

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1



THỰC TẬP CƠ SỞ

ĐỀ TÀI:

Xây Dựng Website Giao Dịch Tiền Mã Hóa Moondotfun

Giảng viên hướng dẫn : TS. Kim Ngọc Bách

Sinh viên thực hiện : Phạm Mạnh Thắng

Khóa : 2022-2027

Hệ : ĐẠI HỌC CHÍNH QUY

Hà Nội, tháng 6 năm 2025

LỜI CẢM ƠN

Đầu tiên, em xin được gửi lời cảm ơn đến Học viện Công nghệ Bưu chính Viễn thông đã tạo điều kiện để em có thể thực hiện học phần Thực tập cơ sở trước khi bắt đầu đi thực tập chính thức.

Em xin bày tỏ lòng biết ơn sâu sắc của em tới TS. Đào Ngọc Phong vì những sự hướng dẫn, hỗ trợ, tạo điều kiện hết sức thuận lợi trong quá trình thực hiện học phần Thực tập cơ sở.

Do thời gian có hạn và kiến thức còn hạn chế nên mặc dù đã rất cố gắng, báo cáo của em vẫn không thể tránh khỏi những thiếu sót. Em rất mong nhận được góp ý từ thầy cô để báo cáo có thể hoàn thiện hơn.

Cuối cùng em xin kính chúc quý thầy, cô và các bạn có được sức khỏe dồi dào, thành công trong sự nghiệp.

Em xin chân thành cảm ơn!

Hà Nội, tháng 6 năm 2025

NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM
(Của người hướng dẫn)

[illegible]

Điểm:..... (bằng chữ:)

....., ngày..... tháng năm 2025

CÁN BỘ GIẢNG VIÊN

MỤC LỤC

LỜI CẢM ƠN	i
NHẬN XÉT, ĐÁNH GIÁ, CHO ĐIỂM	2
CHƯƠNG 1: TỔNG QUAN HỆ THỐNG VÀ CÁC CÔNG NGHỆ SỬ DỤNG.....	5
1.1. Tổng quan hệ thống	5
1.1.1. Giới thiệu đồ án.....	5
1.1.2. Lý do chọn Moondotfun	5
1.1.3. Mục tiêu	5
1.2. Giới thiệu các công nghệ sử dụng trong hệ thống	6
1.2.1. Giới thiệu về ReactJS.....	6
1.2.2. Giới thiệu về Hardhat	7
1.2.3. Giới thiệu về Moralis	8
1.2.4. Giới thiệu Sepolia	9
1.2.5. Giới thiệu thuật toán Bonding Curve.....	9
CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG.....	12
2.1. Lấy yêu cầu hệ thống	12
2.1.1. Mô tả hệ thống bằng ngôn ngữ tự nhiên	12
2.1.2. Lập bảng từ khóa	13
2.1.3. Biểu đồ usecase tổng quan.....	14
2.1.4. Biểu đồ usecase chi tiết	14
2.2. Phân tích hệ thống	16
2.2.1. Screenshot	16
2.3. Thiết kế hệ thống.....	18
2.3.1. Mô hình kiến trúc hệ thống.....	18
2.3.2. Biểu đồ tuần tự.....	18
CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM	20
3.1. Cài đặt hệ thống và triển khai hệ thống.....	20
3.1.1. Cài đặt và triển khai phía BE	20
3.1.2. Cài đặt và triển khai phía FE.....	21
3.1.3. Cài đặt Moralis API,URL	21
3.1.4. Cài đặt code để tạo và lấy Token từ phía Back-end.....	23
3.1.5. Cài đặt thuật toán Exponential Bonding Curve.....	24
3.1.6. Cài đặt hàm mua Token	25
3.1.7. Cài đặt tạo, thêm và burn thanh khoản trên Uniswap	26
3.2. Kết quả cài đặt	27
3.2.1. Triển khai phía Back-end.....	27

3.2.2. Triển khai phía Front-end	28
3.3. Kết luận	29
3.3.1. Kết quả đạt được	29
3.3.2. Hạn chế	31
3.3.3. Định hướng phát triển	32
3.4. TÀI LIỆU THAM KHẢO	32

CHƯƠNG 1: TỔNG QUAN HỆ THỐNG VÀ CÁC CÔNG NGHỆ SỬ DỤNG

1.1. Tổng quan hệ thống

1.1.1. Giới thiệu đề án

Với việc gần đây trong thế giới Web3 nổi lên phong trào memecoin, và rất nhiều người đã kiếm được lợi nhuận từ nó. Nhưng để tìm hiểu và tạo một đồng tiền mã hóa cho riêng mình là không hề dễ dàng gì, đặc biệt là thêm tính thanh khoản để mọi người có thể tham gia giao dịch trên mạng lưới phi tập trung. Đó là lúc Moondotfun ra đời.

