TRƯỜNG ĐẠI HỌC
SƯ PHẠM KỸ THUẬT TP. HỒ CHÍ MINH
HCMC University of Technology and Education
HCMUTE

KHOA CÔNG NGHỆ THÔNG TIN

MÔN: BLOCKCHAIN VÀ ỨNG DỤNG

BÁO CÁO CUỐI KỲ

***

XÂY DỰNG ESCROW DAPP FREELANCETRUST

GVHD: LÊ QUANG THÁI

SVTH:

1. Lê Anh Khoa 22162016
2. Nguyễn Thắng Lợi 22162023
3. Nguyễn Ngọc Đông Phương 22162033
4. Trương Nguyễn Anh Thư 22162047
5. Nguyễn Văn Trường 22162052

Mã lớp: BCAP433280

Thành phố Hồ Chí Minh, Tháng 12 Năm 2025

DANH SÁCH THÀNH VIÊN

HỌC KỲ I NĂM HỌC 2025-2026

Nhóm: 2

Tên đề tài: XÂY DỰNG ESCROW DAPP FREELANCETRUST

STT	HỌ VÀ TÊN SINH VIÊN	MSSV	TỈ LỆ % HOÀN THÀNH
1	Lê Anh Khoa	22162016	100%
2	Nguyễn Thắng Lợi	22162023	100%
3	Nguyễn Ngọc Đông Phương	22162033	100%
4	Trương Nguyễn Anh Thư	22162047	100%
5	Nguyễn Văn Trường	22162052	100%

Ghi chú:

- Tỷ lệ % = 100%: Mức độ phần trăm của từng sinh viên tham gia.

Nhận xét của giáo viên

.....

.....

.....

.....

Ngày 26 tháng 12 năm 2025

GVHD

MỤC LỤC

Phần 1: MỞ ĐẦU

1. Tóm tắt ý tưởng

Đề tài *"Xây dựng Escrow DApp FreelanceTrust"* tập trung vào việc phát triển một ứng dụng phi tập trung dựa trên công nghệ Blockchain Ethereum. Ý tưởng cốt lõi của hệ thống là xây dựng một mô hình Escrow thông minh, trong đó khoản thanh toán của Client sẽ được khóa trong Smart Contract ngay khi hợp đồng được khởi tạo. Số tiền này chỉ được giải ngân cho Freelancer khi công việc đã hoàn thành và được Client xác nhận, hoặc xử lý theo phán quyết của bên thứ ba trong trường hợp phát sinh tranh chấp. Cơ chế này giúp hạn chế gian lận, nâng cao tính minh bạch và đảm bảo quyền lợi cho tất cả các bên tham gia.

Hệ thống được thiết kế với kiến trúc gồm ba thành phần chính: Smart Contract viết bằng ngôn ngữ Solidity và triển khai trên mạng Ethereum, giao diện Web3 cho phép người dùng tương tác với hợp đồng thông minh thông qua trình duyệt, và ví điện tử MetaMask dùng để ký xác nhận cũng như thực hiện các giao dịch trên Blockchain. Ứng dụng hỗ trợ các chức năng cơ bản như tạo hợp đồng, nạp tiền ký quỹ, nộp sản phẩm, xác nhận hoàn thành và xử lý tranh chấp một cách trực quan.

Thông qua quá trình triển khai và thực hiện đề tài, nhóm có cơ hội vận dụng và củng cố kiến thức liên quan đến công nghệ Blockchain, Smart Contract, Web3 cũng như quy trình phát triển ứng dụng phi tập trung (DApp). Bên cạnh đó, góp phần rèn luyện kỹ năng thiết kế hệ thống, lập trình frontend – backend trong môi trường Web3 và triển khai ứng dụng thực tế trên mạng thử nghiệm Sepolia. Đề tài không chỉ có ý nghĩa học thuật mà còn mang tính ứng dụng cao, tạo nền tảng cho việc nghiên cứu và phát triển các hệ thống giao dịch phi tập trung trong tương lai.

2. Lý do chọn đề tài

Trong bối cảnh kinh tế số phát triển mạnh mẽ và mô hình làm việc tự do (Freelance) ngày càng phổ biến, các giao dịch trực tuyến giữa khách hàng và người cùng cấp dịch vụ diễn ra với tần suất ngày càng cao. Tuy nhiên, cùng với sự gia tăng đó là những thách thức nghiêm trọng về lòng tin và an toàn trong việc thanh toán. Trên thực tế, đã ghi nhận nhiều trường hợp Freelancer không nhận được thù lao sau khi hoàn thành công việc, hoặc ngược lại, Client phải đối mặt với rủi ro khi đã thanh toán nhưng không nhận được sản phẩm đúng như cam kết. Những vấn đề này đặc biệt phổ biến trong những giao dịch

không có cơ chế bảo đảm pháp lý chặt chẽ hoặc không thông qua các nền tảng trung gian uy tín.

Bên cạnh đó, các nền tảng Freelance truyền thống tuy đóng vai trò trung gian trong việc quản lý hợp đồng và thanh toán nhưng vẫn tồn tại nhiều hạn chế như chi phí dịch vụ cao, quy trình xử lý tranh chấp phức tạp, thiếu minh bạch và đặc biệt là sự phụ thuộc lớn vào bên thứ ba. Những bất cập này đã ảnh hưởng đến quyền lợi và mức độ tin cậy của các bên tham gia, từ đó đặt ra nhu cầu cấp thiết về một giải pháp thay thế hiệu quả hơn.

Nhận thức được tính cấp thiết thiết và mức độ nghiêm trọng của các rủi ro trên, nhóm quyết định lựa chọn đề tài *"Xây dựng Escrow DApp FreelanceTrust"*. Đề tài này không chỉ giúp nhóm hiểu rõ hơn các kiến thức đã học về Blockchain, Smart Contract và Web3 để giải quyết một bài toán thực tiễn, từ đó củng cố và nâng cao kỹ năng phát triển ứng dụng phi tập trung. Ngoài ra, việc xây dựng một DApp Escrow là một nhu cầu thực tiễn trong môi trường làm việc tự do, nơi lòng tin và an toàn thanh toàn là những thách thức thường gặp.

Quá trình tự tay triển khai hệ thống từ các thành phần cơ bản như hợp đồng thông minh, giao diện người dùng và cơ chế xử lý tranh chấp không chỉ là một thử thách kỹ thuật mà còn giúp sinh viên nâng cao khả năng phân tích, giải quyết vấn đề thực tiễn và phát triển ứng dụng phi tập trung trong môi trường thực tế. Tóm lại, đề tài không chỉ có tính thực tiễn trong bối cảnh kinh tế số, mà còn mang giá trị học thuật rõ rệt, phù hợp với mục tiêu và nội dung môn học, đồng thời góp phần củng cố năng lực nghiên cứu và kỹ năng thực hành của sinh viên trong lĩnh vực công nghệ Blockchain.

3. Mục tiêu đề tài

Mục tiêu của đề tài là xây dựng một ứng dụng phi tập trung (DApp) Escrow có khả năng giữ và phân phối tiền giữa khách hàng (Client) và người làm việc tự do (Freelancer) một cách tự động, minh bạch và đáng tin cậy trên nền tảng Blockchain Ethereum. Thông qua đề tài này, nhóm mong muốn áp dụng các kiến thức đã học về Blockchain, Smart Contract và cơ chế Escrow để quản lý các giao dịch phi tập trung, đồng thời mở rộng hiểu biết về phát triển ứng dụng Web3 và tích hợp ví điện tử MetaMask phục vụ cho các giao dịch. Hệ thống DApp cần đảm bảo tự động quản lý

trạng thái hợp đồng, thực thi thanh toán khi công việc được xác nhận, xử lý tranh chấp thông qua cơ chế Arbiter, đồng thời bảo toàn tính toàn vẹn và minh bạch của dữ liệu.

Bên cạnh đó, ứng dụng cần được thiết kế với giao diện trực quan, cho phép người dùng thực hiện các thao tác cơ bản như tạo hợp đồng, nạp tiền ký quỹ, nộp sản phẩm và xác nhận thanh toán một cách thuận tiện. Đồng thời, kiến trúc của DApp phải đảm bảo khả năng mở rộng, dễ bảo trì và vận hành ổn định, nhằm hướng tới việc phát triển một sản phẩm hoàn chỉnh, có tính thực tiễn cao và sẵn sàng áp dụng trong môi trường Blockchain thực tế.

Tóm lại, thông qua quá trình thực hiện đề tài, nhóm sẽ không chỉ hoàn thiện Escrow DApp FreelanceTrust mà còn rèn luyện các kỹ năng phân tích, thiết kế và lập trình ứng dụng phi tập trung, đồng thời nắm bắt sâu hơn cách vận hành thực tế của Smart Contract, giao diện Web3 và môi trường Blockchain.

4. Đối tượng và phạm vi nghiên cứu

Đối tượng nghiên cứu của đề tài là quá trình thiết kế và triển khai các thành phần cốt lõi của một DApp Escrow trên nền tảng Blockchain Ethereum. Nghiên cứu tập trung vào việc phân tích và xây dựng Smart Contract để tự động hóa cơ chế thanh toán, cơ chế xử lý tranh chấp thông qua Arbiter, cũng như thiết kế giao diện Web3 giúp người dùng tương tác với ví điện tử MetaMask. Bên cạnh đó, đề tài nghiên cứu cách vận hành và quản lý trạng thái hợp đồng, xử lý các sự kiện phát sinh trong giao dịch, đảm bảo tính minh bạch, bất biến và an toàn của dữ liệu trên Blockchain.

Phạm vi nghiên cứu giới hạn trong việc xây dựng và triển khai DApp trên mạng thử nghiệm Sepolia, nhằm đánh giá hiệu năng, tính bảo mật và khả năng mở rộng của hệ thống. Đề tài không đi sâu vào việc phát triển DApp trên các mạng Blockchain khác, không tích hợp các loại tiền điện tử ngoài Ethereum, và không mở rộng các chức năng phức tạp như quản lý danh sách Freelancer/Client toàn cầu hay kết nối với các nền tảng freelance hiện có. Mục tiêu là tập trung nghiên cứu các yếu tố cốt lõi của cơ chế Escrow, từ phát triển Smart Contract đến giao diện người dùng và xử lý tranh chấp, đảm bảo tính khả thi, phù hợp với thời gian và kiến thức của sinh viên trong khuôn khổ môn học Blockchain.

Với phạm vi và đối tượng nghiên cứu như đã nêu, đề tài tập trung làm rõ quá trình xây dựng và vận hành DApp Escrow trên Ethereum, đồng thời đảm bảo tính khả thi,

phù hợp với khối lượng kiến thức và thời gian thực hiện trong khuôn khổ môn học Blockchain.

5. Phương pháp nghiên cứu

Để thực hiện đề tài *"Xây dựng Escrow DApp FreelanceTrust",* nhóm đã áp dụng tổng hợp nhiều phương pháp nghiên cứu khác nhau nhằm đảm bảo tính khoa học và thực tiễn. Trước tiên, nhóm sử dụng *phương pháp nghiên cứu tài liệu* để thu thập kiến thức liên quan đến Blockchain, Smart Contract, cơ chế Escrow và phát triển ứng dụng phi tập trung (DApp). Nhóm đã tham khảo các tài liệu chính thống từ sách chuyên khảo và hướng dẫn chính thức của Ethereum, tài liệu từ Web3.js, các bài viết kỹ thuật từ các diễn đàn lập trình như Stack Overflow, GitHub, cùng nhiều bài viết chuyên sâu từ cộng đồng mã nguồn mở.

Tiếp theo, nhóm áp dụng *phương pháp thực nghiệm* trong suốt quá trình thiết kế và triển khai DApp. Cụ thể, nhóm tiến hành thử nghiệm các Smart Contract để đánh giá khả năng thực thi cơ chế giữ và giải ngân thanh toán tự động, triển khai cơ chế xử lý tranh chấp thông qua Arbiter và tích hợp các chức năng tương tác với ví MetaMask. Đồng thời, nhóm thực hiện các đoạn mã thử nghiệm nhỏ để kiểm tra quá trình quản lý trạng thái hợp đồng, xử lý các sự kiện phát sinh trong giao dịch và đảm bảo tính bất biến, minh bạch của dữ liệu trên Blockchain.

Sử dụng *phương pháp phân tích – thiết kế – lập trình – kiểm thử* theo hướng tiếp cận tuần tự. Trong giai đoạn thiết kế, hệ thống được chia thành các mô-đun riêng biệt: Smart Contract, giao diện Web3, cơ chế quản lý hợp đồng và xử lý tranh chấp. Sau đó từng phần sẽ được triển khai bằng ngôn ngữ lập trình phù hợp và kiểm thử độc lập để tối ưu hóa trước khi tích hợp vào hệ thống phi tập trung tổng thể, nhằm đảm bảo tính ổn định, hiệu quả và bảo mật của hệ thống.

Cuối cùng, nhóm tiến hành *kiểm thử hệ thống* trên mạng thử nghiệm Sepolia, mô phỏng các tình huống thực tế như: hợp đồng bị hủy giữa chừng, tranh chấp phát sinh, hoặc lỗi thao tác từ người dùng. Qua đó, nhóm ghi nhận các lỗi, phân tích nguyên nhân và khắc phục để nâng cao tính ổn định, độ tin cậy và khả năng vận hành của DApp.

Phần 2: NỘI DUNG

CHƯƠNG 1: CƠ SỞ LÝ THUYẾT NỀN TẢNG

1.1. Tổng quan về Blockchain và nền tảng Ethereum

1.1.1. Khái niệm và đặc điểm của Blockchain

Blockchain là một dạng Công nghệ Sổ cái phân tán (Distributed Ledger Technology - DLT), trong đó thông tin được lưu trữ trong các khối (blocks) và liên kết với nhau theo trình tự thời gian bằng các thuật toán mật mã, tạo thành một chuỗi (chain) dữ liệu không thể phá vỡ. Khác với cơ sở dữ liệu truyền thống vốn được quản lý tập trung và cho phép chỉnh sửa dữ liệu, Blockchain hoạt động theo cơ chế "chỉ ghi thêm" (append-only), nghĩa là một khi dữ liệu đã được xác thực và ghi vào chuỗi, nó không thể bị sửa đổi hoặc xóa bỏ. Nhờ cơ chế này, Blockchain tạo ra một môi trường chia sẻ dữ liệu minh bạch và đáng tin cây mà không cần đến sự kiểm soát của một tổ chức trung tâm.

Một trong những đặc điểm quan trọng nhất của Blockchain là *tính phi tập trung* (decentralization). Thay vì phụ thuộc vào một máy chủ trung tâm để lưu trữ và kiểm soát dữ liệu, Blockchain hoạt động trên nền tảng mạng ngang hàng, nơi mỗi nút đều có quyền lưu trữ bản sao của sổ cái và tham gia vào quá trình xác thực giao dịch. Cách tiếp cận này giúp hệ thống giảm thiểu rủi ro bị tấn công hoặc gián đoạn do lỗi tại một điểm đơn lẻ, đồng thời tăng cường tính minh bạch và độ tin cậy trong quá trình vận hành.

Bên cạnh đó, Blockchain còn nổi bật với *tính bất biến* (immutability) của dữ liệu. Một khi thông tin đã được ghi vào Blockchain và được mạng lưới xác nhận thông qua cơ chế đồng thuận, dữ liệu đó sẽ không thể bị chỉnh sửa hoặc xóa bỏ. Mỗi khối mới đều chứa hàm băm của khối trước, tạo ra mới liên kết chặt chẽ giữa các khối trong chuỗi. Nếu một khối bị thay đổi, toàn bộ chuỗi sau đó sẽ mất tính hợp lệ, khiến hành vi gian lận dễ dàng bị phát hiện. Nhờ vậy, Blockchain đảm bảo tính toàn vẹn và độ tin cậy của dữ liệu trong suốt vòng đời của hệ thống.

1.1.2. Nền tảng Ethereum

1.1.2.1. Khái niệm

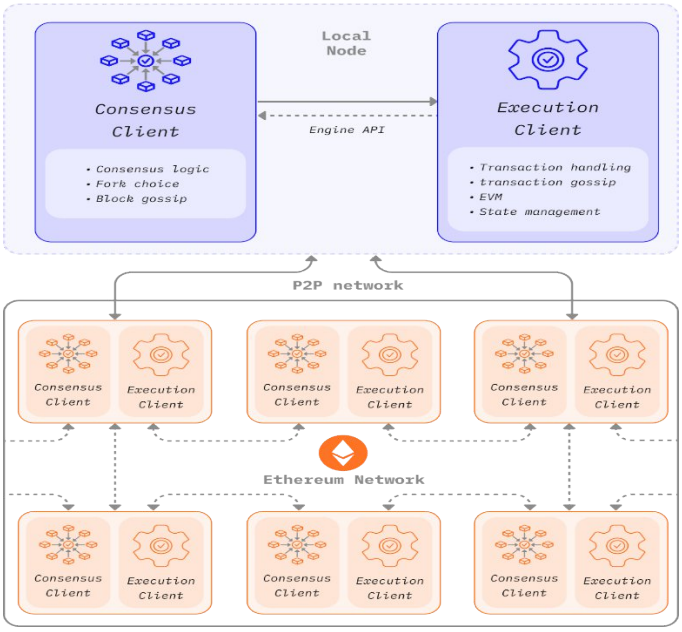
Ethereum là một nền tảng Blockchain phi tập trung, mã nguồn mở, được xây dựng nhằm thiết lập một mạng ngang hàng cho phép thực thi và xác minh các giao dịch cũng như mã ứng dụng một cách an toàn mà không cần đến bên trung gian. Mạng Ethereum

vận hành dựa trên cơ chế đồng thuận, trong đó dữ liệu được ghi nhận công khai, phân tán và có tính bất biến, giúp đảm bảo tính minh bạch và độ tin cậy của hệ thống.

Khác với Bitcoin – vốn được thiết kế chủ yếu để phục vụ mục đích chuyển và lưu trữ giá trị – Ethereum hướng tới vai trò là một nền tảng điện toán phân tán tổng quát, cho phép mở rộng khả năng ứng dụng của công nghệ Blockchain. Ethereum cung cấp môi trường cho các nhà phát triển triển khai nhiều loại ứng dụng khác nhau trên cùng một hạ tầng chung, từ đó hình thành hệ sinh thái ứng dụng phi tập trung phong phú.

Trên nền tảng Ethereum, các lập trình viên có thể xây dựng và triển khai ứng dụng phi tập trung (DApp) và hợp đồng thông minh mà không cần sự can thiệp của bên thứ ba, giúp loại bỏ sự phụ thuộc vào máy chủ trung tâm và tăng cường tính minh bạch trong vận hành. Các giao dịch trên mạng Ethereum được thực hiện thông qua các tài khoản do người dùng tạo và sử dụng Ether (ETH) làm đơn vị thanh toán phí xử lý giao dịch. Nhờ kiến trúc phân tán và cơ chế vận hành minh bạch, Ethereum đóng vai trò nền tảng cốt lõi cho sự phát triển của các hệ thống và ứng dụng Blockchain trong kỷ nguyên Web3.

1.1.2.2. Kiến trúc và cơ chế hoạt động



Ethereum được hình thành từ tập hợp các máy tính tham gia mạng, gọi là các nút (node), hoạt động theo mô hình ngang hàng (peer-to-peer). Mỗi node duy trì một bản sao của sổ cái phân tán, trong đó ghi nhận toàn bộ lịch sử giao dịch và trạng thái của

hệ thống. Nhờ cơ chế này, Ethereum đảm bảo dữ liệu được chia sẻ công khai, minh bạch và không thể bị thay đổi sau khi đã được xác nhận.

Để tham gia vào mạng lưới Ethereum, các node cần cài đặt phần mềm khách (Ethereum client) như Geth hoặc Nethermind. Các client này cho phép node kết nối với mạng, đồng bộ dữ liệu blockchain và tham gia vào quá trình xác thực giao dịch. Bên trong mỗi node, Ethereum vận hành một môi trường thực thi chung gọi là Máy ảo Ethereum (Ethereum Virtual Machine – EVM), đóng vai trò đảm bảo mọi đoạn mã và giao dịch được xử lý thống nhất trên toàn mạng.

Hoạt động của mạng Ethereum dựa trên cơ chế đồng thuận Proof of Stake (PoS). Trong cơ chế này, các node tham gia xác thực, gọi là validator, cần đặt cọc một lượng Ether (ETH) để có quyền đề xuất và xác nhận các khối mới. Các validator được khuyến khích hành xử trung thực thông qua cơ chế thưởng và phạt: nếu xác thực đúng, họ nhận phần thưởng; ngược lại, hành vi gian lận sẽ bị cắt giảm số Ether đã đặt cọc. Cơ chế này giúp mạng lưới đạt được sự đồng thuận mà không cần tiêu tốn nhiều tài nguyên tính toán như cơ chế Proof of Work trước đây.

Mọi giao dịch và hoạt động trên Ethereum đều yêu cầu một khoản phí xử lý gọi là Gas, được thanh toán bằng đồng tiền mã hóa gốc của mạng là Ether. Gas đóng vai trò kiểm soát tài nguyên mạng, ngăn chặn các hành vi lạm dụng và đảm bảo rằng các giao dịch được thực thi một cách công bằng. Khi giao dịch được xác thực và ghi vào khối, dữ liệu sẽ trở nên bất biến và có thể được kiểm tra công khai thông qua các công cụ như Etherscan.

