

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**KHOA CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO ĐỒ ÁN BLOCKCHAIN**

**Nhóm 3**

**Môn học: Các công nghệ mới trong phát triển phần mềm**

**Giảng viên:**

**Trần Văn Quý**

Thành phố Hồ Chí Minh – 2024

## Thành viên

MSSV	Họ Tên
20120383	Nguyễn Đức Tiến
20120447	Trịnh Quốc Cường
20120633	Viên Hải Yên
20120634	Lê Minh Trí

**Contents**

I.	Ý nghĩa đề tài: .....	4
II.	Quá trình thực hiện: .....	5
1.	Xây dựng smart contract: .....	5
2.	Xây dựng giao diện của hệ thống.....	10
III.	Cách sử dụng: .....	11
IV.	Cách cài đặt: .....	17
V.	Tài liệu tham khảo: .....	21

## **I. Ý nghĩa đề tài:**

- Nhóm chọn đề tài “Xây dựng hệ thống gọi vốn IDO (Decentralized Crowdfunding)” để thực hiện đề án cuối kỳ Blockchain.
- Hệ thống gọi vốn IDO là một phương thức huy động vốn mới nổi trong lĩnh vực blockchain, cho phép các dự án có thể tiếp cận trực tiếp với cộng đồng nhà đầu tư mà không cần qua trung gian. Điều này giảm thiểu chi phí và tăng tính minh bạch.
- Hệ thống IDO không chỉ giúp các dự án khởi nghiệp dễ dàng huy động vốn mà còn tạo cơ hội cho các nhà đầu tư nhỏ lẻ tham gia vào các dự án tiềm năng từ giai đoạn đầu.
- Đề tài giúp nhóm tìm hiểu và ứng dụng các công nghệ blockchain, hợp đồng thông minh (smart contracts).

## II. Quá trình thực hiện:

### 1. Xây dựng smart contract:

- Nhóm sử dụng ngôn ngữ solidity để tạo ra smart contract của đề án
- Trong smart contract sẽ bao gồm các thành phần sau:
  - + Hàm createProject để tạo một dự án mới. Nó nhận một đối tượng CreateProjectDTO và kiểm tra các điều kiện đầu vào. Nếu tất cả đều hợp lệ, nó tạo một dự án mới và thêm vào danh sách dự án.

```
function createProject(CreateProjectDTO calldata dto) validSender public returns (Project memory) {
    require(!dto.name.equal(""), "Project name is required");
    require(!dto.shortDescription.equal(""), "Project headline is required");
    require(!dto.description.equal(""), "Project description is required");
    require(dto.maxAllocation > 0, "Max allocation must larger than 0");
    require(dto.totalRaise > 0, "Total raise must larger than 0");
    require(!dto.tokenSymbol.equal(""), "Missing token symbol");

    require(dto.opensAt > block.timestamp * 1000, "Open date should be a date in future");
    require(dto.endsAt > dto.opensAt, "Allocation end date should be larger than open date");

    globalProjectIdCount++;
    // create slug
    string memory projectSlug = createSlug(dto.slug);
    // slugPool.push(projectSlug);
    uniqueSlugMap[projectSlug] = true;

    Project memory project = Project(
        globalProjectIdCount,
        msg.sender,
        dto.name,
        projectSlug,
        dto.shortDescription,
        dto.description,
        dto.logoUrl,
        dto.coverBackgroundUrl,
        TokenInformation(dto.tokenSymbol, dto.tokenSwapRatio),
        ProjectSchedule(block.timestamp * 1000, dto.opensAt, dto.endsAt),
        ProjectAllocation(dto.maxAllocation, dto.totalRaise),
        0,
        0,
        new address[](0)
    );
    projectList.push(project);

    return project;
}
```

- + Hàm getProjectList để trả về danh sách tất cả các dự án.

```
function getProjectList() public view returns (Project[] memory) {
    return projectList;
}
```

+ Hàm `stakingInProject` cho phép người dùng đầu tư vào dự án. Nó kiểm tra các điều kiện và cập nhật thông tin dự án cũng như nhà đầu tư.

```
function stakingInProject(uint256 projectId) public payable validSender {
    int256 index = findIndexOFProject(projectId);
    require(index > -1, "project_not_found");

    Project storage project = projectList[uint256(index)];
    uint256 userStakeInWei = msg.value;

    // check owner self staking
    require(msg.sender != project.owner, "project_owner");

    // check stake time is early or late
    require(block.timestamp * 1000 >= project.schedule.opensAt && block.timestamp * 1000 <= project.schedule.endsAt, "staking_not_open");

    // check is full of staking or not
    require(project.allocation.totalRaise > project.currentRaise, "target_reached");

    // check min allocation
    require(userStakeInWei > 0, "not_enough");

    // check valid allocation
    VestingInfor[] storage vestingList = projectToInvestorMap[project.id][msg.sender];

    uint256 totalStakeInWei = 0;
    for (uint256 i = 0; i < vestingList.length; i++) {
        totalStakeInWei += vestingList[i].stakeAmountInWei;
    }

    require(totalStakeInWei + userStakeInWei <= project.allocation.maxAllocation, "max_allocation");
    require(project.currentRaise + userStakeInWei <= project.allocation.totalRaise, "too_much");

    // add amount to storage
    project.currentRaise += userStakeInWei;

    // add investor to project
    if (!uniqueProjectInvestorMap[project.id][msg.sender]) {
        project.totalParticipants++;
        uniqueProjectInvestorMap[project.id][msg.sender] = true;
        project.investors.push(msg.sender);
    }

    vestingList.push(VestingInfor(userStakeInWei, block.timestamp * 1000));

    // count global participant
    if (!uniqueParticipantMap[msg.sender]) {
        uniqueParticipantMap[msg.sender] = true;
        globalUniqueParticipantCount++;
    }
}
```

