

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



BÁO CÁO LAB 3

|Giáo viên hướng dẫn|

Nguyễn Đình Thúc

Ngô Đình Hy

Trần Hà Sơn

Tên đề án: Solidity Smart Contract Development

Học phần: Blockchain và Ứng dụng

Nhóm thực hiện: Nhóm 7

Thành phố Hồ Chí Minh, ngày 24 tháng 12 năm 2023

MỤC LỤC

I. BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ THÀNH VIÊN	1
II. NỘI DUNG BÁO CÁO	1
1. Mục tiêu.....	1
2. Cơ sở lý thuyết.....	1
3. Kịch bản cài đặt:	2
a. Front-end	2
b. Back-end	3
4. Hướng dẫn cài đặt và Demo :.....	5
5. Kiến thức và kỹ năng thu hoạch được.....	5
a. Kiến thức:.....	5
b. Kỹ năng:.....	5
6. Tài liệu tham khảo:	5

I. BẢNG PHÂN CÔNG VÀ ĐÁNH GIÁ THÀNH VIÊN

MSSV	Họ tên	Phân công	Hoàn thành
20120056	Trần Quốc Đình	Tìm hiểu, viết báo cáo	100%
20120060	Nguyễn Trí Đức	Tìm hiểu, viết báo cáo	100%
20120078	Nguyễn Thế Hiển	Cài đặt ứng dụng: backend và frontend	100%
20120116	Phạm Lê Quốc Khánh	Tìm hiểu, viết báo cáo	100%
20120313	Phan Tấn Kiệt	Cài đặt ứng dụng: backend và frontend	100%

II. NỘI DUNG BÁO CÁO

1. Mục tiêu

Hiểu được một số kiến thức cơ bản về Solidity, Smart Contract và Ethereum, sau đó ứng dụng cài đặt hệ thống Decentralized Voting sử dụng Smart Contract.

2. Cơ sở lý thuyết

Solidity: là ngôn ngữ lập trình cấp cao được thiết kế đặc thù để tạo các hợp đồng thông minh trên nền tảng Ethereum.

Ethereum: là một nền tảng Blockchain phi tập trung cho phép tạo ra các ứng dụng phi tập trung (Dapp) sử dụng smart contract.

Smart Contract: có thể hiểu đơn giản là chương trình được lưu trữ trên Blockchain (Ethereum) và sẽ tự động thực thi khi các điều kiện trong hợp đồng được đáp ứng.

Hệ thống bỏ phiếu phi tập trung – Decentralized Voting System:

1. Giới thiệu:

Ở bài lab này, chúng em phát triển một hệ thống bỏ phiếu phi tập trung sử dụng hợp đồng thông minh hoạt động trên nền tảng Ethereum cho phép người dùng bỏ phiếu an toàn và minh bạch.

Smart Contract được sử dụng trong hệ thống nhằm ngăn chặn việc bỏ phiếu hai lần và đảm bảo tính toàn vẹn của quá trình bỏ phiếu.

2. Tương tác của Smart Contract với Ethereum:

Triển khai Smart Contract trên mạng Ethereum (ví dụ trên ví MetaMask) bằng cách biên dịch file DAO.sol chứa phần cài đặt Smart Contract, sau đó deploy lên Ethereum thành một giao dịch-Transaction, và sử dụng địa chỉ của Smart Contract và ABI của nó để chạy hệ thống Decentralized Voting bên ngoài.

3. Cách Smart Contract giải quyết bài toán Decentralized Voting:

Sau khi tạo một mạng Private Ethereum blockchain, người dùng sẽ đăng nhập vào ví (ví dụ MetaMask) với một tài khoản ảo sử dụng Private Key được tạo cục bộ, triển khai Dapp để tương tác với Smart Contract đã được deploy trên mạng Ethereum đó,

Người dùng có thể tạo một đề xuất (Proposal) với các thông tin như: tên proposal, mô tả về proposal, và thời gian bỏ phiếu hợp lệ. Trong khoảng thời gian bỏ phiếu hợp lệ, các người dùng trong mạng có thể thực hiện bỏ phiếu cho các proposal được hiển thị tại dashboard. Sau khi kết thúc thời gian bỏ phiếu, dựa vào số phiếu Yes và số phiếu No để quyết định proposal đó có được đồng thuận - Adopt hay thất bại - Fail hay từ chối - Rejected.

