

UI for DApps



**Objective/Aim:**  
  
 To design and implement a User Interface (UI) for a decentralized application (DApp) that connects to the blockchain and interacts with deployed smart contracts.

**Apparatus/Software Used:**

* Laptop / PC
* Node.js & npm (for package management)
* React.js or HTML/CSS/JavaScript (for UI development)
* Web3.js or Ethers.js (for blockchain interaction)
* MetaMask wallet (for transaction signing)

**Theory/Concept:**

* What are DApps?  
   Decentralized Applications (DApps) are applications that run on blockchain networks using smart contracts for backend logic and a web-based frontend for user interaction.
* UI Role in DApps:
* Provides an easy way for users to connect wallets and interact with smart contracts.
* Displays blockchain data (balances, NFT ownership, transactions).
* Facilitates actions like sending tokens, minting NFTs, or voting in DAOs.
* UI Flow in a DApp:

1. User opens DApp in browser.
2. DApp connects to MetaMask or another wallet provider.
3. User interacts with UI buttons/inputs.
4. Web3.js/Ethers.js calls smart contract functions.
5. Blockchain confirms and updates UI state.

.



**Procedure:**

Set Up Project

* Initialize a frontend project (e.g., React or simple HTML/JS).
* Install Web3.js or Ethers.js:

npm install web3

Connect Wallet

* Add a “Connect Wallet” button.
* Use MetaMask API to request account access:

if (window.ethereum) {

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

console.log("Connected:", accounts[0]);

}

.

**Observation Table:**

* Users can connect/disconnect wallets from the DApp UI.
* Read operations instantly reflect contract data in the UI.
* Write operations trigger MetaMask confirmation before execution.
* UI updates automatically after successful blockchain transactions.