Nhờ sự kết hợp giữa mạng ngang hàng, cơ chế đồng thuận PoS, máy ảo EVM và hệ thống phí Gas, Ethereum vận hành như một nền tảng blockchain phi tập trung, an toàn và minh bạch, tạo nền tảng cho việc phát triển và triển khai các ứng dụng phi tập trung trong môi trường kinh tế số.

1.2. Hợp đồng thông minh (Smart Contract) và ngôn ngữ lập trình Solidity

1.2.1. Tổng quan về Smart Contract

1.2.1.1. Khái niệm

Hợp đồng thông minh (Smart Contract) là một khái niệm mới xuất hiện cùng với sự phát triển của công nghệ Blockchain và hiện nay vẫn chưa có một định nghĩa pháp lý chính thức, thống nhất tại Việt Nam. Trên phương diện lý luận và học thuật, thuật ngữ

này được đề xuất lần đầu bởi nhà khoa học máy tính người Mỹ - Nick Szabo, theo đó hợp đồng thông minh được hiểu là "một tập hợp các lời hứa được thiết lập dưới dạng kỹ thuật số, bao gồm các giao thức cho phép các bên thực hiện những lời hứa đó". Cách tiếp cận này nhấn mạnh vai trò của công nghệ trong việc tự động hóa việc thực hiện hợp đồng, qua đó giảm sự phụ thuộc vào bên trung gian và hạn chế rủi ro vi phạm nghĩa vụ. Bên cạnh đó, Phòng Thương mại Kỹ thuật số Hoa Kỳ (Chamber of Digital Commerce, 2018) định nghĩa hợp đồng thông minh là "mã máy tính, mà khi xảy ra một điều kiện cụ thể hoặc những điều kiện, có thể vận hành tự động theo các chức năng được chỉ định trước. Mã nguồn có thể được lưu trữ và xử lý trên một sổ cái phân tán và sẽ ghi bất kỳ kết quả thay đổi nào vào sổ cái phân tán".

Sự ra đời của nền tảng Ethereum đã góp phần hoàn thiện và mở rộng khái niệm hợp đồng thông minh, nhà đồng sáng lập Vital Buterin đã đưa ra một định nghĩa mới về hợp đồng thông minh: "Hợp đồng thông minh là một chương trình máy tính, trực tiếp kiểm soát một số loại tài sản kỹ thuật số, nó có thể thay thế hợp đồng pháp lý thông thường, khi chuyển tài sản kỹ thuật số thành một chương trình, chương trình sẽ tự động mã hóa thông qua một số điều kiện và xác định tài sản phải được chuyển cho một người hay trả lại cho người kia, hoặc liệu nó phải được hoàn trả ngay lập tức cho người đã gửi hoặc một số kết hợp khác" (Hays, 2018).

Từ các quan điểm trên, có thể thấy nhiều học giả, tổ chức quốc tế và quốc gia đã đưa ra các cách tiếp cận khác nhau nhằm làm rõ bản chất của loại hợp đồng này. Qua đó, có thể khái quát rằng hợp đồng thông minh là một thỏa thuận giữa các bên được thể hiện dưới dạng mã máy tính, được lưu trữ và vận hành trên nền tảng blockchain hoặc sổ cái phân tán, có khả năng tự động thực thi các điều khoản đã được lập trình mà không cần sự can thiệp của bên trung gian.

1.2.1.2. Đặc điểm

Hợp đồng thông minh (Smart Contract) là loại hợp đồng được thiết lập và vận hành trên nền tảng công nghệ blockchain, vì vậy mang những đặc điểm đặc thù xuất phát từ chính cơ chế hoạt động của công nghệ này. Các đặc điểm của hợp đồng thông minh không chỉ thể hiện tính chất kỹ thuật mà còn ảnh hưởng trực tiếp đến phương thức giao kết và thực hiện hợp đồng trong thực tiễn.

Thứ nhất, tính phi tập trung là đặc điểm cơ bản của hợp đồng thông minh. Thông tin và dữ liệu của hợp đồng không được lưu trữ tại một hệ thống trung tâm mà được phân tán và đồng bộ trên nhiều nút mạng trong hệ thống blockchain. Nhờ cơ chế này, hợp đồng thông minh không chịu sự kiểm soát của một chủ thể duy nhất, góp phần bảo đảm tính minh bạch, khách quan và giảm nguy cơ bị can thiệp trái phép vào nội dung hợp đồng.

Thứ hai, hợp đồng thông minh có tính tự động trong việc thực hiện. Các điều khoản của hợp đồng được mã hóa dưới dạng các câu lệnh điều kiện theo logic "If – Then". Khi các điều kiện đã được lập trình sẵn được thỏa mãn, hợp đồng sẽ tự động kích hoạt và thực hiện các nghĩa vụ tương ứng mà không cần sự can thiệp của con người. Cơ chế này giúp bảo đảm việc thực hiện hợp đồng diễn ra đúng theo các thỏa thuận đã được xác lập.

Thứ ba, tính bất biến là một đặc điểm quan trọng của hợp đồng thông minh. Sau khi được triển khai và ghi nhận trên blockchain, nội dung hợp đồng gần như không thể bị sửa đổi hoặc xóa bỏ. Mọi thay đổi, nếu có, chỉ có thể được thực hiện thông qua cơ chế đồng thuận của hệ thống. Đặc điểm này góp phần bảo đảm sự ổn định và độ tin cậy của hợp đồng, đồng thời hạn chế các hành vi gian lận trong quá trình thực hiện.

Thứ tư, hợp đồng thông minh có độ tin cậy và tính bảo mật cao. Các giao dịch và trạng thái thực hiện hợp đồng đều được ghi nhận trên blockchain với dấu thời gian cụ thể, giúp việc kiểm tra, xác thực và truy vết trở nên dễ dàng. Niềm tin của các bên tham gia được đặt vào hệ thống công nghệ và mã nguồn của hợp đồng, thay vì phụ thuộc vào sự giám sát của các chủ thể trung gian.

Thứ năm, hợp đồng thông minh có tính minh bạch trong quá trình vận hành. Các bên tham gia có quyền truy cập và theo dõi quá trình thực hiện hợp đồng theo phạm vi được cho phép. Toàn bộ lịch sử giao dịch được lưu trữ công khai trên blockchain, góp phần nâng cao tính công khai và hạn chế các tranh chấp phát sinh do thiếu thông tin.

1.2.2. Ngôn ngữ lập trình Solidity

Solidity là ngôn ngữ lập trình bậc cao được thiết kế chuyên biệt để phát triển các hợp đồng thông minh (Smart Contract) trên nền tảng Ethereum. Ngôn ngữ này được đề xuất và phát triển bởi Gavin Wood và nhóm Ethereum Foundation, với mục tiêu cung

cấp một công cụ lập trình mạnh mẽ, linh hoạt và an toàn để hiện thực hóa các logic nghiệp vụ phi tập trung.

Về mặt cú pháp, Solidity chịu ảnh hưởng lớn từ các ngôn ngữ lập trình phổ biến như JavaScript, C++ và Python, nhờ đó giúp lập trình viên dễ dàng tiếp cận và làm quen. Solidity hỗ trợ các cấu trúc lập trình cơ bản như biến, hàm, vòng lặp, điều kiện rẽ nhánh, cũng như các kiểu dữ liệu đặc thù phục vụ cho môi trường blockchain như address, mapping, struct và enum. Ngoài ra, Solidity cung cấp các cơ chế kiểm soát truy cập và xử lý giao dịch như modifier, event và error handling, giúp tăng cường tính an toàn, minh bạch và khả năng theo dõi hoạt động của hợp đồng thông minh.

Các chương trình viết bằng Solidity sau khi được biên dịch sẽ tạo ra bytecode và được triển khai lên blockchain Ethereum để thực thi thông qua Máy ảo Ethereum (Ethereum Virtual Machine – EVM). Do đặc tính bất biến của Blockchain, mã nguồn hợp đồng thông minh sau khi triển khai sẽ không thể chỉnh sửa trực tiếp, đòi hỏi quá trình thiết kế, kiểm thử và triển khai phải được thực hiện một cách cẩn trọng nhằm hạn chế rủi ro và lỗi logic. Ngôn ngữ này cũng cho phép quản lý trạng thái hợp đồng một cách rõ ràng thông qua các biến trạng thái được lưu trữ trực tiếp trên blockchain. Nhờ những đặc điểm trên, Solidity đã trở thành ngôn ngữ lập trình chủ đạo trong hệ sinh thái Ethereum, được sử dụng rộng rãi để phát triển các ứng dụng phi tập trung (DApp), các giao thức tài chính phi tập trung (DeFi), NFT và các mô hình hợp đồng tự động khác.

1.3. Cơ chế Escrow trên Blockchain

1.3.1. Khái niệm và đặc điểm

Cơ chế Escrow (ký quỹ) là một phương thức quản lý và bảo đảm giao dịch, trong đó một bên thứ ba trung lập (Escrow Agent) tạm thời giữ tài sản, tiền bạc hoặc các tài sản kỹ thuật số cho đến khi tất cả các điều kiện thỏa thuận giữa các bên được đáp ứng đầy đủ. Trong ngữ cảnh truyền thống, cơ chế này giúp giảm rủi ro gian lận, đảm bảo rằng người bán chỉ nhận thanh toán khi người mua nhận được sản phẩm/dịch vụ, và ngược lại. Escrow thường được áp dụng trong các giao dịch bất động sản, thương mại hoặc tài chính, nơi việc xây dựng lòng tin giữa các bên là yếu tố quan trọng.

Trên nền tảng Blockchain, cơ chế Escrow được nâng cấp thành dạng Escrow phi tập trung (Decentralized Escrow), chủ yếu thông qua Smart Contract – các hợp đồng thông minh tự động thực thi theo các điều kiện đã lập trình. Smart Contract thay thế vai trò

của bên thứ ba truyền thống, hoạt động hoàn toàn dựa trên mã code bất biến. Khi các điều kiện được lập trình sẵn thỏa mãn, tài sản sẽ tự động được giải phóng. Một số đặc điểm nổi bật của Escrow trên Blockchain bao gồm:

- *Phi tập trung và không cần tin tưởng (Trustless):* Loại bỏ sự phụ thuộc vào các bên thứ ba trung gian, giảm rủi ro gian lận hoặc thiên vị.
- *Tự động hóa:* Smart Contract thực hiện việc giữ và giải ngân tài sản một cách chính xác, nhanh chóng, không thể can thiệp thủ công.
- *Minh bạch:* Mọi giao dịch và điều kiện đều được ghi nhận trên blockchain, cho phép các bên tham gia kiểm tra công khai.
- *Bảo mật và bất biến:* Dữ liệu không thể thay đổi hoặc xóa bỏ sau khi xác nhận, bảo vệ quyền lợi của các bên.
- *Tiết kiệm chi phí và thời gian:* Loại bỏ phí trung gian và hỗ trợ giao dịch xuyên biên giới nhanh chóng.

1.3.2. Cơ chế hoạt động

Về mặt kỹ thuật, cơ chế Escrow trên Blockchain được triển khai thông qua hợp đồng thông minh (Smart Contract), hoạt động như một chương trình máy tính chứa các câu lệnh điều kiện. Khi các điều kiện đã được lập trình sẵn được thỏa mãn, hợp đồng sẽ tự động thực hiện các hành vi như giữ tiền, giải ngân tài sản hoặc cập nhật trạng thái giao dịch mà không cần bên thứ ba can thiệp. Quá trình vận hành cơ chế Escrow trên Blockchain thường trải qua bốn giai đoạn cơ bản:

Thứ nhất, giai đoạn tạo lập hợp đồng Escrow.

Các bên tham gia giao dịch (Client và Freelancer) thương lượng và thống nhất về điều khoản giao dịch, quyền và nghĩa vụ của mỗi bên. Nội dung thỏa thuận ban đầu

được thể hiện bằng ngôn ngữ tự nhiên và sau đó được lập trình thành mã máy tính phù hợp với nền tảng Blockchain, thường sử dụng ngôn ngữ Solidity trên Ethereum.

Thứ hai, giai đoạn triển khai hợp đồng Escrow.

Sau khi được các bên xác nhận, hợp đồng sẽ được triển khai lên blockchain. Mã hợp đồng được phân phối và sao chép trên nhiều nút (node) trong mạng lưới, đảm bảo tính minh bạch, phi tập trung và bất biến. Đồng thời, khoản tiền ký quỹ hoặc tài sản liên quan sẽ bị khóa trong hợp đồng cho đến khi các điều kiện được đáp ứng.

Thứ ba, giai đoạn thực thi hợp đồng Escrow.

Khi các điều kiện được lập trình trước xảy ra (ví dụ: Freelancer nộp sản phẩm, Client xác nhận hoàn thành), hợp đồng Escrow sẽ tự động kích hoạt. Tiền ký quỹ hoặc tài sản được giải ngân cho bên cung cấp dịch vụ theo các điều khoản đã thỏa thuận. Trong trường hợp xảy ra tranh chấp, các arbiter hoặc cơ chế xác thực bên ngoài (oracle) sẽ tham gia để quyết định phân phối tài sản.

Thứ tư, giai đoạn hoàn tất giao dịch.

Sau khi các điều kiện thực thi đầy đủ, trạng thái của hợp đồng và các bên tham gia được cập nhật. Tài sản kỹ thuật số được giải phóng theo thỏa thuận, đồng thời toàn bộ lịch sử giao dịch được lưu trữ vĩnh viễn trên blockchain, đảm bảo khả năng kiểm tra, truy xuất thông tin và tính minh bạch.

Nhờ cơ chế này, Escrow trên Blockchain không chỉ giảm thiểu rủi ro gian lận mà còn loại bỏ sự phụ thuộc vào trung gian, tiết kiệm chi phí, thời gian và tạo sự tin cậy trong các giao dịch phi tập trung.

1.4. Ứng dụng phi tập trung (Decentralized Application – DApp)

1.4.1. Khái niệm

Ứng dụng phi tập trung (Decentralized Application – DApp) là một hệ thống phần mềm hoạt động trên mạng lưới máy tính phân tán, thường là Blockchain, nhằm loại bỏ sự kiểm soát của một tổ chức trung tâm duy nhất (Zheng et al., 2017). Khác biệt căn bản so với các ứng dụng truyền thống là DApp tận dụng các đặc tính cốt lõi của công nghệ sổ cái phân tán để đảm bảo tính minh bạch, bất biến và tự động hóa trong các quy trình nghiệp vụ.

Về mặt kiến trúc, DApp tích hợp ba thành phần chính: Smart Contract (Hợp đồng thông minh) hoạt động như logic nghiệp vụ cốt lõi và được triển khai trên Blockchain

(ví dụ: Ethereum); một giao diện người dùng phía máy khách (Frontend) thường được xây dựng trên công nghệ Web3, cho phép người dùng tương tác với Smart Contract; và mạng lưới Blockchain phía máy chủ (Backend) thực hiện việc lưu trữ dữ liệu và xử lý giao dịch. Trong bối cảnh dự án FreelanceTrust, DApp đóng vai trò là một người giám sát tự động, nơi khoản ký quỹ được khóa trong Smart Contract ngay khi hợp đồng được khởi tạo, đảm bảo rằng việc giải ngân chỉ xảy ra khi các điều kiện đã được thỏa mãn (Client xác nhận) hoặc theo phán quyết của bên thứ ba (Arbiter).

Do hoạt động trên cơ chế đồng thuận của Blockchain, dữ liệu của DApp mang tính bất biến (Immutability), nghĩa là mọi giao dịch được ghi lại sẽ không thể bị thay đổi hay xóa bỏ, qua đó củng cố độ tin cậy và tính toàn vẹn của dữ liệu trong suốt vòng đời của hệ thống. Bên cạnh đó, tính phi tập trung (Decentralization) cũng là nền tảng để DApp Escrow FreelanceTrust giải quyết các vấn đề về lòng tin và rủi ro thanh toán trong mô hình làm việc tự do (Freelance), tạo ra một môi trường giao dịch công bằng, minh bạch và an toàn hơn so với các nền tảng trung gian truyền thống.

1.4.2. Cấu trúc cơ bản

Cấu trúc cơ bản của một ứng dụng phi tập trung (DApp) được phân chia thành ba tầng kiến trúc chính, hoạt động phối hợp để tạo nên tính phi tập trung và khả năng tự động hóa của hệ thống (Zheng et al., 2017). Mối liên kết giữa các tầng này là yếu tố then chốt giúp DApp khác biệt so với các ứng dụng tập trung truyền thống.

- *Tầng Hợp đồng thông minh (Smart Contract Layer):* Đây là lõi nghiệp vụ của DApp, được viết bằng các ngôn ngữ lập trình như Solidity và triển khai trên nền tảng Blockchain, cụ thể trong dự án FreelanceTrust là mạng Ethereum. Hợp đồng thông minh lưu trữ logic điều hành toàn bộ quy trình, chẳng hạn như cơ chế Escrow giữ tiền ký quỹ và tự động giải ngân khi có xác nhận, hoặc xử lý các yêu cầu tranh chấp thông qua cơ chế Arbiter. Tầng này đảm bảo tính bất biến (immutability) và minh bạch của dữ liệu, vì mọi quy tắc và trạng thái hợp đồng đều được ghi vĩnh viễn trên chuỗi khối, không một bên nào có thể đơn phương thay đổi.
- *Tầng Giao diện người dùng (Frontend/Client Layer):* Tầng này đóng vai trò là giao diện tương tác giữa người dùng (Client, Freelancer) và Smart Contract. Thường được xây dựng bằng công nghệ Web3, giao diện này cho phép người

dùng thực hiện các thao tác như tạo hợp đồng, nạp tiền ký quỹ hay xác nhận hoàn thành công việc một cách trực quan. Để kết nối và thực hiện các giao dịch (ký xác nhận, chuyển tiền) trên mạng Blockchain, DApp phải tích hợp ví điện tử như MetaMask, hoạt động như một cầu nối giữa trình duyệt và mạng lưới phi tập trung.

- *Tầng Mạng lưới Blockchain (Backend/Protocol Layer):* Tầng dưới cùng là mạng lưới Blockchain nền tảng (ví dụ: Ethereum) hoạt động như một sổ cái phân tán (DLT) và là máy chủ phi tập trung cho toàn bộ ứng dụng. Tầng này chịu trách nhiệm lưu trữ dữ liệu giao dịch, quản lý trạng thái của các hợp đồng thông minh và sử dụng các cơ chế đồng thuận để xác thực các giao dịch một cách an toàn và chống lại sự can thiệp từ bên ngoài.

Sự phân tách và kết nối chặt chẽ giữa ba thành phần này – logic cốt lõi tự động hóa bởi Smart Contract, khả năng tiếp cận qua giao diện Web3, và tính bảo mật được đảm bảo bởi mạng lưới Blockchain – tạo nên một kiến trúc linh hoạt, đáng tin cậy và hoàn toàn phi tập trung cho Escrow DApp FreelanceTrust.

1.5. Công nghệ Web3 và thư viện Ether.js

1.5.1. Công nghệ Web3

1.5.1.1. Khái niệm

Sự tiến hóa của World Wide Web thường được phân chia thành các thế hệ dựa trên năng lực tương tác và mô hình dữ liệu. Nếu Web 1.0 là kỷ nguyên của nội dung tĩnh (Read-only) và Web 2.0 đánh dấu sự bùng nổ của mạng xã hội cùng các nền tảng tương tác (Read-Write), thì Web3 đại diện cho một tầm nhìn tái cấu trúc Internet, chuyển dịch từ mô hình "lấy tổ chức làm trung tâm" (organization-centric) sang "lấy người dùng làm trung tâm" (user-centric) (Yaga & Mell, 2025).

Khác với những hiểu lầm phổ biến đánh đồng Web3 với Web 3, Web3 không tập trung vào việc làm cho dữ liệu trở nên dễ hiểu với máy móc, mà tập trung vào quyền sở hữu và tính phi tập trung (Yaga & Mell, 2025). Về mặt kỹ thuật, Web3 không loại bỏ các giao thức Internet hiện hành (như TCP/IP, TLS) mà tích hợp chúng với các công nghệ mới như Blockchain, tiền mã hóa (cryptocurrency), và danh tính phi tập trung (Decentralized Identity).

Theo Viện Tiêu chuẩn và Công nghệ Quốc gia Hoa Kỳ (NIST), đặc trưng cốt lõi của Web3 nằm ở việc trao quyền kiểm soát dữ liệu lại cho người dùng cuối. Thay vì dữ liệu bị lưu trữ và kiểm soát bởi các máy chủ tập trung của các "gã khổng lồ" công nghệ – nơi người dùng có nguy cơ bị khóa tài khoản hoặc mất quyền truy cập vào tài sản số của chính mình – Web3 sử dụng sổ cái phân tán để đảm bảo tính minh bạch và bất biến (Yaga & Mell, 2025).

