```solidity

// SPDX-License-Identifier: MIT

pragma solidity ^0.8.18;

import "@openzeppelin/contracts/token/ERC20/ERC20.sol";

import "@openzeppelin/contracts/access/Ownable.sol";

contract Mytti is ERC20, Ownable {

// Tax rates

uint256 public taxRate = 1;

uint256 public marketingTaxRate = 3;

uint256 public devTaxRate = 55;

uint256 public liquidityTaxRate = 3;

// Wallet addresses

address public marketingWallet;

address public devWallet;

address public liquidityPool;

// Token split

bool public tokenSplitActivated;

uint256 public tokenSplitPriceThreshold = 135 \* 10\*\*16; // $1.35

uint256 public tokenSplitStartTime;

// Holder tracking

mapping(address => bool) private \_isHolder;

address[] private \_holders;

// Events

event Airdropped(address indexed recipient, uint256 amount);

event TokenSplitActivated();

event TokenSplitCompleted();

constructor(address \_marketingWallet, address \_devWallet, address \_liquidityPool) ERC20("My-tti", "MYTI") {

\_mint(msg.sender, 100000000000000000 \* 10\*\*decimals());

marketingWallet = \_marketingWallet;

devWallet = \_devWallet;

liquidityPool = \_liquidityPool;

}

function transfer(address recipient, uint256 amount) public override returns (bool) {

\_transferWithTax(msg.sender, recipient, amount);

return true;

}

function transferFrom(address sender, address recipient, uint256 amount) public override returns (bool) {

\_transferWithTax(sender, recipient, amount);

uint256 currentAllowance = allowance(sender, \_msgSender());

require(currentAllowance >= amount, "ERC20: transfer amount exceeds allowance");

\_approve(sender, \_msgSender(), currentAllowance - amount);

return true;

}

function airdrop(address[] memory recipients, uint256[] memory amounts) public onlyOwner {

require(recipients.length == amounts.length, "Mytti: Invalid input arrays");

for (uint256 i = 0; i < recipients.length; i++) {

\_transferWithTax(msg.sender, recipients[i], amounts[i]);

emit Airdropped(recipients[i], amounts[i]);

}

}

function \_transferWithTax(address sender, address recipient, uint256 amount) private {

uint256 totalTaxRate = taxRate + marketingTaxRate + devTaxRate + liquidityTaxRate;

require(totalTaxRate <= 100, "Mytti: Total tax rate cannot exceed 100%");

uint256 tax = (amount \* taxRate) / 100;

uint256 marketingTax = (amount \* marketingTaxRate) / 100;

uint256 devTax = (amount \* devTaxRate) / 100;

uint256 liquidityTax = (amount \* liquidityTaxRate) / 100;

uint256 amountAfterTax = amount - tax - marketingTax - devTax - liquidityTax;

super.\_transfer(sender, recipient, amountAfterTax);

super.\_transfer(sender, marketingWallet, marketingTax);

super.\_transfer(sender, devWallet, devTax);

super.\_transfer(sender, liquidityPool, liquidityTax);

// Redistribution

if (tax > 0) \_redistribute(sender, tax);

// Update holders list

\_updateHolders(recipient);

if (balanceOf(sender) == 0) \_removeHolder(sender);

}

function \_redistribute(address sender, uint256 tax) private {

for (uint256 i = 0; i < \_holders.length; i++) {

address holder = \_holders[i];

if (holder != sender && balanceOf(holder) > 0) {

uint256 holderShare = (balanceOf(holder) \* tax) / totalSupply();

if (holderShare > 0) \_mint(holder, holderShare);

}

}

}

function activateTokenSplit() public onlyOwner {

require(!tokenSplitActivated, "Mytti: Token split already activated");

tokenSplitActivated = true;

tokenSplitStartTime = block.timestamp;

emit TokenSplitActivated();

}

function processTokenSplit() public onlyOwner {

require(tokenSplitActivated, "Mytti: Token split not activated");

require(block.timestamp >= tokenSplitStartTime + 90 days, "Mytti: 90 days not passed since activation");

for (uint256 i = 0; i < \_holders.length; i++) {

address holder = \_holders[i];

uint256 holderBalance = balanceOf(holder);

if (holderBalance > 0) \_mint(holder, holderBalance / 2);

}

tokenSplitActivated = false;

emit TokenSplitCompleted();

}

function setTaxRates(uint256 \_taxRate, uint256 \_marketingTaxRate, uint256 \_devTaxRate, uint256 \_liquidityTaxRate) public onlyOwner {

taxRate = \_taxRate;

marketingTaxRate = \_marketingTaxRate;

devTaxRate = \_devTaxRate;

liquidityTaxRate = \_liquidityTaxRate;

}

function setWalletAddresses(address \_marketingWallet, address \_devWallet, address \_liquidityPool) public onlyOwner {

marketingWallet = \_marketingWallet;

devWallet = \_devWallet;

liquidityPool = \_liquidityPool;

}

function \_updateHolders(address holder) private {

if (!\_isHolder[holder]) {

\_isHolder[holder] = true;

\_holders.push(holder);

}

}

function \_removeHolder(address holder) private {

if (\_isHolder[holder]) {

\_isHolder[holder] = false;

for (uint256 i = 0; i < \_holders.length; i++) {

if (\_holders[i] == holder) {

\_holders[i] = \_holders[\_holders.length - 1];

\_holders.pop();

break;

}

}

}

}

}

```

This updated contract:

1. Limits total tax to 100%.

2. Adjusts `\_redistribute()` to avoid minting zero tokens.

3. Ensures gas costs are managed, though further optimizations may be necessary for large holder lists.