clear overview document for your supply chain management project involving block chain technology tools like Ether.js, Web3Modal, Next.js, MetaMask, and Hardhat is essential for communicating the project's purpose, components, and implementation details. Here's a structured outline for your overview document:

**Title:**

**Supply Chain Management with Block chain Technology**

The project aims to implement a supply chain management system utilizing block chain technology. It leverages Ethereum block chain and associated tools to ensure transparency, traceability, and security in the supply chain processes.

**Key Components:**

**1. Block chain Infrastructure:**

**Ethereum Block chain:** Utilized as the underlying block chain platform for executing smart contracts and storing immutable transaction records.

**2. Smart Contracts:**

Developed using Solidity: Smart contracts are programmed in Solidity, Ethereum's native programming language, to automate and enforce business logic within the supply chain. Smart contracts are self-executing contracts with the terms of the agreement directly written into code. Deployed on the block chain, smart contracts automatically enforce predefined rules and conditions, eliminating the need for intermediaries and ensuring trustless execution of transactions.

Smart contracts maintain an immutable record of all transactions and interactions on the block chain. Every transaction, such as product movements, deliveries, or payments, is securely recorded on the block chain, providing an auditable and tamper-proof trail of activities throughout the supply chain lifecycle. This transparency enhances accountability, traceability, and compliance with regulatory requirements.

**3. Frontend Application:**

**Built with Next.js:** The frontend interface is developed using Next.js, a React framework, for building fast and scalable web applications.

**4. Block chain Interactions:**

Ether.js and Web3Modal: These tools facilitate interactions with the Ethereum block chain, enabling functionalities such as contract deployment, transaction execution, and data retrieval.

**5. User Authentication:**

MetaMask Integration: MetaMask is a popular Ethereum wallet browser extension that allows users to manage their Ethereum accounts, securely store digital assets, and interact with decentralized applications (daps) directly from their web browser. Integrating MetaMask with your supply chain management system enables seamless authentication and transaction signing, empowering users to engage with the block chain securely.

MetaMask facilitates transaction signing, allowing users to authorize and execute block chain transactions directly from the frontend application. When performing actions such as product transfers, ownership changes, or transaction approvals within the supply chain, users can review and sign transactions using their MetaMask wallet. This ensures that transactions are securely executed on the block chain, providing transparency and audit ability.

MetaMask acts as a bridge between the frontend application and the Ethereum block chain, enabling secure communication and data exchange. Through MetaMask's APIs and communication protocols, the frontend application can interact with the Ethereum network to retrieve data, execute transactions, and listen for events. This integration ensures that sensitive information, such as private keys and transaction data, remains encrypted and protected throughout the communication process.

**6. Development Environment:**

**Hardhat Framework:** Used for Ethereum smart contract development, testing, and deployment. Hardhat streamlines the development workflow by providing essential tools and utilities. Hardhat supports scriptable tasks and plugins, allowing developers to extend its functionality and automate common development tasks. Developers can create custom tasks to perform actions such as compiling contracts, deploying contracts, running tests, and interacting with the Ethereum block chain programmatically. Additionally, Hardhat's plugin ecosystem provides a wide range of integrations and extensions for integrating with external tools, libraries, and services.

**Features:**

**Product Traceability:** Every product within the supply chain is assigned a unique identifier stored on the block chain, enabling end-to-end traceability.

**Immutable Records:** Transaction records, including product movements, transfers, and ownership changes, are securely stored on the block chain, ensuring tamper-proof data integrity.

**Real-time Updates:** Participants in the supply chain can access real-time updates and insights into the status of products and transactions through the frontend interface.

**Secure Transactions:** Transactions within the supply chain are executed securely using block chain technology, minimizing the risk of fraud, counterfeit, and data manipulation.

**Implementation:**

**Smart Contract Design:** Describe the design and functionalities of the smart contracts, including contract structure, data models, and business logic.

**Frontend Development**: Provide an overview of the frontend application's design, user interface components, and interaction flows.

**Integration with Block chain Tools:** Explain how Ether.js, Web3Modal, and MetaMask are integrated into the application to enable block chain interactions and user authentication.

**Testing and Deployment:** Outline the testing strategies employed for smart contracts and frontend components, as well as the deployment process using the Hardhat framework.

