* **Implementation of First Module**

**Module Name**: Smart Contract for Crowdfunding Campaign

**Objective**

The first module focuses on developing and deploying the **smart contract** that handles:

* Campaign creation
* Accepting contributions

**Tools and Technologies Used**

* **Solidity** (Smart contract language)
* **Hardhat** (Development framework)
* **MetaMask** (Wallet for testing)
* **Ethereum Testnet** (Goerli or Sepolia)
* **Node.js** (Environment for Hardhat)

**Key Steps**

**1.Smart Contract Design**

**2.Compile and Test Smart Contract**

**3.Deploy Smart Contract**

**Expected Output**

* A working smart contract that can:
  + Accept user contributions.
  + Track the total funds.
  + Allow the project creator to withdraw funds if the goal is reached.
* **Implementation of Second Module**

**Module Name**: Frontend Interface for Campaign Interaction

**Objective**

The second module focuses on building the **user interface** where users can:

* Connect their crypto wallets
* View active crowdfunding campaigns
* Contribute to campaigns
* Check funding status and withdraw funds/refunds

This module connects the **smart contract** developed in the first module to a **web frontend**.

**Tools and Technologies Used**

* **React.js** (Frontend Framework)
* **Ethers.js** or **Web3.js** (Blockchain Interaction)
* **MetaMask** (Wallet for transactions)

**Key Steps**

1. **Setting Up React App**
2. **Connecting to MetaMask**
3. **Interacting with the Smart Contract**
   * Load the deployed smart contract using Ethers.js:

* **Check Funding Status / Withdraw / Refund** similarly by calling corresponding smart contract methods.

1. **Building UI Pages**
   * **Home Page**: View active campaigns and their details.
   * **Contribution Page**: Simple form to contribute ETH to a campaign.
   * **Dashboard (for Creators)**: Withdraw funds if the goal is reached.
2. **Styling**
   * Use Tailwind CSS or Bootstrap to make the pages clean and responsive.

**Expected Output**

* A working frontend DApp where:
  + Users can connect their wallets.
  + View campaign status (goal, amount raised, deadline).
  + Contribute ETH to a campaign.
* **Implementation of Third Module**

**Module Name**: Campaign Management and Fund Handling (Post-Campaign Phase)

**Objective**

The third module focuses on managing campaigns **after** the fundraising deadline:

* Checking if the funding goal was achieved.
* Allowing **project creators** to withdraw raised funds.
* Allowing **contributors** to claim **refunds** if the campaign failed.

This ensures complete automation without the need for manual verification, using blockchain-based trust.

**Tools and Technologies Used**

* **Solidity** (for smart contract goal checking logic)
* **React.js / Frontend** (for fund withdrawal and refund UI)
* **Ethers.js** (to interact with blockchain)
* **MetaMask** (for signing transactions)

**Key Steps**

1. **Smart Contract Functions (Already Created in First Module)**
2. **Frontend Integration**
   * **Goal Checking**:

* **Withdraw Funds (Creator Only)**:
* **Requst Refund (Contributor Only)**:

1. **Access Control**
   * Verify wallet addresses:
     + Only **campaign creator** can withdraw.
     + Any **contributor** can request a refund if the campaign failed.
2. **Testing**
   * Test all flows:
     + Contribution and goal checking.
     + Successful withdrawal by the creator.
     + Refund claim by contributors.

**Expected Output**

* Automated **goal verification** after campaign deadline.
* **Fund withdrawal** option for creators if successful.
* Smooth and transparent post-campaign experience for users.