Moon.fun (Moondotfun) là một website giúp người dùng Web3 có thể tự tạo và tự giao dịch các hợp đồng thông minh (smart contract) trên mạng thử nghiệm Sepolia. Người dùng có thể mua bán, lựa chọn những token yêu thích của bản thân hoặc có thể tự tạo cho mình một token riêng và mua bán token đó. Bằng cách sử dụng cơ chế Exponential BondingCurve (Cơ chế mở rộng của Bonding Curve), điều này sẽ giúp cho giao dịch mua bán và đúc token trở nên dễ dàng hơn.

1.1.2. Lý do chọn Moondotfun

Đề tài “Xây dựng website giao dịch tiền mã hóa Moondotfun” được em lựa chọn làm đề tài thực tập cơ sở ngành Công nghệ Thông tin Chất Lượng Cao nhằm giải quyết các bài toán của người dùng web3 nói chung và các nhà giao dịch tiền ảo nói riêng. Thị trường hiện nay, đặc biệt là khi càng ngày web3 càng lúc tiếp cận vào cuộc sống thường ngày của người dùng, vậy nên việc lừa đảo diễn ra càng ngày càng phổ biến trên mạng lưới web3 đặc biệt là hình thức lừa đảo mua đồng tiền điện tử. Do đó, việc xây dựng một website mà giúp người dùng web3 đặc biệt là những người mới hoặc những nhà phát triển thực thụ có thể tạo token mà không cần lo lắng về vấn đề scam, remove liquid hoặc rug pool.

Qua đề tài này, em không chỉ mong muốn tạo ra một sản phẩm có tính ứng dụng cao mà còn có cơ hội để phát triển và hoàn thiện các kỹ năng lập trình web, đồng thời áp dụng các công nghệ hiện đại hơn vào lĩnh vực phát triển phần mềm. Việc sử dụng công nghệ ReactJS cho Front-end, Hardhat cho Back-end và Moralis cho RPC sẽ giúp em hiểu rõ hơn để tạo ra một ứng dụng web3 đầy đủ tính năng và hiệu quả. ReactJS mang lại cho người dùng trải nghiệm mượt mà, linh hoạt, đồng thời đảm bảo khả năng mở rộng khi ứng dụng phát triển. Hardhat là một công cụ hỗ trợ lập trình ngôn ngữ Solidity trên local để người dùng thoải mái test mà không tốn bất kì chi phí gas hoặc token.

1.1.3. Mục tiêu

Giúp người dùng Web3 tự do tạo và giao dịch token yêu thích của mình.

Tạo một nơi người dùng hệ sinh thái Monad có thể tự tạo token của họ.

Tránh khỏi việc dính vào những đồng token không minh bạch (Mintable, Khóa Liquid, Khóa Buy/Sell,...).

Bảo vệ được quyền sở hữu của tác giả, token minh bạch và được truy vết bởi mạng lưới Blockchain.

Giúp giải quyết vấn đề đồng tiền mất giá khi các quốc gia in tiền vô tội vạ,..

1.2. Giới thiệu các công nghệ sử dụng trong hệ thống

1.2.1. Giới thiệu về ReactJS

ReactJS là một thư viện JavaScript phổ biến được sử dụng để xây dựng giao diện người dùng (UI) động và phản ứng nhanh chóng với các thay đổi trong dữ liệu. Dưới đây là lý do tại sao ReactJS là sự lựa chọn chính cho frontend:

Component-based Architecture (Kiến trúc dựa trên component):

ReactJS cho phép chia nhỏ giao diện thành các component độc lập và tái sử dụng, giúp phát triển giao diện một cách linh hoạt và dễ bảo trì. Ví dụ, các phần như giỏ hàng, danh sách tranh, thông tin chi tiết có thể được xây dựng thành các component riêng biệt, giúp dễ dàng tái sử dụng và cập nhật.