+ Hàm `getBalance` trả về số dư hiện tại của hợp đồng.

```
function getBalance() public view returns (uint256) {
    return address(this).balance;
}
```

+ Hàm `getDexMetris` trả về các chỉ số của nền tảng, bao gồm tổng số dự án, số lượng người tham gia duy nhất, và tổng số tiền huy động được.

```
function getDexMetris() public view returns (DexMetrics memory) {
    uint256 totalRaised = 0;

    for (uint256 i = 0; i < projectList.length; i++) {
        totalRaised += projectList[i].currentRaise;
    }

    return DexMetrics(projectList.length, globalUniqueParticipantCount, totalRaised);
}

function getProjectDetail(string calldata slug) public view returns (Project memory) {
    int256 index = findIndexOfProject(slug);

    require(index > -1, "not_found");

    Project memory project = projectList[uint256(index)];
    return project;
}
```

+ Hàm `getProjectDetail` trả về chi tiết của một dự án dựa trên slug của nó.

```
function getProjectDetail(string calldata slug) public view returns (Project memory) {
    int256 index = findIndexOfProject(slug);

    require(index > -1, "not_found");

    Project memory project = projectList[uint256(index)];
    return project;
}
```

+ Hàm `getProjectStakingByInvestor` trả về thông tin staking của một nhà đầu tư trong một dự án cụ thể.

```
function getProjectStakingByInvestor(uint256 projectId) public view returns (VestingInfor[] memory) {
    require(uniqueProjectInvestorMap[projectId][msg.sender], "staking_not_found");

    return projectToInvestorMap[projectId][msg.sender];
}
```

+ Hàm `getStakedProjectByInvestor` trả về danh sách các dự án mà nhà đầu tư đã tham gia.

```
function getStakedProjectByInvestor() public view returns (Project[] memory) {
    address investor = msg.sender;

    uint256 count = 0;
    for (uint256 i = 0; i < projectList.length; i++) {
        if (uniqueProjectInvestorMap[projectList[i].id][investor]) {
            count++;
        }
    }

    Project[] memory list = new Project[](count);
    for (uint256 i = 0; i < projectList.length; i++) {
        if (uniqueProjectInvestorMap[projectList[i].id][investor]) {
            list[i] = projectList[i];
        }
    }

    return list;
}
```

+ Hàm `automateDeliverMoney` để tự động phân phối tiền sau khi giai đoạn huy động vốn kết thúc. Nếu dự án đạt được mục tiêu, tiền sẽ được gửi cho chủ dự án. Nếu không, tiền sẽ được hoàn lại cho các nhà đầu tư.



```
function automateDeliverMoney(string calldata slug) public onlyOwner returns (string memory) {
    uint256 index = findIndexOfProject(slug);
    require(index > -1, "not_found");

    Project memory project = projectList[uint256(index)];

    // Project funding still in progress
    require(block.timestamp * 1000 >= project.schedule.endsAt, "Funding period still in progress");

    // Check funding condition
    bool isFullyFunded = project.currentRaise >= project.allocation.totalRaise * 90 / 100;

    if (isFullyFunded) {
        require(address(this).balance >= project.currentRaise, "Contract balance is not enough");

        (bool sent,) = payable(project.owner).call{value: project.currentRaise}("");
        if (!sent) {
            return "Sending money to owner failed";
        }

        return "Successfully sent money to owner";
    }

    address[] memory investors = project.investors;
    uint count = 0;
    for (uint256 i = 0; i < project.totalParticipants; i++) {
        VestingInfor[] memory infor = projectToInvestorMap[project.id][investors[i]];

        // this investor is refunded
        if (projectToInvestorRefundMap[project.id][investors[i]]) {
            continue;
        }

        uint256 total = 0;
        for (uint256 j = 0; j < infor.length; j++) {
            total += infor[j].stakeAmountInWei;
        }

        (bool sent,) = payable(investors[i]).call{value: total}("");
        if (sent) {
            projectToInvestorRefundMap[project.id][investors[i]] = true;
            count++;
        }
    }

    return string.concat("Successfully sent money to ", Strings.toString(count), " investors");
}
```

