**Dictator Game (Blockchain-Based Experiment)**

The **Simplified Dictator Game** is a behavioural economic experiment implemented as a smart contract on the Ethereum blockchain, supported by a user-friendly HTML/JavaScript frontend. The objective of this game is to simulate a one-way resource allocation scenario, where one participant the dictator determines how a fixed amount of ETH should be split between themselves and another participant the recipient.

**Smart Contract Design (Solidity)**

The smart contract, written in **Solidity (v0.8.0)**, governs the game logic and ensures transparent and immutable fund allocation.

The Dictator Game smart contract involves two key participants: the **dictator**, who is the player deploying the contract and funding the game with ETH, and the **recipient**, whose address is provided during the deployment process. The contract's constructor plays a crucial role by initializing the dictator as the sender and registering the recipient’s address. Additionally, it enforces that a non-zero amount of ETH must be sent at the time of deployment, ensuring there is a pot available for the game to function properly. This setup lays the foundation for a fair and secure execution of the game's mechanics.

**Function dictate Split(uint256 amount to Recipient):**

The dictate Split() function is designed with clear and secure logic. Firstly, it can only be executed by the designated dictator, ensuring strict access control and preventing unauthorized interactions. Once triggered, the function facilitates the transfer of the specified amount of ETH to the recipient’s address. After this allocation, any remaining balance in the contract is automatically returned to the dictator. This setup reinforces the dictator's sole authority in determining how the funds are distributed while maintaining transparency and safeguarding the proper flow of ETH within the contract.

**Frontend Interface (HTML + JavaScript + Web3.js)**

The web interface is designed to be clean, intuitive, and mobile-friendly:

The visual design features a gradient background and a modern, card-style user interface created with CSS. It includes a form input for users to enter the amount of ETH they wish to allocate to the recipient and a button that triggers the dictate Split() function. Functionally, the application leverages Web3.js to connect with MetaMask and interact with the deployed smart contract. It establishes a connection to the user's wallet, retrieves the current account, converts the ETH input into Wei, and calls the appropriate smart contract method. Transaction outcomes or error messages are then displayed in a designated status box. This integration effectively bridges the blockchain backend with a user-friendly frontend, enabling real-time interaction with the smart contract.