**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**Báo cáo bài tập lớn**

**Đề tài:  Ứng dụng cho vay phi tập trung trên nền tảng blockchain ethereum**

**Tô Duy Tường**

[tuong.td252043M@sis.hust.edu.vn](mailto:tuong.td252043M@sis.hust.edu.vn)

**Nguyễn Ngọc Ánh**

[Anh.NN252032M@sis.hust.edu.vn](mailto:Anh.NN252032M@sis.hust.edu.vn)

**Cao Thành Duy**

[Duy.CT252038M@sis.hust.edu.vn](mailto:Duy.CT252038M@sis.hust.edu.vn)

**Module Chuyển đổi số**

**Chuyên ngành Công nghệ thông tin (KT)**

|  |  |
| --- | --- |
| **Giảng viên hướng dẫn:** | TS. Trịnh Tuấn Đạt |
| **Bộ môn:** | Công nghệ và nền tảng chuỗi khối |
| **Khoa:** | Khoa học máy tính |

**Lời cảm ơn**

Chúng em xin chân thành cảm ơn thầy Trịnh Tuấn Đạt đã tận tình chỉ bảo, định hướng cũng như giải đáp các thắc mắc để chúng em có thể hoàn thành bài tập lớn này một cách trọn vẹn nhất.

**Tóm tắt nội dung bài tập lớn**

Bài tập lớn xây dựng hệ thống Lending DApp (ứng dụng cho vay phi tập trung) trên blockchain Ethereum, kết nối người cho vay và người vay qua smart contract. Hệ thống gồm hai phần: smart contract (Solidity) triển khai trên Hardhat localhost và giao diện web (Nuxt 3) với PrimeVue và Tailwind CSS. Người dùng có thể gửi ETH làm tài sản thế chấp, vay token (MCK) với tỷ lệ LTV 80%, trả nợ, rút tài sản, và cung cấp thanh khoản để nhận lãi. Hệ thống sử dụng Linear Interest Rate Model để điều chỉnh lãi suất theo tỷ lệ sử dụng pool, tính toán Health Factor để đảm bảo an toàn, và hỗ trợ thanh lý tự động khi Health Factor < 1. Giao diện tích hợp MetaMask qua Ethers.js để kết nối ví và thực hiện giao dịch, hỗ trợ đa ngôn ngữ (Tiếng Việt/English) và dark mode. Hệ thống được thiết kế phi tập trung, minh bạch, không cần trung gian, đảm bảo tính bất biến của dữ liệu trên blockchain và hỗ trợ quản lý tài sản DeFi.

Sinh viên thực hiện

Ký và ghi rõ họ tên

**MỤC LỤC**

[CHƯƠNG 1. TỔNG QUAN HỆ THỐNG](#_Toc219920432)

[1.1 Giới thiệu](#_Toc219920433)

[1.2 Mục tiêu](#_Toc219920434)

[1.3 Phạm vi](#_Toc219920435)

[1.4 Cấu trúc báo cáo [1] [2]](#_Toc219920436)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT](#_Toc219920437)

[2.1 Tổng quan về Blockchain](#_Toc219920438)

[2.2 Smart Contract và Solidity](#_Toc219920439)

[2.3 Ứng dụng phi tập trung (DApp)](#_Toc219920440)

[2.4 DeFi và Lending Protocol](#_Toc219920441)

[2.5 Mô hình lãi suất động (Interest Rate Model)](#_Toc219920442)

[2.6 Hệ thống thế chấp và Health Factor](#_Toc219920443)

[2.7 Web3 và tích hợp ví (MetaMask, Ethers.js)](#_Toc219920444)

[CHƯƠNG 3. PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG](#_Toc219920445)

[3.1 Yêu cầu hệ thống](#_Toc219920446)

[3.2 Kiến trúc tổng thể hệ thống](#_Toc219920447)

[3.3 Phân tích các actor và use cases](#_Toc219920448)

[3.4 Thiết kế Smart Contract (LendingPool)](#_Toc219920449)

[3.5 Thiết kế giao diện người dùng](#_Toc219920450)

[CHƯƠNG 4. TRIỂN KHAI HỆ THỐNG](#_Toc219920451)

[4.1 Môi trường triển khai](#_Toc219920452)

[4.2 Triển khai Smart Contract (Hardhat, Solidity)](#_Toc219920453)

[4.3 Triển khai các chức năng cốt lõi (Deposit, Borrow, Repay, Withdraw)](#_Toc219920454)

[4.4 Triển khai hệ thống lãi suất động](#_Toc219920455)

[4.5 Triển khai cơ chế thanh lý (Liquidation)](#_Toc219920456)

[4.6 Tích hợp MetaMask và xử lý giao dịch](#_Toc219920457)

[CHƯƠNG 5. KẾT QUẢ](#_Toc219920458)

[5.1 Kết quả triển khai](#_Toc219920459)

[5.2 Các vấn đề gặp phải và giải pháp](#_Toc219920460)

[CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN](#_Toc219920461)

[a. Hạn chế của đồ án](#_Toc219920462)

[b. Hướng phát triển](#_Toc219920463)

# TỔNG QUAN HỆ THỐNG

## Giới thiệu

Trong thời đại công nghệ số, blockchain và tài chính phi tập trung (DeFi) đang thay đổi cách tiếp cận các dịch vụ tài chính truyền thống. DeFi cho phép thực hiện các giao dịch tài chính mà không cần trung gian, tăng tính minh bạch và giảm chi phí.

Lending Protocol là một trong những ứng dụng DeFi phổ biến, cho phép người dùng vay và cho vay tài sản mã hóa một cách tự động thông qua smart contract. Khác với hệ thống truyền thống, các giao thức này hoạt động phi tập trung, minh bạch và không cần KYC phức tạp.

Tuy nhiên, việc xây dựng một lending protocol an toàn và hiệu quả đòi hỏi hiểu rõ về smart contract, cơ chế quản lý rủi ro, mô hình lãi suất động và tích hợp với các công cụ Web3. Điều này gây khó khăn cho việc phát triển và triển khai các ứng dụng DeFi hoàn chỉnh.

Đồ án này tập trung xây dựng một hệ thống Lending DApp hoàn chỉnh, cho phép người dùng thế chấp ETH để vay token, cung cấp thanh khoản để nhận lãi, và tự động quản lý rủi ro thông qua cơ chế thanh lý. Hệ thống được xây dựng trên nền tảng Ethereum với smart contract được viết bằng Solidity và giao diện web hiện đại sử dụng Nuxt 3.

## Mục tiêu

Mục tiêu chính của đồ án bao gồm:

1. **Xây dựng Smart Contract:** Thiết kế và triển khai smart contract LendingPool trên Ethereum, hỗ trợ các chức năng cốt lõi: deposit, borrow, repay, withdraw, liquidate và supply.
2. **Triển khai mô hình lãi suất động:** Xây dựng Linear Interest Rate Model điều chỉnh lãi suất theo tỷ lệ sử dụng pool (utilization rate), đảm bảo cân bằng cung cầu và khuyến khích thanh khoản.
3. **Quản lý rủi ro:** Triển khai hệ thống Health Factor và cơ chế thanh lý tự động để đảm bảo an toàn vốn cho giao thức, với tỷ lệ LTV (Loan-to-Value) 80% và bonus thanh lý 10%.
4. **Xây dựng giao diện người dùng:** Phát triển dashboard web hiện đại với Nuxt 3, PrimeVue và Tailwind CSS, hỗ trợ kết nối MetaMask, hiển thị thông tin thời gian thực và thực hiện giao dịch.
5. **Tích hợp Web3:** Tích hợp Ethers.js để kết nối frontend với blockchain, xử lý giao dịch, quản lý trạng thái ví và xử lý lỗi một cách trực quan.