Bên cạnh đó, Smart Contract còn được thiết kế nhằm chống lại tình trạng một account thực hiện vote cho một proposal nhiều hơn một lần (bằng cách kiểm tra chữ kí của người vote trong danh sách voted của proposal). Điều này sẽ giúp ngăn chặn tình trạng bỏ phiếu kép.

Ngoài ra, mỗi lượt bỏ phiếu đối với mỗi proposal sau khi được ghi nhận và thêm vào blockchain đều không thể thay đổi. Điều này giúp bảo vệ tính toàn vẹn của kết quả bỏ phiếu (vì không có cách nào có thể can thiệp và sửa đổi).

Cuối cùng, với mỗi lượt vote bao gồm các thông tin như: địa chỉ tài khoản voter, thông tin vote đều sẽ được mã hóa. Điều này giúp đảm bảo được tính an toàn của Decentralized Voting System.

4. Tầm quan trọng và những cải tiến trong tương lai:

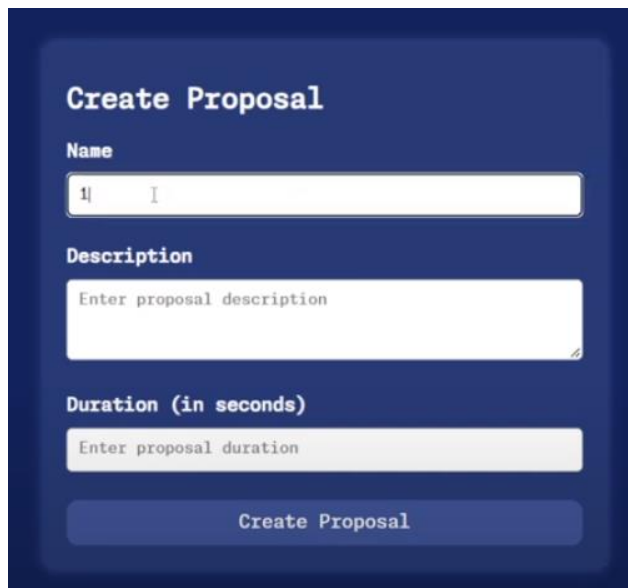
Với những ưu điểm và tiện lợi của Blockchain cũng như Smart Contract, Decentralized Voting System mang lại nhiều lợi ích cho người dùng như tính trung thực và minh bạch, độ hiệu quả và chính xác, loại bỏ các vấn đề gian lận và thao túng gây khó khăn cho hệ thống bỏ phiếu truyền thống.

Trong tương lai, hệ thống có thể sẽ phát triển thêm những tính năng khác như tích hợp lưu trữ an toàn trên đám mây, tích hợp nhận dạng khuôn mặt. Thậm chí, Decentralized Voting System còn có khả năng phát triển thành một ứng dụng bầu cử lãnh đạo thay thế cho phương pháp bầu cử truyền thống trong bối cảnh những thách thức về an ninh và tính minh bạch đang ngày càng phức tạp như hiện nay.

3. Kịch bản cài đặt:

a. Front-end

- Giao diện được cài đặt bởi ngôn ngữ JavaScript và sử dụng thư viện như web3, react-router-dom...
- Các cài đặt giao diện được đặt trong thư mục components:
 - Thư mục utils chứa các cài đặt hàm hỗ trợ:
 - Hàm getStatusString và getStatusColor trả về thông tin trạng thái của Proposal dưới dạng chuỗi.
 - Hàm formatTimestamp và TimestampToTime trả về thông tin của thời điểm timestamp được nhập vào, trả về dưới dạng chuỗi.
 - Thư mục style chứa các file css thiết kế các page của giao diện người dùng.
 - Thư mục pages chứa các file js cài đặt các page của giao diện người dùng:
 - newProposal.js: chứa cài đặt của chức năng tạo Proposal mới, người dùng sẽ phải nhập đúng hết các trường thông tin, nếu không thì sẽ báo lỗi cho người dùng.



