**12.** **Integration Test:** Write an integration test for your smart contracts, demonstrating the creation of new ERC-20 token assigning a batch ID, transferring tokens and handling token expiry.

Ans:

Creating an integration test involves

1. deploying your smart contracts to a test blockchain,
2. interacting with them, and
3. verifying that the expected behavior occurs.

Below is a simple example using the Truffle framework for testing Ethereum smart contracts.

Steps:

Install Truffle:

* npm install -g truffle

Create a Truffle project:

* mkdir TokenTest
* cd TokenTest
* truffle init

Create contracts/Token.sol with your ExpirableERC20Token.sol and TokenFactory.sol contracts.

Create a new test file, test/TokenTest.js

const ExpirableERC20Token = artifacts.require("ExpirableERC20Token");

const TokenFactory = artifacts.require("TokenFactory");

contract("TokenTest", (accounts) => {

let tokenInstance;

let factoryInstance;

before(async () => {

factoryInstance = await TokenFactory.new({ from: accounts[0] });

});

it("should create a new ERC-20 token, assign a batch ID, transfer tokens, and handle token expiry", async () => {

// Create a new ERC-20 token with a batch ID

const initialSupply = 100;

const expiryDate = Math.floor(Date.now() / 1000) + 3600; // 1 hour from now

const batchId = 1;

await factoryInstance.createToken("TestToken", "TT", initialSupply, expiryDate, batchId, initialSupply);

// Retrieve the deployed token address

const tokenAddress = await factoryInstance.getTokenAddress(batchId);

tokenInstance = await ExpirableERC20Token.at(tokenAddress);

// Check the initial balances

const ownerBalance = await tokenInstance.balanceOf(accounts[0]);

const recipientBalance = await tokenInstance.balanceOf(accounts[1]);

assert.equal(ownerBalance.toNumber(), initialSupply, "Owner should have initial supply");

assert.equal(recipientBalance.toNumber(), 0, "Recipient should have zero balance");

// Transfer tokens using batch ID

const transferAmount = 50;

await tokenInstance.transferByBatch(accounts[1], transferAmount, batchId);

// Check balances after transfer

const newOwnerBalance = await tokenInstance.balanceOf(accounts[0]);

const newRecipientBalance = await tokenInstance.balanceOf(accounts[1]);

assert.equal(newOwnerBalance.toNumber(), initialSupply - transferAmount, "Owner balance should decrease");

assert.equal(newRecipientBalance.toNumber(), transferAmount, "Recipient balance should increase");

// Wait for the token to expire

await new Promise((resolve) => setTimeout(resolve, 3600000)); // 1 hour

// Attempt transfer after token expiry

try {

await tokenInstance.transferByBatch(accounts[0], 10, batchId);

assert.fail("Transfer should fail after token expiry");

} catch (error) {

assert.include(error.message, "Token has expired", "Transfer should fail after token expiry");

}

});

});

To run the test, use the following command:

Perform the Test:

truffle test

This will deploy the contracts to a local test blockchain (Ganache by default) and execute the specified test script.