## Phạm vi

Đồ án tập trung vào các phạm vi sau:

**Trong phạm vi:**

* Xây dựng smart contract LendingPool với các chức năng: deposit ETH, borrow token (MCK), repay, withdraw, liquidate và supply token
* Triển khai Linear Interest Rate Model với cơ chế tính lãi theo thời gian thực (per-second accrual)
* Hệ thống quản lý rủi ro với Health Factor và cơ chế thanh lý tự động
* Giao diện web dashboard với các trang: Overview, Market, My Assets, Liquidation và Settings
* Tích hợp MetaMask và xử lý giao dịch qua Ethers.js
* Hỗ trợ đa ngôn ngữ (Tiếng Việt/English) và dark mode
* Triển khai trên Hardhat localhost network để phát triển và kiểm thử

**Ngoài phạm vi:**

* Triển khai trên mainnet Ethereum
* Oracle Chainlink hoặc các oracle phức tạp
* Audit bảo mật chuyên nghiệp

## Cấu trúc báo cáo [1] [2]

Báo cáo được tổ chức thành 6 chương:

* **Chương 1**: Giới thiệu tổng quan
* **Chương 2**: Trình bày cơ sở lý thuyết và các công nghệ liên quan
* **Chương 3**: Phân tích yêu cầu và thiết kế hệ thống
* **Chương 4**: Triển khai hệ thống
* **Chương 5**: Trình bày kết quả và đánh giá
* **Chương 6**: Đưa ra kết luận và hướng phát triển

# CƠ SỞ LÝ THUYẾT

## Tổng quan về Blockchain

### Khái niệm Blockchain

Blockchain là một công nghệ sổ cái phân tán (Distributed Ledger Technology - DLT) cho phép lưu trữ và truyền tải thông tin dưới dạng các khối (blocks) được liên kết với nhau bằng mã hóa. Mỗi khối chứa thông tin về các giao dịch, timestamp và hash của khối trước đó, tạo thành một chuỗi không thể thay đổi.

### Đặc điểm của Blockchain

* **Phi tập trung (Decentralization)**: Không có cơ quan trung ương kiểm soát, dữ liệu được lưu trữ trên nhiều node
* **Bất biến (Immutability)**: Dữ liệu đã được ghi vào blockchain không thể sửa đổi hoặc xóa bỏ
* **Minh bạch (Transparency)**: Tất cả giao dịch đều công khai và có thể kiểm tra
* **Bảo mật cao**: Sử dụng mã hóa và cơ chế đồng thuận để đảm bảo tính toàn vẹn dữ liệu
* **Không cần tin cậy (Trustless)**: Các bên có thể giao dịch mà không cần tin tưởng lẫn nhau

### Ethereum Blockchain

Ethereum là một nền tảng blockchain mã nguồn mở hỗ trợ smart contract. Khác với Bitcoin chỉ tập trung vào chuyển tiền, Ethereum cho phép xây dựng các ứng dụng phi tập trung (DApp) trên nền tảng của nó.

* **EVM (Ethereum Virtual Machine)**: Máy ảo thực thi smart contract
* **Gas**: Đơn vị đo chi phí tính toán, người dùng phải trả phí gas cho mỗi giao dịch
* **ETH**: Đồng tiền mã hóa gốc của Ethereum, dùng để trả phí giao dịch

## Smart Contract và Solidity

### Khái niệm Smart Contract

Smart Contract (hợp đồng thông minh) là các chương trình tự động thực thi các điều khoản được mã hóa sẵn khi đáp ứng các điều kiện nhất định. Chúng được triển khai trên blockchain và chạy trên EVM, đảm bảo tính minh bạch và không thể thay đổi.

### Ngôn ngữ Solidity

Solidity là ngôn ngữ lập trình hướng đối tượng được thiết kế đặc biệt để viết smart contract trên Ethereum và các blockchain tương thích EVM.

**Đặc điểm:**

* Cú pháp tương tự JavaScript và C++
* Hỗ trợ kiểu dữ liệu: uint256, address, bool, mapping, array
* Các modifier như public, private, internal, external để kiểm soát truy cập
* Events để ghi log và thông báo cho frontend
* Hỗ trợ inheritance và libraries

**Ví dụ cơ bản:**

*pragma solidity ^0.8.0;*

*contract SimpleStorage {*

*uint256 public storedData;*

*function set(uint256 x) public {*

*storedData = x;*

*}*

*function get() public view returns (uint256) {*

*return storedData;*

*}*

*}*

### OpenZeppelin Contracts

OpenZeppelin là thư viện các smart contract đã được audit và kiểm thử, cung cấp các thành phần bảo mật và chuẩn:

* **ReentrancyGuard**: Bảo vệ khỏi tấn công reentrancy
* **Ownable**: Quản lý quyền sở hữu contract
* **ERC20**: Chuẩn token có thể thay thế
* **SafeMath**: Xử lý số học an toàn (tích hợp trong Solidity 0.8+)

## Ứng dụng phi tập trung (DApp)

### Khái niệm DApp

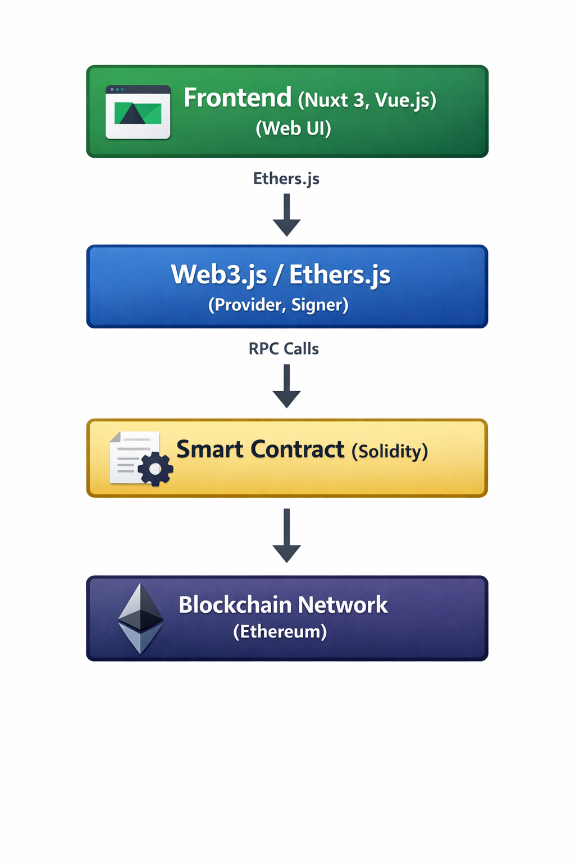
DApp (Decentralized Application) là ứng dụng chạy trên mạng blockchain phi tập trung, không phụ thuộc vào máy chủ trung tâm. DApp thường gồm:

* **Frontend**: Giao diện người dùng (web/mobile)
* **Smart Contract**: Logic nghiệp vụ trên blockchain
* **Blockchain**: Lưu trữ dữ liệu và trạng thái

### So sánh App truyền thống vs DApp

| **Đặc điểm** | **App truyền thống** | **DApp** |
| --- | --- | --- |
| Kiến trúc | Client-Server | Peer-to-Peer |
| Dữ liệu | Lưu trên server tập trung | Lưu trên blockchain phân tán |
| Quyền kiểm soát | Tập trung | Phi tập trung |
| Chi phí | Phí hosting/server | Phí gas |
| Bảo mật | Phụ thuộc server | Dựa trên mã hóa blockchain |

### Kiến trúc Dapp



## DeFi và Lending Protocol

### Tổng quan về DeFi

DeFi (Decentralized Finance) là hệ sinh thái tài chính được xây dựng trên blockchain, cung cấp các dịch vụ tài chính truyền thống (cho vay, vay, giao dịch) mà không cần trung gian.**Các ứng dụng DeFi phổ biến:**

* **Lending/Borrowing**: Aave, Compound, MakerDAO
* **DEX (Decentralized Exchange)**: Uniswap, SushiSwap
* **Stablecoin**: DAI, USDC
* **Yield Farming**: Cung cấp thanh khoản để nhận lợi nhuận

