NAMESPACE
INFINITE HORIZONS ONE COMMUNITY

namespacecomm.in/
namespace@bpitindia.com
linktr.ee/namespacecomm

TECHX 3.0 ASSIGNMENT

BLOCKCHAIN ASSIGNMENT

TASK 1: SMART CONTRACT CREATION AND DEPLOYMENT

Goal: Develop and deploy a basic ERC-20 token smart contract on an Ethereum testnet.

1. Create a Basic Token Smart Contract:

- Using Solidity, define an ERC-20 token with parameters like name, symbol, decimals, and totalSupply.
- Implement core ERC-20 functions: balanceOf, transfer, approve, and transferFrom for token transactions.
- Make sure the token has a fixed supply, assigned to the contract deployer.

2. Deploy the Contract:

Use a development framework like Hardhat or Truffle to compile and deploy the contract on a testnet (e.g., Ropsten, Goerli, or Polygon Mumbai).

Document the contract address and deployment steps.

Bonus: Integrate OpenZeppelin's ERC-20 library to manage standard implementations easily.

TASK 2: DAPP INTERFACE DEVELOPMENT

Goal: Develop a simple front-end interface for interacting with the deployed token smart contract, including wallet connectivity and token transaction capabilities.

1. **Set Up Wallet Connection:**

- Create a front-end interface with Web3.js or Ethers.js that connects to MetaMask.
- Display the connected wallet's address and balance of your token.

2. **Token Transfer Functionality:**

- Provide an input field for a recipient's address and the token amount to send.
- Upon submission, execute the transfer function of your smart contract to send tokens.
- Display transaction feedback to indicate success or failure.

Optional Extension: Add a display for recent transactions or balances.

SUBMISSION REQUIREMENTS

- Upload the smart contract and front-end code to a GitHub repository.
- Include setup instructions and screenshots showing the DApp in action.

RESOURCES

- **Solidity Documentation:** For understanding Solidity basics and ERC-20 tokens.
- **ERC-20 Standard by OpenZeppelin:** Pre-built ERC-20 contract templates, including detailed explanations and usage.
- **Hardhat Setup and Deployment Guide:** A framework for compiling, deploying, and testing smart contracts. **Hardhat Website** **Hardhat Tutorial**
- **Polygon Testnet (Mumbai) Faucet:** To get free test MATIC tokens for deployment on the Polygon Mumbai network.
- **MetaMask Guide:** For setting up MetaMask and connecting it to a test network.
- **Ethers.js Library:** For connecting to the Ethereum blockchain and interacting with your smart contract.
- **Web3.js Library:** Another option for blockchain interaction, particularly popular for wallet connection.
- **Connecting Frontend to Smart Contract with Ethers.js:** This guide walks through using Ethers.js for wallet integration and contract interaction.
- **Front-End Development with React (optional):** React setup guide for creating the user interface.