BÁO CÁO HÀNG TUẦN

Dự án: Web3 Crowdfunding
Tuần: 3
Thời gian: 23/03/2025 – 29/03/2025
Người thực hiện: Lê Đức Thiện – B22DCCN823

---

1. TỔNG QUAN

Trong tuần này, em tập trung vào việc học cách viết unit test cho smart contract bằng Foundry. Bên cạnh đó, em đã bắt đầu xây dựng các smart contract chính cho dự án Web3 Crowdfunding, trong đó đã hoàn thành hai chức năng cơ bản là gửi tiền và rút tiền.

2. CÔNG VIỆC TRONG TUẦN

2.1. Mục tiêu

- Học cách viết unit test cho smart contract bằng Foundry.
- Viết smart contract với các chức năng chính của dự án Web3 Crowdfunding.
- Tối ưu hóa code và đảm bảo tính bảo mật cho hợp đồng thông minh.

2.2. Công việc đã hoàn thành

Học về Foundry

- Cài đặt Foundry và thiết lập môi trường làm việc trên máy tính cá nhân.
- Tìm hiểu về cấu trúc dự án trong Foundry, bao gồm cách tổ chức file, thư mục và cách biên dịch smart contract.
- Nghiên cứu về cách sử dụng Forge, công cụ CLI của Foundry, để biên dịch và chạy test.
- Đọc tài liệu hướng dẫn về cách viết test trong Foundry, bao gồm cách giả lập giao dịch và cách kiểm tra kết quả test.
- Thử nghiệm chạy các test mẫu có sẵn để hiểu cách hoạt động nhưng chưa tự viết được test riêng cho smart contract của dự án.

Thực hành viết một số chức năng chính của dự án Web3 Crowdfunding

- Chức năng gửi tiền vào hợp đồng.
- Chức năng rút tiền từ hợp đồng.

```
// SPDX-License-Identifier: MIT
pragma solidity 0.8.26;

contract Crowdfunding {
    error NotOwner();
    uint public constant MINIMUM_FUND = 0.001 ether;
    address public immutable i_owner;

    receive() external payable {
        fund();
    }

    fallback() external payable {
        fund();
    }

    constructor() {
        i_owner = msg.sender ;
    }

    modifier onlyOwner() {
        if(i_owner != msg.sender) {
            revert NotOwner();
        }
        _;
    }

    function fund() public payable {
        uint fundAmount = msg.value;
        require(fundAmount >= MINIMUM_FUND, "Insufficient funds!");
    }

    function withdraw() public onlyOwner{
        (bool sent, ) = payable(i_owner).call{value:
address(this).balance}("");
        require(sent, "Failed to send ether");
    }
}
```

Tối ưu và kiểm tra bảo mật

- Sử dụng các modifier để kiểm soát quyền truy cập.

3. ĐÁNH GIÁ KẾT QUẢ

- Đã học về Foundry nhưng chưa thực hành viết test.
- Đã hoàn thành hai chức năng cơ bản của smart contract.
- Hiểu rõ hơn về các vấn đề bảo mật trong lập trình Solidity.
- Cần tiếp tục phát triển các chức năng khác của smart contract.

4. THÁCH THỨC & GIẢI PHÁP

4.1. Thách thức

- Việc viết unit test cho smart contract còn nhiều khó khăn do chưa quen với Foundry.
- Việc triển khai các chức năng phức tạp cần nhiều thời gian kiểm thử.

4.2. Giải pháp

- Tiếp tục thực hành và nghiên cứu thêm về Foundry.
- Tham khảo các smart contract mẫu từ các dự án Web3 khác.
- Kiểm thử nhiều tình huống hơn để đảm bảo tính toàn vẹn của hợp đồng.

5. MỤC TIÊU TUẦN TỚI

- Viết unit test cho smart contract bằng Foundry.
- Triển khai thử nghiệm smart contract trên Sepolia Testnet.

6. GHI CHÚ KHÁC

Trong tuần tới, em sẽ tập trung hoàn thiện smart contract, bổ sung thêm chức năng và tiến hành kiểm thử trên Sepolia Testnet để chuẩn bị cho giai đoạn tiếp theo của dự án. Rất mong nhận được góp ý của thầy để cải thiện hơn nữa trong tuần tiếp theo.