**Architecture for Block chain-Based Supply Chain Management**

**Runtime Environment:**

* Utilizes Node.js for server-side execution of JavaScript.
* Manages packages with npm to handle project dependencies.

**Frontend Framework:**

* Leverages Next.js for backend management and frontend rendering.
* Employs React for building interactive user interfaces.

**UI Interaction:**

* Integrates with Web3Modal for connecting block chain wallets to the application.

**Wallet Integration:**

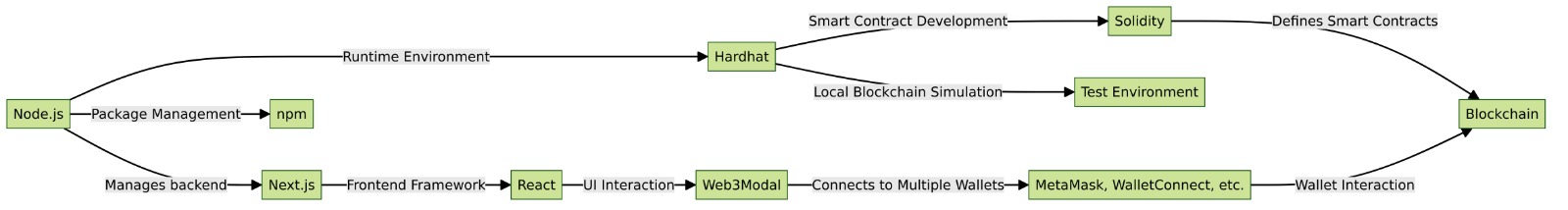
* Supports multiple wallets such as MetaMask and Wallet Connect for user transactions.

**Smart Contract Development:**

* Develops smart contracts with Solidity in the Hardhat environment.
* Simulates a local block chain for testing smart contracts.

**Block chain Deployment:**

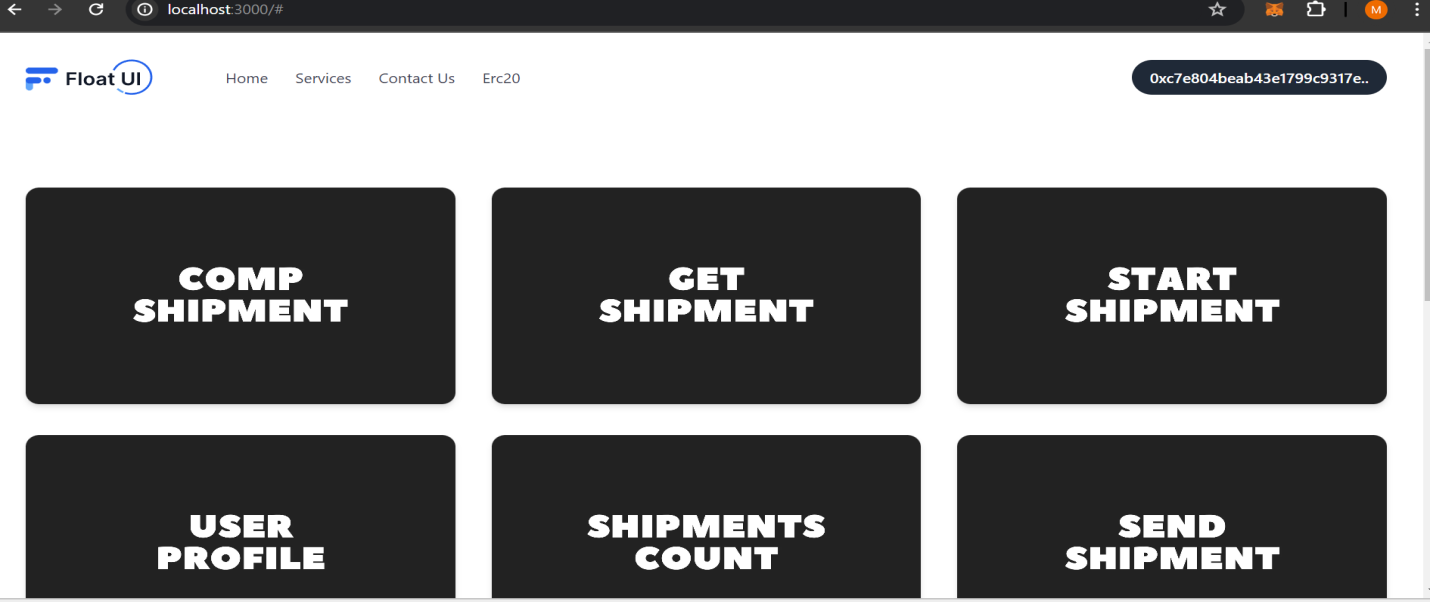
* Tests smart contracts thoroughly in a controlled environment before deployment.
* Deploys smart contracts to the block chain, ensuring immutable record-keeping and transaction verification in supply chain management.