+ Các hàm createSlug, findIndexOffProject để tạo slug duy nhất và tìm chỉ số của dự án trong danh sách dự án dựa trên ID hoặc slug.

```
function createSlug(string calldata str) private view returns (string memory) {
    string memory slug = str;
    while (uniqueSlugMap[slug]) {
        uint256 randNum = uint256(keccak256(abi.encodePacked(block.timestamp, msg.sender)));
        slug = string.concat(str, "-", Strings.toString(randNum));
    }

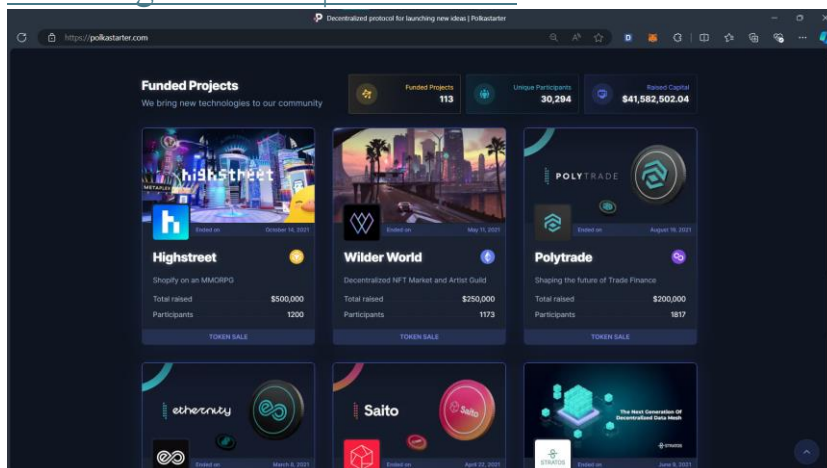
    return slug;
}

function findIndexOffProject(uint256 projectId) private view returns (int256) {
    for (uint256 i = 0; i < projectList.length; i++) {
        if (projectList[i].id == projectId) {
            return int256(i);
        }
    }
    return -1;
}

function findIndexOffProject(string calldata slug) private view returns (int256) {
    for (uint256 i = 0; i < projectList.length; i++) {
        if (projectList[i].slug.equal(slug)) {
            return int256(i);
        }
    }
    return -1;
}
```

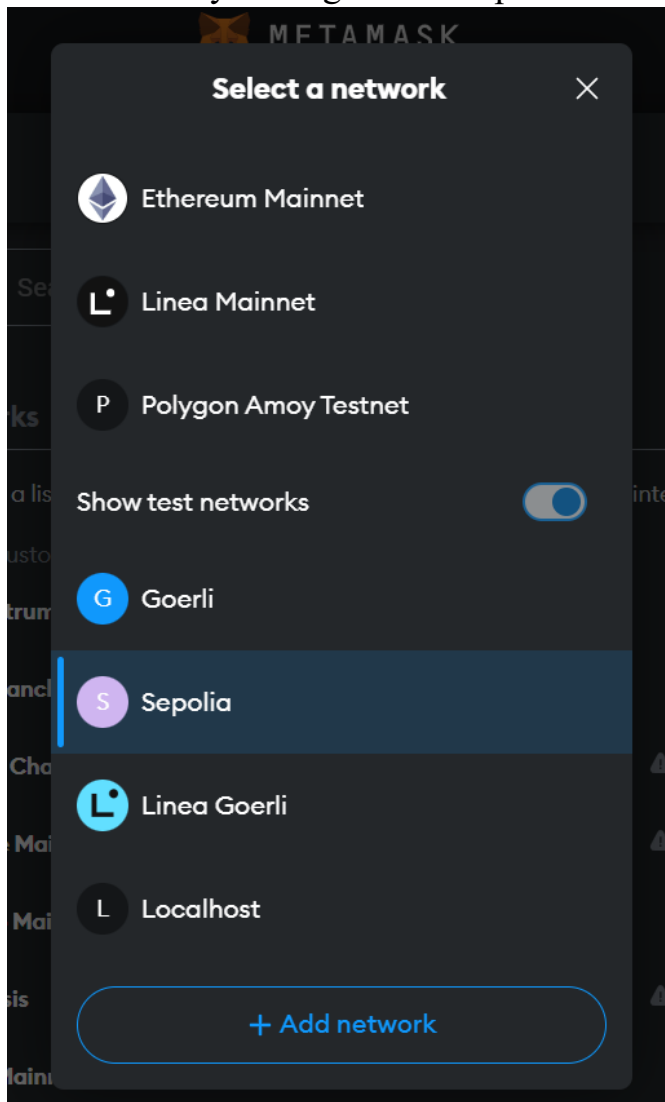
## 2. Xây dựng giao diện của hệ thống

- Nhóm xây dựng giao diện dựa trên một website: [Decentralized protocol for launching new ideas | Polkastarter](https://polkastarter.com)