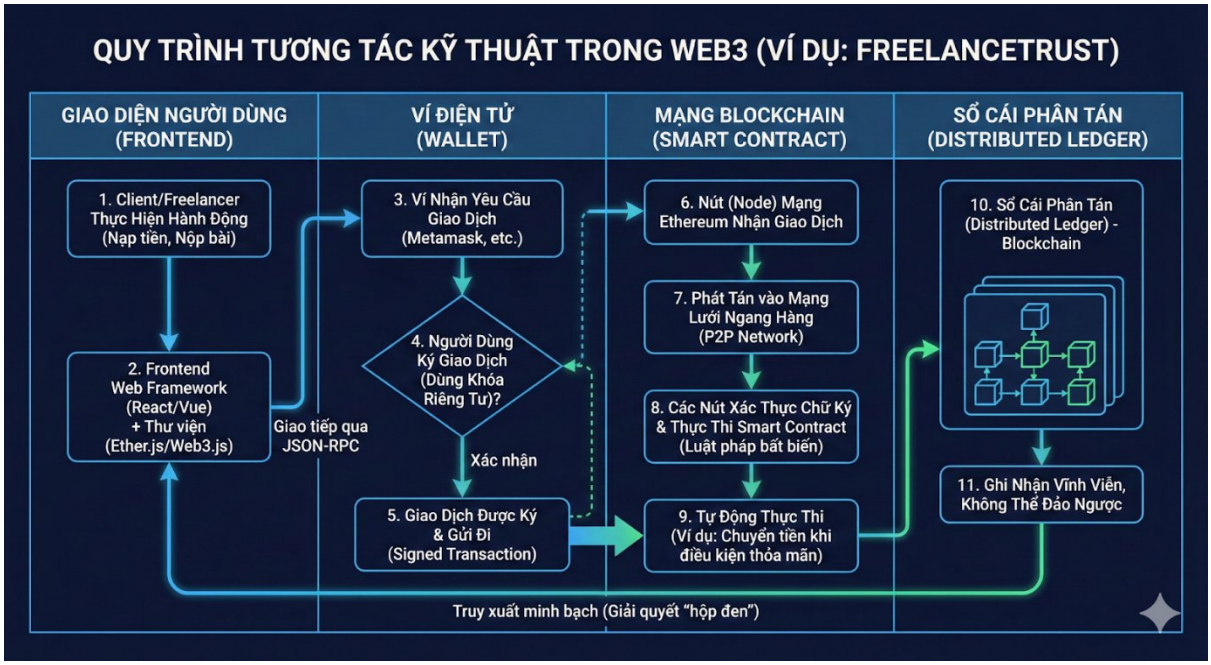
Tóm lại, Web3 trong đề tài này được định nghĩa là môi trường kỹ thuật cho phép xây dựng các ứng dụng phi tập trung, nơi giao diện người dùng tương tác trực tiếp với logic nghiệp vụ trên Blockchain thông qua ví điện tử, tạo ra một hệ sinh thái giao dịch minh bạch, an toàn và trao quyền tự chủ tối đa cho các bên tham gia

1.5.1.2. Nguyên lý hoạt động của Web3

Khác với kiến trúc Client-Server truyền thống của Web 2.0, nơi trình duyệt gửi yêu cầu đến một máy chủ trung tâm để truy xuất và thao tác dữ liệu, nguyên lý hoạt động của Web3 dựa trên sự tương tác trực tiếp giữa người dùng và mạng lưới Blockchain phi tập trung. Trong kiến trúc này, không có một cơ sở dữ liệu duy nhất nào nắm giữ thông tin; thay vào đó, trạng thái của ứng dụng được duy trì và đồng bộ hóa trên toàn bộ mạng lưới các nút (nodes).

Cơ chế vận hành cốt lõi của Web3 xoay quanh việc định danh phi tập trung và thực thi giao dịch thông qua chữ ký số. Theo Yaga và Mell (2025), đây là sự chuyển dịch mô hình từ “xác thực dựa trên tổ chức” sang “xác thực dựa trên sở hữu”. nơi người dùng nắm quyền kiểm soát hoàn toàn danh tính và tài sản số của mình mà không phụ thuộc vào bên thứ ba.

Quá trình tương tác trong Web3 diễn ra theo quy trình kỹ thuật như sau:



Đầu tiên, giao diện người dùng (Frontend) - được viết bằng các framework web tiêu chuẩn - sẽ đóng vai trò là cổng kết nối. Tuy nhiên, thay vì gọi API đến máy chủ backend truyền thống, Frontend sẽ sử dụng các thư viện chuyên dụng (Ether.js hay Web.js) để giao tiếp với một nút (Node) trong mạng Ethereum thông qua giao thức JSON-RPC.

Khi một hành động thay đổi dữ liệu được thực hiện, chẳng hạn như Client nạp tiền vào hợp đồng Escrow, ứng dụng sẽ yêu cầu cầu ví điện tử ký vào một giao dịch (Transaction) bằng khóa riêng tư của người dùng. Giao dịch này sau đó được phát tán (broadcast) vào mạng lưới ngang hàng. Tại đây, các nút trong mạng sẽ xác thực chữ ký và thực thi mã lệnh trong Smart Contract. Kết quả của quá trình này là sự thay đổi trạng thái của sổ cái phân tán (Distributed Ledger), được ghi nhận vĩnh viễn và không thể đảo ngược (Zheng et al., 2017).

Đối với dự án FreelanceTrust, nguyên lý này đảm bảo tính "Trustless" (không cần niềm tin). Mã nguồn của Smart Contract đóng vai trò là luật pháp bất biến; khi các điều kiện lập trình được thỏa mãn, mạng lưới sẽ tự động thực thi việc chuyển tiền mà không một các nhân hay tổ chức nào có thể can thiệp hay ngăn chặn. Dữ liệu về hợp đồng và trạng thái thanh toán được lưu trữ công khai trên Blockchain, cho phép truy xuất minh bạch bất cứ lúc nào, giải quyết vấn đề "hộp đen" dữ liệu thường thất ở các nền tảng Web 2.0.

1.5.2. Thư viện Ether.js

Để hiện thực hóa cơ chế tương tác trực tiếp giữa giao diện người dùng và mạng lưới Blockchain như đã trình bày ở phần nguyên lý hoạt động Web3, dự án *FreelanceTrust* sử dụng thư viện Ether.js. Đây là một thư viện JavaScript mã nguồn mở, gọn nhẹ và bảo mật, đóng vai trò là lớp phần mềm trung gian (middleware) cho phép ứng dụng phía máy khách (Client-side) giao tiếp với chuỗi khối Ethereum thông qua giao thức JSON-RPC (Moore, 2024).

Trong kiến trúc của DApp *FreelanceTrust*, Ether.js không chỉ đơn thuần là công cụ gửi yêu cầu, mà còn chịu trách nhiệm quản lý các trạng thái cục bộ của ví, định dạng dữ liệu và xử lý các sự kiện từ Blockchain. Thư viện này hoạt động dựa trên ba khái niệm trừu tượng cốt lõi, tương ứng với các chức năng nghiệp vụ của hệ thống:

- *Provider (Nhà cung cấp):* Đây là thành phần đảm nhiệm việc kết nối "chỉ-đọc" (read-only) tới mạng lưới Ethereum. Provider đóng vai trò như một lăng kính giúp ứng dụng truy xuất dữ liệu trạng thái của sổ cái phân tán, chẳng hạn như kiểm tra số dư tài khoản, đọng trạng thái hiện tại của hợp đồng Escrow hoặc truy vấn lịch sử giao dịch mà không làm thay đổi dữ liệu trên chuỗi.
- *Signer (Người ký):* Để thực hiện các thao tác "ghi" (write) làm thay đổi trạng thái hệ thống (như khởi tạo hợp đồng, nạp tiền ký quỹ), ứng dụng cần quyền truy cập vào khóa riêng tư (Private Key) của người dùng. Ether.js cung cấp lớp đối tượng Signer để tương tác an toàn với ví điện tử (MetaMask). Theo cơ chế bảo mật của Web3, Signer không lưu trữ khóa riêng tư trực tiếp mà gửi yêu cầu ký giao dịch tới ví, đảm bảo rằng quyền kiểm soát tài sản luôn nằm trong tay người dùng (Yaga & Mell, 2025).
- *Contract (Hợp đồng):* Đây là cầu nối lập trình giữa mã nguồn Frontend và Smart Contract đã triển khai trên Blockchain. Ether.js sử dụng tệp ABI (Application Binary Interface) - bản mô tả cấu trúc của Smart Contract - để chuyển đổi các hàm gọi từ JavaScript sang mã bytecode mà máy ảo Ethereum (EVM) có thể hiểu được và ngược lại, giải mã các dữ liệu trả về từ Blockchain thành định dạng mà giao diện người dùng có thể hiển thị.

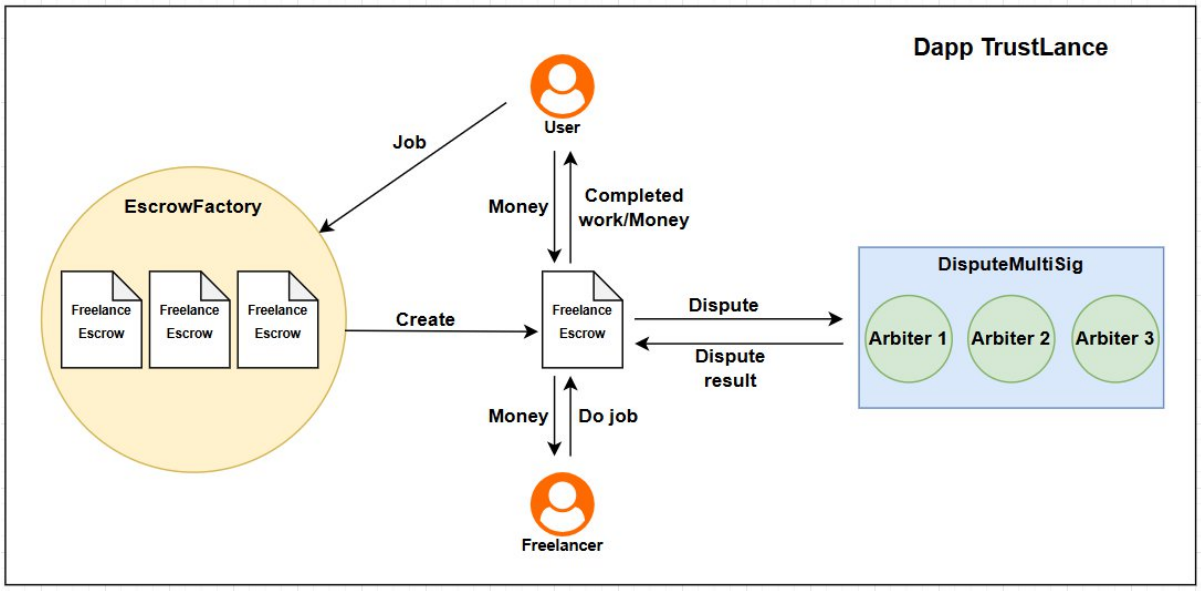
Việc lựa chọn Ether.js thay vì các thư viện khác (như Web3.js) trong dự án này xuất phát từ tính ổn định, khả năng định kiểu dữ liệu mạnh mẽ (hỗ trợ tốt cho TypeScript) và cơ chế xử lý sự kiện (Event listening) hiệu quả. Trong bối cảnh của *FreelanceTrust*,

tính năng lắng nghe sự kiện của Ether.js là cực kỳ quan trọng; nó cho phép giao diện người dùng tự động cập nhật trạng thái ngay khi một sự kiện quan trọng (ví dụ: PaymentReleased hoặc DisputeInitiated) được xác nhận trên Blockchain, mang lại trải nghiệm người dùng mượt mà và minh bạch theo thời gian thực.

CHƯƠNG 2: THIẾT KẾ HỆ THỐNG

2.1. Kiến trúc về hệ thống

Hệ thống Escrow cho giao dịch freelance được thiết kế theo mô hình phân tách rõ ràng vai trò và trách nhiệm của từng Smart Contract cũng như các tác nhân tham gia. Cách tiếp cận này giúp tăng khả năng mở rộng, dễ quản lý và đảm bảo tính minh bạch trong toàn bộ vòng đời của một giao dịch freelance. Hệ thống gồm 3 thành phần chính: EscrowFactory, FreelanceEscrow, DisputeMultiSig.



1/ EscrowFactory

EscrowFactory đóng vai trò là hợp đồng nhà máy (factory contract), chịu trách nhiệm khởi tạo và quản lý các hợp đồng escrow cho từng giao dịch freelance riêng biệt. Thay vì triển khai một hợp đồng duy nhất cho toàn bộ hệ thống, mô hình factory cho phép mỗi giao dịch được cô lập trong một hợp đồng độc lập, từ đó giảm thiểu rủi ro và đơn giản hóa việc quản lý trạng thái.

Các chức năng chính của EscrowFactory bao gồm:

- Tạo mới hợp đồng FreelanceEscrow khi client khởi tạo một yêu cầu giao dịch freelance.
- Lưu trữ và quản lý danh sách các hợp đồng Escrow đã được triển khai, phục vụ cho việc truy xuất và theo dõi.

Cung cấp các hàm truy vấn nhằm lấy thông tin các Escrow theo tiêu chí như client, freelancer hoặc trạng thái giao dịch, hỗ trợ cho giao diện người dùng và các thành phần ngoài chuỗi (off-chain).

2/ FreelanceEscrow

FreelanceEscrow đại diện cho một giao dịch freelance cụ thể giữa client và freelancer. Đây là hợp đồng trung tâm, chịu trách nhiệm quản lý dòng tiền ký quỹ và trạng thái công việc trong suốt vòng đời của giao dịch.

Các chức năng chính của FreelanceEscrow bao gồm:

- Nhận và giữ tiền ký quỹ từ client ngay khi giao dịch được khởi tạo.
- Quản lý trạng thái công việc theo từng giai đoạn, chẳng hạn như: chờ thực hiện, đang thực hiện, đã hoàn thành hoặc đang tranh chấp.
- Cho phép freelancer xác nhận hoàn thành công việc và client xác nhận kết quả bàn giao.
- Thực hiện giải ngân tiền ký quỹ cho freelancer khi công việc được xác nhận hoàn thành, hoặc hoàn trả cho client trong trường hợp hủy hoặc thất bại hợp lệ.
- Kích hoạt và liên kết với hợp đồng DisputeMultiSig khi phát sinh tranh chấp giữa các bên.

3/ DisputeMultiSig

DisputeMultiSig là hợp đồng chịu trách nhiệm xử lý tranh chấp thông qua cơ chế đa chữ ký (multi-signature), nhằm đảm bảo tính công bằng và minh bạch trong quá trình phân xử.

Các chức năng chính của DisputeMultiSig bao gồm:

- Được kích hoạt bởi hợp đồng FreelanceEscrow khi một trong hai bên yêu cầu tranh chấp.
- Quản lý danh sách các arbiter (trọng tài) có quyền tham gia bỏ phiếu.
- Thu thập và tổng hợp phiếu bầu từ các arbiter để đưa ra quyết định cuối cùng về việc phân bổ tiền ký quỹ.
- Đảm bảo rằng kết quả tranh chấp không phụ thuộc vào một cá nhân đơn lẻ mà dựa trên sự đồng thuận của nhiều trọng tài độc lập.

2.2. Các tác nhân tham gia hệ thống

Bên cạnh các Smart Contract chịu trách nhiệm thực thi tự động logic nghiệp vụ, hệ thống còn bao gồm các tác nhân tham gia với vai trò và quyền hạn được xác định rõ ràng. Các tác nhân này tương tác với nhau thông qua các hợp đồng thông minh, trong đó mọi hành vi đều được ràng buộc bởi các điều kiện đã được mã hóa sẵn, nhằm đảm bảo tính minh bạch, khả năng kiểm chứng và giảm thiểu sự phụ thuộc vào niềm tin giữa các bên.

Client là bên khởi tạo giao dịch freelance và đóng vai trò mở đầu cho toàn bộ vòng đời của hợp đồng escrow. Client chịu trách nhiệm xác định các điều khoản công việc, giá trị thanh toán và thời hạn thực hiện trước khi yêu cầu hệ thống tạo mới hợp đồng FreelanceEscrow tương ứng. Thông qua việc nạp tiền ký quỹ vào hợp đồng thông minh, client thể hiện cam kết tài chính đối với Freelancer, đồng thời đảm bảo rằng khoản thanh toán sẽ được giải ngân tự động khi các điều kiện đã thỏa thuận được đáp ứng. Trong quá trình thực hiện, client có quyền theo dõi trạng thái công việc, xác nhận hoàn thành hoặc kích hoạt cơ chế tranh chấp khi phát sinh bất đồng, qua đó bảo vệ quyền lợi của mình thông qua các quy tắc được thực thi trên blockchain.

Freelancer là bên cung cấp dịch vụ và trực tiếp thực hiện công việc theo thỏa thuận đã được thiết lập trong hợp đồng Escrow. Sau khi chấp nhận giao dịch, Freelancer tiến hành thực hiện nhiệm vụ với kỳ vọng rằng khoản thanh toán đã được ký quỹ sẽ được bảo toàn cho đến khi công việc hoàn tất. Khi hoàn thành nhiệm vụ, freelancer có thể gửi yêu cầu xác nhận lên Smart Contract, từ đó kích hoạt quy trình đánh giá và phê duyệt từ phía client. Trong trường hợp client không phản hồi hoặc từ chối xác nhận một cách không hợp lý, freelancer có quyền yêu cầu mở tranh chấp để chuyển việc phân xử sang cơ chế trung gian. Mô hình escrow giúp freelancer giảm thiểu rủi ro không được thanh toán, đồng thời tạo ra môi trường làm việc công bằng và minh bạch hơn.

Arbiter là các bên thứ ba trung lập, chỉ tham gia vào hệ thống khi phát sinh tranh chấp giữa client và freelancer. Các arbiter không trực tiếp tham gia vào giao dịch ban đầu mà chỉ đóng vai trò phân xử khi hai bên không đạt được sự đồng thuận. Thông qua hợp đồng DisputeMultiSig, arbiter thực hiện việc đánh giá tranh chấp và bỏ phiếu một cách độc lập để đưa ra quyết định cuối cùng về việc phân bổ tiền ký quỹ. Cơ chế đa chữ ký đảm bảo rằng kết quả tranh chấp không phụ thuộc vào ý kiến của một cá nhân duy

nhất mà dựa trên sự đồng thuận của nhiều trọng tài, từ đó nâng cao tính khách quan, minh bạch và công bằng của toàn bộ hệ thống.

Nhìn chung, mối quan hệ giữa Client, Freelancer và Arbiter được điều phối hoàn toàn thông qua các Smart Contract, trong đó quyền hạn và nghĩa vụ của mỗi tác nhân được xác định rõ ràng và không thể thay đổi sau khi triển khai. Cách tiếp cận này giúp hạn chế các hành vi gian lận, giảm thiểu tranh chấp ngoài chuỗi và tăng cường mức độ tin cậy cho các giao dịch freelance trong môi trường phi tập trung.

CHƯƠNG 3: TRIỂN KHAI VÀ THỰC NGHIỆM

2.

3.

3.1. Xây dựng Smart Contract

3.1.1. Cấu trúc và các thành phần chính của Smart Contract

Hệ thống TrustLance được thiết kế dựa trên triết lý phi tập trung, tận dụng sức mạnh của hợp đồng thông minh (Smart Contract) để đảm bảo tính minh bạch và tự động hóa trong các giao dịch dịch vụ tự do. Hệ thống sử dụng ngôn ngữ lập trình Solidity phiên bản 0.8.20 và triển khai trên nền tảng Ethereum Virtual Machine (EVM). Kiến trúc Smart Contract của dự án được tổ chức theo mô hình module hóa với ba thành phần chính phối hợp chặt chẽ, đảm bảo tính bảo mật và khả năng quản lý dòng tiền tối ưu.

1/ FreelanceEscrow Contract – Quản trị giao dịch

Đây là thành phần cốt lõi của hệ thống, đóng vai trò là một thực thể trung gian (escrow) quản lý tài sản và điều phối logic nghiệp vụ giữa người thuê (Client) và người thực hiện (Freelancer).

- *Quản lý trạng thái (State Management):* Hợp đồng sử dụng kiểu dữ liệu Enum Status để định nghĩa vòng đời của một dự án, từ giai đoạn khởi tạo (Created) đến khi hoàn tất thanh toán (Released) hoặc hoàn tiền (Refunded).

```
enum Status {
    Created,    // 0 - Vừa tạo, chờ freelancer nhận việc
    Accepted,   // 1 - Freelancer đã nhận việc
    Submitted,  // 2 - Freelancer đã nộp kết quả
    Disputed,   // 3 - Đang trong tranh chấp
    Released,   // 4 - Đã thanh toán cho freelancer
    Refunded    // 5 - Đã hoàn tiền cho client
}
```

Enum Status - Các trạng thái của hợp đồng

- *Biến trạng thái:* Các thông tin quan trọng như địa chỉ ví của hai bên, số tiền ETH được khóa, thời hạn công việc (deadline) và địa chỉ trọng tài (arbiter) được lưu trữ trực tiếp trên chuỗi (on-chain).

Biến	Kiểu dữ liệu	Mô tả
`factory`	`address`	Địa chỉ của contract Factory đã tạo escrow này
`client`	`address payable`	Địa chỉ ví của người thuê
`freelancer`	`address payable`	Địa chỉ ví của người làm việc
`amount`	`uint256`	Số tiền ETH được khóa trong hợp đồng
`deadline`	`uint256`	Thời hạn hoàn thành công việc (Unix timestamp)
`arbiter`	`address`	Địa chỉ contract DisputeMultiSig
`status`	`Status`	Trạng thái hiện tại của hợp đồng

Các biến trạng thái (State Variables) trong FreelanceEscrow

- *Kiểm soát truy cập (Access Control):* Hệ thống áp dụng các modifiers nghiêm ngặt (onlyClient, onlyFreelancer, onlyArbiter) để ngăn chặn các truy cập trái phép vào các hàm thay đổi trạng thái nhạy cảm.

```
modifier onlyFactory()     // Chỉ Factory contract có thể gọi
modifier onlyClient()      // Chỉ người thuê có thể gọi
modifier onlyFreelancer()  // Chỉ người làm việc có thể gọi
modifier onlyArbiter()     // Chỉ contract trọng tài có thể gọi
```

Các Modifiers kiểm soát quyền truy cập

2/ EscrowFactory Contract – Mô hình quản lý tập trung

Contract EscrowFactory đóng vai trò là nhà máy sản xuất các hợp đồng Escrow, áp dụng Factory Pattern trong thiết kế phần mềm.

