Experiment No. 07

Semester	B.E. Semester VII – Computer Engineering
Subject	Blockchain Lab (CSDL7022)
Subject Professor In-charge	Prof. Swapnil S. Sonawane
Academic Year	2024-25

Student Name	Deep Salunkhe
Roll Number	21102A0014

Title: Integrating Metamask With DApp

Theory:
MetaMask is a cryptocurrency wallet and browser extension that enables users to manage Ethereum-based assets and interact with decentralized applications (dApps) seamlessly.

Key Features:

1. Wallet Functionality: Allows users to create, import, and manage Ethereum wallets securely.
2. Transaction Management: Users can send and receive ETH and tokens, with options to customize gas fees for transactions.
3. dApp Integration: Provides a bridge to interact with dApps directly through the browser, allowing users to engage with smart contracts easily.

User Interface:
MetaMask provides a user-friendly interface that allows users to view their account balances, transaction history, and manage assets without needing deep technical knowledge.

Connecting to dApps:
When visiting a dApp, MetaMask prompts users to connect their wallet, enabling interaction with smart contracts. This includes sending transactions and querying contract data.

Security Features:

- Users control their private keys, and MetaMask encrypts them locally. It supports hardware wallets for enhanced security.
- Users receive notifications for transaction requests, ensuring they can review actions before approval.

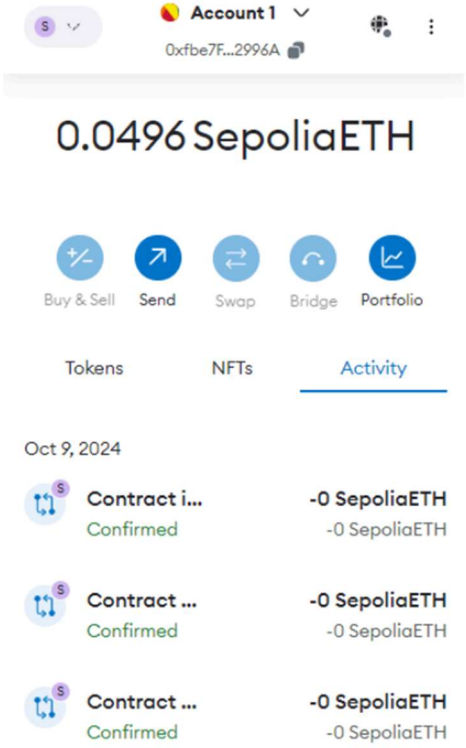
Network Switching:
MetaMask allows users to switch between different Ethereum networks (mainnet, testnets like Sepolia) and custom networks, facilitating development and testing.

Smart Contract (DApp):

```
// SPDX-License-Identifier: MIT
contract TodoList {
struct Todo {
string text;
bool isDone;
}
Todo[] private todos;
function addTodo(string memory _text) public {
todos.push(Todo(_text,false));
}
function removeTodo(uint _index) public {
todos[_index] = todos[todos.length-1];
todos.pop();
}
function getAllTodos() public view returns(Todo[] memory ){
return todos;
}
function completeTodo(uint _index) public {
todos[_index].isDone = true;
}
function unCompleteTodo(uint _index ) public {
todos[_index].isDone = false;
}
}
```

Output:

Before Deploying and Interacting With Smart Contracts



After Deploying and Interacting With Smart Contracts