Virtual DOM (DOM ảo):

React sử dụng cơ chế Virtual DOM, giúp cải thiện hiệu suất khi thay đổi dữ liệu. Khi có sự thay đổi, React chỉ cập nhật những phần tử DOM cần thiết thay vì toàn bộ giao diện, giúp giảm thời gian render và mang đến trải nghiệm người dùng mượt mà.

React Router:

React Router được sử dụng để điều hướng giữa các trang khác nhau trên website (ví dụ: trang chủ, trang chi tiết sản phẩm, giỏ hàng, thanh toán). Điều này giúp xây dựng ứng dụng một trang (Single Page Application - SPA) mà không cần tải lại toàn bộ trang.

State Management (Quản lý trạng thái):

Để quản lý dữ liệu giữa các component, React sử dụng state và props. Đối với các ứng dụng phức tạp, có thể sử dụng các thư viện như Redux hoặc React Context API để quản lý trạng thái toàn cục (global state), ví dụ như thông tin người dùng đăng nhập, giỏ hàng, hay lịch sử đơn hàng.

Responsive Design (Thiết kế đáp ứng):

ReactJS kết hợp tốt với các công cụ như CSS Grid, Flexbox, và Media Queries để tạo ra một giao diện web có thể tự động điều chỉnh phù hợp với các thiết bị khác nhau (máy tính để bàn, máy tính bảng, điện thoại di động).

React Hooks

Hooks là một cải tiến lớn trong React, cho phép sử dụng state và các tính năng khác trong functional components mà không cần dùng class. Một số hooks phổ biến:

- `useState`: Quản lý state.
- `useEffect`: Thực hiện các tác vụ phụ như gọi API, cập nhật DOM.
- `useContext`: Chia sẻ dữ liệu giữa các components.

Unidirectional Data Flow:

- React sử dụng mô hình luồng dữ liệu một chiều (data flows down).
- Dữ liệu từ component cha được truyền xuống component con thông qua props.
- Điều này giúp quản lý dữ liệu trong ứng dụng dễ dàng và hạn chế lỗi.

Lợi ích của ReactJS

Hiệu suất cao

- Sử dụng Virtual DOM giúp React xử lý các thay đổi nhanh chóng và tối ưu hóa hiệu suất.
- Việc cập nhật DOM thật được giảm thiểu đáng kể.

Tái sử dụng mã

- Các thành phần được viết dưới dạng components giúp tái sử dụng dễ dàng.
- Điều này tiết kiệm thời gian, giảm thiểu lỗi và tăng năng suất lập trình.

Hệ sinh thái phong phú

- React không hoạt động độc lập mà có thể kết hợp với nhiều thư viện như Redux, React Router, Axios để xây dựng các ứng dụng phức tạp.

Dễ tích hợp

- React có thể tích hợp vào các ứng dụng hiện có mà không cần viết lại toàn bộ mã.
- Điều này lý tưởng cho việc nâng cấp hoặc mở rộng các ứng dụng cũ.

Ứng dụng thực tế của ReactJS

React được sử dụng trong nhiều lĩnh vực khác nhau, từ các ứng dụng nhỏ đến hệ thống lớn:

- Single Page Applications (SPA): Các ứng dụng chạy mượt mà, không cần tải lại trang.
- Mạng xã hội: Facebook, Instagram đều được xây dựng một phần bằng React.
- Dashboard: Các ứng dụng quản trị phức tạp, hiển thị dữ liệu thời gian thực.
- Ứng dụng di động: Với React Native, React được sử dụng để phát triển ứng dụng di động native.

1.2.2. Giới thiệu về Hardhat

Hardhat là một môi trường phát triển mã nguồn mở để xây dựng và thử nghiệm các hợp đồng thông minh trên chuỗi khối Ethereum. Nó cung cấp các công cụ và tính năng mạnh mẽ, bao gồm trình biên dịch Solidity tích hợp, khung thử nghiệm, công cụ gỡ lỗi, công cụ triển khai và hệ thống plugin.

