Treasury Smart Contract

The Treasury smart contract is designed to manage funds by depositing USDC tokens into Uniswap and Aave protocols. The deposited funds are then swapped for USDT and DAI tokens, respectively. The contract allows the owner to set the distribution ratios for Uniswap and Aave, deposit funds, withdraw funds, and calculate the yield.

Constructor

The constructor initializes the contract with the following parameters:

- **_uniswapRouter**: Address of the Uniswap V2 Router
- **_aaveLendingPool**: Address of the Aave Lending Pool
- **_usdcToken**: Address of the USDC token
- **_usdtToken**: Address of the USDT token
- **_daiToken**: Address of the DAI token

These addresses are stored in the contract as public variables, allowing external parties to interact with them.

Functions

1. **setRatios**: Allows the contract owner to set the distribution ratios for Uniswap and Aave. The sum of both ratios must equal 100. This function ensures that the total allocation of funds is always 100%, preventing any loss or mismanagement of funds.

```
function setRatios(
    uint256 _uniswapRatio,
    uint256 _aaveRatio
) external onlyOwner {
    require(_uniswapRatio + _aaveRatio == 100, "Ratios must add up to 100");
    uniswapRatio = _uniswapRatio;
    aaveRatio = _aaveRatio;
}
```

2. **deposit**: Accepts an amount of USDC tokens from the sender, calculates the distribution amounts based on the set ratios, and deposits the funds into Uniswap and Aave. The USDC tokens are then swapped for USDT and DAI tokens, respectively.

```
function deposit(uint256 amount) external {
    // Transfer USDC from sender to the contract
    IERC20(usdcToken).safeTransferFrom(msg.sender, address(this), amount);

    // Calculate distribution amounts based on ratios
    uint256 uniswapAmount = (amount * uniswapRatio) / 100;
    uint256 aaveAmount = (amount * aaveRatio) / 100;

    // Approve Uniswap and AAVE to spend USDC
    IERC20(usdcToken).safeApprove(address(uniswapRouter), uniswapAmount);
```

```
    IERC20(usdcToken).safeApprove(address(aaveLendingPool), aaveAmount);

    // Deposit funds into Uniswap and swap for USDT
    address[] memory path = new address[](2);
    path[0] = usdcToken;
    path[1] = usdtToken;

    uniswapRouter.swapExactTokensForTokens(
        uniswapAmount,
        0,
        path,
        address(this),
        block.timestamp + 600
    );

    // Deposit funds into AAVE and swap for DAI
    aaveLendingPool.deposit(usdcToken, aaveAmount, address(this), 0);
}
```

3. **withdraw**: Allows the contract owner to withdraw a specified amount of USDT and DAI tokens based on the set ratios. This function ensures that the owner can retrieve their funds when needed.

```
function withdraw(uint256 amount) external onlyOwner {
    IERC20(usdtToken).safeTransfer(
        msg.sender,
        (amount * uniswapRatio) / 100
    );
    IERC20(daiToken).safeTransfer(msg.sender, (amount * aaveRatio) / 100);
```

```
}
```

4. **calculateYield**: Returns the total balance of USDT and DAI tokens held by the contract. This function allows the owner to monitor the performance of their investments.

```
function calculateYield() public view returns (uint256) {
    uint256 usdtBalance = IERC20(usdtToken).balanceOf(address(this));
    uint256 daiBalance = IERC20(daiToken).balanceOf(address(this));

    return usdtBalance + daiBalance;
}
```

Interaction Flow

1. The contract owner sets the distribution ratios for Uniswap and Aave using the setRatios function.

2. Users deposit USDC tokens into the Treasury smart contract using the deposit function.

3. The deposited USDC tokens are split based on the set ratios and sent to Uniswap and Aave.

4. In Uniswap, USDC tokens are swapped for USDT tokens.

5. In Aave, USDC tokens are deposited and swapped for DAI tokens.

6. The contract owner can withdraw USDT and DAI tokens using the withdraw function.

7. The contract owner can check the total yield (USDT and DAI token balances) using the calculateYield function.

Block Diagram

In this diagram, you can see the following interactions:

1. The Treasury smart contract deposits USDC tokens into Uniswap V2 Router to swap for USDT tokens.
2. The Treasury smart contract deposits USDC tokens into Aave Lending Pool, which returns DAI tokens.
3. The Treasury smart contract can withdraw USDT and DAI tokens.

This diagram provides a comprehensive view of the interactions between the Treasury smart contract, Uniswap V2 Router, Aave Lending Pool, and the respective tokens involved.