**1. Homepage Overview:**

The homepage of the 'Float UI' web application presents a dashboard with six primary options for users to interact with: 'COMP SHIPMENT', 'GET SHIPMENT', 'START SHIPMENT', 'USER PROFILE', 'SHIPMENTS COUNT', and 'SEND SHIPMENT'.

The interface uses a clean and minimalistic design with a black background and white text to emphasize the options available.

****

**2.** **Form Title:** Track product, Create Shipment

Use this form to input tracking information for a new shipment.

**Input Fields:**

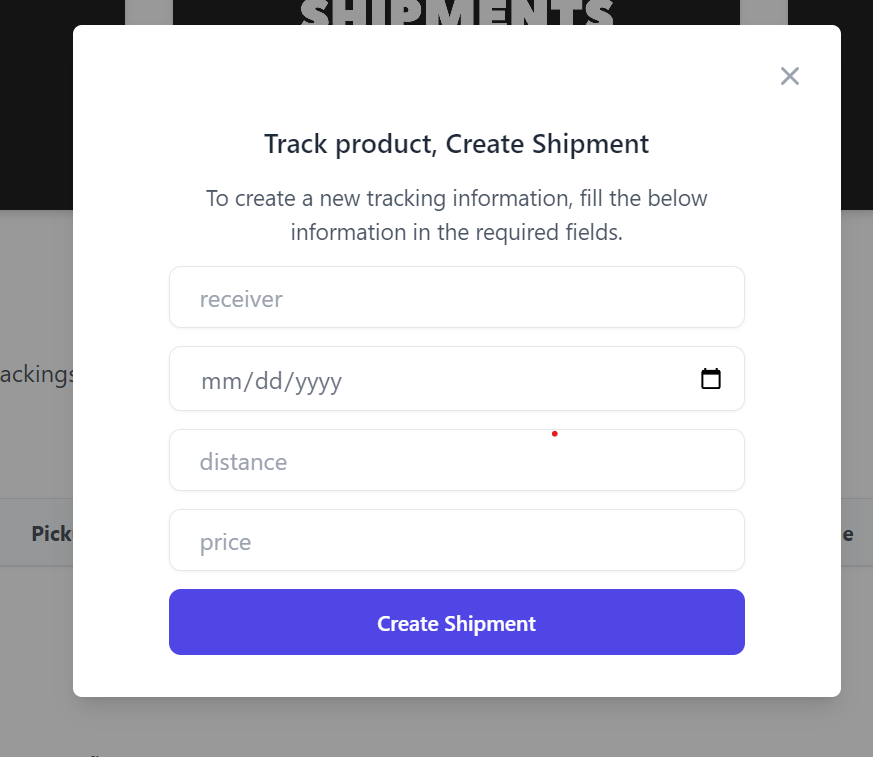
**Receiver:** Enter the name of the person or company receiving the shipment.

**Date:** Select or enter the date for when the shipment is created or sent.

**Distance (required):** Enter the shipping distance. This field is mandatory.

**Price:** Enter the cost of shipping.

**Create Shipment:** Click to finalize and submit the shipment details.



**3. Form Title:** Complete Shipment

This form is used to update the status of a shipment to "Complete".