### Lending Protocol

Lending Protocol cho phép người dùng:

* **Cho vay (Lend/Supply)**: Gửi tài sản vào pool để nhận lãi
* **Vay (Borrow)**: Thế chấp tài sản để vay tài sản khác
* **Thanh lý (Liquidation)**: Thanh lý các vị thế rủi ro để bảo vệ protocol

**Cơ chế hoạt động:**

1. Người cho vay gửi tài sản vào pool
2. Người vay thế chấp tài sản và vay từ pool
3. Lãi suất được tính dựa trên tỷ lệ sử dụng pool
4. Khi giá trị thế chấp giảm, vị thế có thể bị thanh lý

### Các thành phần chính

* **Liquidity Pool**: Pool chứa tài sản cho vay
* **Collateral**: Tài sản thế chấp
* **Loan-to-Value (LTV)**: Tỷ lệ cho vay so với giá trị thế chấp (ví dụ: 80%)
* **Interest Rate Model**: Mô hình tính lãi suất động
* **Liquidation**: Cơ chế thanh lý khi vị thế không còn an toàn

## Mô hình lãi suất động (Interest Rate Model)

### Khái niệm

Interest Rate Model là cơ chế tự động điều chỉnh lãi suất cho vay và vay dựa trên tỷ lệ sử dụng pool (utilization rate). Khi nhu cầu vay tăng, lãi suất tăng để khuyến khích người cho vay và hạn chế người vay.

### Utilization Rate

Utilization Rate (U) được tính bằng công thức:

* **U = 0%**: Không có ai vay, lãi suất thấp
* **U = 100%**: Pool hết thanh khoản, lãi suất cao nhất

### Linear Interest Rate Model

Mô hình tuyến tính được sử dụng trong project này:

**Borrow Rate:**

**Supply Rate:**

Trong đó:

* **Base Rate**: Lãi suất cơ bản (~5% APY)
* **Multiplier**: Hệ số nhân (~20% APY tại 100% utilization)
* **U**: Utilization rate (0-1)

**Ví dụ:**

* Nếu U = 50%, Base = 5%, Multiplier = 20%
* Borrow Rate = 5% + (20% × 0.5) = 15% APY
* Supply Rate = 15% × 0.5 = 7.5% APY

### Cơ chế tính lãi

Lãi được tính theo thời gian thực (per-second accrual) và được cộng dồn mỗi khi có giao dịch:

// Tính lãi theo thời gian

interestFactor = ratePerSecond × timeDelta

newBalance = oldBalance × (1 + interestFactor)

## Hệ thống thế chấp và Health Factor

### Loan-to-Value (LTV) Ratio

LTV là tỷ lệ giữa số tiền vay và giá trị tài sản thế chấp. Trong project này, LTV được set ở mức 80%, nghĩa là người dùng chỉ có thể vay tối đa 80% giá trị thế chấp.

**Ví dụ:**

* Thế chấp: 10 ETH
* Giá ETH: 2000 MCK/ETH
* Giá trị thế chấp: 20,000 MCK
* Số tiền vay tối đa: 20,000 × 80% = 16,000 MCK

### Health Factor

Health Factor (HF) là chỉ số đo mức độ an toàn của vị thế vay:

**Ý nghĩa:**

* **HF > 1.0**: Vị thế an toàn
* **HF = 1.0**: Vị thế ở ngưỡng an toàn
* **HF < 1.0**: Vị thế có thể bị thanh lý

### Cơ chế Thanh lý (Liquidation)

Khi Health Factor < 1.0, vị thế có thể bị thanh lý:

1. **Liquidator** (người thanh lý) trả nợ thay cho borrower
2. Liquidator nhận lại collateral tương ứng + **bonus** (thường 5-10%)
3. Borrower mất một phần collateral nhưng được giải phóng khỏi nợ

**Ví dụ:**

* Debt: 16,000 MCK
* Collateral: 10 ETH (giá 2000 MCK/ETH)
* Liquidator trả: 16,000 MCK
* Liquidator nhận: 8 ETH + 0.8 ETH bonus = 8.8 ETH
* Borrower còn lại: 1.2 ETH

## Web3 và tích hợp ví (MetaMask, Ethers.js)

### Web3 là gì?

Web3 là thế hệ tiếp theo của internet, tập trung vào tính phi tập trung và quyền sở hữu dữ liệu của người dùng. Trong bối cảnh blockchain, Web3 đề cập đến các công cụ và thư viện cho phép ứng dụng web tương tác với blockchain.

### MetaMask

MetaMask là ví tiền mã hóa dạng browser extension, cho phép:

* Lưu trữ và quản lý private key
* Ký giao dịch
* Kết nối với các DApp
* Quản lý nhiều tài khoản và mạng

**Cách hoạt động:**

1. Người dùng cài đặt MetaMask extension
2. DApp yêu cầu kết nối qua window.ethereum
3. Người dùng phê duyệt kết nối
4. DApp có thể đọc địa chỉ ví và gửi giao dịch

### Ethers.js

Ethers.js là thư viện JavaScript để tương tác với Ethereum blockchain.

**Các thành phần chính:**

**Provider**: Kết nối với blockchain node

*const provider = new ethers.JsonRpcProvider("http://localhost:8545");*

**Signer**: Đại diện cho tài khoản có thể ký giao dịch

*const signer = await provider.getSigner();*

*const address = await signer.getAddress();*

**Contract**: Tương tác với smart contract

*const contract = new ethers.Contract(*

*contractAddress,*

*abi,*

*signer*

*);*

*const result = await contract.deposit({ value: ethers.parseEther("1.0") });*

**2.7.4 Luồng giao dịch trong DApp**

1. **Kết nối ví**: window.ethereum.request({ method: 'eth\_requestAccounts' })
2. **Tạo Provider**: new ethers.BrowserProvider(window.ethereum)
3. **Lấy Signer**: await provider.getSigner()
4. **Tạo Contract instance**: new ethers.Contract(address, abi, signer)
5. **Gọi function**: await contract.functionName(params)
6. **Chờ confirmation**: await tx.wait()
7. **Cập nhật UI**: Đọc lại dữ liệu từ contract

**2.7.5 Xử lý lỗi và Events**

* **User Rejection**: Người dùng từ chối giao dịch (code 4001)
* **Insufficient Funds**: Không đủ ETH để trả gas
* **Contract Revert**: Smart contract từ chối giao dịch
* **Network Error**: Lỗi kết nối mạng

**Events**: Smart contract emit events để frontend có thể lắng nghe:

*event Deposit(address indexed user, uint256 amount);*

*contract.on("Deposit", (user, amount) => {*

*console.log(`User ${user} deposited ${amount}`);*

*});*

# PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## Yêu cầu hệ thống

### Yêu cầu chức năng

**RF1: Quản lý thế chấp (Collateral Management)**

* **RF1.1**: Người dùng có thể gửi ETH vào pool làm tài sản thế chấp
* **RF1.2**: Hệ thống lưu trữ số dư thế chấp của từng người dùng
* **RF1.3**: Người dùng có thể rút ETH thế chấp khi không còn nợ hoặc Health Factor >= 1

**RF2: Vay tài sản (Borrowing)**

* **RF2.1**: Người dùng có thể vay token (MCK) dựa trên giá trị thế chấp
* **RF2.2**: Hệ thống chỉ cho phép vay tối đa 80% giá trị thế chấp (LTV = 80%)
* **RF2.3**: Hệ thống kiểm tra thanh khoản pool trước khi cho vay
* **RF2.4**: Hệ thống tự động tính lãi suất theo thời gian thực

**RF3: Trả nợ (Repayment)**

* **RF3.1**: Người dùng có thể trả một phần hoặc toàn bộ nợ
* **RF3.2**: Hệ thống tự động cập nhật số dư nợ sau khi trả
* **RF3.3**: Sau khi trả hết nợ, người dùng có thể rút toàn bộ thế chấp