Here's a description of the main components and their relationships:

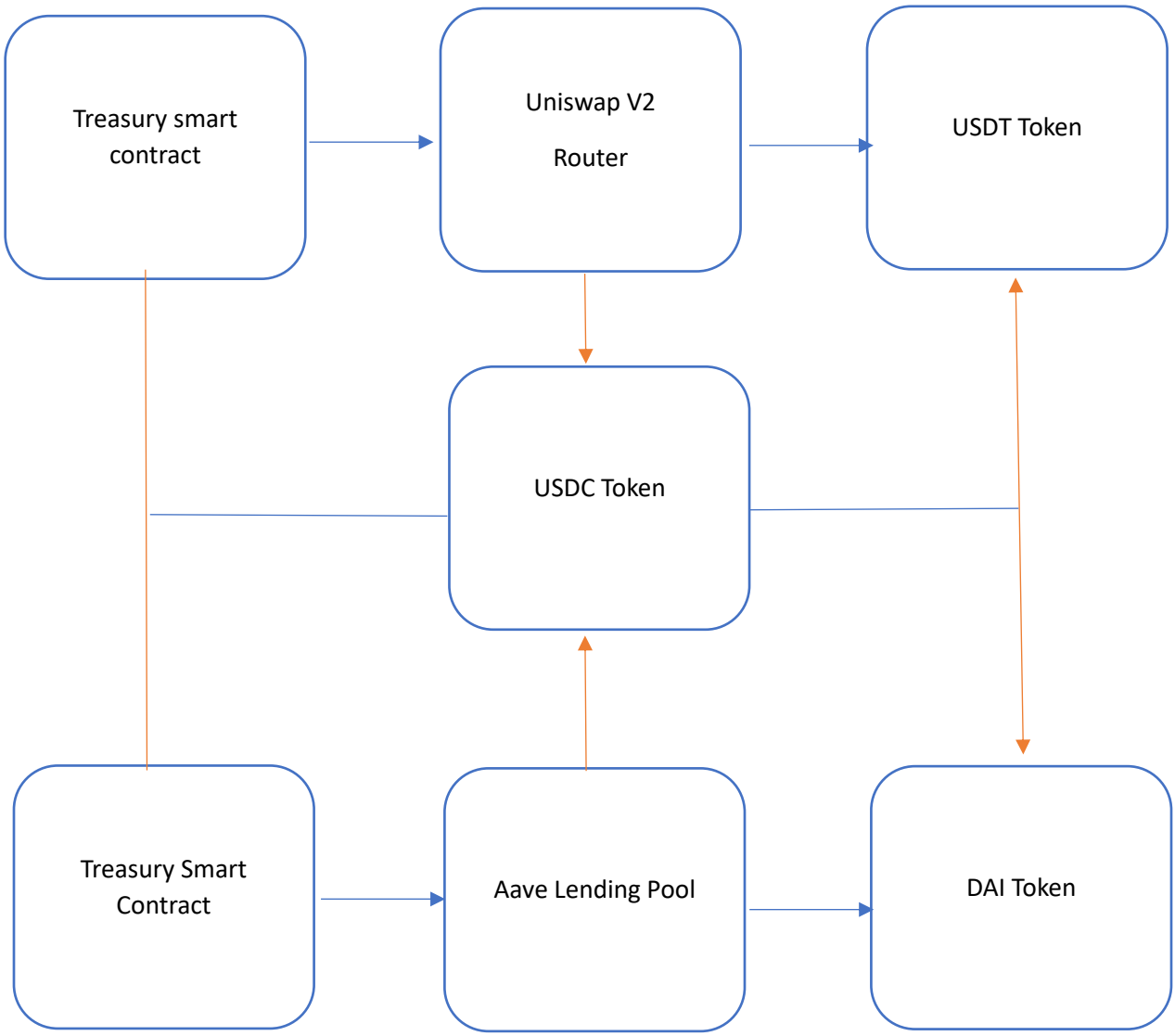
1. User: Interacts with the Treasury smart contract by depositing USDC tokens and calling the `deposit` function.

2. Treasury Smart Contract: Holds the deposited USDC tokens and distributes them between Uniswap and Aave based on the set ratios.

3. Uniswap V2 Router: The Treasury smart contract interacts with the Uniswap V2 Router to swap USDC tokens for USDT tokens.

4. Aave Lending Pool: The Treasury smart contract interacts with the Aave Lending Pool to deposit USDC tokens and receive DAI tokens in return.

5. USDC, USDT, and DAI Tokens: These are the ERC20 tokens involved in the transactions within the Treasury smart contract.


Treasury smart contract
Uniswap V2 Router
USDT Token
USDC Token
Treasury Smart Contract
Aave Lending Pool
DAI Token