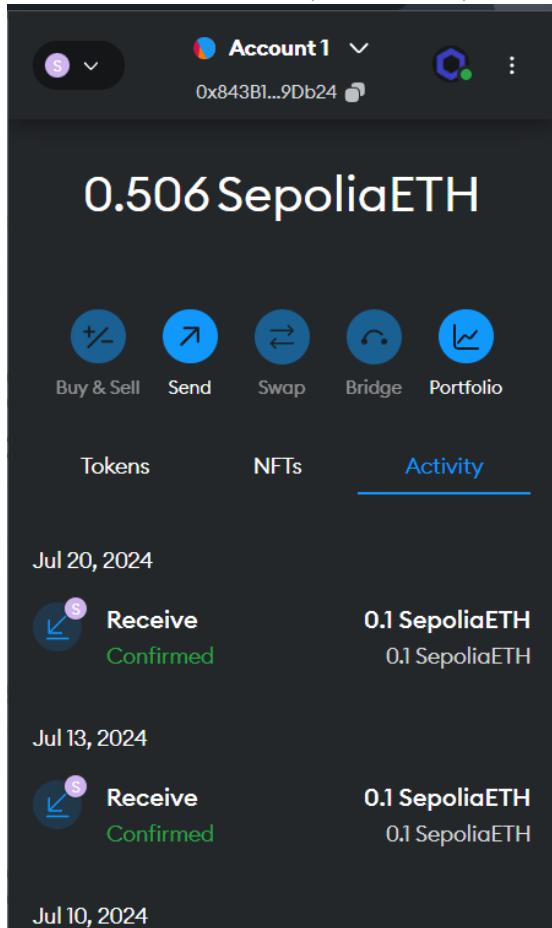


### III. Cách sử dụng:

- B1: Tạo ví metamask.
- B2: Ta sẽ chuyển sang testnet Sepolia

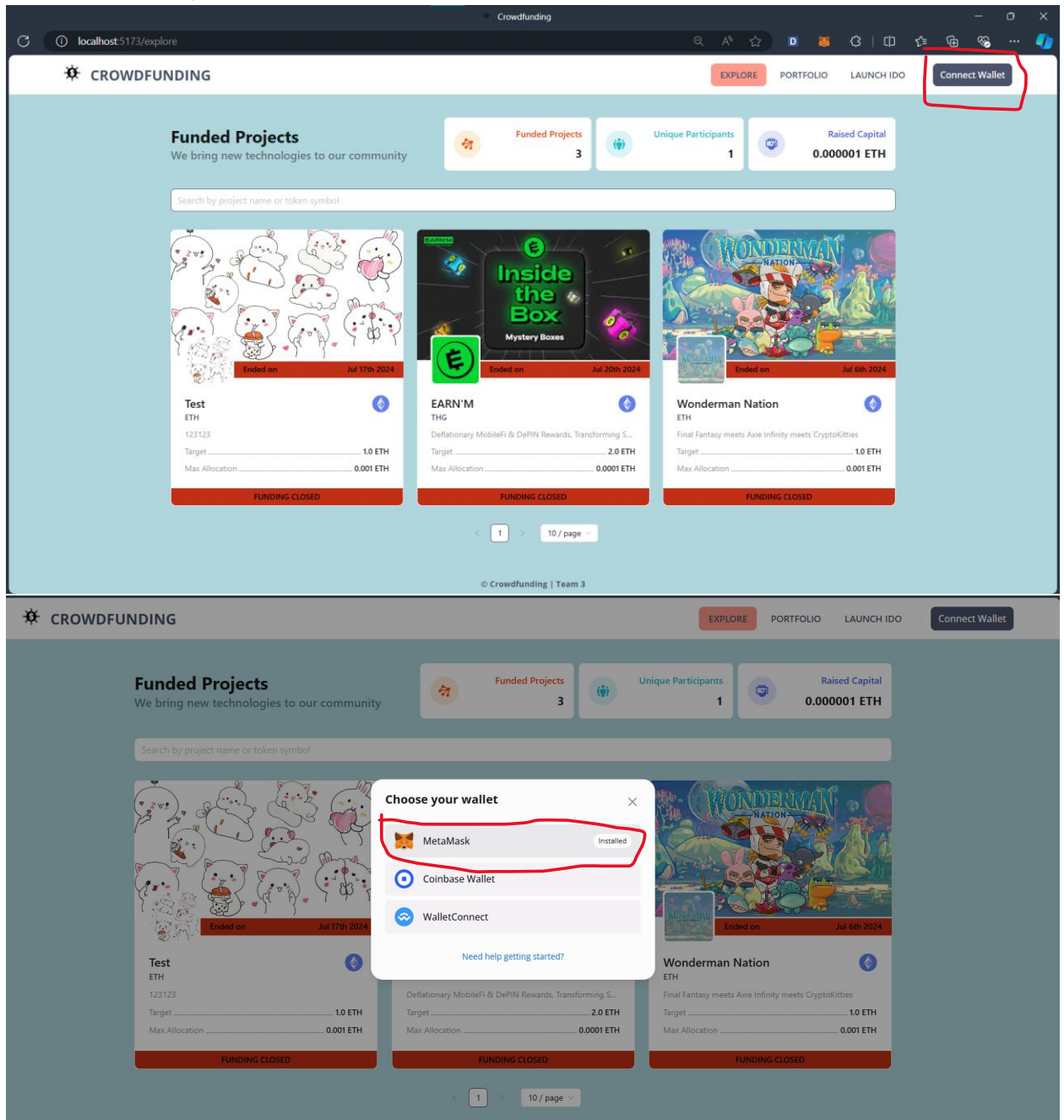


- B3: Ta sẽ đi lấy ETH free cho testnet sepolia tại trang web: [Sepolia Faucet - Get Testnet Tokens \(chain.link\)](#)