- *Tính mở rộng:* Cho phép hệ thống tạo ra hàng loạt các hợp đồng trung gian một cách tự động và chuẩn hóa.
- *Truy xuất dữ liệu:* Lưu trữ danh sách toàn bộ các công việc (jobs) và ánh xạ (mapping) để theo dõi các dự án của từng Client cụ thể, giúp việc truy vấn từ Frontend trở nên hiệu quả hơn.

Biến	Kiểu dữ liệu	Mô tả
`arbiter`	`address`	Địa chỉ contract DisputeMultiSig
`jobs`	`address[]`	Mảng lưu trữ tất cả địa chỉ Escrow đã tạo
`jobsByClient`	`mapping(address => address[])`	Ánh xạ từ địa chỉ client đến danh sách jobs

```
event JobCreated(
    address indexed escrow,   // Địa chỉ escrow vừa tạo
    address indexed client,   // Địa chỉ người thuê
    uint256 amount,           // Số tiền khóa
    uint256 deadline,         // Thời hạn
    uint256 timestamp         // Thời điểm tạo
);
```

Các biến trạng thái và event trong EscrowFactory

3/ DisputeMultiSig Contract - Cơ chế giải quyết tranh chấp

Để đảm bảo tính công bằng, hệ thống triển khai cơ chế Multi-Signature Voting (Bỏ phiếu đa chữ ký) cho việc xử lý khiếu nại.

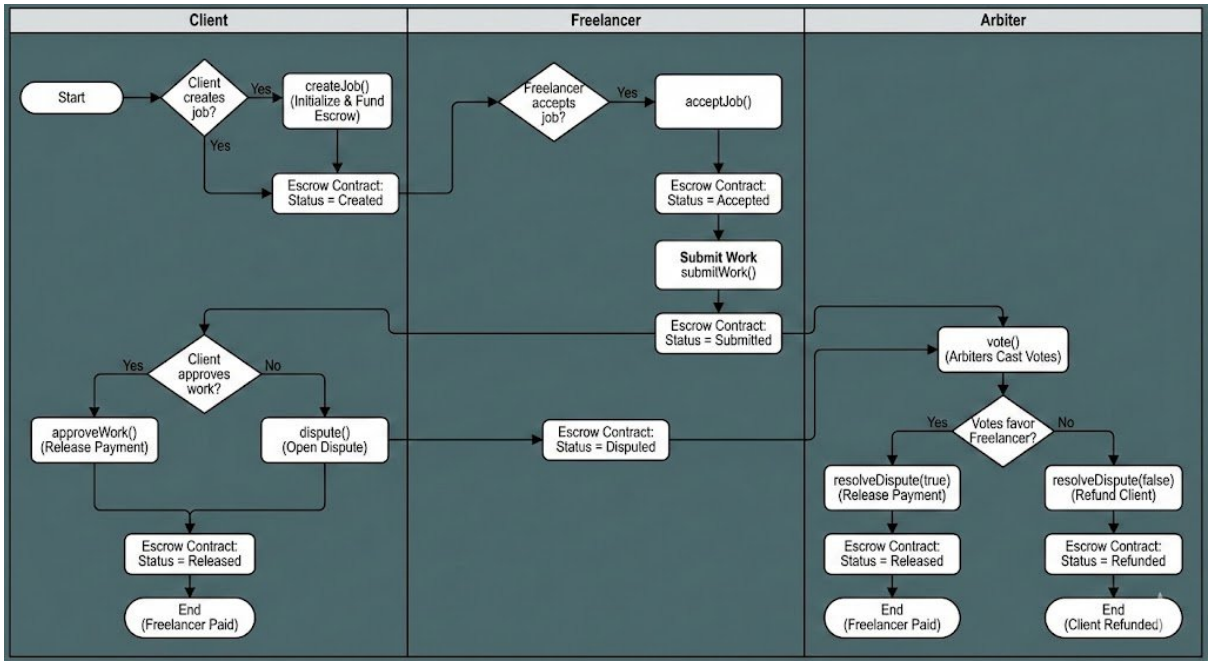
- *Cấu trúc dữ liệu:* Sử dụng struct VoteState để ghi lại số phiếu ủng hộ cho từng bên và trạng thái giải quyết của từng vụ tranh chấp.
- *Quy tắc ra quyết định:* Hệ thống yêu cầu một số lượng phiếu bầu tối thiểu (required) từ hội đồng trọng tài để thực thi quyết định cuối cùng, loại bỏ rủi ro thao túng từ một cá nhân duy nhất.

Biến	Kiểu dữ liệu	Mô tả
`arbiters`	`address[]`	Danh sách địa chỉ các trọng tài
`isArbiter`	`mapping(address => bool)`	Kiểm tra một địa chỉ có phải trọng tài không
`required`	`uint256`	Số phiếu bầu tối thiểu để đưa ra quyết định
`disputes`	`mapping(address => VoteState)`	Trạng thái bỏ phiếu cho từng escrow

```
struct VoteState {
    uint256 votesForFreelancer;          // Số phiếu ủng hộ freelancer
    uint256 votesForClient;              // Số phiếu ủng hộ client
    bool resolved;                       // Tranh chấp đã được giải quyết chưa
    mapping(address => bool) hasVoted;   // Trọng tài đã bỏ phiếu chưa
}
```

3.1.2. Các hàm chức năng chính của Smart Contract

Quy trình làm việc trên Smart Contract của dự án được thiết kế theo một luồng logic chặt chẽ, đảm bảo an toàn tài chính cho cả hai bên tham gia.



Quy trình nghiệp vụ và Logic chức năng

1/ Khởi tạo và Thiết lập (Initialization): Khi một Client tạo công việc mới thông qua createJob(), hệ thống sẽ:

- Khởi tạo một instance mới của FreelanceEscrow.
- Chuyển số tiền ETH tương ứng từ ví Client vào hợp đồng thông qua hàm init() với thuộc tính payable.
- Thiết lập các thông số về thời hạn và gán quyền trọng tài.

2/ Vòng đời thực hiện công việc bao gồm:

- Tiếp nhận (Accept): Freelancer thực hiện hàm acceptJob() để xác nhận tham gia dự án. Hệ thống chỉ cho phép nhận việc khi trạng thái là Created.
- Nộp kết quả (Submit): Sau khi hoàn thành, Freelancer gọi hàm submitWork() để thông báo kết quả. Trạng thái hợp đồng chuyển sang Submitted, sẵn sàng cho bước kiểm duyệt.

3/ Nghiệm thu và Giải quyết tranh chấp: Đây là giai đoạn then chốt trong việc bảo vệ quyền lợi người dùng

- Thanh toán tự nguyện: Nếu Client hài lòng, hàm approveWork() sẽ kích hoạt cơ chế _payFreelancer() để chuyển ETH trực tiếp cho Freelancer.
- Kích hoạt tranh chấp: Trong trường hợp sản phẩm không đạt yêu cầu, Client có quyền gọi hàm dispute() để khóa trạng thái và mời trọng tài vào cuộc.

- Thực thi quyết định: Dựa trên kết quả bỏ phiếu từ DisputeMultiSig, hàm resolveDispute() sẽ tự động phân phối lại tài sản (thanh toán cho Freelancer hoặc hoàn tiền cho Client).

4/ Cơ chế bảo mật trong thanh toán: Hệ thống sử dụng hàm call cấp thấp thay vì transfer hoặc send trong các hàm nội bộ _payFreelancer() và _refundClient(). Đây là một kỹ thuật quan trọng nhằm:

- Tương thích với các loại ví Smart Contract có logic phức tạp.
- Tránh các lỗi liên quan đến giới hạn gas (gas limit) khi thực hiện giao dịch chuyển tiền.

3.2. Phát triển giao diện người dùng

Để đảm bảo trải nghiệm người dùng trực quan và tương tác hiệu quả với hệ thống Blockchain, đội ngũ phát triển đã xây dựng ứng dụng Web Frontend sử dụng thư viện React kết hợp với Ethers.js phiên bản 6. Giao diện được thiết kế để kết nối trực tiếp với ví MetaMask, cho phép người dùng thực hiện các giao dịch ký gửi (escrow) và bỏ phiếu phi tập trung một cách minh bạch.

3.2.1. Tích hợp thư viện Ethers.js

Thư viện Ethers.js đóng vai trò là cầu nối giao tiếp giữa giao diện web và mạng lưới Ethereum (Blockchain). Hệ thống sử dụng Ethers.js để khởi tạo các đối tượng hợp đồng (Contract Instances), đọc trạng thái dữ liệu và gửi các giao dịch đã ký.

1/ Cấu hình Contract Instances

Việc khởi tạo các đối tượng hợp đồng được tập trung hóa trong module để đảm bảo tính tái sử dụng và dễ dàng bảo trì. Các hàm factory được định nghĩa để tạo ra các instance tương ứng với ba loại smart contract của hệ thống:

- getFactory(signer): Khởi tạo instance cho EscrowFactory tại địa chỉ cố định FACTORY_ADDRESS. Đối tượng signer được truyền vào để cho phép thực hiện các giao dịch thay đổi trạng thái (như tạo job mới).
- getEscrow(address, runner): Khởi tạo instance cho một hợp đồng FreelanceEscrow cụ thể dựa trên địa chỉ động được truyền vào. Tham số runner có thể là provider (chỉ đọc) hoặc signer (đọc/ghi).
- getMultiSig(runner): Khởi tạo instance cho hệ thống trọng tài DisputeMultiSig tại địa chỉ MULTISIG_ADDRESS.

2/ Đọc dữ liệu từ Blockchain

Quy trình đồng bộ dữ liệu từ Blockchain về giao diện người dùng được thực hiện thông qua ethers.BrowserProvider. Ví dụ, hàm loadJobs thực hiện truy vấn danh sách tất cả các hợp đồng escrow đã được tạo:

1. Gọi hàm factory.getAllJobs() để lấy mảng địa chỉ các hợp đồng.
2. Duyệt qua từng địa chỉ và khởi tạo instance FreelanceEscrow tương ứng.
3. Sử dụng Promise.all để truy xuất song song các thông tin chi tiết như: người thuê (client), người làm việc (freelancer), số tiền (amount), và trạng thái (status). Kỹ thuật này giúp tối ưu hóa thời gian tải trang so với việc truy vấn tuần tự.

3/ Gửi giao dịch lên Blockchain

Các hành động thay đổi trạng thái hệ thống (như tạo công việc, thanh toán) yêu cầu người dùng phải ký xác nhận giao dịch. Hàm createJob minh họa quy trình này:

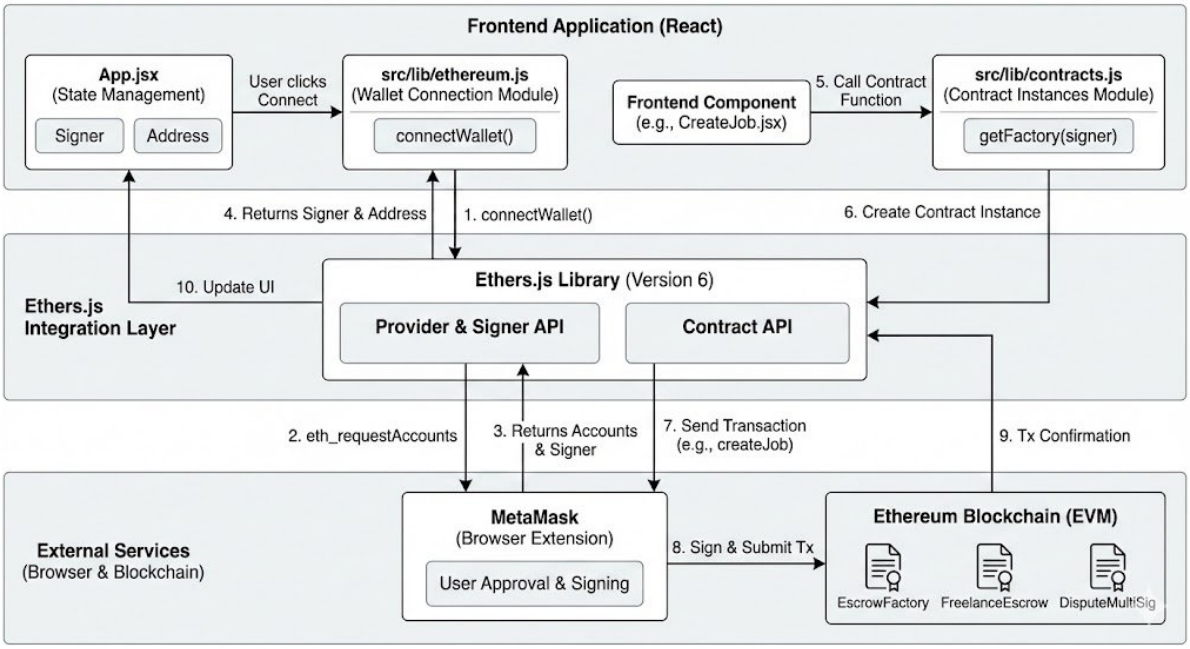
- Sử dụng ethers.parseEther(amount) để chuyển đổi số tiền từ đơn vị ETH sang Wei (đơn vị nhỏ nhất của Ethereum).
- Gọi hàm factory.createJob() kèm theo giá trị ETH (value) gửi vào hợp đồng.
- Sử dụng await tx.wait() để đợi giao dịch được xác nhận (mined) trên blockchain trước khi cập nhật giao diện, đảm bảo tính nhất quán của dữ liệu.

3.2.2. Xử lý kết nối ví MetaMask

Việc quản lý định danh người dùng và xác thực giao dịch được ủy quyền hoàn toàn cho ví MetaMask thông qua chuẩn EIP-1193. Module ethereum.js chịu trách nhiệm xử lý logic kết nối:

- *Kiểm tra môi trường:* Xác minh sự tồn tại của đối tượng window.ethereum để đảm bảo trình duyệt đã cài đặt MetaMask.
- *Yêu cầu quyền truy cập:* Gọi phương thức eth_requestAccounts để yêu cầu người dùng cấp quyền đọc địa chỉ ví.
- *Khởi tạo Signer:* Trả về đối tượng signer và địa chỉ ví (address) để ứng dụng sử dụng cho các tác vụ tiếp theo.

Trong component gốc App, trạng thái kết nối được quản lý thông qua React Hooks (useState). Giao diện sẽ hiển thị nút "Connect Wallet" khi chưa có signer và chuyển sang giao diện chính (MainApp) sau khi kết nối thành công. Điều này đảm bảo người dùng không thể tương tác với các tính năng sâu của hệ thống nếu chưa xác thực ví.



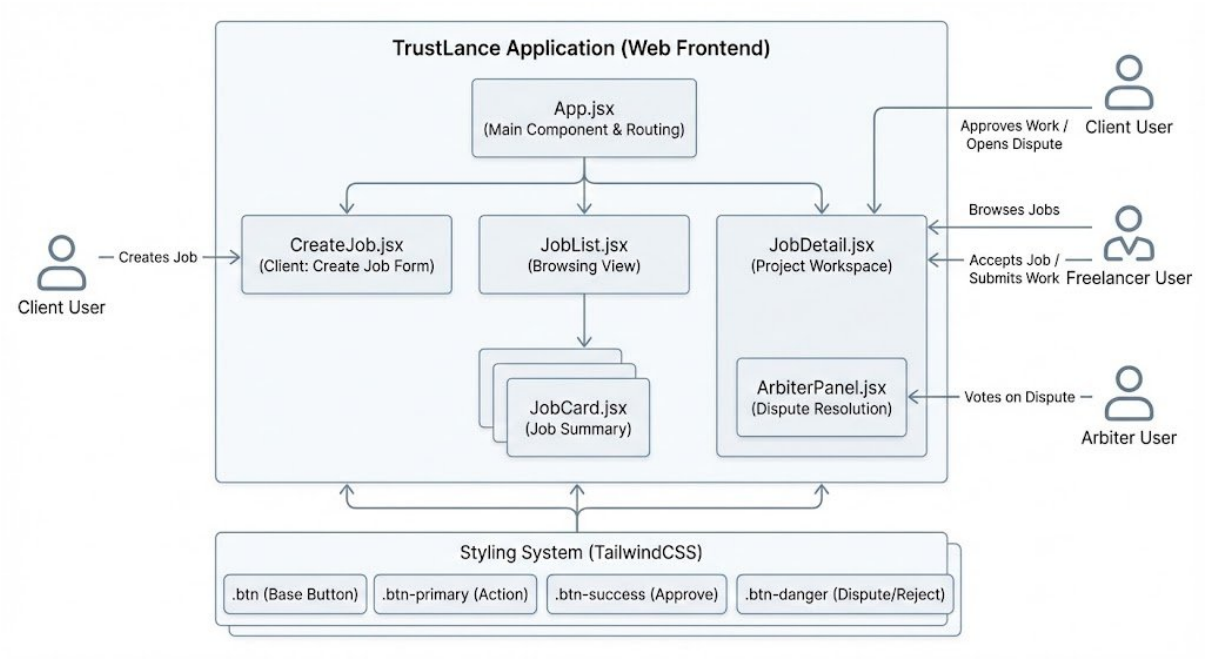
Sơ đồ tích hợp Ethers.js và liên kết ví

3.2.3. Thiết kế giao diện và tương tác người dùng

Kiến trúc Frontend được chia thành các Component độc lập, tối ưu hóa cho từng vai trò người dùng (Client, Freelancer, Arbiter). Cấu trúc phân cấp của ứng dụng bao gồm 4 thành phần chính:

- *CreateJob:* Form dành cho Client để khởi tạo hợp đồng mới.
- *JobList & JobCard:* Hiển thị danh sách công việc dưới dạng lưới (grid), giúp người dùng dễ dàng lướt xem.
- *JobDetail:* Màn hình chi tiết, nơi diễn ra hầu hết các tương tác nghiệp vụ (nộp bài, thanh toán, tranh chấp).
- *ArbiterPanel:* Component chuyên biệt dành riêng cho trọng tài để thực hiện bỏ phiếu giải quyết tranh chấp.

Component CreateJob là giao diện đơn giản hóa việc nhập liệu với các trường số tiền và thời hạn. Hệ thống tự động chuyển đổi thời gian người dùng chọn sang Unix timestamp để tương thích với Smart Contract. Component ArbiterPanel cung cấp giao diện trực quan cho quy trình bỏ phiếu Multi-Sig. Các thanh tiến trình (progress bar) hiển thị tỷ lệ phiếu bầu hiện tại (ví dụ: "Pay Freelancer: 1/2"), giúp trọng tài nắm bắt nhanh tình hình tranh chấp. Các nút bỏ phiếu sẽ tự động bị vô hiệu hóa (disabled) sau khi trọng tài đã thực hiện quyền của mình.



Giao diện và trải nghiệm người dùng

Giao diện được xây dựng dựa trên TailwindCSS, sử dụng các lớp tiện ích (utility classes) để thiết kế nhanh chóng và nhất quán. Các component như nút bấm (.btn) được định nghĩa lại với các biến thể màu sắc rõ ràng để chỉ dẫn hành động:

- *btn-primary:* Dùng cho các hành động chính (Tạo Job, Nộp bài).
- *btn-success:* Dùng cho hành động tích cực (Thanh toán).
- *btn-danger:* Dùng cho hành động cảnh báo (Mở tranh chấp, Hoàn tiền).

3.3. Quy trình triển khai hệ thống

Quy trình triển khai hệ thống được chia làm hai giai đoạn chính: thiết lập môi trường Blockchain cục bộ và cấu hình ứng dụng Web để tương tác với mạng lưới này.

3.3.1. Triển khai Smart Contract lên mạng local

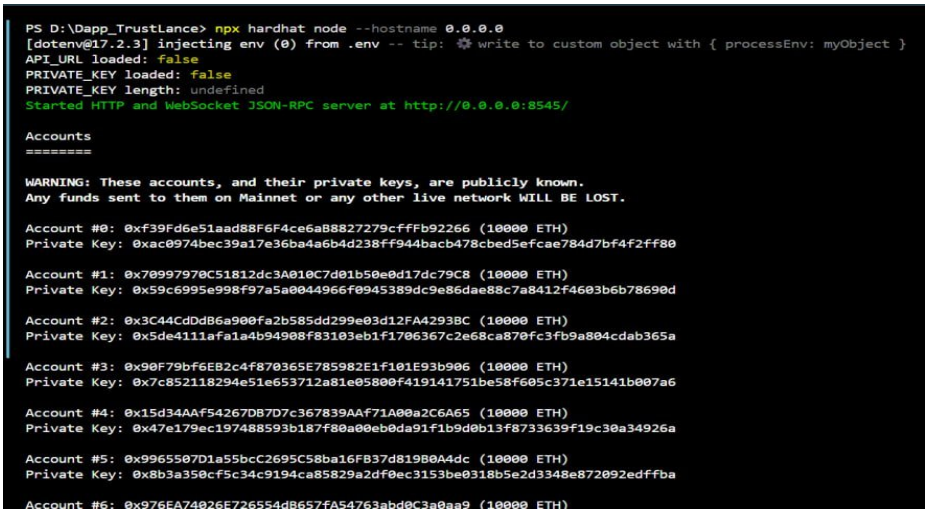
Việc triển khai Smart Contract được thực hiện thông qua công cụ Hardhat, một môi trường phát triển Ethereum chuyên nghiệp giúp biên dịch, triển khai, thử nghiệm và gỡ lỗi phần mềm. Tệp cấu hình hardhat.config.js được thiết lập để kết nối với mạng cục bộ tại cổng 8545. Cấu hình này định nghĩa phiên bản Solidity 0.8.24 và Chain ID 31337 để đảm bảo tính nhất quán trong quá trình ký giao dịch.