**RF4: Cung cấp thanh khoản (Supply Liquidity)**

* **RF4.1**: Người dùng có **thể** gửi token (MCK) vào pool để cung cấp thanh khoản
* **RF4.2**: Người cung cấp thanh khoản nhận lãi suất dựa trên tỷ lệ sử dụng pool
* **RF4.3**: Người dùng có thể rút token đã cung cấp (nếu pool có đủ thanh khoản)

**RF5: Thanh lý (Liquidation)**

* **RF5.1**: Khi Health Factor < 1.0, bất kỳ ai cũng có thể thanh lý vị thế
* **RF5.2**: Liquidator trả nợ thay cho borrower và nhận collateral + 10% bonus
* **RF5.3**: Hệ thống tự động tính toán số lượng collateral cần thanh lý

**RF6: Quản lý lãi suất (Interest Rate Management)**

* **RF6.1**: Hệ thống tự động tính lãi suất dựa trên Linear Interest Rate Model
* **RF6.2**: Lãi suất được cập nhật theo thời gian thực (per-second accrual)
* **RF6.3**: Admin có thể điều chỉnh tham số lãi suất (Base Rate, Multiplier)

**RF7: Quản lý giá (Price Management)**

* **RF7.1**: Admin có thể cập nhật giá ETH (mock oracle)
* **RF7.2**: Giá ETH được sử dụng để tính giá trị thế chấp và Health Factor

**RF8: Giao diện người dùng**

* **RF8.1**: Dashboard hiển thị tổng quan tài sản, nợ, Health Factor
* **RF8.2**: Trang Market hiển thị thông tin pool, lãi suất, utilization rate
* **RF8.3**: Trang My Assets hiển thị chi tiết vị thế của người dùng
* **RF8.4**: Trang Liquidation hiển thị danh sách vị thế có thể thanh lý
* **RF8.5**: Hỗ trợ đa ngôn ngữ (Tiếng Việt/English) và dark mode

### Yêu cầu phi chức năng

**RNF1: Bảo mật**

* **RNF1.1**: Smart contract phải được bảo vệ khỏi reentrancy attack (sử dụng ReentrancyGuard)
* **RNF1.2**: Tất cả các function quan trọng phải có input validation
* **RNF1.3**: Chỉ owner mới có thể cập nhật giá và tham số lãi suất
* **RNF1.4**: Kiểm tra Health Factor trước khi cho phép rút thế chấp

**RNF2: Hiệu năng**

* **RNF2.1**: Giao dịch phải được xử lý trong vòng 15 giây (trên localhost)
* **RNF2.2**: Frontend phải cập nhật dữ liệu real-time sau mỗi giao dịch
* **RNF2.3**: Tối ưu hóa gas cost cho các operations thường dùng

**RNF3: Tính khả dụng (Usability)**

* **RNF3.1**: Giao diện phải trực quan, dễ sử dụng
* **RNF3.2**: Hiển thị thông báo lỗi rõ ràng khi giao dịch thất bại
* **RNF3.3**: Hỗ trợ xem trước (preview) trước khi thực hiện giao dịch
* **RNF3.4**: Hiển thị trạng thái loading khi đang xử lý giao dịch

**RNF4: Tính mở rộng (Scalability)**

* **RNF4.1**: Kiến trúc cho phép dễ dàng thêm các loại tài sản mới
* **RNF4.2**: Smart contract có thể nâng cấp thông qua proxy pattern (tương lai)

**RNF5: Tính tin cậy (Reliability)**

* **RNF5.1**: Hệ thống phải xử lý được các edge cases (ví dụ: pool hết thanh khoản)
* **RNF5.2**: Có cơ chế fallback khi oracle không hoạt động
* **RNF5.3**: Events được emit đầy đủ để frontend có thể theo dõi

## Kiến trúc tổng thể hệ thống

### Sơ đồ kiến trúc

Ảnh có chứa văn bản, đồ điện tử, ảnh chụp màn hình, Trang web

Nội dung do AI tạo ra có thể không chính xác.

### Các thành phần chính

**1. Frontend Layer (Nuxt 3)**

* **Dashboard**: Trang chủ hiển thị tổng quan
* **Market**: Thông tin pool, lãi suất, utilization
* **My Assets**: Chi tiết vị thế của người dùng
* **Liquidation**: Danh sách vị thế có thể thanh lý
* **Settings**: Cài đặt ngôn ngữ, theme
* **Guide**: Hướng dẫn sử dụng

**2. Web3 Integration Layer**

* **Ethers.js**: Thư viện tương tác với blockchain
* **MetaMask**: Ví tiền mã hóa
* **lendingService.ts**: Service xử lý các giao dịch
* **useWeb3.ts**: Composable quản lý kết nối Web3

**3. Smart Contract Layer**

* **LendingPool.sol**: Contract chính xử lý logic lending
* **MockToken.sol**: ERC20 token dùng để vay
* **OpenZeppelin**: Thư viện bảo mật và chuẩn

**4. Blockchain Layer**

* **Hardhat Network**: Mạng blockchain local để phát triển
* **EVM**: Máy ảo thực thi smart contract

## Phân tích các actor và use cases

### Các Actor trong hệ thống

**1. Borrower (Người vay)**

* Mô tả: Người dùng muốn vay token bằng cách thế chấp ETH
* Mục tiêu: Vay token để sử dụng, đầu tư hoặc trading

**2. Supplier (Người cung cấp thanh khoản)**

* Mô tả: Người dùng gửi token vào pool để nhận lãi
* Mục tiêu: Kiếm lợi nhuận từ lãi suất

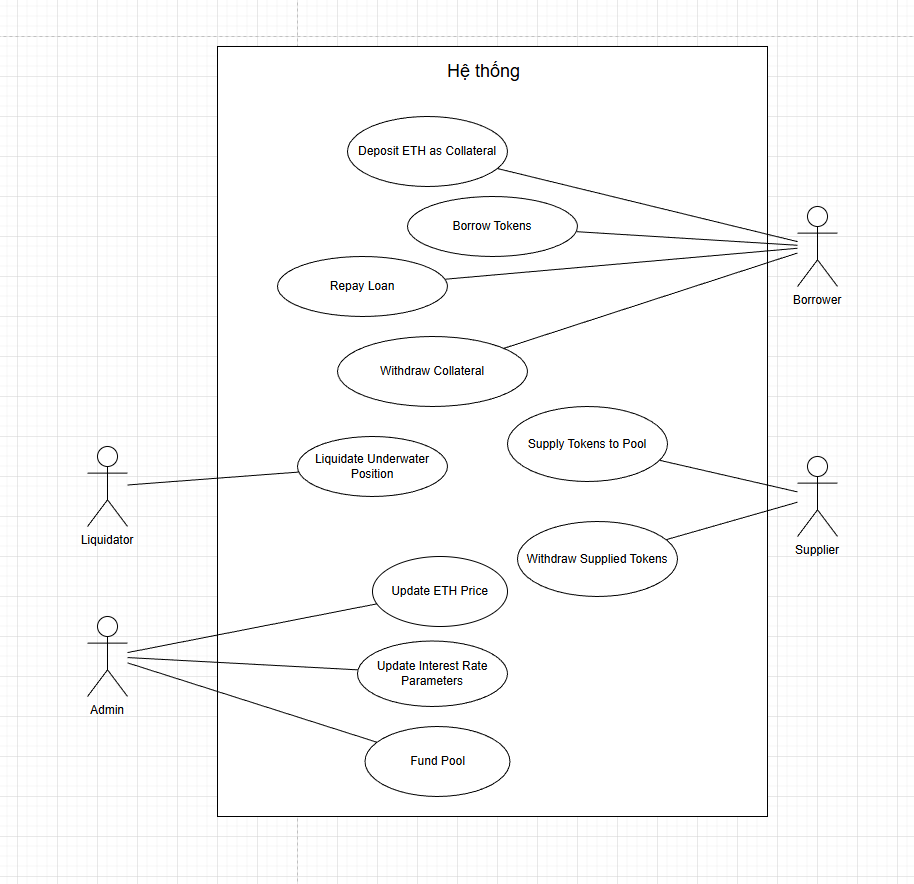
**3. Liquidator (Người thanh lý)**

* Mô tả: Người dùng thanh lý các vị thế rủi ro để nhận bonus
* Mục tiêu: Kiếm lợi nhuận từ liquidation bonus

