Sarthak Kaushik: 101471600

BCDV-4028 Lab 3 Submission

Ganache Time Traveller

ganache-time-traveler is a tool that extends the functionality of Ganache. It allows us to manipulate the blockchain's timestamp, which can be useful for testing time-dependent smart contracts. This tool enables us to move the blockchain's timestamp forward or backward to simulate various time-related scenarios, such as expiration of time-based events.

Here's an example of how to use ganache-time-traveler to test a time-dependent smart contract. In this example, we'll create a simple crowdfunding contract where contributors can only withdraw their contributions after a certain time period has passed.

Installation:

npm install --save-dev ganache-time-traveler

Contract:

```
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.0;

contract Crowdfunding {
    address public owner;
    uint256 public fundingEndTime;
    uint256 public totalFunds;

    constructor(uint256 _duration) {
        owner = msg.sender;
        fundingEndTime = block.timestamp + _duration;
    }

    modifier onlyOwner() {
        require(msg.sender == owner, "Only the contract owner can call this function");
        _;
    }

    modifier onlyAfterFundingEnd() {
        require(block.timestamp >= fundingEndTime, "Funding period is not over yet");
        _;
    }

    function contribute() external payable {
        require(block.timestamp < fundingEndTime, "Funding period is over");
        require(msg.value > 0, "You must send Ether to contribute");
        totalFunds += msg.value;
    }

    function withdrawFunds() external onlyOwner onlyAfterFundingEnd {
        payable(owner).transfer(totalFunds);
        totalFunds = 0;
    }
}
```

This contract allows contributors to send Ether, but the owner can only withdraw the funds after the funding period has ended.

Tests:

```
const Crowdfunding = artifacts.require("Crowdfunding");
const { time } = require("ganache-time-traveler");

contract("Crowdfunding", (accounts) => {
    let crowdfundingContract;
    const owner = accounts[0];
    const contributor = accounts[1];

    beforeEach(async () => {
        crowdfundingContract = await Crowdfunding.new(3600); // 1 hour funding period
    });

    it("should allow contributors to send Ether during the funding period", async () => {
        await crowdfundingContract.contribute({ from: contributor, value: web3.utils.toWei("1",
"ether") });
        const contractBalance = await web3.eth.getBalance(crowdfundingContract.address);
        assert.equal(contractBalance, web3.utils.toWei("1", "ether"));
    });

    it("should not allow contributors to withdraw funds before the funding period ends", async
() => {
        try {
            await crowdfundingContract.withdrawFunds({ from: owner });
            assert.fail("Withdrawal should fail before funding period ends");
        } catch (error) {
            assert(error.message.includes("Funding period is not over yet"), "Expected error
message");
```

```
        }
    });

    it("should allow the owner to withdraw funds after the funding period ends", async () => {
        await time.increase(3601); // Move time forward by 1 hour and 1 second
        await crowdfundingContract.withdrawFunds({ from: owner });
        const contractBalance = await web3.eth.getBalance(crowdfundingContract.address);
        assert.equal(contractBalance, "0");
    });
});
```

In these tests, we use ganache-time-traveler to manipulate the blockchain's timestamp. Specifically, we move time forward by 1 hour and 1 second to simulate the end of the funding period, allowing the owner to withdraw the funds.

Finally, to test, we can use:

```
ganache test
```