- *Triển khai DisputeMultiSig:* Hệ thống ưu tiên triển khai hợp đồng trọng tài trước tiên. Một hội đồng gồm 3 trọng tài được thiết lập với quy tắc đồng thuận là 2/3 phiếu bầu (required = 2).

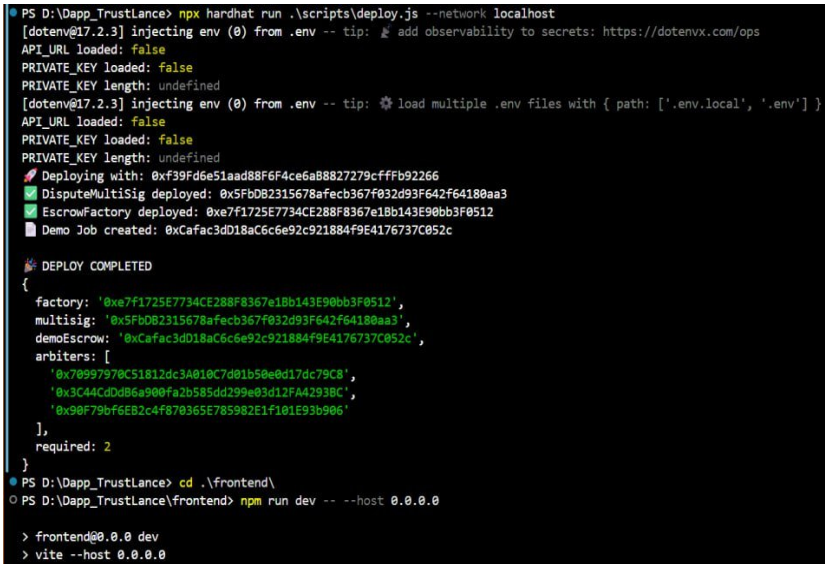
- *Triển khai EscrowFactory:* Sau khi có địa chỉ của DisputeMultiSig, hợp đồng EscrowFactory được triển khai và liên kết với hợp đồng trọng tài này.
- Khởi tạo dữ liệu mẫu (Optional): Script cũng tự động tạo một công việc mẫu (Demo Job) với thời hạn 7 ngày và giá trị 1 ETH để phục vụ việc kiểm thử ngay lập tức.
- *Lưu trữ metadata:* Cuối cùng, các địa chỉ hợp đồng và thông tin cấu hình (factoryAddress, multisigAddress, arbiters) được xuất ra file JSON để Frontend có thể sử dụng.

Quá trình triển khai thực tế diễn ra qua hai bước lệnh trên terminal:

1/ Khởi động node cục bộ: **npx hardhat node**

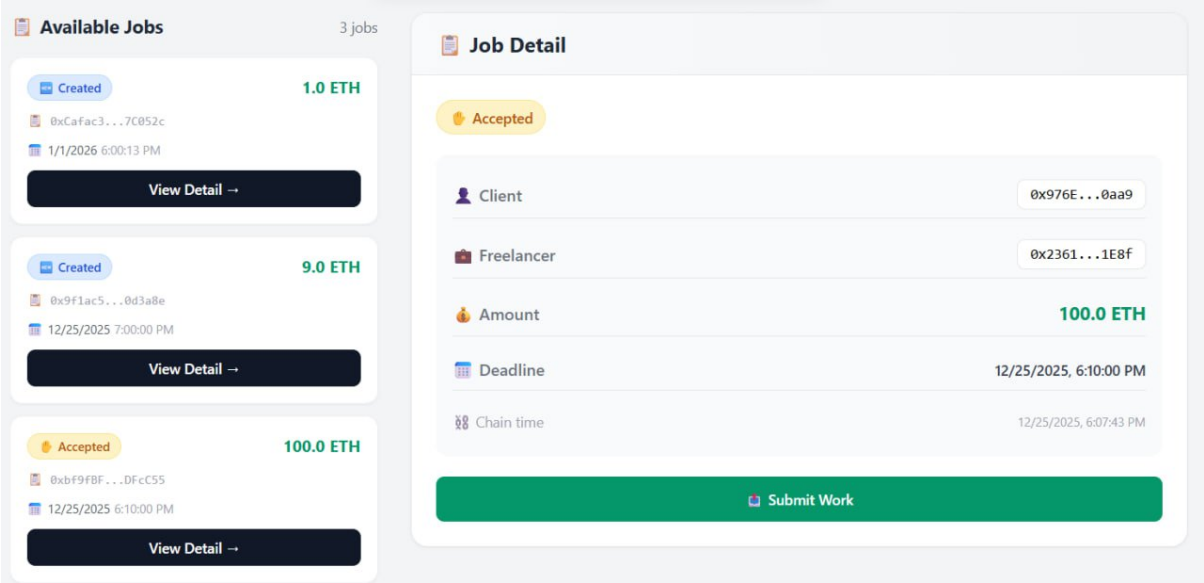


2/ Chạy script triển khai: **npx hardhat run scripts/deploy.js --network localhost**



3.3.2. Triển khai ứng dụng Web Frontend

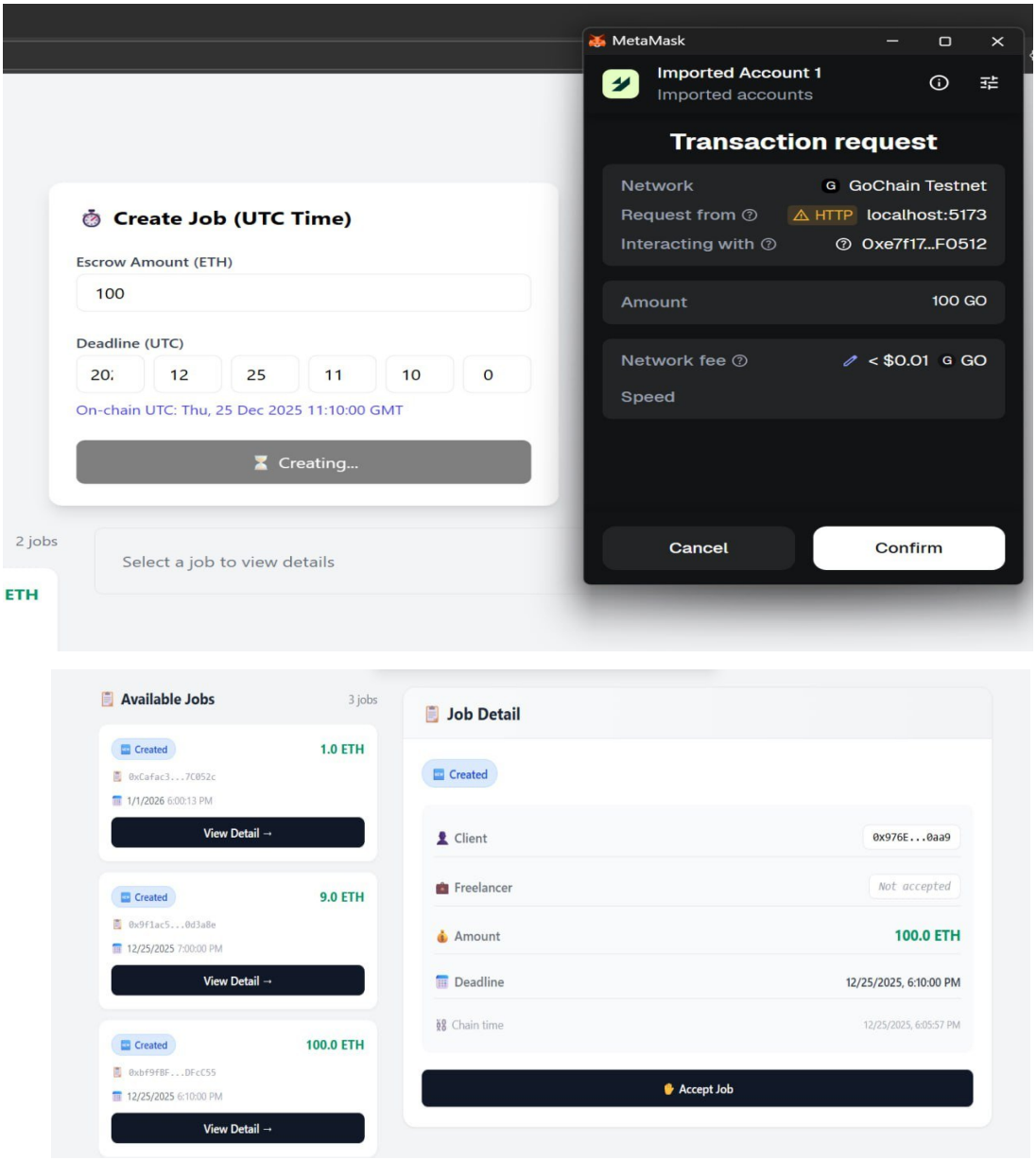
Sau khi Smart Contract đã hoạt động trên mạng local, ứng dụng Frontend cần được cấu hình để kết nối chính xác. File config.js trong mã nguồn Frontend được cập nhật với các địa chỉ contract vừa triển khai. Điều này đảm bảo ứng dụng gọi đúng vào các hợp đồng mới nhất. Để tương tác với mạng Hardhat Local, ví MetaMask cần được cấu hình thủ công với các thông số RPC URL (http://127.0.0.1:8545) và Chain ID (31337). Các tài khoản kiểm thử (Test Accounts) được nhập vào ví sử dụng Private Key do Hardhat cung cấp, mỗi tài khoản được cấp sẵn 10,000 ETH giả lập.



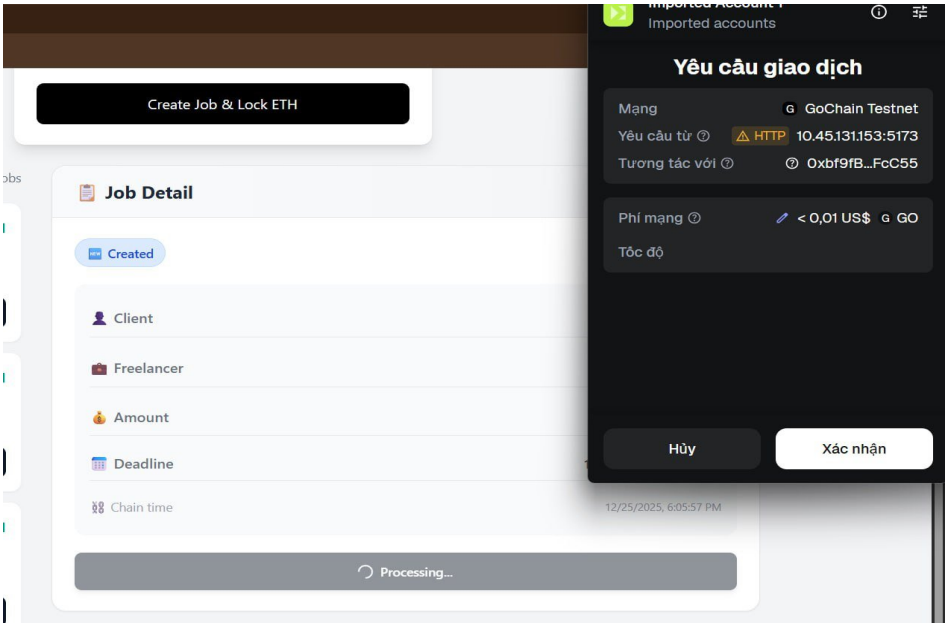
3.4. Thực nghiệm và đánh giá kết quả

3.4.1. Kịch bản 1: Giao dịch đúng hẹn và được thanh toán

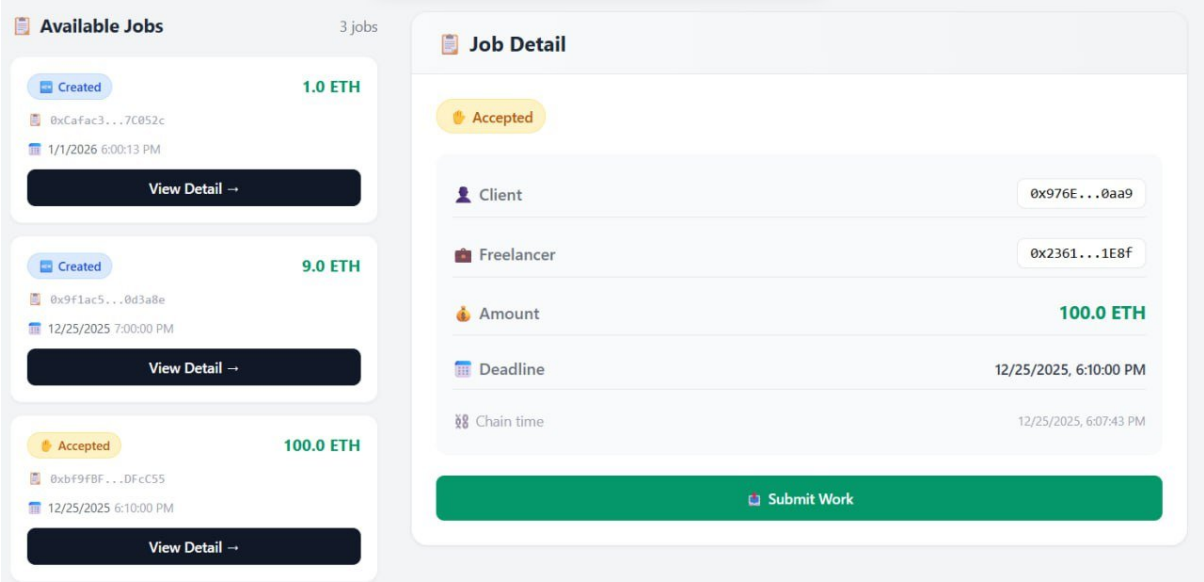
Mô tả: Người thuê sẽ tiến hành tạo một job mới, khai báo các thông tin về thù lao, ngày đến hạn và thông tin về yêu cầu công việc.



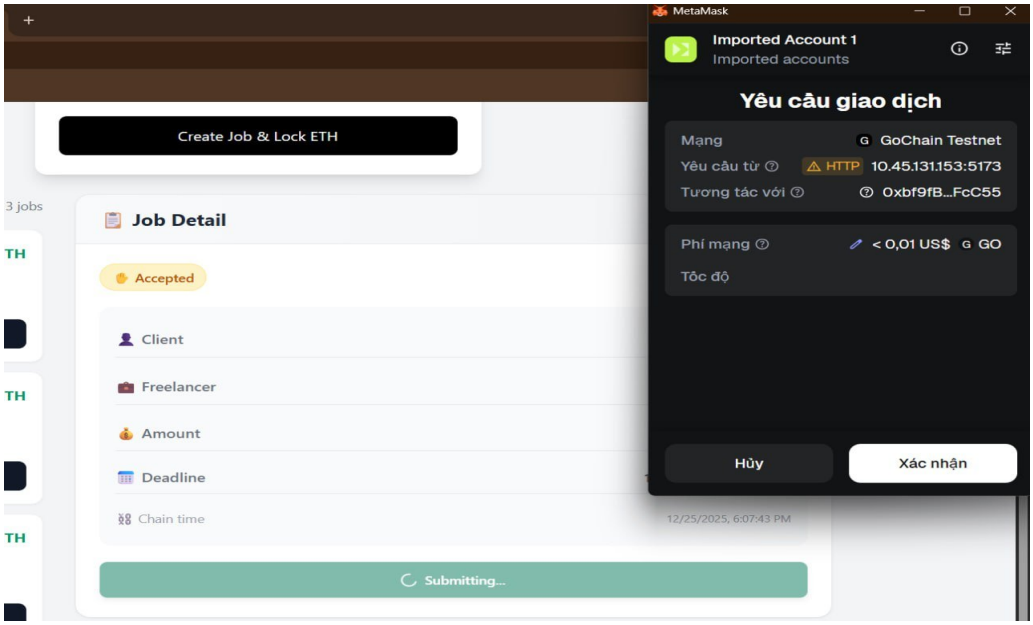
Các freelancer có nhu cầu tìm việc sẽ truy cập nền tảng, click vào các job đang open và tiến hành nhận việc.



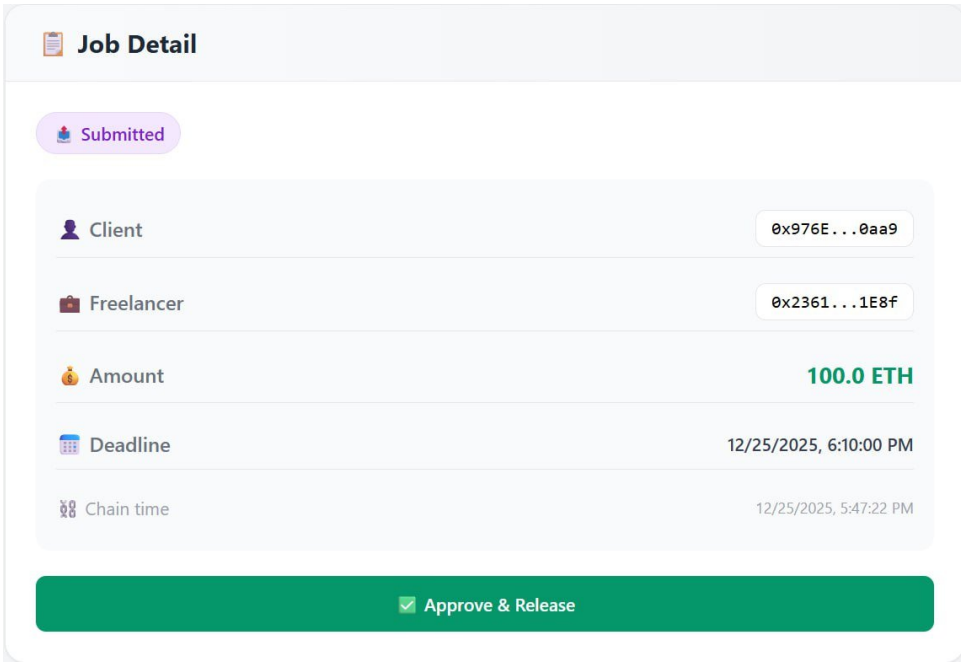
Freelancer sau đó tiến hành thực hiện job đã nhận, sau khi hoàn thành sẽ quay lại hệ thống để tiến hành thông báo kết quả.

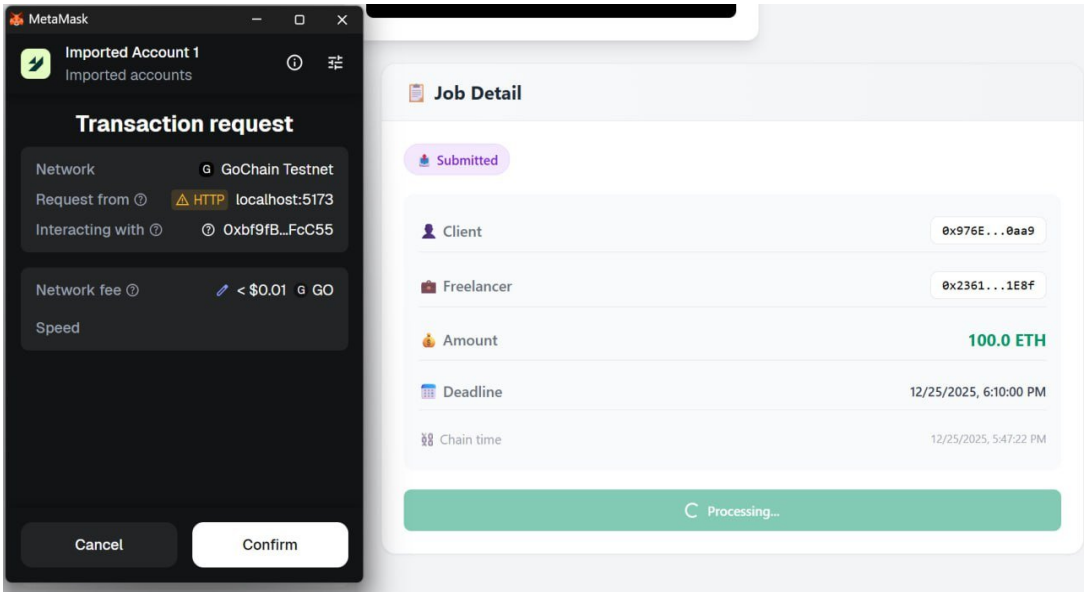


Ấn Submit Work sẽ ghi thông báo hoàn thành công việc vào Blockchain và phía người thuê sẽ có thể vào kiểm tra chất lượng bàn giao.