Create Proposal

Name

1/

Description

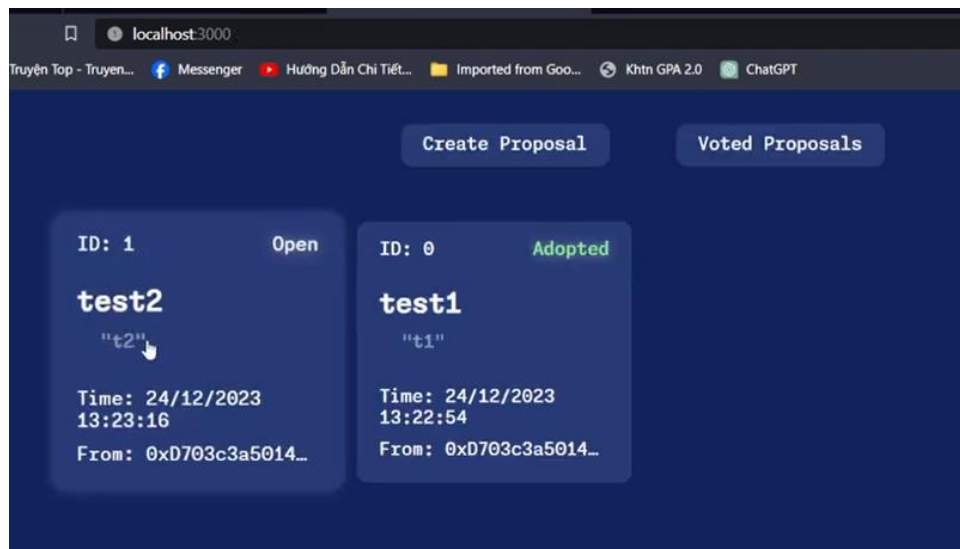
Enter proposal description

Duration (in seconds)

Enter proposal duration

Create Proposal

- Card.js: chứa cài đặt giao diện chính hiển thị thông tin của các Proposal hiện tại, nút tạo Proposal mới và các Proposal đã vote



- Proposal.js: chứa cài đặt của đối tượng Proposal và các hàm xử lý các thao tác trên các đối tượng trên giao diện, hiển thị thông tin của các thông tin của Proposal tại thời điểm hiện tại.



b. Back-end

- Cấu hình Decentralized Autonomous Organization - DAO:
 - Trong file DAO.sol định nghĩa một DAO với chức năng tạo và bỏ phiếu cho các proposal.
 - ‘enum ProposalStatus’: Định nghĩa một kiểu dữ liệu liệt kê (enum) ProposalStatus với các giá trị có thể là ‘Open’, ‘Rejected’, ‘Adopted’, và ‘Fail’. Mỗi giá trị thể hiện giá trị trạng thái của các giao dịch.