**4. Admin (Quản trị viên)**

* Mô tả: Người quản lý hệ thống, cập nhật giá và tham số
* Mục tiêu: Duy trì và vận hành hệ thống

### Use Case Diagram



### Mô tả Use Cases

**Các Usecase chính:**

**UC1: Deposit ETH as Collateral**

* **Actor**: Borrower
* **Mô tả**: Người dùng gửi ETH vào pool làm tài sản thế chấp
* **Precondition**: Người dùng đã kết nối ví và có ETH
* **Main Flow**:

1. Người dùng nhập số lượng ETH muốn gửi
2. Hệ thống kiểm tra số dư ví
3. Người dùng xác nhận giao dịch
4. Smart contract nhận ETH và cập nhật deposits[user]
5. Hệ thống hiển thị số dư thế chấp mới

**UC2: Borrow Tokens**

* **Actor**: Borrower
* **Mô tả**: Người dùng vay token dựa trên giá trị thế chấp
* **Precondition**: Người dùng đã có thế chấp ETH
* **Main Flow**:

1. Người dùng nhập số lượng token muốn vay
2. Hệ thống tính toán max borrow (80% LTV)
3. Hệ thống kiểm tra thanh khoản pool
4. Người dùng xác nhận giao dịch
5. Smart contract chuyển token cho người dùng
6. Hệ thống cập nhật loans[user] và totalBorrowsGlobal

**UC3: Repay Loan**

* **Actor**: Borrower
* **Mô tả**: Người dùng trả nợ để giảm debt và cải thiện Health Factor
* **Precondition**: Người dùng có nợ và có đủ token để trả
* **Main Flow**:

1. Người dùng nhập số lượng token muốn trả
2. Hệ thống kiểm tra số dư token và nợ hiện tại
3. Người dùng approve token (nếu cần)
4. Người dùng xác nhận giao dịch
5. Smart contract cập nhật loans[user] và totalBorrowsGlobal
6. Hệ thống hiển thị số dư nợ mới

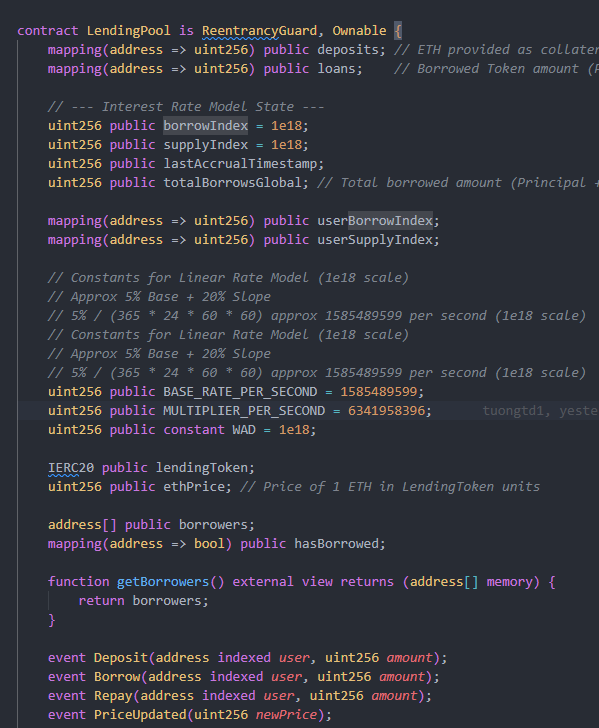
**UC7: Liquidate Underwater Position**

* **Actor**: Liquidator
* **Mô tả**: Thanh lý vị thế khi Health Factor < 1.0
* **Precondition**: Có vị thế với HF < 1.0 và liquidator có đủ token
* **Main Flow**:

1. Hệ thống hiển thị danh sách vị thế có thể thanh lý
2. Liquidator chọn vị thế muốn thanh lý
3. Hệ thống tính toán debt và collateral cần thanh lý
4. Liquidator approve token (nếu cần)
5. Liquidator xác nhận giao dịch
6. Smart contract trả nợ và chuyển collateral + bonus cho liquidator
7. Hệ thống cập nhật trạng thái borrower

## Thiết kế Smart Contract (LendingPool)

### Cấu trúc Contract



### Các Function chính

**1. Core Functions**

| **Function** | **Mô tả** | **Modifiers** |
| --- | --- | --- |
| deposit() | Gửi ETH làm thế chấp | payable, nonReentrant |
| borrow(uint256) | Vay token | nonReentrant |
| repay(uint256) | Trả nợ | nonReentrant |
| withdraw(uint256) | Rút thế chấp | nonReentrant |
| supply(uint256) | Cung cấp thanh khoản | nonReentrant |
| withdrawSupply(uint256) | Rút thanh khoản | nonReentrant |
| liquidate(address) | Thanh lý vị thế | nonReentrant |

**2. Interest Rate Functions**

| **Function** | **Mô tả** |
| --- | --- |
| accrueInterest() | Tính lãi suất và cập nhật indices |
| \_updateUserBorrow(address) | Cập nhật số dư nợ của user |
| \_updateUserSupply(address) | Cập nhật số dư supply của user |

**3. View Functions**

| **Function** | **Mô tả** |
| --- | --- |
| getUserBorrowBalance(address) | Lấy số dư nợ (bao gồm lãi) |
| getUserSupplyBalance(address) | Lấy số dư supply (bao gồm lãi) |
| getBorrowers() | Lấy danh sách người vay |

**4. Admin Functions**

| **Function** | **Mô tả** | **Modifiers** |
| --- | --- | --- |
| updatePrice(uint256) | Cập nhật giá ETH | onlyOwner |
| setInterestParameters(uint256, uint256) | Cập nhật tham số lãi suất | onlyOwner |
| fundPool(uint256) | Nạp token vào pool | onlyOwner |