- B4: Ta sẽ chạy hệ thống theo cách cài đặt

- B5: Ta sẽ bắt tạo connect đến ví



- B6: Ta sẽ tạo 1 project

The image shows a web application for launching a project on a crowdfunding platform. The form is titled "Launch a Project" and includes the following fields and sections:

- Project Title:** MOBLAND
- Project Slug:** mobland
- Short Description:** The Mafia Metaverse (19 / 100 characters)
- Description:** MOBLAND is the first-ever Mafia Metaverse. Here you fight, loot, build and govern through your own syndicate.
- Logo URL:** [https://polkastarter.com/\\_next/image?url=https%3A%2F%2Fencrypted-tbn0.gstatic.com/images?q=tbn:ANdRK...](https://polkastarter.com/_next/image?url=https%3A%2F%2Fencrypted-tbn0.gstatic.com/images?q=tbn:ANdRK...)
- Cover Image URL:** <https://encrypted-tbn0.gstatic.com/images?q=tbn:ANdRK...>
- Max Allocation:** 0.001 ETH
- Target:** 1 ETH
- Token Symbol:** THG
- Token Swap Ratio:** 0.001
- Funding Period:** 2024/07/20 20:41:00 - 2024/07/31 23:59:59









Below the form, there is a "Before you submit" section with the following text:

- Before you submit
- After submitting, you cannot edit or delete your project on this site.
- When the funding period ends, if your project is not staked at 90%, all money will be sent back to investors!

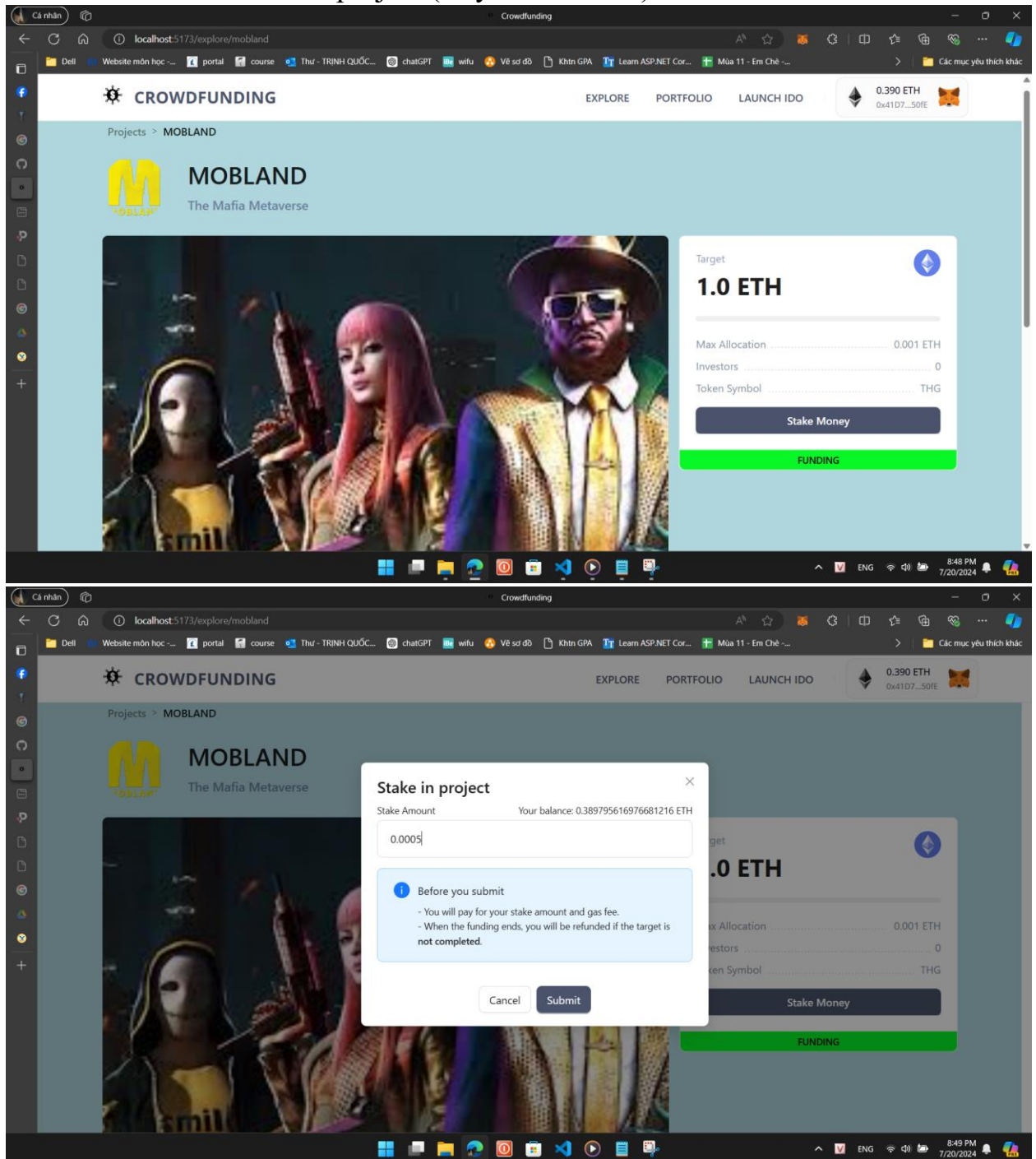
The form has "Cancel" and "Submit" buttons. A MetaMask transaction confirmation overlay is visible on the right side of the screen, showing the following details:

- Account 1
- Estimated fee: 0.03724575 SepoliaETH
- Max fee: 0.03724575 SepoliaETH
- Total: 0.03724575 SepoliaETH
- Amount + gas fee: 0.03724575 SepoliaETH
- Buttons: Reject, Confirm

- Ta có thể thấy được project đã tạo

Your projects						
Project name	Participants	Total raised	Swap ratio	Status	Ends at	Chain
 Test ETH	0	0 ETH	1 ETH = 0.05 ETH	FUNDING CLOSED	July 17th 2024 3:46:02 PM+07:00	
 EARN'M THG	1	0.000001 ETH	1 ETH = 0.01 THG	FUNDING CLOSED	July 20th 2024 4:29:36 PM+07:00	
 Wonderman Nation ETH	0	0 ETH	1 ETH = 1 ETH	FUNDING CLOSED	July 6th 2024 1:32:23 AM+07:00	
 MOBLAND THG	0	0 ETH	1 ETH = 0.001 THG	UPCOMING	July 31st 2024 11:59:59 PM+07:00	
< 1 >						

- B7: Ta có thể đầu tư vào 1 project (Đây là 1 ví khác)



- Khi ta đầu tư thì ta sẽ có token cho cái dự án này. Khi dự án thành công thì người đầu tư có thể bán token đó để thu lợi nhuận



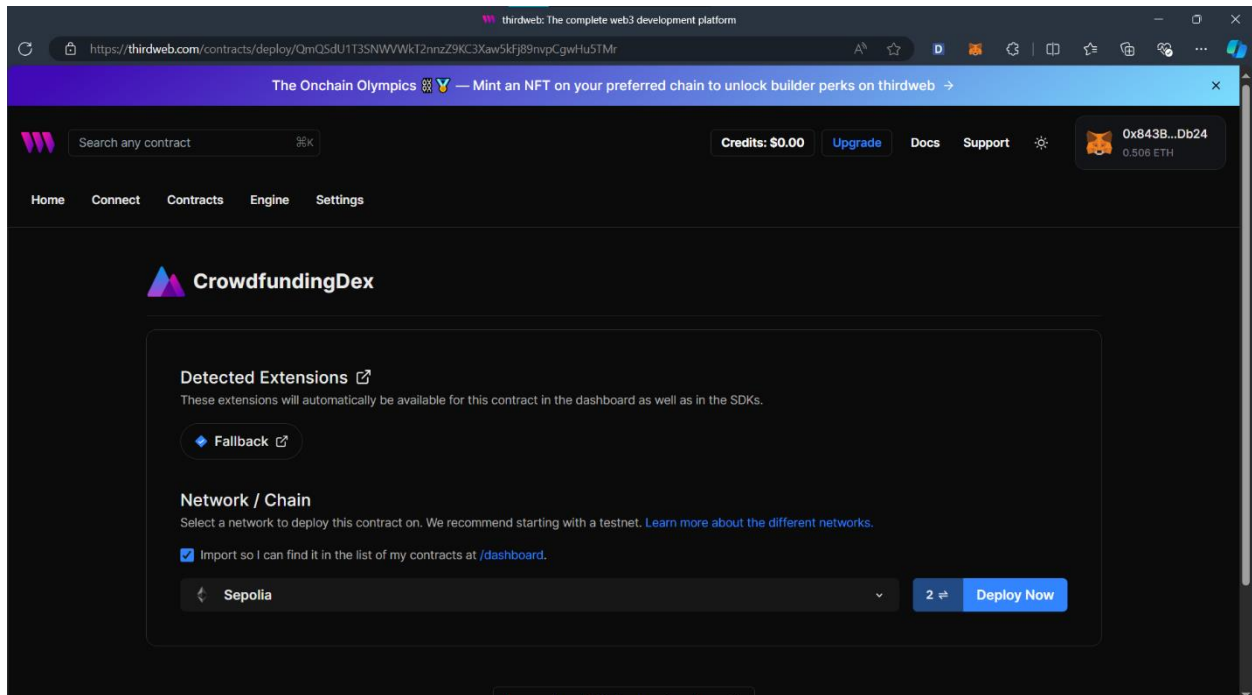
#### IV. Cách cài đặt:

- B1: Ta sẽ deploy smart contract lên thirdweb
  - o Ta phải tạo file .env trong thư mục contract dựa trên .env.template

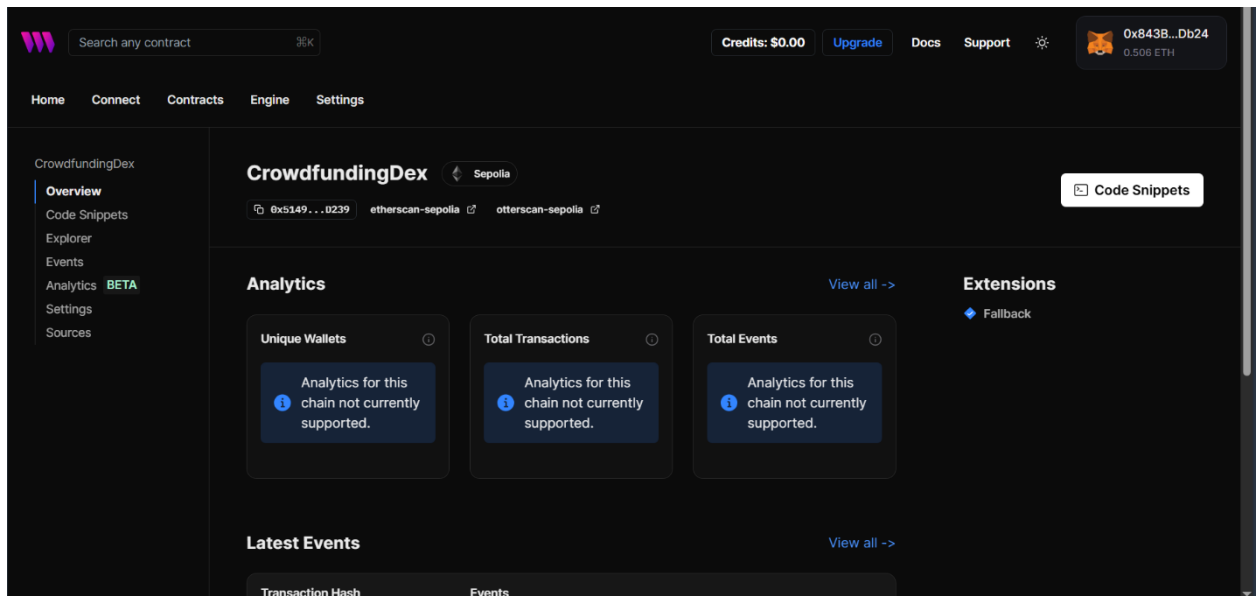
[illegible]