Với Hardhat, các nhà phát triển có thể viết, biên dịch, thử nghiệm và triển khai hợp đồng thông minh của họ một cách an toàn và hiệu quả. Giao diện thân thiện với người dùng và tài liệu toàn diện khiến nó trở thành lựa chọn phổ biến trong số các nhà phát triển, cho dù họ mới bắt đầu hay đã có nhiều năm kinh nghiệm trong phát triển Ethereum.

Tính năng nổi bật của Hardhat:

- Hardhat Network (Blockchain local tích hợp):
 - Cho phép bạn chạy một mạng blockchain mô phỏng ngay trong máy

- tính.
 - Tốc độ cực nhanh, hỗ trợ debug chi tiết.
 - Tự động "fork" trạng thái mainnet để test trong môi trường thật.
- Console.log cho Smart Contract: Hardhat hỗ trợ console.log() trong Solidity (thông qua plugin), giúp debug dễ dàng như khi lập trình JavaScript.
- Plugin System
 - Dễ dàng mở rộng tính năng với các plugin như:
 - hardhat-ethers: tích hợp với thư viện ethers.js.
 - hardhat-waffle: hỗ trợ test với Waffle.
 - hardhat-deploy: chuẩn hóa quá trình deploy hợp đồng.
 - hardhat-gas-reporter: báo cáo gas trong quá trình test.
- Test nhanh với Mocha/Chai: Viết unit test cho smart contract bằng JavaScript với Mocha và Chai rất dễ dàng.
- Tự động biên dịch Solidity:
 - Hardhat theo dõi các file .sol trong thư mục contracts/.
 - Tự động biên dịch khi có thay đổi.
 - Hỗ trợ nhiều phiên bản Solidity cùng lúc (multi-version compiler).
- Tích hợp Ethers.js để gọi function Solidity:
 - Khi contract Solidity được deploy, Hardhat tự động tạo ABI & address để bạn gọi qua Ethers.js.
 - Hỗ trợ fake account, signer, gas estimate,...

1.2.3. Giới thiệu về Moralis

Moralis là một nền tảng phát triển Web3 (Web3 development platform) giúp lập trình viên xây dựng ứng dụng blockchain nhanh hơn, dễ hơn và chuyên nghiệp hơn.

Thay vì phải tự xây dựng node, index dữ liệu on-chain, hoặc viết backend phức tạp để tương tác với smart contract, bạn có thể dùng Moralis để làm tất cả điều đó chỉ với vài dòng code.

Tính năng nổi bật của Moralis:

- API Web3 mạnh mẽ (Moralis Web3 API):
 - Truy vấn balance, token holdings, NFTs, transaction history của bất kỳ ví nào.
 - Truy xuất dữ liệu token prices, events từ smart contract, NFT metadata,...
 - Hỗ trợ multi-chain (Ethereum, BNB, Polygon, Solana, Arbitrum, Optimism,...)
- Authentication Web3 (Đăng nhập bằng ví):
 - Đăng nhập bằng MetaMask, WalletConnect,...
 - Xác thực người dùng qua message signing (không cần mật khẩu!)
- Realtime Events & Sync Smart Contract:
 - Tự động theo dõi events từ smart contract.
 - Ghi log vào database backend (Moralis DB) để sử dụng trong app của

bạn.

- Cực kỳ phù hợp cho DApps, GameFi, NFT Marketplace, DeFi...

1.2.4. Giới thiệu Sepolia

Sepolia là một Ethereum testnet – một mạng thử nghiệm mô phỏng Ethereum mainnet nhưng không dùng tiền thật.

Nó được dùng để: Kiểm thử smart contract, Deploy thử nghiệm DApps, Phát triển Web3 không tốn phí, Tránh rủi ro trước khi deploy lên mainnet.

Sepolia là testnet chính thức, ổn định và được khuyến dùng cho smart contract.