### Events

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

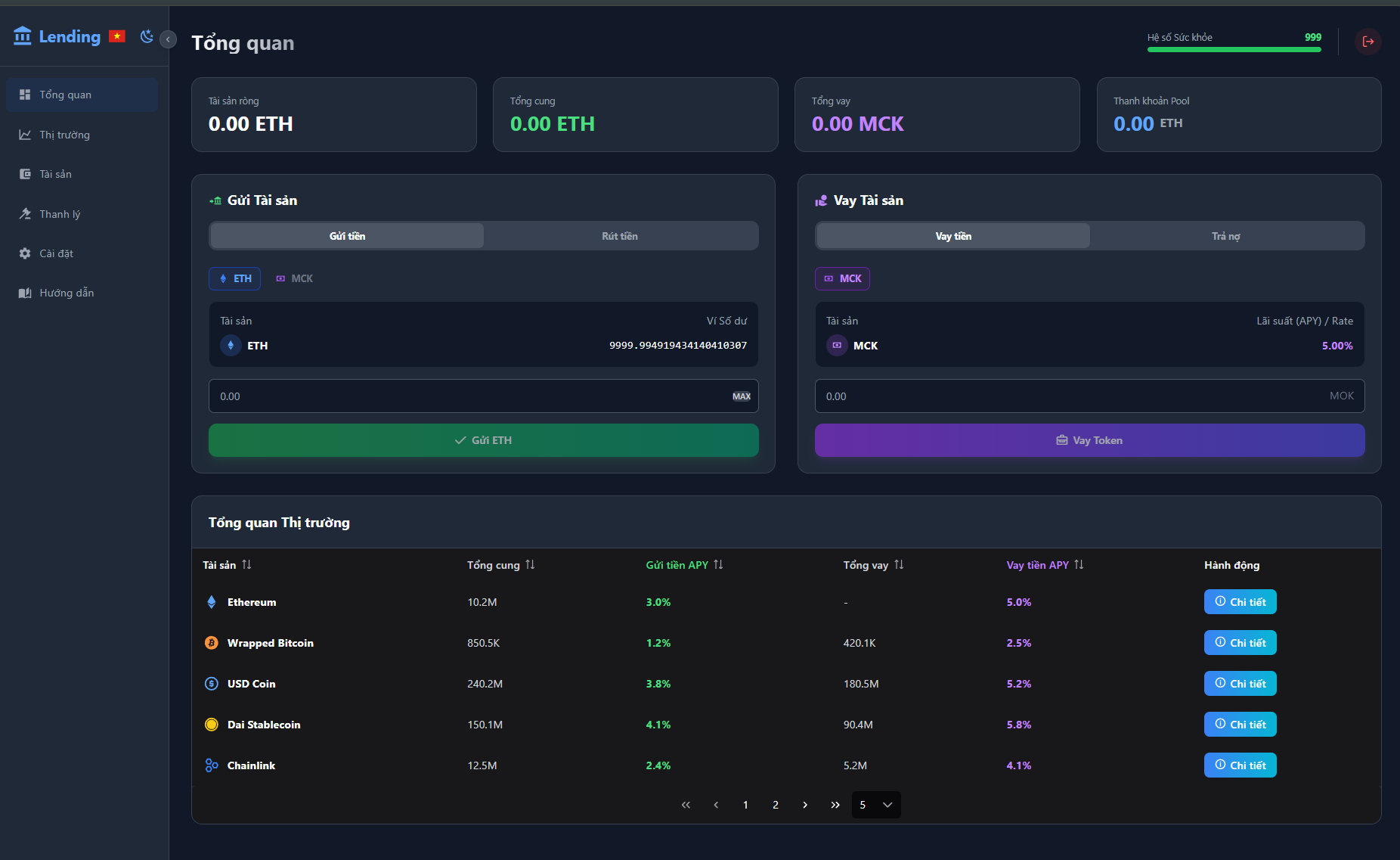
Nội dung do AI tạo ra có thể không chính xác.

Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.

## Thiết kế giao diện người dùng

1. **Dashboard (Trang chủ)**



Hình 1: Giao diện Dashboard

Dashboard là trang chính, cung cấp cái nhìn tổng quan về tài sản và hoạt động của người dùng. Khi vào trang, người dùng thấy ngay các thẻ tổng quan (Overview Cards) hiển thị tổng giá trị tài sản đã gửi, tổng nợ hiện tại, và Health Factor. Health Factor được hiển thị nổi bật với màu sắc cảnh báo: xanh khi an toàn, vàng khi cần chú ý, đỏ khi có nguy cơ thanh lý.

Phần Quick Actions đặt các form thao tác nhanh: deposit, borrow, repay, withdraw. Người dùng có thể thực hiện các giao dịch chính mà không cần chuyển trang. Mỗi form có validation và hiển thị preview trước khi xác nhận.

Market Summary cung cấp thông tin tổng quan về pool như tổng thanh khoản (TVL), lãi suất hiện tại và tỷ lệ sử dụng, giúp đánh giá nhanh thị trường.

1. **Market**

**Ảnh có chứa ảnh chụp màn hình, văn bản, Phần mềm đa phương tiện, phần mềm

Nội dung do AI tạo ra có thể không chính xác.**

Hình 2: Giao diện Market

Market cung cấp thông tin chi tiết về các pool và tài sản. Market Table liệt kê các tài sản với APY cho vay/vay, tỷ lệ sử dụng, và tổng thanh khoản. Người dùng có thể so sánh và chọn tài sản phù hợp.

Charts hiển thị biểu đồ TVL và utilization rate theo thời gian, giúp nhận biết xu hướng và đưa ra quyết định. Pool Details cung cấp thông tin chi tiết từng pool như lịch sử lãi suất, số lượng người vay, và các chỉ số khác

1. **My Assets**

**Ảnh có chứa ảnh chụp màn hình, văn bản, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.**

Hình 3: Giao diện My Assets

My Assets tập trung vào vị thế cá nhân. Supplied Assets liệt kê các tài sản đã cung cấp, số dư hiện tại, lãi đã tích lũy và APY. Borrowed Assets hiển thị các khoản vay, số nợ hiện tại (bao gồm lãi), lãi suất và thời gian vay.

Health Factor được hiển thị nổi bật với cảnh báo rõ ràng. Khi Health Factor < 1.0, hệ thống cảnh báo nguy cơ thanh lý và gợi ý hành động. Actions cung cấp các nút Repay và Withdraw để quản lý vị thế, với thông tin preview trước khi thực hiện.

1. **Liquidation**

**Ảnh có chứa Phần mềm đa phương tiện, phần mềm, Phần mềm đồ họa, văn bản

Nội dung do AI tạo ra có thể không chính xác.**

Hình 4: Giao diện Liquidation

Liquidation dành cho liquidator để tìm và thanh lý các vị thế rủi ro. Liquidatable Positions hiển thị bảng các vị thế có Health Factor < 1.0, sắp xếp theo mức độ rủi ro. Mỗi hàng hiển thị địa chỉ borrower, số nợ, giá trị thế chấp và Health Factor.

Position Details cung cấp thông tin chi tiết từng vị thế, bao gồm lịch sử giao dịch và các chỉ số liên quan. Liquidation Calculator tính toán lợi nhuận khi thanh lý, bao gồm số token cần trả, số ETH nhận được và bonus 10%, giúp liquidator đánh giá trước khi thực hiện.

Liquidate Button cho phép thực hiện thanh lý. Hệ thống yêu cầu xác nhận và hiển thị rõ ràng các thông tin liên quan trước khi thực hiện.

1. **Settings**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phần mềm đa phương tiện

Nội dung do AI tạo ra có thể không chính xác.

Hình 5: Giao diện Settings

Settings cho phép tùy chỉnh trải nghiệm. Language Toggle cho phép chuyển đổi giữa Tiếng Việt và English, áp dụng ngay lập tức cho toàn bộ giao diện.

Theme Toggle cho phép chuyển đổi giữa Light mode và Dark mode, lưu tùy chọn và áp dụng cho các lần truy cập sau.

Network Info hiển thị thông tin mạng đang kết nối (tên mạng, Chain ID, RPC URL), giúp xác nhận đang kết nối đúng mạng và cảnh báo nếu sai.

Account Info hiển thị địa chỉ ví đang kết nối (có thể copy), số dư ETH và token, và nút disconnect để ngắt kết nối an toàn.

# TRIỂN KHAI HỆ THỐNG

## Môi trường triển khai

### Yêu cầu hệ thống

**Phần mềm cần thiết:**

* **Node.js**: Phiên bản 16.0.0 trở lên (khuyến nghị 18.x hoặc 20.x)
* **npm** hoặc **yarn**: Quản lý package
* **MetaMask**: Extension trình duyệt để kết nối ví
* **Git**: Quản lý phiên bản code (tùy chọn)

**Môi trường phát triển:**

* **Hardhat**: Framework phát triển và test smart contract
* **Hardhat Network**: Mạng blockchain local (ChainID: 1337)
* **Nuxt 3**: Framework frontend
* **Trình duyệt**: Chrome, Firefox, hoặc Edge (hỗ trợ MetaMask)

### Cấu trúc thư mục dự án

Ảnh có chứa văn bản, ảnh chụp màn hình, số, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

### Cấu hình mạng blockchain

**Hardhat Network Configuration:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

**MetaMask Network Setup:**

* **Network Name**: Hardhat Local
* **RPC URL**:  http://127.0.0.1:8545
* **Chain ID**: 1337
* **Currency Symbol**: ETH

## Triển khai Smart Contract (Hardhat, Solidity)

### Thiết lập môi trường Hardhat

**Bước 1: Khởi tạo project**

*cd blockchain*

*npm init -y*

*npm install --save-dev hardhat @nomicfoundation/hardhat-toolbox*

*npm install @openzeppelin/contracts*

**Bước 2: Khởi tạo Hardhat project**

*npx hardhat init*

**Bước 3: Cấu hình Hardhat**

File hardhat.config.js được cấu hình với Solidity 0.8.20 và mạng localhost với ChainID 1337.