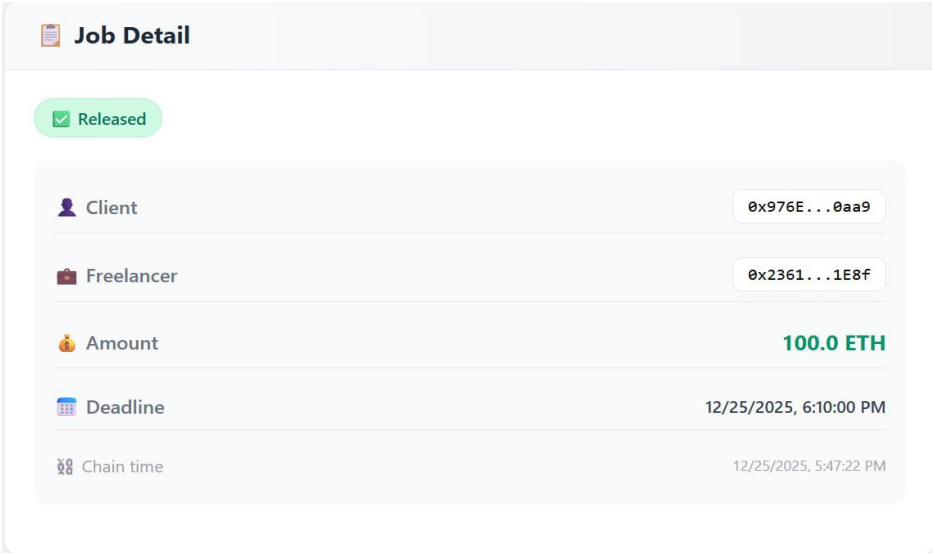


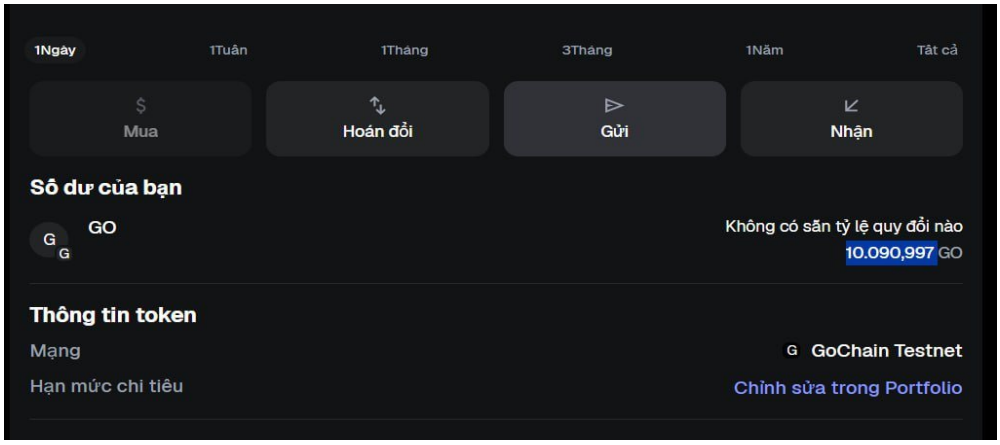
Người thuê nếu không phát sinh khiếu nại và chấp nhận sản phẩm thì sẽ ấn chấp nhận thanh toán để hệ thống tự động chuyển tiền đến ví của freelancer.





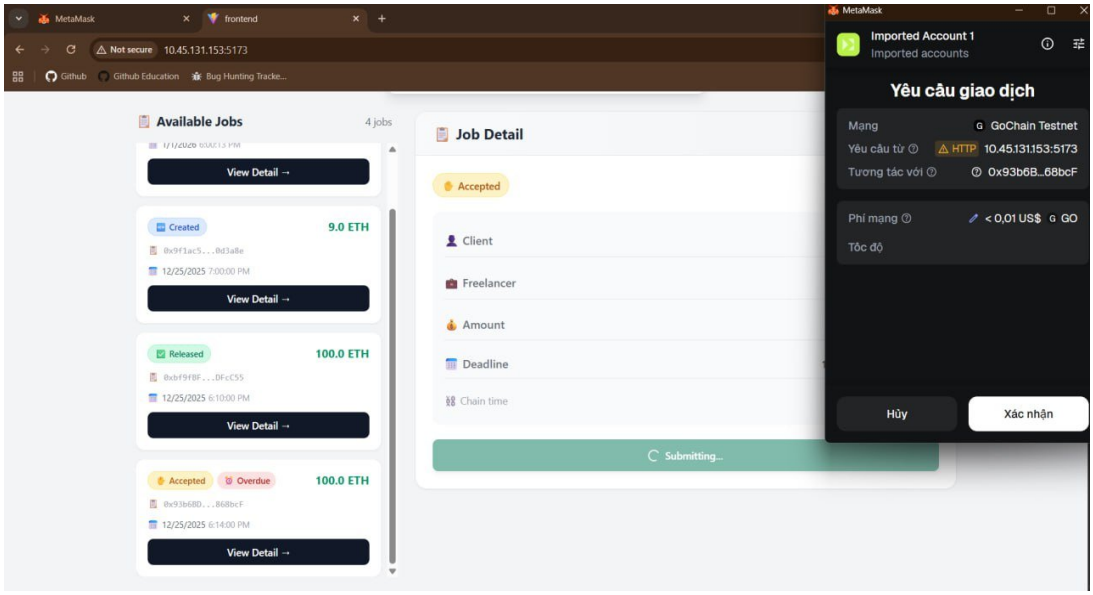
Hợp đồng hoàn tất và freelancer kiểm tra ví sẽ thấy tài khoản đã được nhận tiền theo thù lao đã nêu trong hợp đồng. Lưu ý số tiền không đủ 100 ETH vì các thao tác đều phát sinh một ít phí gas cho việc lưu trữ thông tin trên blockchain.



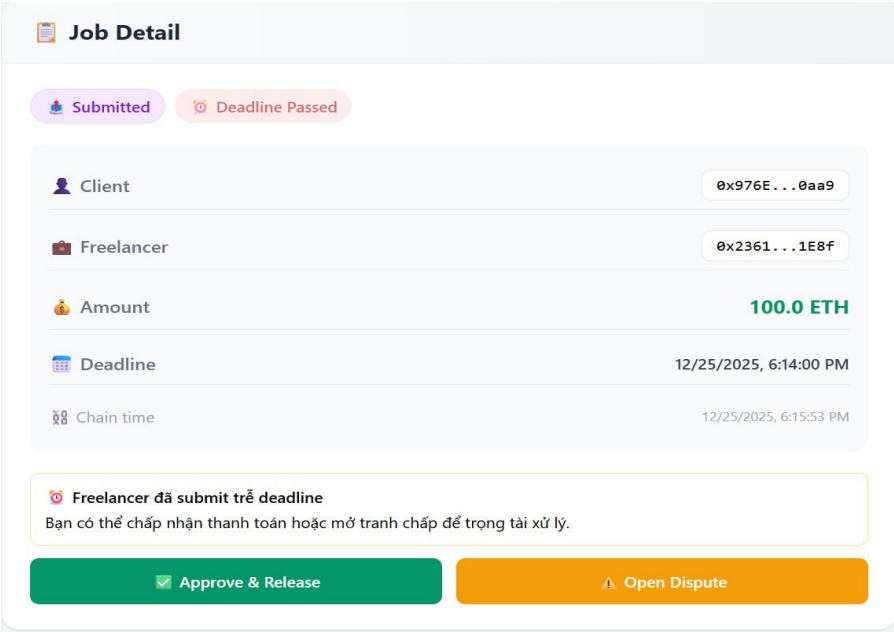


3.4.2. Kịch bản 2: Giao dịch trễ hẹn và được thanh toán

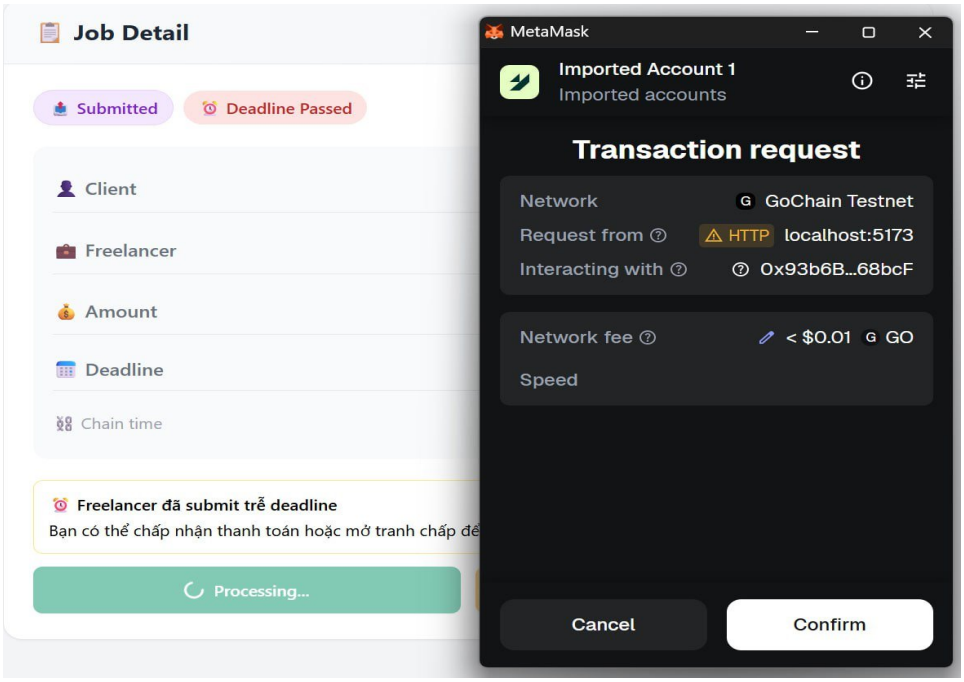
Mô tả: Tình huống 2 mô phỏng kịch bản freelancer hoàn thành trễ hẹn so với thông tin deadline được ghi trong hợp đồng. Hệ thống hiển thị thông tin cảnh báo quá hạn.



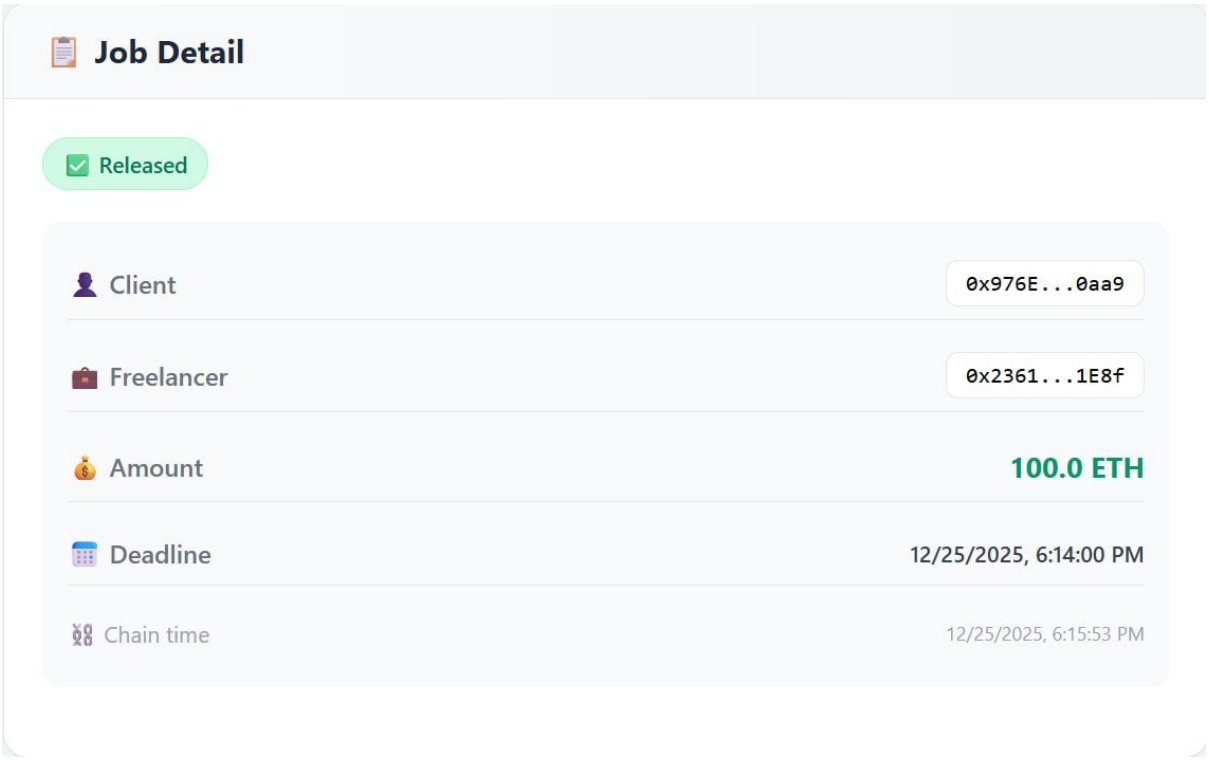
Lúc này người thuê sẽ chọn giữa việc chấp nhận thanh toán hoặc có thể mở tranh chấp để yêu cầu hoàn tiền.

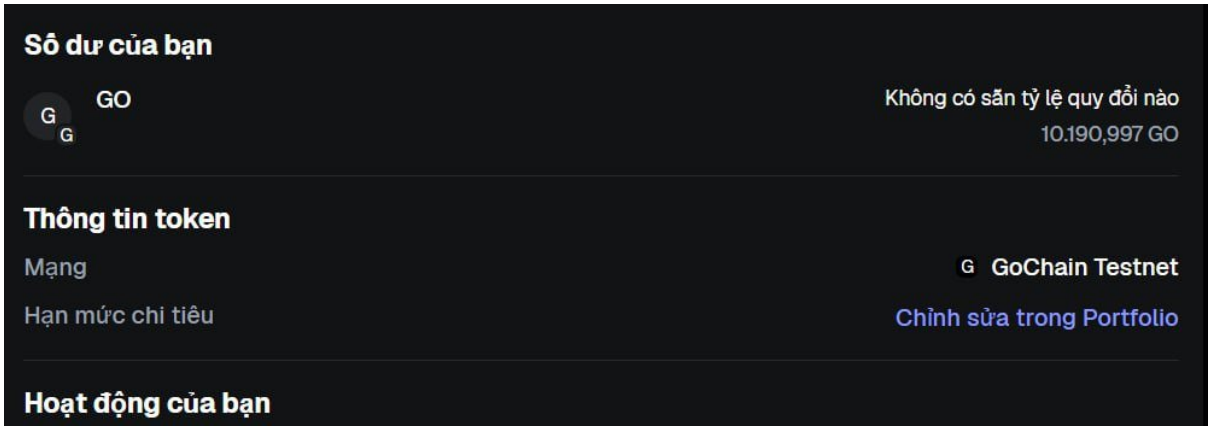


Người thuê quyết định chấp nhận thanh toán dù đã trễ hẹn



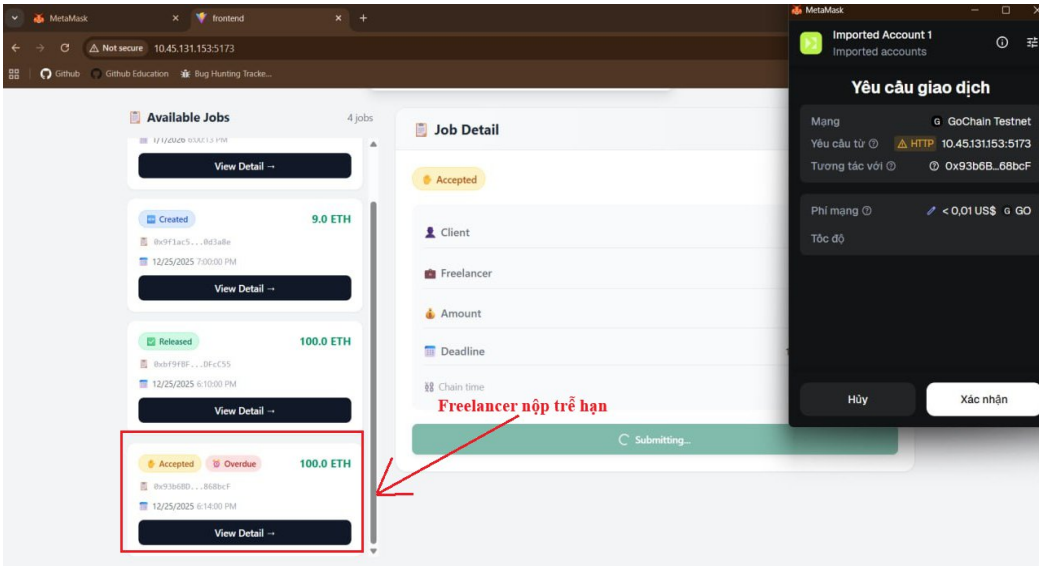
Hệ thống sẽ xử lý chuyển tiền đến ví của freelancer và hoàn tất hợp đồng như bình thường.



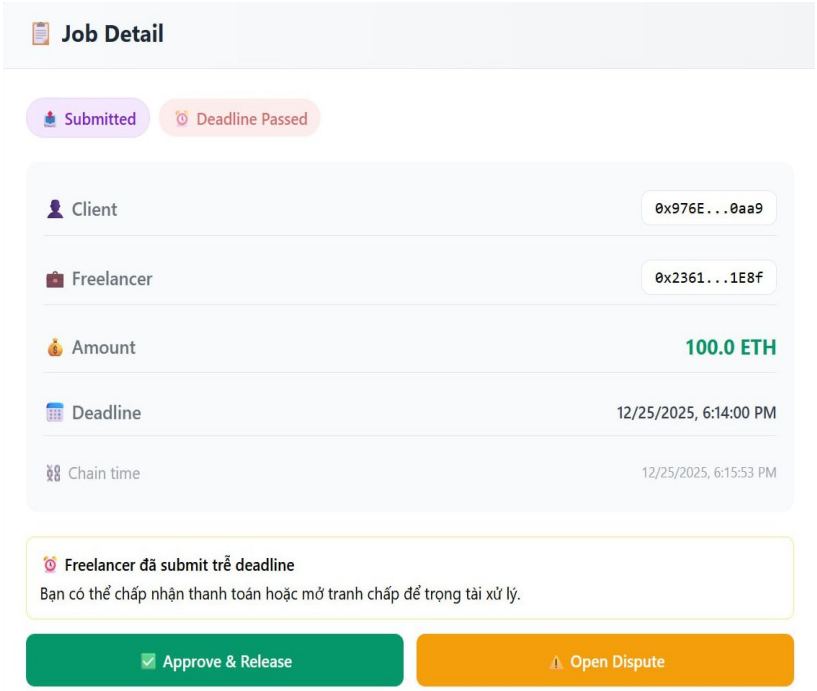


3.4.3. Kịch bản 3: Giao dịch trễ hẹn và phát sinh tranh chấp

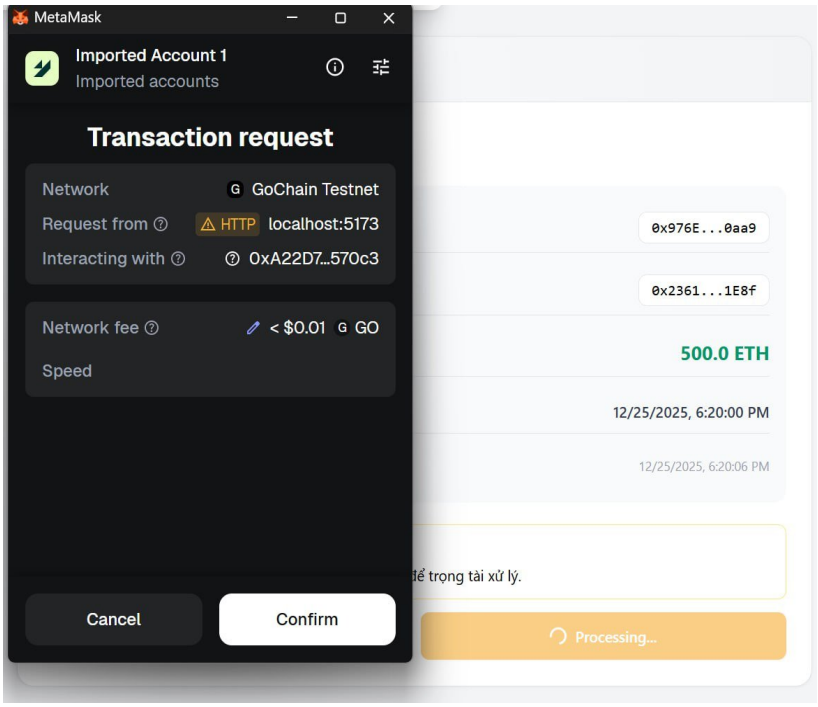
Mô tả: Trong tình huống 3 freelancer hoàn thành công việc trễ hạn và người thuê không đồng tình vào chuyện này nên đã phát sinh tranh chấp.