Đặc điểm của Sepolia:

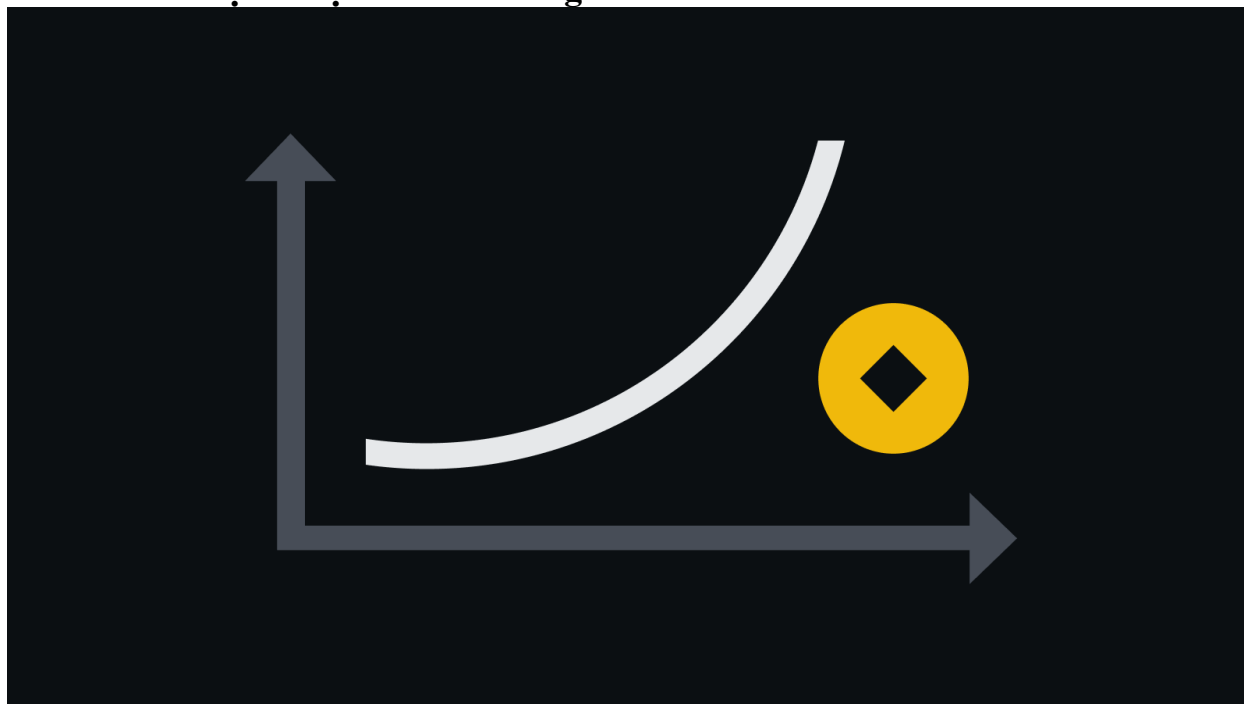
Chain ID: 11155111

Đồng tiền: Sepolia ETH (chỉ dùng test)

Consensus: Proof of Stake (giống mainnet)

EVM: Tương thích hoàn toàn

1.2.5. Giới thiệu thuật toán Bonding Curve.



Bằng cách liên kết cung và cầu, bonding curve cung cấp một khuôn khổ toán học cho ngành crypto và có thể được sử dụng để tự động hóa việc định giá và thanh khoản.

Các dự án có thể tùy chỉnh giá token và cách phân phối bằng cách áp dụng nhiều dạng curve khác nhau, bao gồm linear, exponential, logarithmic và step-function curves.

Dù không đảm bảo hoàn toàn tính tự bền vững do biến động token và rủi ro, các nền tảng như pump.fun cho thấy cách bonding curve giúp tạo ra việc phát hành token có thể dự đoán và khuyến khích người dùng tham gia sớm vào thị trường.

Khi có nhiều người mua token, giá có xu hướng tăng. Khi token bị bán hoặc rút khỏi lưu thông, giá thường sẽ giảm. Đây là mô hình bonding curve truyền thống, một cơ chế thường có lợi cho những người tham gia thị trường sớm.

Bonding curve là một khuôn khổ toán học quan trọng trong tokenomics. Các nền

tăng phổ biến như pump.fun dựa vào cơ chế bonding curve để tự động hóa việc định giá, thanh khoản và phân phối token.

Cơ chế định giá này được quản lý bởi smart contract, đảm bảo việc thực thi tự động, minh bạch và phi tập trung trên blockchain.

Cách Bonding Curve hoạt động: Càng có nhiều người mua token → càng nhiều token lưu hành → giá càng tăng. Ngược lại, khi người dùng bán token → nguồn cung giảm → giá giảm.

