Combine with metamask using the hardhat accounts as beneficiary addresses. Here the 0 address is that of the consortium and the others are all beneficiaries except for the last one that could be the public

To switch the **addressAccount** to the one currently connected to MetaMask, you can use the **ethereum** object provided by MetaMask. Here's how you can modify the code to achieve this:

import Web3 from 'web3';

import detectEthereumProvider from '@metamask/detect-provider';

const handleTransfer = async () => {

// Instantiate the contract

const contract = new web3.eth.Contract(CarbonChainJSON.abi, contractAddress);

// Get the currently selected Ethereum address from MetaMask

**const accounts = await window.ethereum.request({ method: 'eth\_requestAccounts' });**

**const addressAccount = accounts[0];**

// Perform the transfer using the selected Ethereum address

await contract.methods.transfer(inputAddress, inputCredits, inputCID).send({ from: addressAccount });

// Additional code...

};:

1. We use **window.ethereum.request({ method: 'eth\_requestAccounts' })** to request the currently selected accounts from MetaMask. This will prompt the user to select an account if they haven't already.
2. Once the user selects an account, **ethereum.request** resolves with an array of selected accounts. We extract the first account (**accounts[0]**) and use it as the **from** address when calling the **transfer** method on the contract.

Make sure to handle user rejections and errors appropriately when requesting accounts from MetaMask.