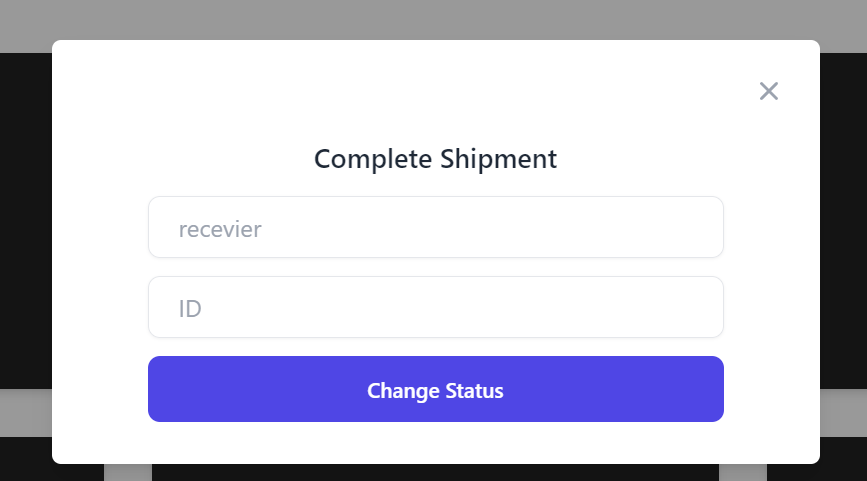
**Fields to Fill:**

**Receiver:** Type in the name of the person or entity that the shipment was sent to.

**ID:** Enter the unique identification number of the shipment.

**Action Button:**

Change Status: After entering the required information, press this button to update the shipment's status.

****

**4. Purpose of the Form:**

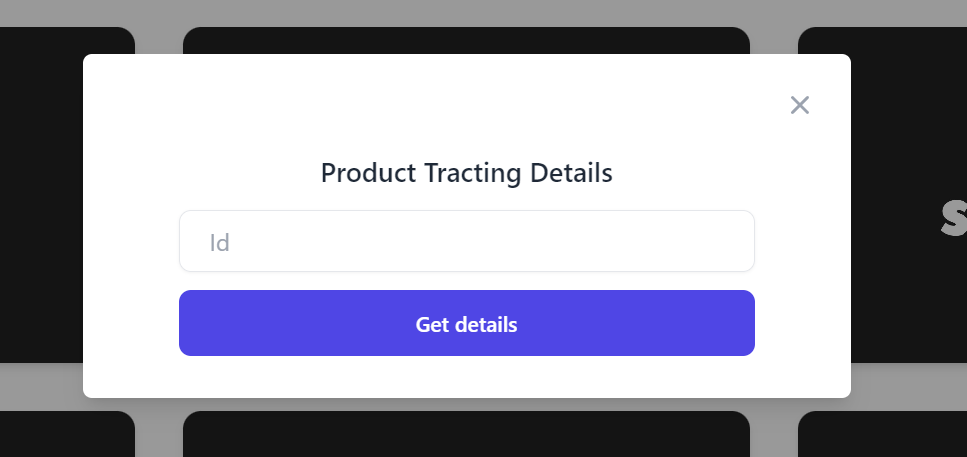
This form is intended to retrieve details about a product's shipment by using its unique identifier.