Tính tự động của bonding curve giúp duy trì liquidity khi token được mua hoặc bán. Các dự án có thể tùy chỉnh tokenomics bằng cách xác định các mô hình toán học riêng cho bonding curve. Không có giới hạn cố định về dạng curve, nhưng phổ biến nhất là: Linear bonding curve, Exponential bonding curve, Logarithmic bonding curve.

1.2.6. Giới thiệu về Exponential Bonding Curve

Exponential Bonding Curve là một mô hình toán học được sử dụng để xác định giá token trong các hệ thống phi tập trung như DeFi hoặc các nền tảng phát hành token. Trong mô hình này, giá của mỗi token sẽ tăng theo cấp số nhân khi số lượng token đang lưu hành (circulating supply) tăng lên.

Hàm giá (Price Function):

Hàm giá của token trong mô hình Exponential Bonding Curve được định nghĩa như sau:

We will use an exponential bonding curve. A simple exponential function for the price could be:

$$P(C) = P_0 \cdot e^{kC}$$

Trong đó:

$P(C)$: Giá của token tại thời điểm nguồn cung hiện tại là C .

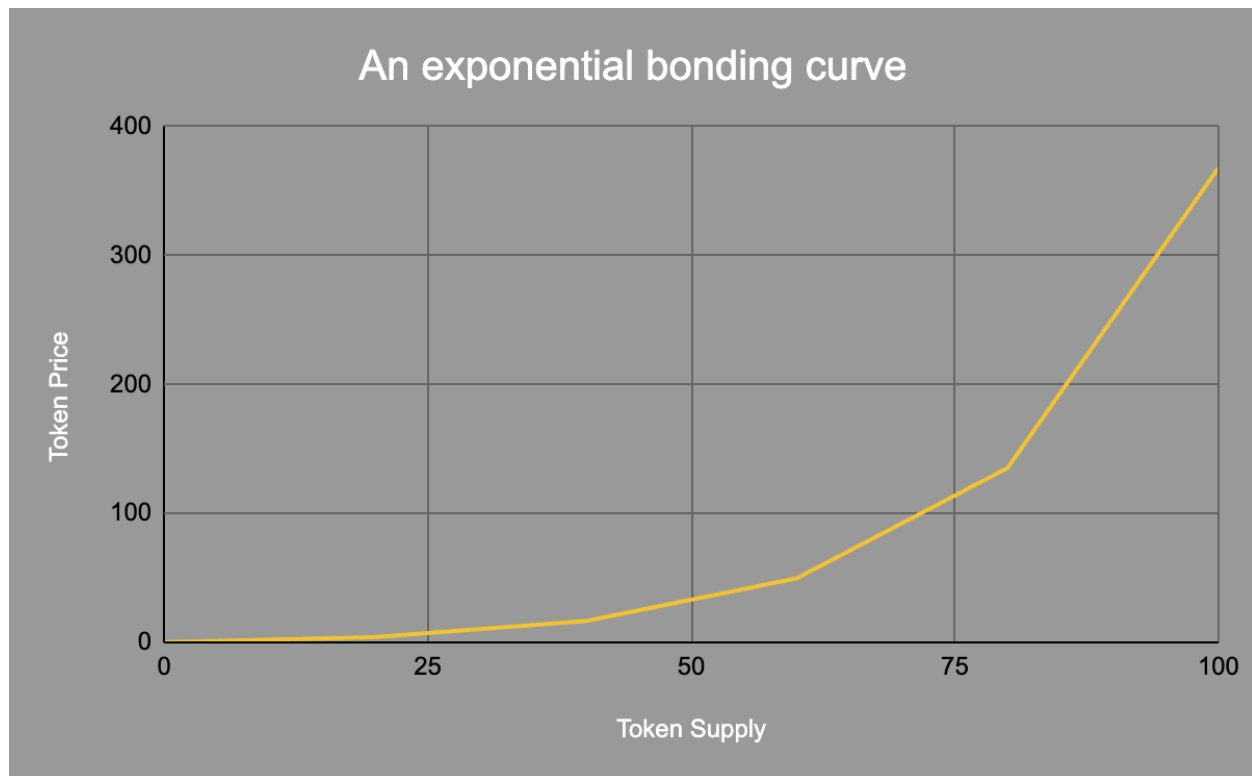
P_0 : Giá ban đầu của token (tính theo đơn vị như wei).

k : Hệ số tăng trưởng, càng lớn thì giá token càng tăng nhanh.