### Triển khai MockToken Contract

**Contract MockToken:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

**Deployment Script:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

**Kết quả:** MockToken được deploy và trả về địa chỉ contract để sử dụng trong LendingPool.

### Triển khai LendingPool Contract

**Deployment Process:**

*1. Deploy LendingPool với MockToken address và ETH price*

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

*2. Fund pool với 10,000 MCK tokens*

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.Lưu địa chỉ contract:**

Script tự động lưu địa chỉ vào file deployed\_addresses.json và copy ABI sang frontend:

*Lưu addresses*

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

*Copy ABI to frontend*

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

### Quy trình deployment hoàn chỉnh

**1. Khởi động Hardhat node:**

*cd blockchain*

*npx hardhat node*

Lệnh này khởi động local blockchain node tại http://127.0.0.1:8545 với 20 tài khoản test, mỗi tài khoản có 10,000 ETH.

**2. Deploy contracts (terminal mới):**

*cd blockchain*

*npm run deploy*

Script sẽ:

* Deploy MockToken
* Deploy LendingPool với MockToken address và ETH price = 2000
* Fund pool với 10,000 MCK
* Lưu addresses và copy ABI

**3. Kết quả:**

MockToken deployed to: 0x5FbDB2315678afecb367f032d93F642f64180aa3

LendingPool deployed to: 0xe7f1725E7734CE288F8367e1Bb143E90bb3F0512

LendingPool funded with 10000.0 tokens

* ABI files automatically copied to frontend!

## Triển khai các chức năng cốt lõi (Deposit, Borrow, Repay, Withdraw)

### Chức năng Deposit

**Smart Contract Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

**Frontend Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

**Luồng xử lý:**

1. User nhập số lượng ETH trong form
2. Frontend kiểm tra số dư ví
3. Gọi depositETH() với amount đã parse
4. MetaMask hiển thị popup xác nhận
5. User ký giao dịch
6. Smart contract nhận ETH và cập nhật deposits[user]
7. Frontend chờ confirmation và cập nhật UI

**Xử lý lỗi:**

* Kiểm tra số dư trước khi gửi
* Hiển thị thông báo khi user từ chối
* Xử lý lỗi network hoặc contract revert

### Chức năng Borrow

**Smart Contract Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

**Frontend Pre-validation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.*

**Đặc điểm:**

* Pre-validation ở frontend để cải thiện UX
* Kiểm tra LTV 80% trước khi cho phép vay
* Kiểm tra thanh khoản pool
* Tự động cập nhật lãi suất trước khi vay

### Chức năng Repay

**Smart Contract Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.*

**Frontend Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

**Tính năng đặc biệt:**

* Tự động approve token nếu cần
* Hỗ trợ "Repay All" để trả toàn bộ nợ (bao gồm dust debt)
* Cập nhật Health Factor sau khi trả

### Chức năng Withdraw

**Smart Contract Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.***Đặc điểm:**

* Kiểm tra Health Factor trước khi cho phép rút
* Chỉ cho phép rút nếu Health Factor >= 1.0 sau khi rút
* Sử dụng call để chuyển ETH an toàn

## Triển khai hệ thống lãi suất động

### Linear Interest Rate Model

**Công thức tính lãi suất:**

*// Utilization Rate*

*uint256 util = (borrows \* WAD) / totalLiquidity;*

*// Borrow Rate = Base + (Multiplier \* Util)*

*uint256 borrowRate = BASE\_RATE\_PER\_SECOND +*

*(MULTIPLIER\_PER\_SECOND \* util) / WAD;*

*// Supply Rate = Borrow Rate \* Util*

*uint256 supplyRate = (borrowRate \* util) / WAD;*

**Tham số mặc định:**

* **BASE\_RATE\_PER\_SECOND**: 1585489599 (tương đương ~5% APY)
* **MULTIPLIER\_PER\_SECOND**: 6341958396 (tương đương ~20% APY tại 100% util)

### Hàm accrueInterest()

**Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.*

**Đặc điểm:**

* Được gọi tự động trước mỗi giao dịch quan trọng
* Tính lãi theo thời gian thực (per-second)
* Cập nhật global indices (borrowIndex, supplyIndex)
* Emit event để frontend theo dõi

### Cập nhật số dư người dùng

**Update User Borrow Balance:**

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.*

**Update User Supply Balance:**

**Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ, phần mềm

Nội dung do AI tạo ra có thể không chính xác. View Functions để hiển thị số dư real-time:**

Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.

## Triển khai cơ chế thanh lý (Liquidation)

### Điều kiện thanh lý

**Health Factor Calculation:**

*// Health Factor < 1.0 khi:*

*totalDebt > maxBorrow*

*// Trong đó:*

*collateralValue = deposits \* ethPrice*

*maxBorrow = collateralValue \* 80% / 100%*

### Hàm liquidate()

**Implementation:**

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

**Cơ chế:**

1. Kiểm tra Health Factor < 1.0
2. Tính số ETH cần thanh lý: debt / ethPrice
3. Tính bonus 10%: baseCollateral \* 10%
4. Liquidator trả nợ bằng token
5. Liquidator nhận ETH (base + bonus)
6. Borrower mất một phần collateral nhưng được giải phóng khỏi nợ

### Frontend Integration

**Lấy danh sách vị thế có thể thanh lý:**

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

**Thực hiện thanh lý:**

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

## Tích hợp MetaMask và xử lý giao dịch

### Kết nối MetaMask

**useWeb3 Composable:**

*// composables/useWeb3.ts*

*export const useWeb3 = () => {*

*const account = useState<string | null>('web3\_account', () => null);*

*const isConnected = useState<boolean>('web3\_isConnected', () => false);*

*const provider = useState<ethers.BrowserProvider | null>('web3\_provider', () => null);*

*const signer = useState<ethers.JsonRpcSigner | null>('web3\_signer', () => null);*

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

### Xử lý Events

**Setup Event Listeners:**

Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

### Xử lý giao dịch

**Transaction Flow:**

*// services/lendingService.ts*

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.*

*Ảnh có chứa văn bản, ảnh chụp màn hình, Phông chữ

Nội dung do AI tạo ra có thể không chính xác.****Error Handling:***

*Ảnh có chứa văn bản, ảnh chụp màn hình, phần mềm, màn hình

Nội dung do AI tạo ra có thể không chính xác.*

### Auto-reconnect và Persistence

**Auto-reconnect on page load:**

*Ảnh có chứa văn bản, ảnh chụp màn hình

Nội dung do AI tạo ra có thể không chính xác.*

# KẾT QUẢ

## Kết quả triển khai

### Tổng quan hệ thống

Hệ thống Lending DApp đã được triển khai với các chức năng chính:

**Smart Contract Layer:**

* LendingPool contract với các chức năng: deposit, borrow, repay, withdraw, supply, liquidate
* MockToken contract (ERC20) để test
* Linear Interest Rate Model hoạt động đúng
* Hệ thống Health Factor và cơ chế thanh lý
* Bảo mật với ReentrancyGuard và input validation

**Frontend Layer:**

* Dashboard với overview cards và quick actions
* Market page với bảng thông tin và biểu đồ
* My Assets page hiển thị vị thế người dùng
* Liquidation page để thanh lý vị thế rủi ro
* Settings page với dark mode và đa ngôn ngữ
* Tích hợp MetaMask và xử lý giao dịch

### Kết quả triển khai Smart Contract

**Deployment thành công:**

* MockToken và LendingPool được deploy trên Hardhat localhost
* Contract address được lưu tự động và ABI được copy sang frontend
* Pool được fund với 10,000 MCK tokens để test