**Field Description:**

**Id:** Enter the unique ID assigned to the product shipment you want to track.

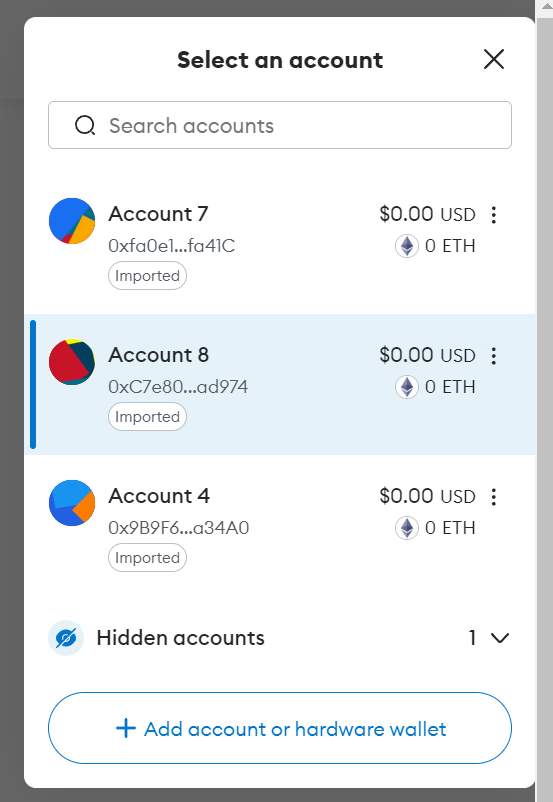
**Button Functionality:**

**Get details:** Click this button after entering the product ID to fetch and display the shipment details.



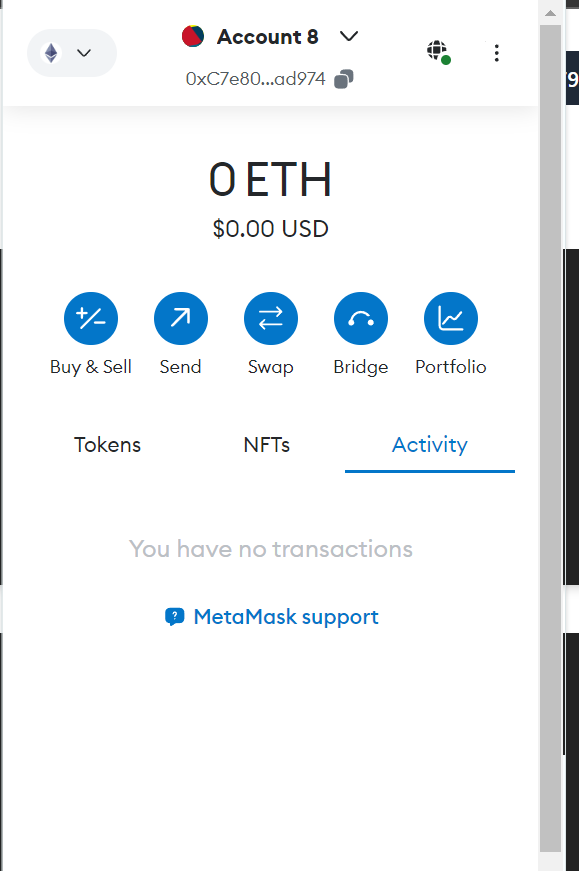
5. **Selecting a Wallet Account:**

* Users can choose an account from a list, where each account shows the balance in USD and in ETH (Ethereum), a crypto currency.
* There is a search feature to find accounts quickly.
* Users can add new accounts or hardware wallets for managing their transactions.



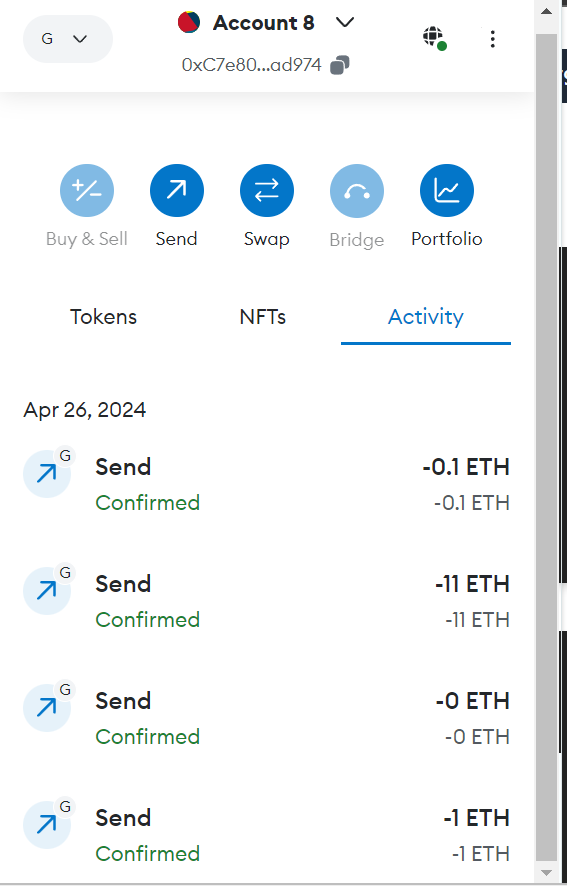
**6. Account Overview:**

* After selecting an account, users can see their current balance in both USD and ETH.
* There are options to buy & sell crypto currency, send funds to another address, swap currencies, and view the portfolio.
* The account page also has tabs for Tokens, NFTs (Non-Fungible Tokens), and Activity.