C : Lượng token đang lưu hành (circulating supply).

Mô hình này đảm bảo rằng những người mua token sớm sẽ mua với giá rẻ hơn, trong khi người mua sau phải trả giá cao hơn.

Đồ thị minh họa:



Trong hình thứ hai, ta thấy:

- Trục X: thể hiện lượng token đang được lưu hành (Supply)
- Trục Y: thể hiện giá token (Token Price)
- Đường cong hàm mũ biểu diễn mối quan hệ giữa giá và lượng cung
- Khi supply tăng, giá tăng nhanh chóng theo đường cong

Hàm tính tổng chi phí:

To find the cost for purchasing x tokens starting from a circulating supply C , we need to integrate the price function:

$$\text{Cost}(C, x) = \int_C^{C+x} P(s) ds = \int_C^{C+x} P_0 \cdot e^{ks} ds$$

The integral of $P_0 \cdot e^{ks}$ is:

$$\text{Cost}(C, x) = \left[\frac{P_0}{k} \cdot e^{ks} \right]_C^{C+x}$$

This simplifies to:

$$\text{Cost}(C, x) = \frac{P_0}{k} \left(e^{k(C+x)} - e^{kC} \right)$$

Ưu điểm của Exponential Bonding Curve:

- Khuyến khích người dùng mua sớm: Vì giá rẻ hơn ở giai đoạn đầu.

- Tự động điều chỉnh giá: Dựa vào cung, không cần sổ lệnh (order book).
- Tạo thanh khoản liên tục: Thông qua smart contract mà không cần bên trung gian.

CHƯƠNG 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

2.1. Lấy yêu cầu hệ thống

2.1.1. Mô tả hệ thống bằng ngôn ngữ tự nhiên

Chức năng cho User:

User là người dùng phổ thông trên hệ thống, có thể khám phá và tương tác với các token đã được tạo trên nền tảng.

Đăng nhập bằng ví Web3: Người dùng có thể đăng nhập vào hệ thống bằng các ví Web3 phổ biến như Phantom, OKX, v.v.

Sau khi đăng nhập, địa chỉ ví sẽ được sử dụng làm danh tính người dùng.

Tương tác với các token: Có thể thực hiện giao dịch mua hoặc bán bất kỳ token nào đã được niêm yết trên nền tảng với giá theo Bonding Curve.

Quản lý tài sản:

Giao diện hiển thị số dư của từng loại token mà người dùng đang nắm giữ.

Xem lịch sử giao dịch đã thực hiện.

Theo dõi giá trị hiện tại của danh mục đầu tư theo thời gian thực.

Chức năng cho Token Creator:

Token Creator là người có quyền khởi tạo và quản lý các token trên hệ thống.

Đăng nhập bằng ví Web3.

Tương tự như User, Token Creator sẽ xác thực qua các ví Web3 như Phantom, OKX, v.v.

Tạo và quản lý token:

Tạo mới token: Nhập tên, kí hiệu, tài ảnh đại diện, và liên kết với các mạng xã hội như X (Twitter) hoặc website cá nhân.

Cập nhật thông tin token bất kỳ lúc nào sau khi tạo.

Xem danh sách token đã tạo và theo dõi hoạt động giao dịch của từng token.

Giao dịch trước niêm yết: Trong giai đoạn chuẩn bị niêm yết, Token Creator có thể mua hoặc bán token của chính mình trước khi nó xuất hiện công khai trên thị trường. Tính năng này giúp thử nghiệm tính thanh khoản và định giá ban đầu trước khi mở bán.

Chức năng cho Admin:

Admin là người vận hành và giám sát toàn bộ hệ thống, có quyền cao nhất.

Thiết lập cơ chế Bonding Curve:

Chọn loại đường cong định giá (Exponential, Linear,...).

2.1.2. Lập bảng từ khóa

a) Xác định các actor của toàn hệ thống

Bảng 2.1 Danh sách các actor trong hệ thống

STT	Tên actor	Mô tả
1	User	Người dùng phổ thông trên hệ thống.
2	Admin	Người quản lý hệ thống.
3	Token Creator	Người có quyền khởi tạo và quản lý các token trên hệ thống.