- Sau đó ta chạy câu lệnh npm run deploy tại thư mục Contract

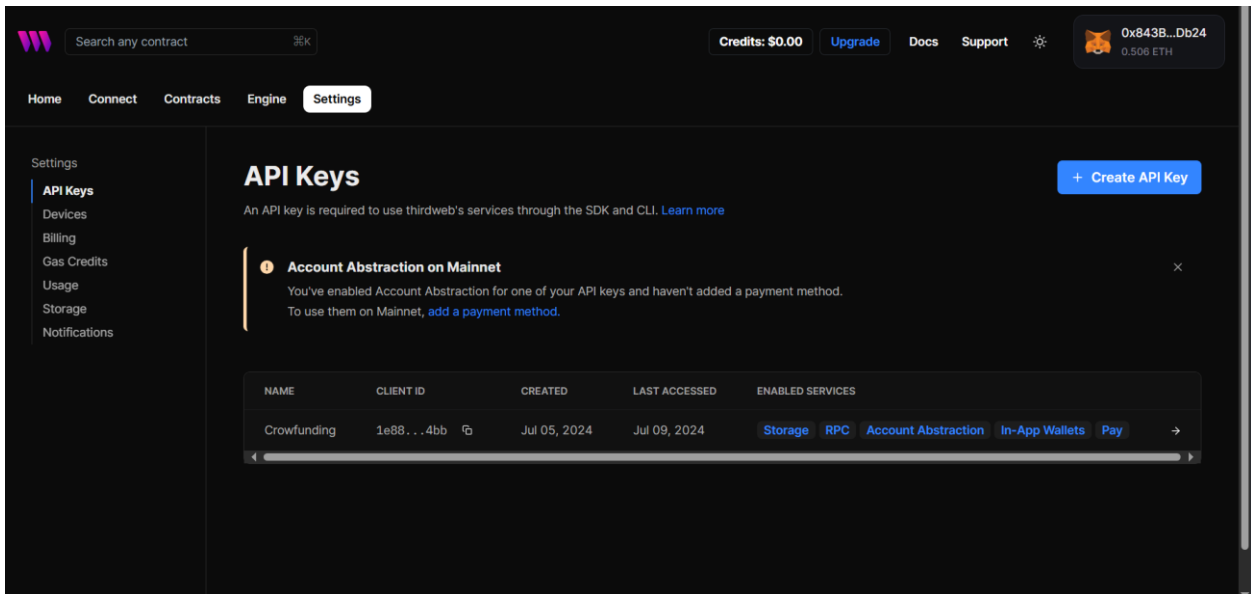
- Ta sẽ bấm vào link để deploy



- Lưu ý là khi ta deploy thì ta sẽ cần sử dụng testnet là sepolia và trong ví của ta phải có tiền ETH thì ta mới có thể deploy. Sau khi deploy thành công ta có thể copy được địa chỉ của contract



- B2: Ta sẽ tạo API key trên thirdweb

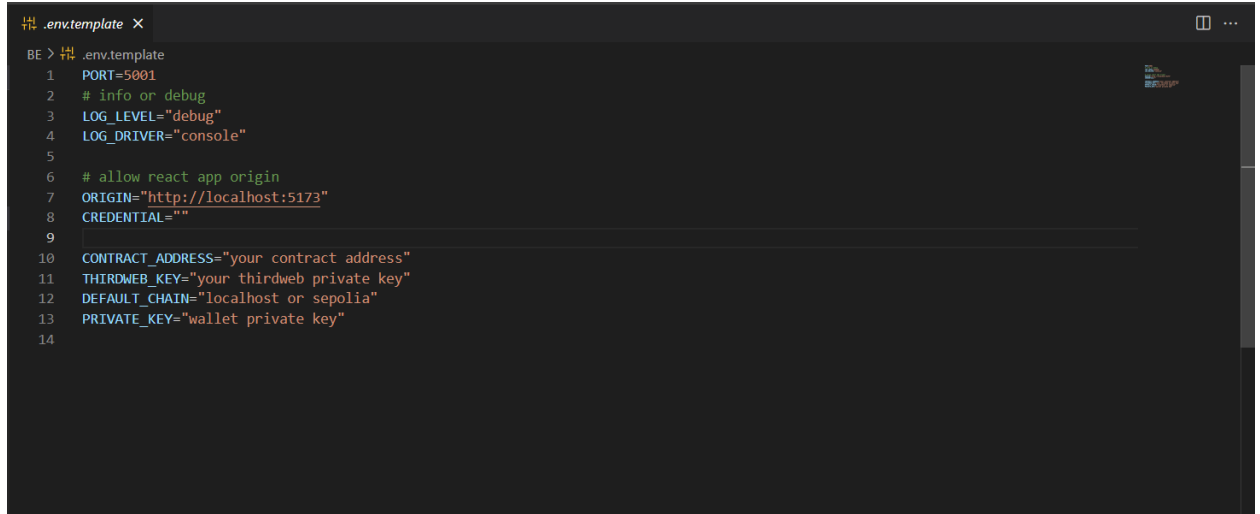


- B3: Ta sẽ copy contract address và ClientID của API key để tạo .env cho thư mục front end, active\_chain ta sẽ sử dụng sepolia vì ta sử dụng testnet là sepolia

```
.envxtemplate X
FE > .envxtemplate
1 VITE_APP_NAME="Crowdfunding"
2
3 VITE_THIRDWEB_CLIENT_ID="your thirdweb client id"
4 VITE_CONTRACT_ADDRESS="contract address"
5 VITE_ACTIVE_CHAIN="localhost or sepolia"
6 # url of the automation backend app
7 VITE_API_URL="http://localhost:5001"
```

- B4: Ta có thể npm install để cài các dependencies
- B5: Ta bắt đầu chạy hệ thống bằng npm run dev và truy cập <http://localhost:5173/> để sử dụng hệ thống

- B6: Ta sẽ tạo .env cho thư mục back end. Ta có thể lấy private key của ví trên metamask



```
.env.template X
BE > .env.template
1  PORT=5001
2  # info or debug
3  LOG_LEVEL="debug"
4  LOG_DRIVER="console"
5
6  # allow react app origin
7  ORIGIN="http://localhost:5173"
8  CREDENTIAL=""
9
10 CONTRACT_ADDRESS="your contract address"
11 THIRDSWEB_KEY="your thirdweb private key"
12 DEFAULT_CHAIN="localhost or sepolia"
13 PRIVATE_KEY="wallet private key"
14
```

- B7: Ta npm install để cài các dependencies
- B8: Ta npm run start để chạy back-end

## **V. Tài liệu tham khảo:**

Viết 1 Smart Contract đơn giản sử dụng ngôn ngữ Solidity dựa trên mạng lưới của Ethereum (viblo.asia)

Decentralized protocol for launching new ideas | Polkastarter

Github Repo: Cuongparish/Crowdfunding (github.com)

Demo: Blockchain project : Crowdfunding - Team 3 - YouTube