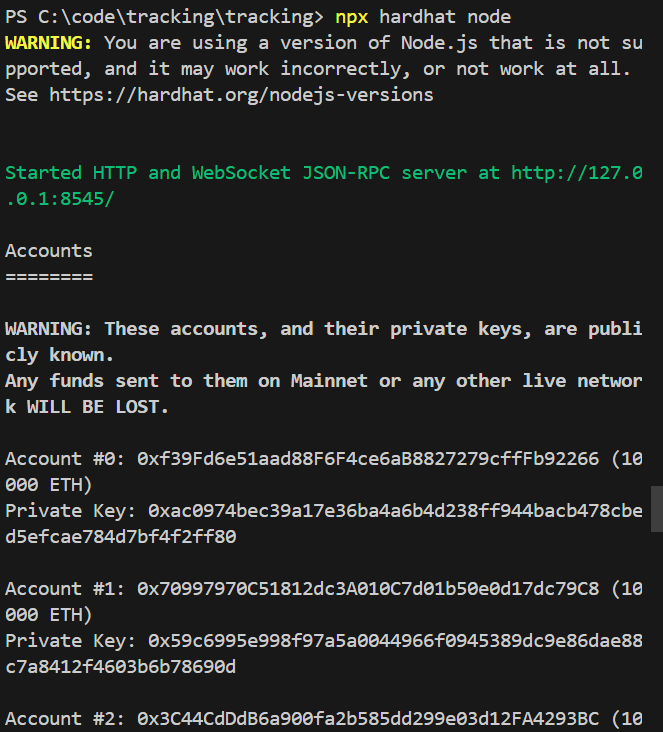
**7. Transaction History:**

* Users can review their transaction history, which shows past transfers of ETH including the amount and confirmation status.
* Transactions are dated, making it easy to track when each activity took place.



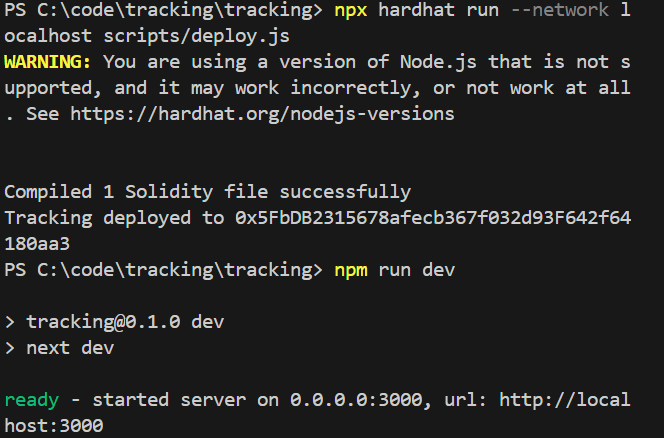
**8. Block chain Development Environment:**

* This image is from a command line interface showing the setup of a local block chain development environment using Hardhat, which is a popular development tool for Ethereum.
* The screen shows a list of accounts with Ethereum addresses and their respective private keys. This is a development-only setup, and it's essential to note that these accounts are not for production use as their private keys are publicly exposed.
* The warning message indicates that Node.js is not supported, suggesting a need to update or change the Node.js version to one compatible with Hardhat.
* It also shows the JSON-RPC server started, meaning the local block chain is ready to accept requests at the shown URL.



**9. Deployment of Smart Contracts:**

* This image captures the deployment of a Solidity smart contract, as indicated by the "Compiled 1 Solidity files successfully" message.
* A contract is then deployed to a given address on the block chain, which is likely a test or local network given the context.
* The "npm run dev" command is used to start a development server, and the "ready" message confirms that the server is running, with the local server URL provided.
* The same Node.js version warning appears which should be noted in the documentation for developers to use a compatible version when setting up their development environment.



**10. Conclusion:**

In conclusion, the project demonstrates the potential of block chain technology in revolutionizing supply chain management by enhancing transparency, efficiency, and trust among stakeholders. The combination of Ethereum block chain and associated tools provides a robust foundation for building scalable and secure supply chain solutions.