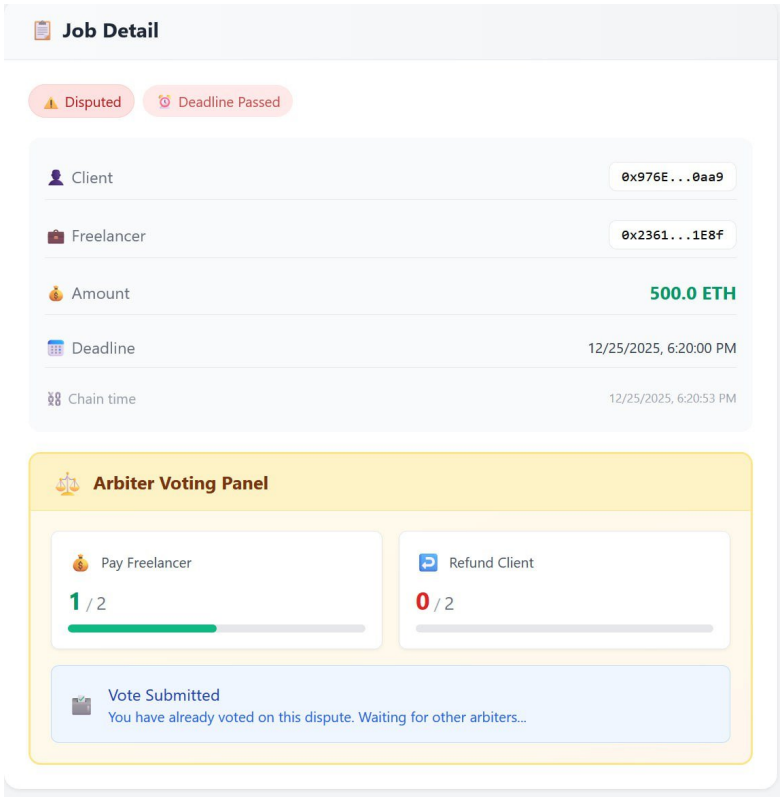
Khi đó thì người thuê có thể chọn lựa giữa việc trả tiền hoặc gọi trọng tài để có thể phân xử trong tình huống này



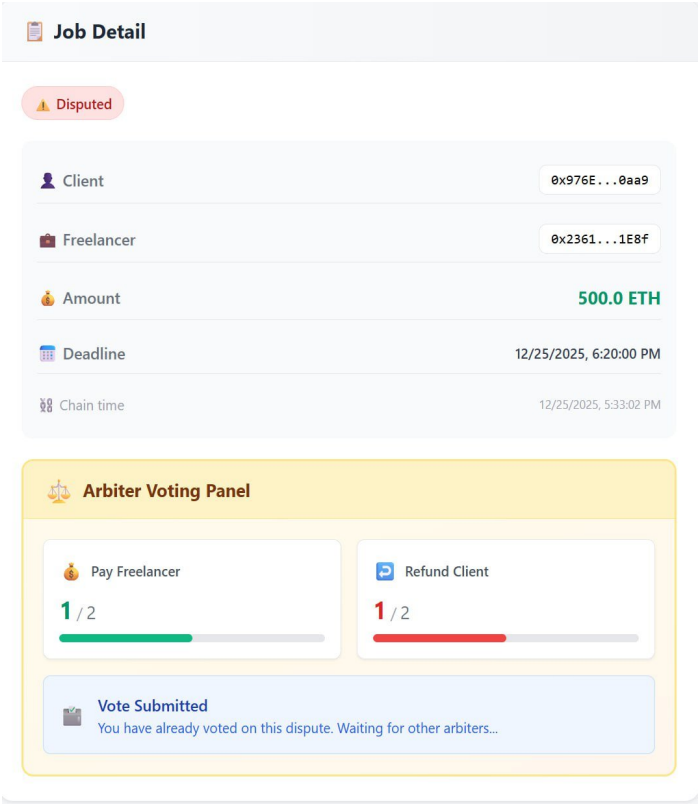
Người thuê quyết định gọi trọng tài để xử lý tranh chấp này



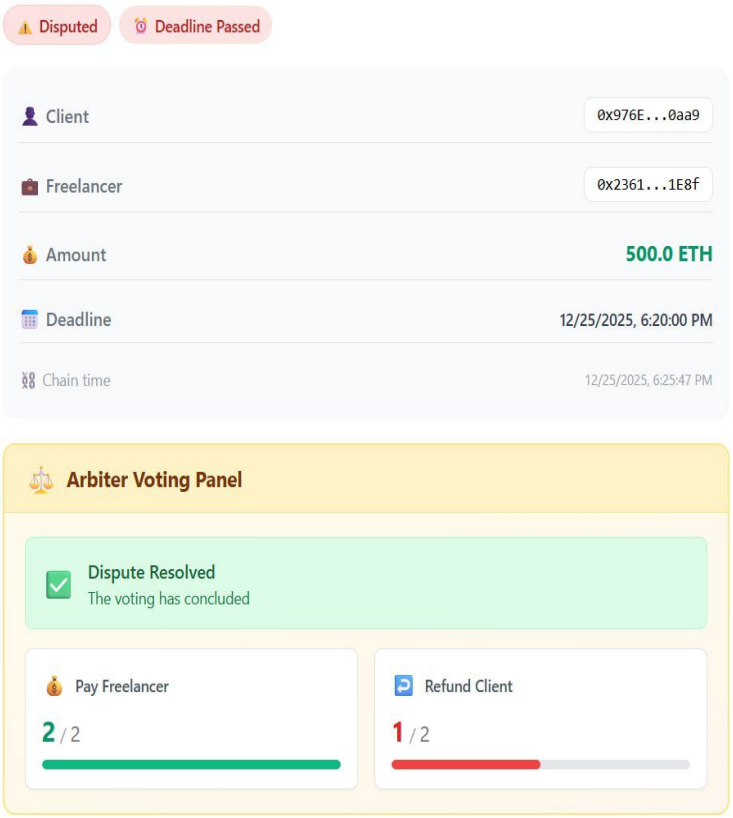
Trọng tài 1 đã quyết định vote cho freelancer



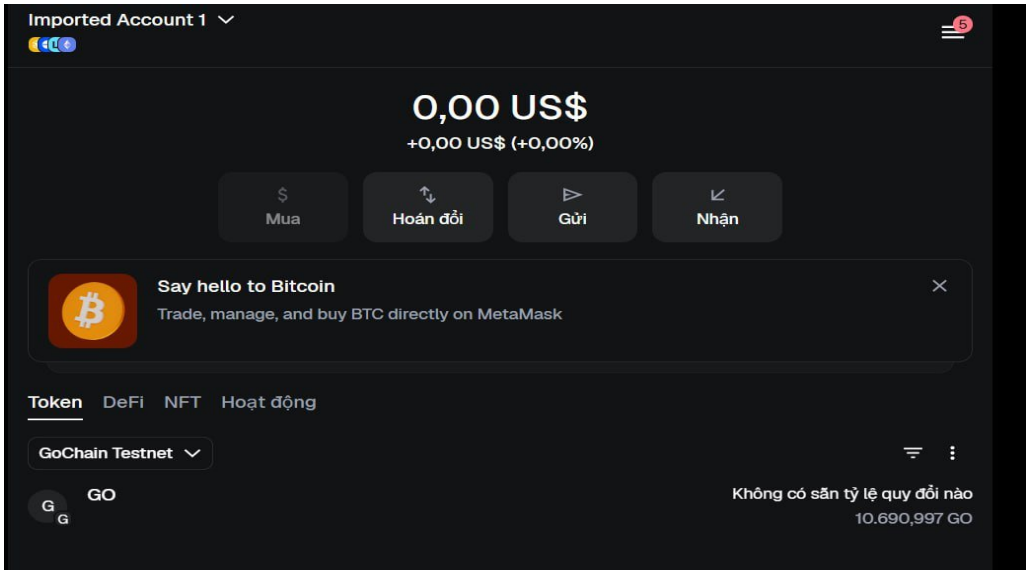
Trọng tài 2 thì quyết định vote cho người thuê



Và trọng tài 3 quyết định vote cho freelancer, với số phiếu 2/3 trọng tài thì freelancer đã được quyết định để thắng cuộc tranh chấp này

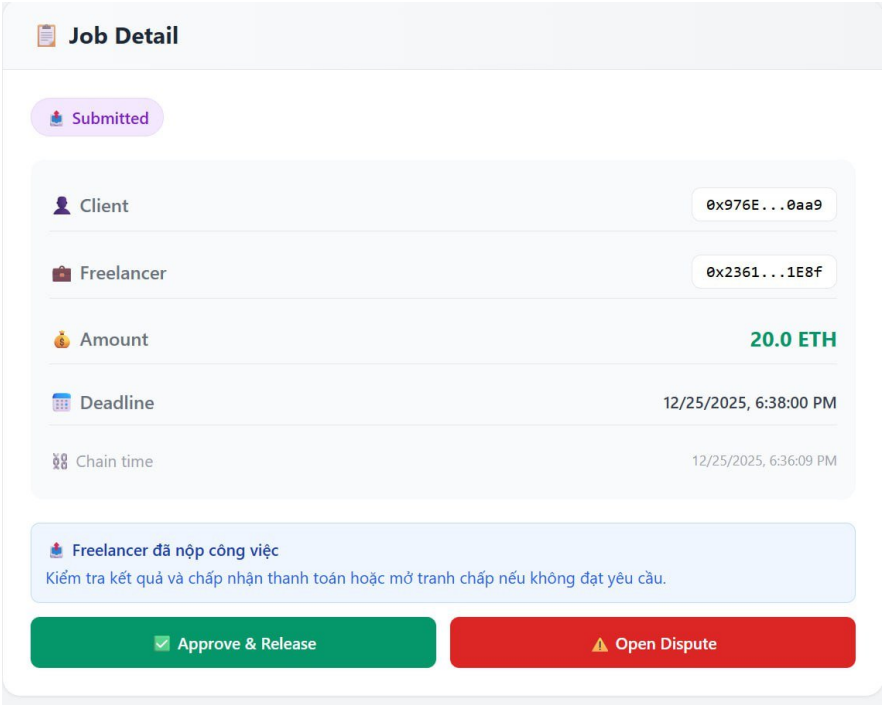


Cuối cùng, tiền đã được trả về cho freelancer

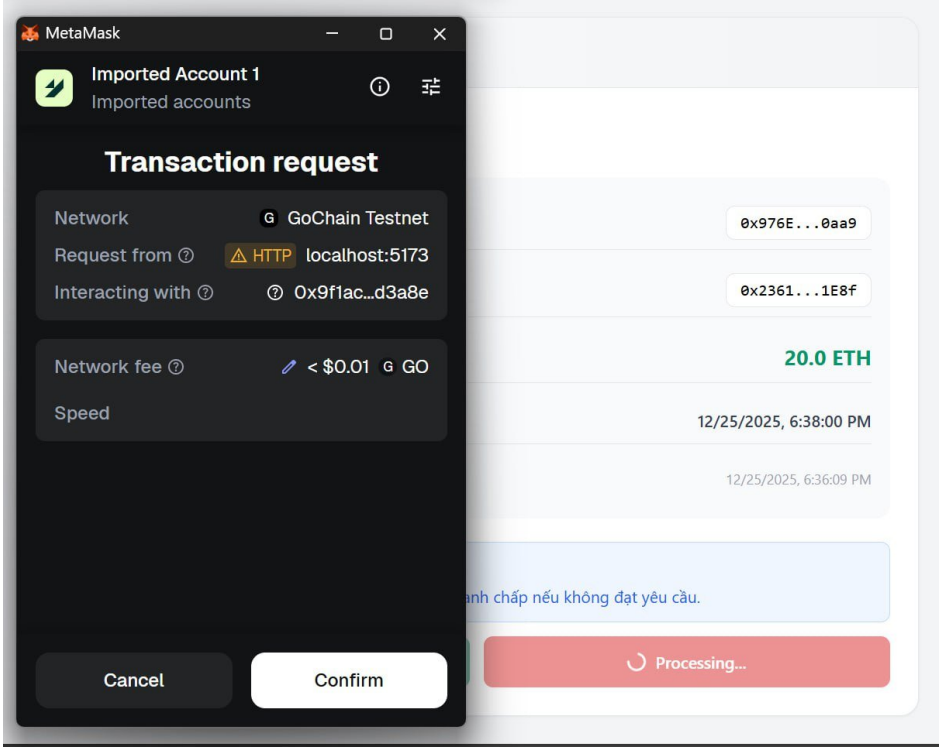


3.4.4. Kịch bản 4: Giao dịch đúng hẹn và phát sinh tranh chấp

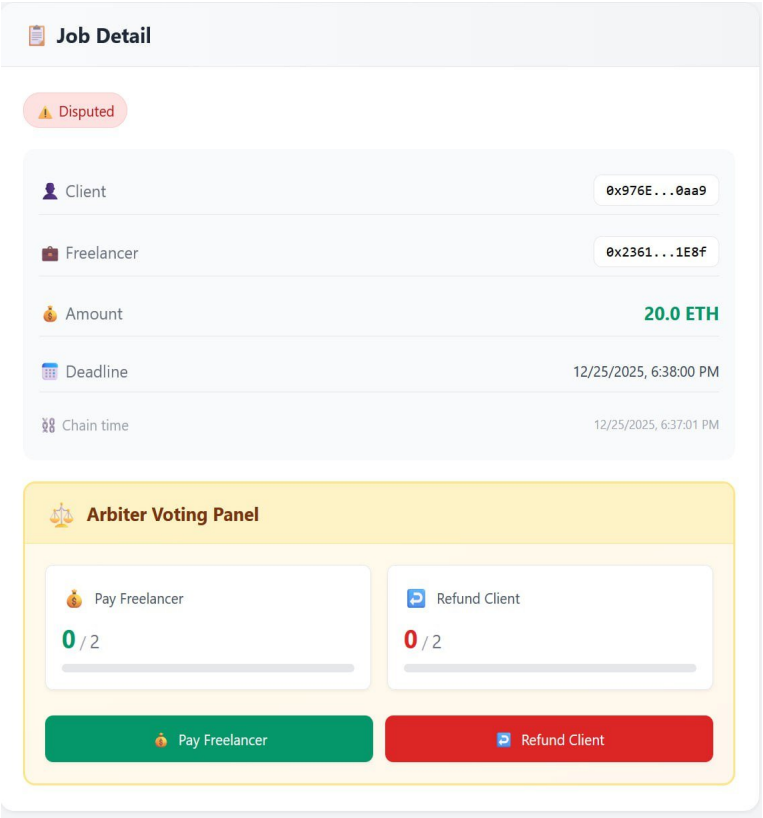
Mô tả: Trong tình huống này, dù nộp đúng hẹn nhưng phía người thuê cho rằng kết quả bàn giao của freelancer không đạt yêu cầu công việc và muốn mở tranh chấp.



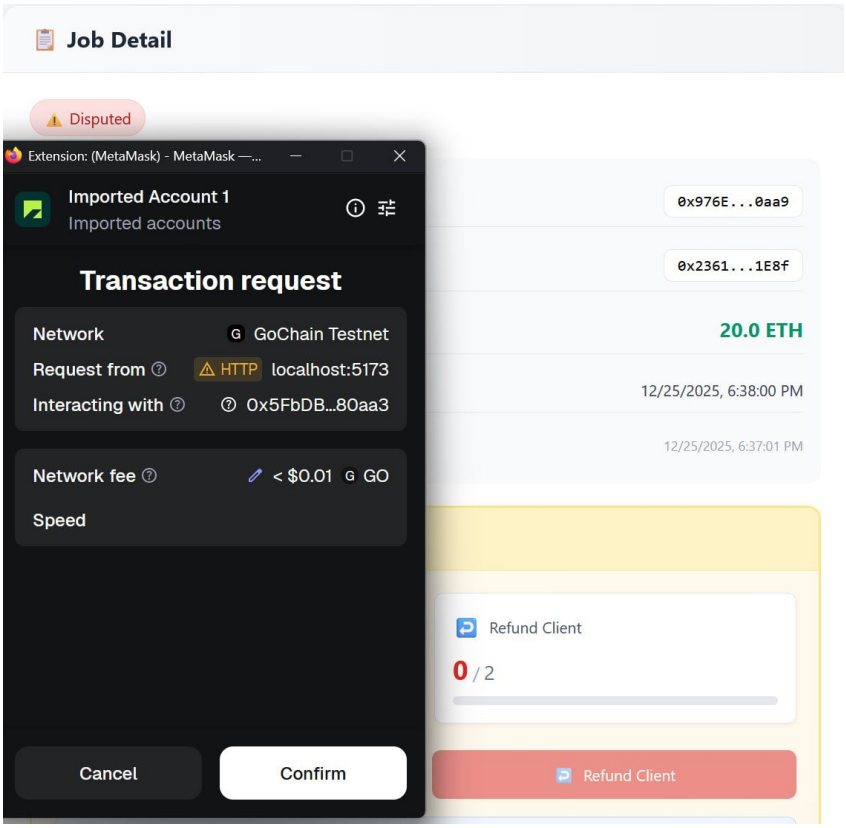
Người thuê đã quyết định mở tranh chấp, gọi trọng tài bằng cách ấn nút "Open Dispute"

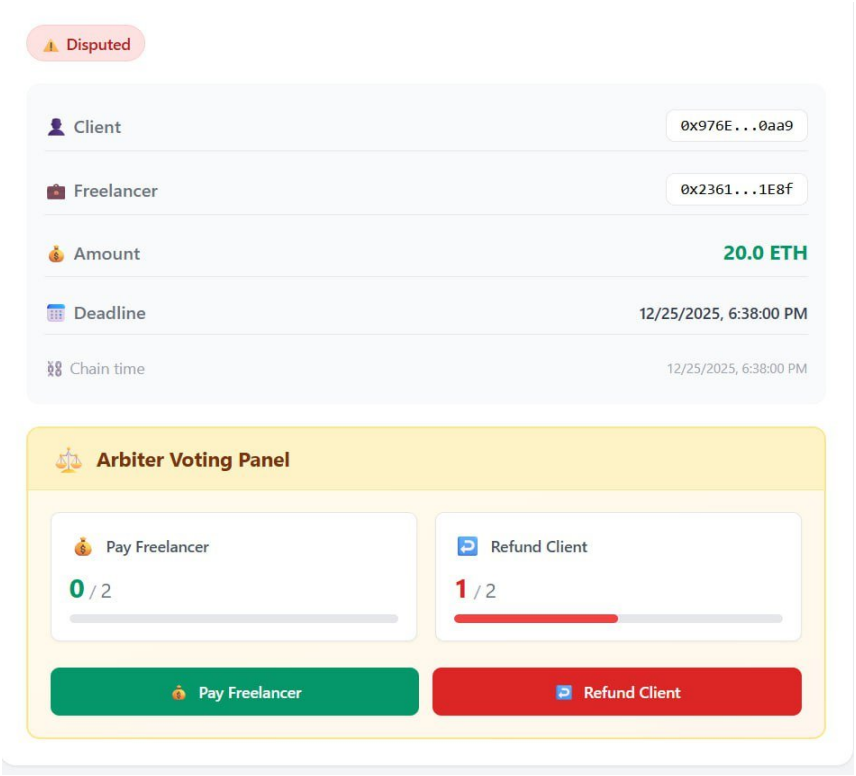


Cuộc tranh chấp bắt đầu, các trọng tài bắt đầu vào cuộc

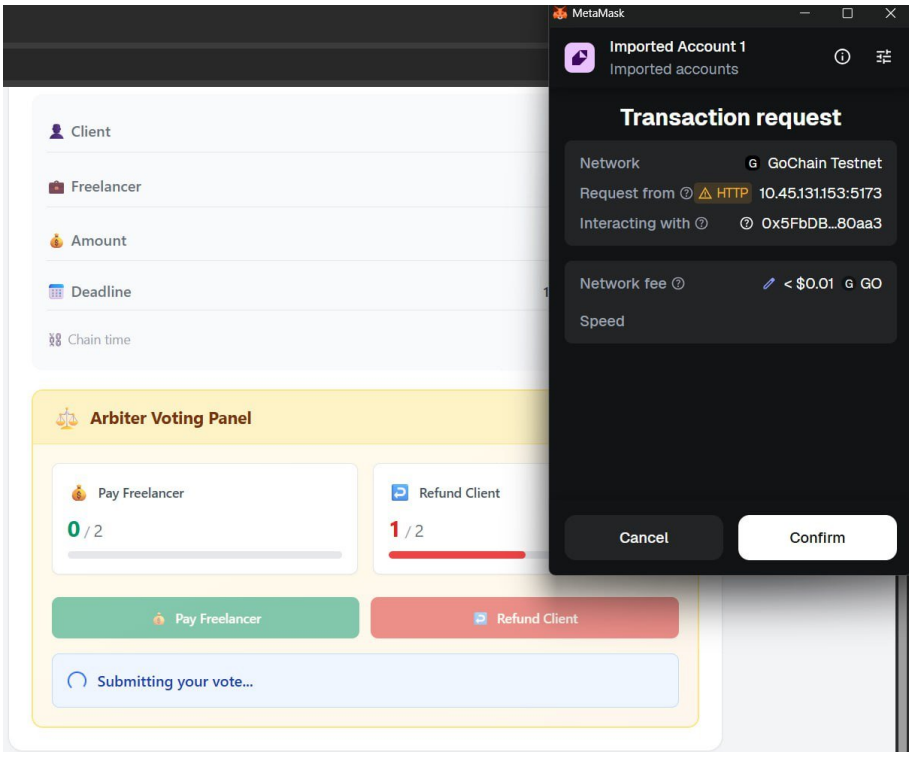


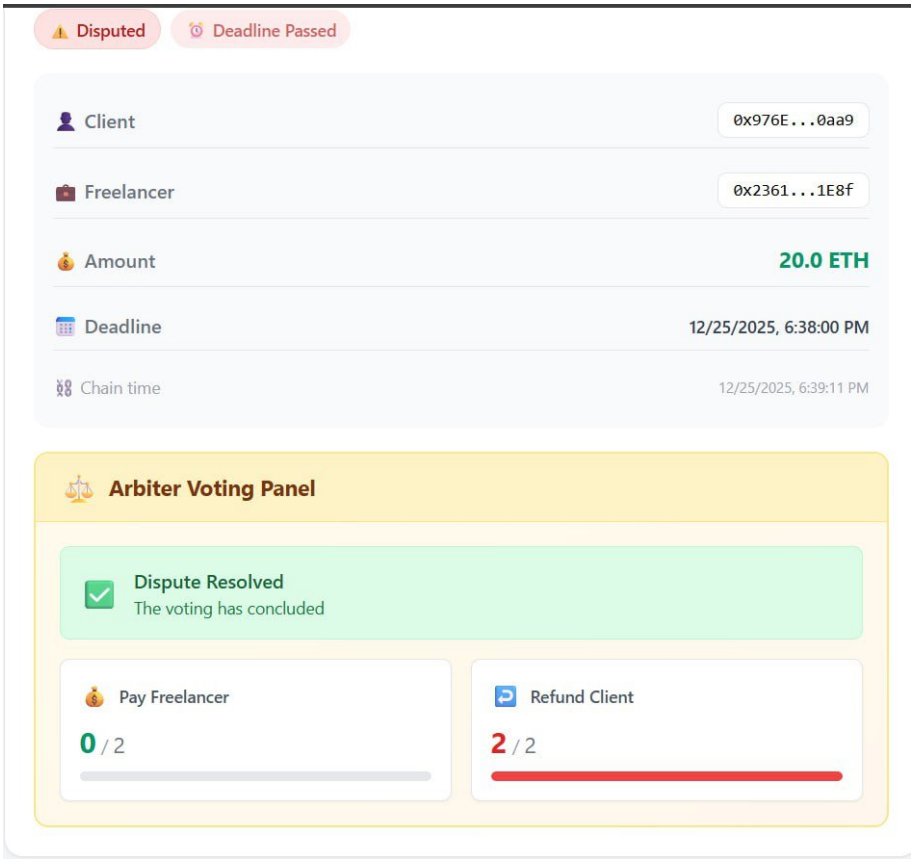
Trọng tài đầu tiên đã quyết định vote cho người thuê