**Các chức năng đã test:**

* Deposit ETH: Hoạt động đúng, ETH được lưu làm thế chấp
* Borrow Token: Vay token với LTV 80%, kiểm tra thanh khoản pool
* Repay Loan: Trả nợ thành công, tự động approve token nếu cần
* Withdraw ETH: Rút thế chấp với kiểm tra Health Factor
* Supply Token: Cung cấp thanh khoản và nhận lãi suất
* Liquidate: Thanh lý vị thế với Health Factor < 1.0
* Interest Accrual: Tính lãi theo thời gian thực

### Kết quả triển khai Frontend

**Dashboard Page:**

* Overview Cards hiển thị tổng quan tài sản, nợ, Health Factor
* Quick Actions cho phép thực hiện các giao dịch chính
* Real-time updates sau mỗi giao dịch
* Toast notifications cho success/error

**Market Page:**

* Market Table hiển thị thông tin pool, APY, utilization rate
* Charts hiển thị TVL và utilization rate theo thời gian
* Dữ liệu được cập nhật real-time từ smart contract

**My Assets Page:**

* Hiển thị chi tiết tài sản đã cung cấp và đã vay
* Health Factor với cảnh báo màu sắc
* Actions để repay và withdraw

**Liquidation Page:**

* Danh sách vị thế có thể thanh lý (Health Factor < 1.0)
* Liquidation calculator tính toán lợi nhuận
* Thực hiện thanh lý thành công

**Settings Page:**

* Chuyển đổi ngôn ngữ Tiếng Việt/English
* Toggle dark/light mode
* Hiển thị thông tin mạng và tài khoản

### Kết quả tích hợp Web3

**MetaMask Integration:**

* Kết nối ví thành công
* Auto-reconnect khi reload trang
* Xử lý account và chain changes

**Transaction Handling:**

* Gửi giao dịch thành công
* Hiển thị loading state và chờ confirmation
* Cập nhật UI sau giao dịch

**Error Handling:**

* Xử lý user rejection, insufficient funds, contract revert
* Hiển thị thông báo lỗi rõ ràng

## Các vấn đề gặp phải và giải pháp

1. **Vấn đề về quản lý state và event listeners**

**Mô tả vấn đề:**Hệ thống gặp vấn đề memory leak do event listeners không được cleanup đúng cách. Khi component unmount hoặc disconnect, listeners vẫn hoạt động, gây tiêu tốn tài nguyên và có thể gây lỗi khi có nhiều instances.**Nguyên nhân:**

* Event listeners (accountsChanged, chainChanged) được add nhưng không remove
* Nhiều instances của component tạo nhiều listeners cùng lúc
* Không có cơ chế cleanup khi component unmount

**Giải pháp:**

1. **Lưu reference của callback functions:**

* Sử dụng module-level variables để lưu reference
* Đảm bảo có thể remove đúng function đã add

1. **Cleanup listeners:**

* Remove listeners trước khi add mới
* Remove listeners khi disconnect hoặc component unmount
* Sử dụng removeListener() với đúng function reference

1. **Quản lý lifecycle:**

* Setup listeners trong onMounted()
* Cleanup trong onUnmounted() hoặc disconnect()

1. **Kết quả:**

* Không còn memory leak
* Event listeners được quản lý đúng cách
* Hệ thống hoạt động ổn định hơn

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

Đồ án đã xây dựng thành công một hệ thống Lending DApp trên blockchain Ethereum, cho phép người dùng thực hiện các hoạt động cho vay và vay tài sản mã hóa một cách phi tập trung, minh bạch và tự động.

**Về mặt kỹ thuật:**

Đồ án đã triển khai các thành phần chính:

1. **Smart Contract Layer:**

* LendingPool contract với các chức năng: deposit, borrow, repay, withdraw, supply, liquidate
* Linear Interest Rate Model điều chỉnh lãi suất theo tỷ lệ sử dụng pool
* Hệ thống Health Factor và cơ chế thanh lý tự động để quản lý rủi ro
* Áp dụng các biện pháp bảo mật như ReentrancyGuard và input validation

1. **Frontend Layer:**

* Giao diện web với Nuxt 3, PrimeVue và Tailwind CSS
* Dashboard hiển thị tổng quan và quick actions
* Các trang Market, My Assets, Liquidation với đầy đủ chức năng
* Hỗ trợ đa ngôn ngữ (Tiếng Việt/English) và dark mode

1. **Web3 Integration:**

* Tích hợp MetaMask để kết nối ví và ký giao dịch
* Xử lý giao dịch với Ethers.js
* Auto-reconnect và xử lý lỗi

**Về mặt học thuật:**

Đồ án giúp nắm vững:

* Blockchain và smart contract: cách viết, deploy và tương tác với smart contract trên Ethereum
* DeFi: mô hình lending protocol, interest rate model, liquidation mechanism
* Web3: tích hợp frontend với blockchain, xử lý giao dịch và events
* Kiến trúc DApp: thiết kế và triển khai ứng dụng phi tập trung

**Về mặt thực tiễn:**

Hệ thống đã đạt được:

* Các chức năng cốt lõi hoạt động đúng như thiết kế
* Giao diện thân thiện, dễ sử dụng
* Xử lý lỗi và edge cases tốt
* Hiệu năng ổn định trên môi trường localhost

**Giá trị của đồ án:**

Đồ án minh chứng khả năng xây dựng một ứng dụng DeFi hoàn chỉnh, từ smart contract đến giao diện người dùng. Hệ thống có thể mở rộng và áp dụng cho các use case thực tế sau khi được audit và triển khai trên mainnet.

## Hạn chế của đồ án

**1. Phạm vi triển khai:**

* Chỉ triển khai trên Hardhat localhost network, chưa test trên testnet hoặc mainnet
* Chưa có audit bảo mật chuyên nghiệp, chỉ áp dụng best practices cơ bản
* Chưa có test suite đầy đủ cho smart contract và frontend

**2. Oracle và giá cả:**

* Sử dụng mock oracle đơn giản, admin có thể cập nhật giá ETH thủ công
* Chưa tích hợp Chainlink hoặc các oracle phức tạp để lấy giá real-time
* Giá cả có thể bị thao túng nếu triển khai trên mainnet

**3. Tính năng:**

* Chỉ hỗ trợ một loại tài sản thế chấp (ETH) và một loại token vay (MCK)
* Chưa hỗ trợ nhiều loại tài sản như các protocol lớn (Aave, Compound)
* Chưa có governance token và cơ chế DAO để quản trị phi tập trung
* Chưa hỗ trợ flash loans

**4. Bảo mật:**

* Chưa có audit bảo mật chuyên nghiệp
* Chưa có bug bounty program
* Chưa có cơ chế pause/emergency stop trong trường hợp khẩn cấp
* Chưa có insurance fund để bảo vệ người dùng

**5. Hiệu năng và mở rộng:**

* Chưa tối ưu gas cost cho các operations
* Chưa có cơ chế batch operations để giảm số lượng giao dịch
* Chưa hỗ trợ Layer 2 solutions (Arbitrum, Optimism) để giảm phí

## Hướng phát triển

**Mở rộng tính năng**

* **Multi-Asset Support:** Hỗ trợ nhiều loại tài sản thế chấp và token vay (ETH, WBTC, USDC, DAI, etc.)
* **Flash Loans:** Triển khai flash loans để người dùng vay không cần thế chấp trong một transaction
* **Stablecoin Integration:** Tích hợp các stablecoin phổ biến (USDC, DAI, USDT)
* **Cross-Chain Support:** Hỗ trợ các blockchain khác (BSC, Polygon, Arbitrum)

**Oracle và Giá cả**

* **Chainlink Integration:** Tích hợp Chainlink Price Feeds để lấy giá real-time và đáng tin cậy
* **Multiple Oracle Sources:** Sử dụng nhiều nguồn oracle và tính giá trung bình để giảm rủi ro
* **Price Impact Protection:** Thêm cơ chế bảo vệ khỏi price manipulation