# BÁO CÁO HÀNG TUẦN

**Dự án:** Web3 Crowdfunding
**Tuần:** 2
**Thời gian:** 16/03/2025 – 22/03/2025
**Người thực hiện:** Lê Đức Thiện – B22DCCN823

---

## 1. TỔNG QUAN

Tiếp nối mục tiêu tuần trước, tuần này em tập trung học lập trình Solidity, triển khai và tương tác với smart contract trên Remix IDE, đồng thời tìm hiểu các phương pháp tối ưu phí gas khi deploy smart contract.

## 2. CÔNG VIỆC TRONG TUẦN

### 2.1. Mục tiêu

- Học lập trình Solidity cơ bản và nâng cao.
- Viết và triển khai thử nghiệm các smart contract đơn giản.
- Tìm hiểu và thực hành tương tác với smart contract bằng Remix IDE.
- Nghiên cứu và áp dụng các phương pháp tối ưu phí gas.

### 2.2. Công việc đã hoàn thành

**Học và nắm được các kiến thức cơ bản về Solidity**

- Kiểu dữ liệu cơ bản (uint, address, bool, string).
- Cách khai báo biến và sử dụng hằng số (constant, immutable).
- Cấu trúc hàm và phạm vi truy cập (public, private, internal, external).
- Sự kiện (event) và emit event.
- Modifiers và ứng dụng để kiểm soát quyền truy cập.
- Cách sử dụng mapping và struct.
- Hiểu sơ lược về constructor và fallback functions.

**Thực hành viết và triển khai một smart contract đơn giản, triển khai và test bằng Remix IDE**

- SimpleContract: với chức năng lưu trữ một số nguyên và lấy ra giá trị đã lưu.

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

contract SimpleStorage {
    uint256 public number;

    uint256[] public arrayOfNumber;
    uint256[5] public fixArrayOfNumber;

    struct Person {
        uint256 number;
        string name;
    }

    Person[] public dynamicArrayOfPerson;

    mapping(string name => uint256 number) public nameToNumber;

    function setNumber (uint256 newNumber) public {
        number = newNumber;
    }

    function getNumber() public view returns(uint256) {
        return number;
    }

    function pushToArrayOfNumber(uint256 newNumber) public {
        arrayOfNumber.push(newNumber);
    }

    function pushToFixArrayOfNumber(uint256 newNumber) public {
        fixArrayOfNumber[0] = newNumber;
    }

    function pushToDynamicArrayOfPerson(uint256 newNumber, string memory
newName) public {
        dynamicArrayOfPerson.push(Person(newNumber, newName));
    }

    function addNumberForName(string memory name, uint256 newNumber) public {
```

```
        nameToNumber[name] = newNumber;
    }

}
```

- Thực hành viết và tham khảo thêm mã nguồn trò chơi ZombieFactory để hiểu rõ hơn về cách tổ chức code và tối ưu.

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

contract ZombieFactory {

    event NewZombie(uint zombieId, string name, uint dna);

    uint dnaDigits = 16;
    uint dnaModulus = 10 ** dnaDigits;
    uint cooldownTime = 1 days;

    struct Zombie {
      string name;
      uint dna;
      uint32 level;
      uint32 readyTime;
    }

    Zombie[] public zombies;

    mapping (uint => address) public zombieToOwner;
    mapping (address => uint) ownerZombieCount;

    function _createZombie(string memory _name, uint _dna) internal {
        uint id = zombies.push(Zombie(_name, _dna, 1, uint32(now +
cooldownTime))) - 1;
        zombieToOwner[id] = msg.sender;
        ownerZombieCount[msg.sender]++;
        emit NewZombie(id, _name, _dna);
    }

    function _generateRandomDna(string memory _str) private view returns
(uint) {
        uint rand = uint(keccak256(abi.encodePacked(_str)));
        return rand % dnaModulus;
    }
```

```
    function createRandomZombie(string memory _name) public {
        require(ownerZombieCount[msg.sender] == 0);
        uint randDna = _generateRandomDna(_name);
        randDna = randDna - randDna % 100;
        _createZombie(_name, randDna);
    }

}
```

**Tìm hiểu cách tối ưu gas**

- Ưu tiên sử dụng biến memory khi có thể.
    - Thay vì sử dụng mapping để lưu trữ mảng zombie của 1 address, cách này sẽ tốn rất nhiều gas trong quá trình trao đổi zombie giữa các address vì chúng được lưu trực tiếp trên blockchain.
    - Ta có thể sử dụng 1 hàm view để trả về mảng id của zombie ứng với mỗi address, cách này sẽ hiệu quả hơn rất nhiều về mặt chi phí.

```
    function getZombiesByOwner(address _owner) external view returns(uint[]
memory) {
        uint[] memory result = new uint[](ownerZombieCount[_owner]);
        uint counter = 0;
        for (uint i = 0; i < zombies.length; i++) {
            if (zombieToOwner[i] == _owner) {
                result[counter] = i;
                counter++;
            }
        }
        return result;
    }
```

- Tránh lặp không cần thiết trong logic.

**3. ĐÁNH GIÁ KẾT QUẢ**

- Đã nắm chắc kiến thức cơ bản về Solidity.
- Tự tin viết và deploy contract đơn giản trên Remix IDE.
- Đã hiểu được cách tối ưu chi phí gas trong những trường hợp cơ bản.
- Còn gặp khó khăn khi tiếp cận các hợp đồng phức tạp.

## 4. THÁCH THỨC & GIẢI PHÁP

### 4.1. Thách thức

- Việc viết smart contract khá phức tạp do mới làm quen.

### 4.2. Giải pháp

- Tiếp tục thực hành viết thêm nhiều smart contract mẫu để nâng cao kiến thức.
- Tham khảo mã nguồn và ví dụ từ các khóa học và Github.

## 5. MỤC TIÊU TUẦN TỚI

- Học cách viết unit test cho contract bằng Foundry.
- Bắt đầu viết smart contract một vài chức năng chính cho dự án Web3 Crowdfunding.
- Hoàn thiện contract với chức năng tạo chiến dịch, đóng góp và rút tiền.

## 6. GHI CHÚ KHÁC

- Em sẽ cố gắng tìm hiểu nhiều hơn trong tuần tới để có thể hoàn thành tốt dự án này và đúng tiến độ.
- Rất mong nhận được góp ý của thầy về bài báo cáo để em có thể cải thiện hơn trong tuần tới.