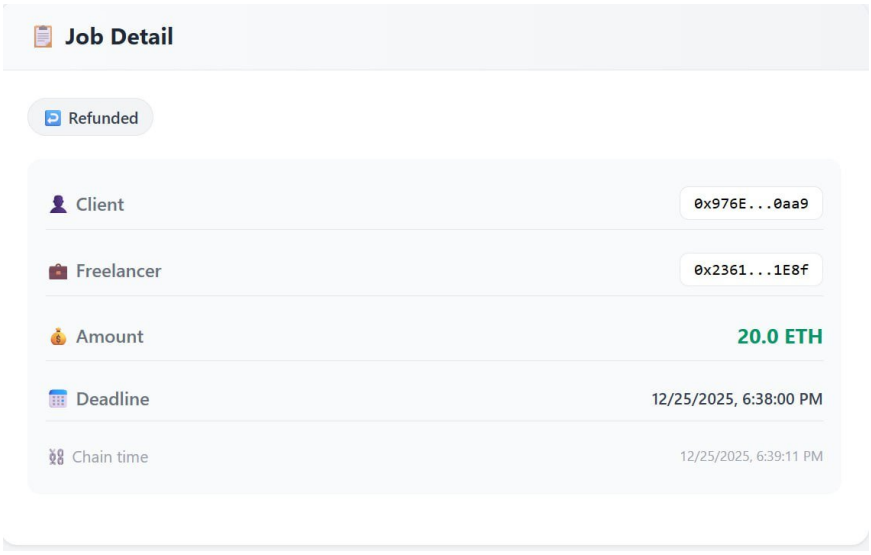


Còn trọng tài 2 cũng quyết định vote cho người thuê

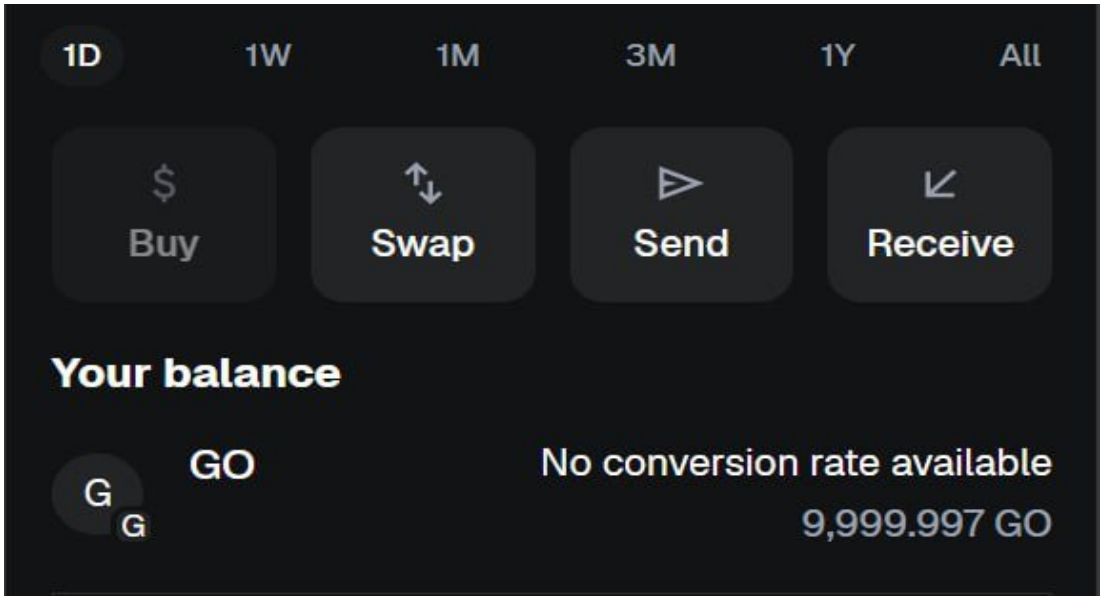




Vậy chỉ sau 2 lượt vote thì kết quả đã được quyết định và người thuê là người thắng cuộc tranh chấp này



Tài khoản người thuê được hoàn lại 20 ETH, chỉ mất một lượng nhỏ phí gas.


1D
1W
1M
3M
1Y
All
Buy
Swap
Send
Receive
Your balance
GO
No conversion rate available
9,999.997 GO

PHẦN 3: KẾT LUẬN

1. Kết luận

Đề tài *"Xây dựng Escrow DApp FreelanceTrust"* là một dự án tiêu biểu cho việc vận dụng kiến thức công nghệ Blockchain và phát triển ứng dụng phi tập trung vào giải quyết các bài toán thực tiễn trong môi trường kinh tế số. Trong quá trình thực hiện, nhóm đã tiếp cận và làm việc với nhiều thành phần cốt lõi của hệ sinh thái Ethereum như Smart Contract, cơ chế Escrow, mạng ngang hàng phi tập trung, cũng như các công nghệ Web3 phục vụ cho việc tương tác giữa người dùng và blockchain.

Việc xây dựng và triển khai Smart Contract bằng ngôn ngữ Solidity giúp nhóm hiểu rõ hơn về cách thức vận hành của các giao dịch phi tập trung, cơ chế quản lý trạng thái hợp đồng, xử lý thanh toán tự động và đảm bảo tính minh bạch, bất biến của dữ liệu trên blockchain. Bên cạnh đó, quá trình tích hợp giao diện Web3 với ví điện tử MetaMask thông qua thư viện Ethers.js giúp tiếp cận gần hơn với mô hình phát triển DApp hiện đại, nơi ứng dụng web truyền thống có thể tương tác trực tiếp với mạng blockchain mà không cần trung gian tập trung.

Trong quá trình triển khai, nhóm cũng phải đối mặt với nhiều thách thức kỹ thuật như thiết kế luồng xử lý Escrow hợp lý, đảm bảo an toàn cho tài sản ký quỹ, xử lý các tình huống tranh chấp thông qua cơ chế Arbiter, cũng như tối ưu trải nghiệm người dùng trong môi trường phi tập trung. Việc triển khai hệ thống trên mạng thử nghiệm Sepolia đã giúp nhóm đánh giá được tính khả thi, hiệu năng và mức độ an toàn của giải pháp trong điều kiện gần với thực tế. Thông qua dự án, bọn em không chỉ củng cố kiến thức chuyên môn về Blockchain và Smart Contract mà còn rèn luyện tư duy phân tích, khả năng thiết kế hệ thống phi tập trung, kỹ năng lập trình Web3 và làm việc nhóm..

Mặc dù vẫn còn những hạn chế nhất định và nhiều hướng mở rộng có thể tiếp tục phát triển, đề tài đã hoàn thành các mục tiêu đề ra và mang lại những kinh nghiệm thực tiễn quý báu. Kết quả của dự án không chỉ đáp ứng yêu cầu học thuật của môn học mà còn tạo tiền đề cho việc nghiên cứu, phát triển và ứng dụng các giải pháp DApp Escrow trong môi trường thực tế sau này.

2. Kết quả đạt được

Thông qua quá trình triển khai và thực nghiệm hệ thống, các kết quả thu được cho thấy mô hình Smart Contract escrow được đề xuất hoạt động đúng theo thiết kế và đáp

ứng các yêu cầu chức năng đã đặt ra. Việc xây dựng và triển khai thành công các hợp đồng thông minh, kết hợp với giao diện người dùng dựa trên nền tảng web, đã chứng minh tính khả thi của việc áp dụng Smart Contract vào bài toán quản lý và thanh toán giao dịch freelance trong môi trường phi tập trung.

Về mặt triển khai kỹ thuật, các Smart Contract được xây dựng với cấu trúc rõ ràng, phân tách hợp lý giữa các thành phần chức năng và cơ chế xử lý tranh chấp. Việc triển khai thử nghiệm trên mạng local cho phép kiểm tra đầy đủ vòng đời của một giao dịch freelance, từ khởi tạo hợp đồng, nạp tiền ký quỹ, thực hiện công việc cho đến giải ngân hoặc hoàn tiền. Kết quả cho thấy các hợp đồng hoạt động ổn định, không xảy ra lỗi nghiêm trọng trong quá trình thực thi các hàm chức năng chính.

Đối với giao diện người dùng, việc tích hợp thư viện Ethers.js và kết nối với ví MetaMask đã giúp hệ thống tương tác hiệu quả với blockchain. Người dùng có thể thực hiện các thao tác như tạo giao dịch mới, nạp tiền ký quỹ, xác nhận hoàn thành công việc hoặc khởi tạo tranh chấp một cách trực quan. Quá trình ký giao dịch và gửi yêu cầu lên blockchain được thực hiện thành công, đảm bảo tính xác thực và an toàn cho các hành động của người dùng.

Trong kịch bản giao dịch đúng hẹn và được thanh toán, hệ thống thể hiện khả năng tự động hóa toàn bộ quy trình thanh toán. Sau khi freelancer hoàn thành công việc và client xác nhận, khoản tiền ký quỹ được giải ngân chính xác và kịp thời cho freelancer mà không cần sự can thiệp của bên trung gian. Điều này cho thấy Smart Contract đã thực thi đúng logic nghiệp vụ và đảm bảo quyền lợi cho cả hai bên.

Ở kịch bản giao dịch trễ hẹn nhưng vẫn được thanh toán, hệ thống vẫn xử lý đúng theo các điều kiện đã được thiết lập trước. Dù thời gian hoàn thành bị kéo dài, quy trình xác nhận và giải ngân vẫn diễn ra chính xác khi client chấp thuận kết quả công việc. Kết quả này cho thấy hệ thống có khả năng linh hoạt trong xử lý các tình huống thực tế, đồng thời không làm gián đoạn hoặc sai lệch luồng xử lý chính của giao dịch.

Trong các trường hợp phát sinh tranh chấp, hệ thống đã chứng minh được vai trò của cơ chế phân xử dựa trên hợp đồng DisputeMultiSig. Khi xảy ra tranh chấp trong cả hai trường hợp giao dịch trễ hẹn hoặc đúng hẹn, Smart Contract escrow chuyển trạng thái giao dịch sang chế độ tranh chấp và kích hoạt quy trình bỏ phiếu của các arbiter. Kết quả bỏ phiếu được ghi nhận minh bạch trên blockchain và được sử dụng để quyết

định việc phân bổ tiền ký quỹ, đảm bảo rằng kết quả cuối cùng không phụ thuộc vào ý chí của một bên đơn lẻ.

Tổng hợp kết quả từ bốn kịch bản thực nghiệm cho thấy hệ thống đáp ứng đầy đủ các yêu cầu về tự động hóa, minh bạch và công bằng trong quản lý giao dịch freelance. Việc sử dụng Smart Contract giúp giảm thiểu rủi ro gian lận, hạn chế tranh chấp ngoài chuỗi và loại bỏ sự phụ thuộc vào bên trung gian tập trung. Qua đó, hệ thống đã chứng minh được tiềm năng ứng dụng thực tế của Smart Contract trong các nền tảng freelance và các mô hình giao dịch tương tự.

3. Ưu và nhược điểm

Hệ thống Smart Contract Escrow được xây dựng và thực nghiệm cho thấy nhiều ưu điểm nổi bật so với các mô hình giao dịch freelance truyền thống. Trước hết, việc sử dụng Smart Contract giúp tự động hóa toàn bộ quy trình quản lý giao dịch và thanh toán, từ khâu ký quỹ đến giải ngân hoặc hoàn tiền. Nhờ đó, hệ thống giảm thiểu sự phụ thuộc vào bên trung gian, hạn chế rủi ro gian lận và nâng cao mức độ tin cậy giữa các bên tham gia. Bên cạnh đó, tính minh bạch và khả năng kiểm chứng là một ưu điểm quan trọng của hệ thống. Mọi giao dịch, trạng thái công việc và quyết định phân xử đều được ghi nhận công khai trên blockchain, cho phép các bên liên quan theo dõi và xác minh độc lập. Điều này đặc biệt có ý nghĩa trong các tình huống phát sinh tranh chấp, khi kết quả xử lý không phụ thuộc vào quyết định chủ quan của một cá nhân hay tổ chức trung gian.

Hệ thống cũng thể hiện tính linh hoạt thông qua việc hỗ trợ nhiều kịch bản giao dịch khác nhau, bao gồm các trường hợp hoàn thành đúng hạn, trễ hạn và phát sinh tranh chấp. Cơ chế DisputeMultiSig với sự tham gia của nhiều arbiter góp phần nâng cao tính công bằng trong quá trình phân xử, đồng thời giảm nguy cơ thiên vị. Ngoài ra, việc tách biệt các giao dịch thành các hợp đồng escrow độc lập giúp hệ thống dễ mở rộng và hạn chế ảnh hưởng lan truyền khi một giao dịch gặp sự cố.

Mặc dù đạt được các kết quả tích cực, hệ thống vẫn tồn tại một số hạn chế nhất định. Trước hết, chi phí giao dịch (gas fee) là một yếu tố cần được cân nhắc khi triển khai trên các mạng blockchain công khai. Trong các kịch bản có nhiều thao tác hoặc phát sinh tranh chấp phức tạp, chi phí thực thi Smart Contract có thể tăng lên đáng kể, ảnh hưởng đến trải nghiệm người dùng. Ngoài ra, Smart Contract sau khi triển khai gần như không thể chỉnh sửa, do đó các lỗi logic hoặc thiết kế nếu tồn tại sẽ gây khó khăn trong việc

khắc phục. Điều này đòi hỏi quá trình phân tích yêu cầu, lập trình và kiểm thử phải được thực hiện một cách nghiêm ngặt, làm tăng độ phức tạp và thời gian phát triển hệ thống. Việc sử dụng ví điện tử như MetaMask và tương tác trực tiếp với blockchain có thể gây khó khăn cho những người dùng chưa có nhiều kinh nghiệm về công nghệ blockchain. Bên cạnh đó, cơ chế tranh chấp dựa trên arbiter vẫn mang yếu tố con người, do đó chất lượng và tính trung lập của arbiter có thể ảnh hưởng đến kết quả phân xử nếu không có cơ chế lựa chọn và giám sát phù hợp.

4. Hướng phát triển

Mặc dù hệ thống Smart Contract escrow cho giao dịch freelance đã đạt được các kết quả khả quan trong giai đoạn thực nghiệm, vẫn còn nhiều hướng phát triển tiềm năng nhằm nâng cao tính hoàn thiện, khả năng mở rộng và tính ứng dụng thực tế của hệ thống trong tương lai.

Trước hết, một hướng phát triển quan trọng là tối ưu chi phí giao dịch và hiệu năng thực thi của Smart Contract. Việc nghiên cứu và áp dụng các kỹ thuật tối ưu hóa mã nguồn, cũng như triển khai hệ thống trên các giải pháp mở rộng như Layer 2 hoặc sidechain, có thể giúp giảm đáng kể chi phí gas và độ trễ giao dịch. Điều này đặc biệt cần thiết khi hệ thống được sử dụng ở quy mô lớn với số lượng giao dịch và tranh chấp tăng cao.

Bên cạnh đó, hệ thống có thể được mở rộng về mặt chức năng bằng cách hỗ trợ các mô hình thanh toán linh hoạt hơn. Thay vì chỉ giải ngân toàn bộ khoản ký quỹ sau khi hoàn thành công việc, Smart Contract có thể được thiết kế để hỗ trợ thanh toán theo từng giai đoạn (milestone-based payment), giữ lại một phần tiền làm bảo chứng chất lượng hoặc tích hợp các điều khoản phạt – thưởng tự động dựa trên tiến độ và mức độ hoàn thành công việc.

Về mặt trải nghiệm người dùng, hệ thống cần được phát triển theo hướng thân thiện hơn đối với người không chuyên về blockchain. Điều này bao gồm việc đơn giản hóa quy trình kết nối ví, giảm số lượng thao tác ký giao dịch và cung cấp các hướng dẫn trực quan trong giao diện người dùng. Đồng thời, việc tích hợp các giải pháp xác thực và phục hồi tài khoản an toàn hơn có thể giúp giảm rủi ro mất quyền truy cập ví đối với người dùng.

Cuối cùng, để tăng khả năng ứng dụng thực tế, hệ thống có thể được mở rộng sang các lĩnh vực khác ngoài freelance, chẳng hạn như quản lý hợp đồng dịch vụ, chuỗi cung ứng, cho thuê tài sản số hoặc các giao dịch thương mại điện tử có yêu cầu ký quỹ. Việc áp dụng cùng một mô hình Smart Contract escrow cho nhiều bối cảnh khác nhau sẽ góp phần khẳng định tính tổng quát và tiềm năng của giải pháp được đề xuất.

TÀI LIỆU THAM KHẢO

[1]. Đình Long (2025). *Ethereum là gì? Những điều bạn cần biết trước khi đầu tư vào thị trường tiền điện tử*, truy cập 21/12/2025. Đường dẫn: ***https://fptshop.com.vn/tin-tuc/danh-gia/ethereum-la-gi-179101***

[2]. Invesco (2025). *What Is Ethereum, and how does this digital asset work?*, truy cập 21/12/2025. Đường dẫn: ***https://www.invesco.com/us/en/insights/what-is-ethereum.html***

[3]. Ethereum Foundation (2023). *Ethereum Developer Documentation*, truy cập 21/12/2025. Đường dẫn: ***https://ethereum.org/developers/docs/***

[4]. Solidity Team (2024). *Solidity Documentation – Version 0.8.33*, truy cập 21/12/2025. Đường dẫn: ***https://docs.soliditylang.org/en/v0.8.33/***

[5]. Caroline Banton (2025). *Understanding Escrow: Protecting Parties in Financial Transactions*, truy cập 21/12/2025. Đường dẫn: ***https://www.investopedia.com/terms/e/escrow.asp***

[6]. Michael Summer (2024). *Blockchain-Based Escrow Services for Creative Works: A Primer*, truy cập 22/12/2025. Đường dẫn: ***https://www.scoredetect.com/blog/posts/blockchain-based-escrow-services-for-creative-works-a-primer***

[7]. James Howell (2024). *Know Everything About Escrow Smart Contract*, truy cập 22/12/2025. Đường dẫn: ***https://101blockchains.com/escrow-smart-contract/***

[8]. Zheng, Z., Xie, S., Dai, H., Chen, X., & Wang, H. (2017). An overview of blockchain technology: Architecture, consensus, and future trends. *Proceedings of the 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 558–566.

[9]. Yaga, D., & Mell, P. (2025). *A Security Perspective on the Web3 Paradigm* (NIST IR 8475), truy cập 23/12/2025. Đường dẫn: ***https://doi.org/10.6028/NIST.IR.8475***

[10]. Voshmgir, S. (2020). *Token Economy: How the Web3 reinvents the Internet*. Token Kitchen.

[11]. Ethereum Foundatio (2024). *Introduction to Web3*, truy cập 23/12/2025. Đường dẫn: ***https://ethereum.org/en/web3/***

[12]. Moore, R. (2024). *Ethers.js Documentation: The Ethers Project*, truy cập 23/12/2025. Đường dẫn: ***https://docs.ethers.org/v6/***

[13]. Ethereum Foundation (2024). *JavaScript Client Libraries*, truy cập 23/12/2025. Đường dẫn: ***https://ethereum.org/en/developers/docs/apis/javascript/***