- ‘struct Proposal’: Định nghĩa một cấu trúc Proposal để biểu diễn thông tin, bao gồm:
 - id: Mã số định danh của proposal.
 - proposer: Địa chỉ của người proposal.
 - name và description: Tên và mô tả của proposal.
 - createTime: Thời gian tạo proposal.
 - duration: Thời gian diễn ra cuộc bỏ phiếu.
 - status: Trạng thái của proposal (sử dụng ProposalStatus).
 - hasVoted: Một ánh xạ (mapping) từ địa chỉ người bỏ phiếu đến trạng thái đã bỏ phiếu hay chưa.
 - yesVotes và noVotes: Số phiếu thuận và phiếu chống.
 - Hàm ‘createProposal’:
 - Cho phép người dùng tạo một proposal mới.
 - Kiểm tra rằng thời gian diễn ra cuộc bỏ phiếu (_duration) phải lớn hơn 0.
 - Tạo một proposal mới và thêm vào mảng proposals.
 - Tăng giá trị của proposalCount.
 - Hàm ‘vote’:
 - Cho phép người dùng bỏ phiếu cho một proposal cụ thể (_proposalId).
 - Kiểm tra xem proposal có tồn tại hay không.
 - Kiểm tra thời gian diễn ra cuộc bỏ phiếu còn lại, nếu đã kết thúc, đóng proposal và không ghi nhận. Ngược lại, ghi lại phiếu và tăng giá trị của phiếu chọn.
 - Hàm ‘getProposalCount’: Trả về số lượng proposal hiện tại.
 - Hàm ‘getProposal’: Trả về thông tin chi tiết của một proposal cụ thể (_proposalId).
 - Hàm ‘getRangeProposals’:
 - Hàm này lấy thông tin của một dãy các proposal được chỉ định bởi startId và endId.
 - Trước tiên, nó kiểm tra xem khoảng đã cho có hợp lệ không, sau đó khởi tạo các mảng để lưu trữ thông tin về mỗi proposal trong khoảng đó.
 - Hàm lặp qua các proposal trong khoảng đã chỉ định và điền thông tin vào các mảng tương ứng.
 - Cuối cùng, nó trả về các mảng chứa chi tiết về các proposal.
 - Hàm ‘hasVotedForProposal’:
 - Hàm này kiểm tra xem một cử tri cụ thể đã bỏ phiếu cho một proposal cụ thể (proposalId) chưa.
 - Trước hết, nó kiểm tra xem ID của proposal có hợp lệ không.
 - Sau đó, nó truy cập vào bảng băm hasVoted trong proposal được chỉ định để xác định xem tài khoản đã bỏ phiếu chưa.
 - Hàm ‘getVoterVotedProposals’:
 - Hàm này trả về một mảng chứa ID của các proposal mà một tài khoản cụ thể đã bỏ phiếu.
 - Nó lặp qua tất cả các proposal và kiểm tra xem tài khoản đã bỏ phiếu cho từng proposal hay không.
 - Nếu tài khoản đã bỏ phiếu cho một proposal, ID của nó sẽ được thêm vào mảng votedProposalIds.
 - Sau khi lặp, hàm điều chỉnh kích thước của mảng để loại bỏ các phần tử không sử dụng và sau đó trả về mảng.
 - Hàm ‘_closeProposal’:
 - Hàm nội bộ này được sử dụng để đóng một proposal và xác định trạng thái cuối cùng của nó dựa trên kết quả bỏ phiếu.
 - Nó so sánh số phiếu "Yes" và "No" và đặt trạng thái của proposal tương ứng:
- Cấu hình file ‘abis.js’:

- File ‘abis.js’ chứa một mô tả của ABI (Application Binary Interface) cho một smart contract trên blockchain Ethereum. ABI định nghĩa các phương thức và cấu trúc dữ liệu trong smart contract để nó có thể tương tác được với các ứng dụng bên ngoài.

4. Hướng dẫn cài đặt và Demo :

- Cần cài đặt các package sau đây để khởi chạy frontend:
 - npm install react-router-dom
 - npm install web3
- Khởi chạy chương trình theo link hướng dẫn: [tại đây](#)

5. Kiến thức và kỹ năng thu hoạch được

a. Kiến thức:

- Hiểu được những kiến thức nền tảng về Solidity, Smart Contract, Dapp và cách chúng tương tác trên nền tảng Blockchain như Ethereum.
- Tự tay cài đặt một Dapp ứng dụng Smart Contract và có tiềm năng phát triển trong tương lai.

b. Kỹ năng:

- Hoàn thiện kỹ năng làm việc nhóm;
- Hoàn thiện kỹ năng research;
- Hoàn thiện kỹ năng đọc hiểu nhanh tài liệu bằng tiếng Anh

6. Tài liệu tham khảo:

- [1] Tài liệu môn học trên Skype và Moodle môn học
- [2] [Secure Voting Website Using Ethereum and Smart Contracts](#)
- [3] [A remote and cost-optimized voting system using blockchain and smart contract](#)
- [4] [References Front-end](#)