School: ..... Campus: .....

Academic Year: ..... Subject Name: ..... Subject Code: .....

Semester: ..... Program: ..... Branch: ..... Specialization: .....

Date: .....

Applied and Action Learning

(Learning by Doing and Discovery)

Name of the Experiement : React Start – DApp Frontend Scaffolding

Objective/Aim:

To develop a decentralized application (DApp) frontend using React.js that allows users to interact with blockchain smart contracts through wallet connections such as MetaMask, and to understand how frontend frameworks communicate with blockchain networks.

Apparatus/Software Used:

1. Visual Studio Code (VS Code)
2. MetaMask Wallet
3. React.js Framework
4. Web3.js / Ethers.js Library
5. Smart Contract (Deployed on Testnet)
6. Hardhat / Truffle (for backend contract)
7. Internet Connection

Theory/Concept:

A **decentralized application (DApp)** integrates a user-friendly frontend with blockchain-based smart contracts. The frontend is typically built using **React.js**, which enables dynamic user interfaces that interact with the blockchain through **Web3.js** or **Ethers.js** libraries.

Key Concepts:

- **React.js:**
  A JavaScript library that enables developers to create reusable UI components and manage application states efficiently.
- **Web3.js / Ethers.js:**
  JavaScript libraries that serve as communication bridges between the DApp frontend and blockchain networks. They handle wallet connections, smart contract calls, and transaction signing.
- **MetaMask:**
  A browser extension that functions as a crypto wallet, allowing users to manage blockchain accounts, sign transactions, and connect to Ethereum-compatible networks.
- **DApp Architecture:**
  1. Frontend: User interface (React.js)
  2. Middleware: Blockchain connector (Web3.js / Ethers.js)
  3. Backend: Smart contract logic (Solidity, deployed on a testnet)

By combining these components, developers can create secure, interactive blockchain-based applications accessible directly from web browsers.

PROCEDURE:

Step 1: Initialize React Project

- Install Node.js and verify version:
- node -v
- npm -v
- Create a new React application:
- npx create-react-app blockchain-frontend
- Navigate into the project folder:
- cd blockchain-frontend

Step 2: Install Dependencies

- Add Web3.js or Ethers.js library:
- npm install web3

or

npm install ethers

Step 3: Connect React with MetaMask

- Open App.js and include wallet connection logic:
- import { useState } from "react";
- import { ethers } from "ethers";
-
- function App() {
- const [account, setAccount] = useState("");
-
- async function connectWallet() {
- if (window.ethereum) {
- const accounts = await window.ethereum.request({ method: "eth_requestAccounts" });
- setAccount(accounts[0]);
- } else {
- alert("MetaMask not detected");
- }
- }
-
- return (
-
- <h2>My DApp Frontend</h2>
- <button onClick={connectWallet}>Connect Wallet</button>
- Connected Account: {account}
-
-);
- }
-
- export default App;

Step 4: Run and Test the Application

- Start the development server:
- npm start
- Open the DApp in your browser:
  http://localhost:3000/
- Connect MetaMask and observe successful wallet integration.

Step 5: Interact with Smart Contract (Optional)

- Import contract ABI and connect to deployed contract address.
- Write and read blockchain data using functions like contract.methods.functionName().call() or Ethers.js equivalents.

Observation Table:

Activity	Tool/Library Used	Result / Output
React project initialization	Node.js, npm	Project structure created successfully
Wallet connection setup	Ethers.js	MetaMask connected to frontend
Smart contract interaction	Web3.js	Functions executed with real-time results
DApp test run (localhost)	React.js	Application loaded and interacted smoothly

:

ASSESSMENT

Rubrics	Full Mark	Marks Obtained	Remarks
Concept	10		
Planning and Execution/ Practical Simulation/ Programming	10		
Interpretation Result and	10		
Record of Applied and Action Learning	10		
Viva	10		
Total	**50**		

Signature of the Faculty:

Signature of the Student:

Name :

Regn. No.