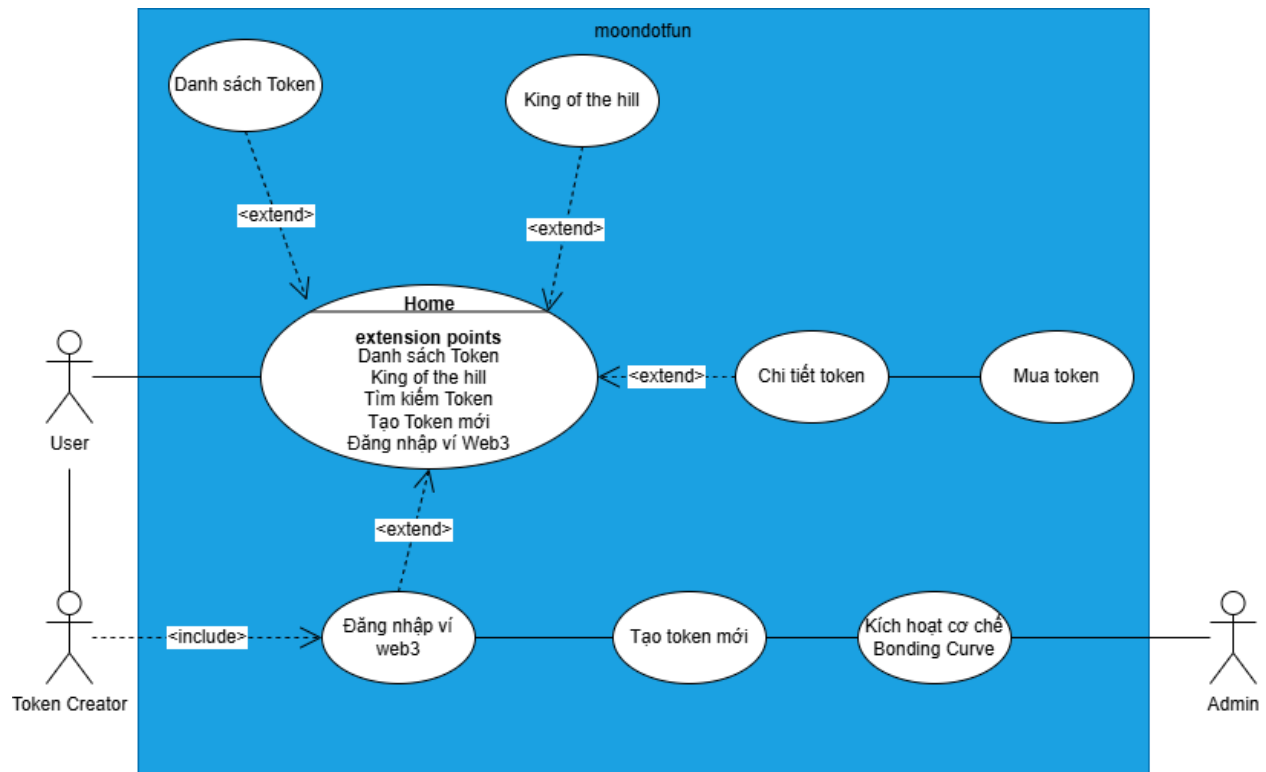
b) Xác định các usecase

Bảng 2.2 Danh sách các usecase trong hệ thống

STT	Tên usecase	Mô tả
1	Đăng nhập ví Web3	Người dùng có thể dùng ví web3 để tiến hành đăng nhập
2	Danh sách token.	Hiển thị danh sách các token được tạo bởi moondotfun trên giao diện người dùng.
3	Tìm kiếm Token.	Cho phép người dùng tìm kiếm token theo token address.
4	King of the hill	Hiển thị token đầu tiên của moondotfunn tạo ra.
5	Chi tiết token	Hiển thị chi tiết một đồng token mà người dùng chọn
6	Tạo token mới	Hiển thị danh sách yêu cầu người dùng điền thông tin để tạo token mới

7	Kích hoạt cơ chế Bonding Curve	Khi người dùng tạo token, Admin kích hoạt cơ chế Bonding Curve để hoàn thiện token.
8	Mua token	Người dùng nhập số lượng token muốn mua và tiến hành giao dịch.

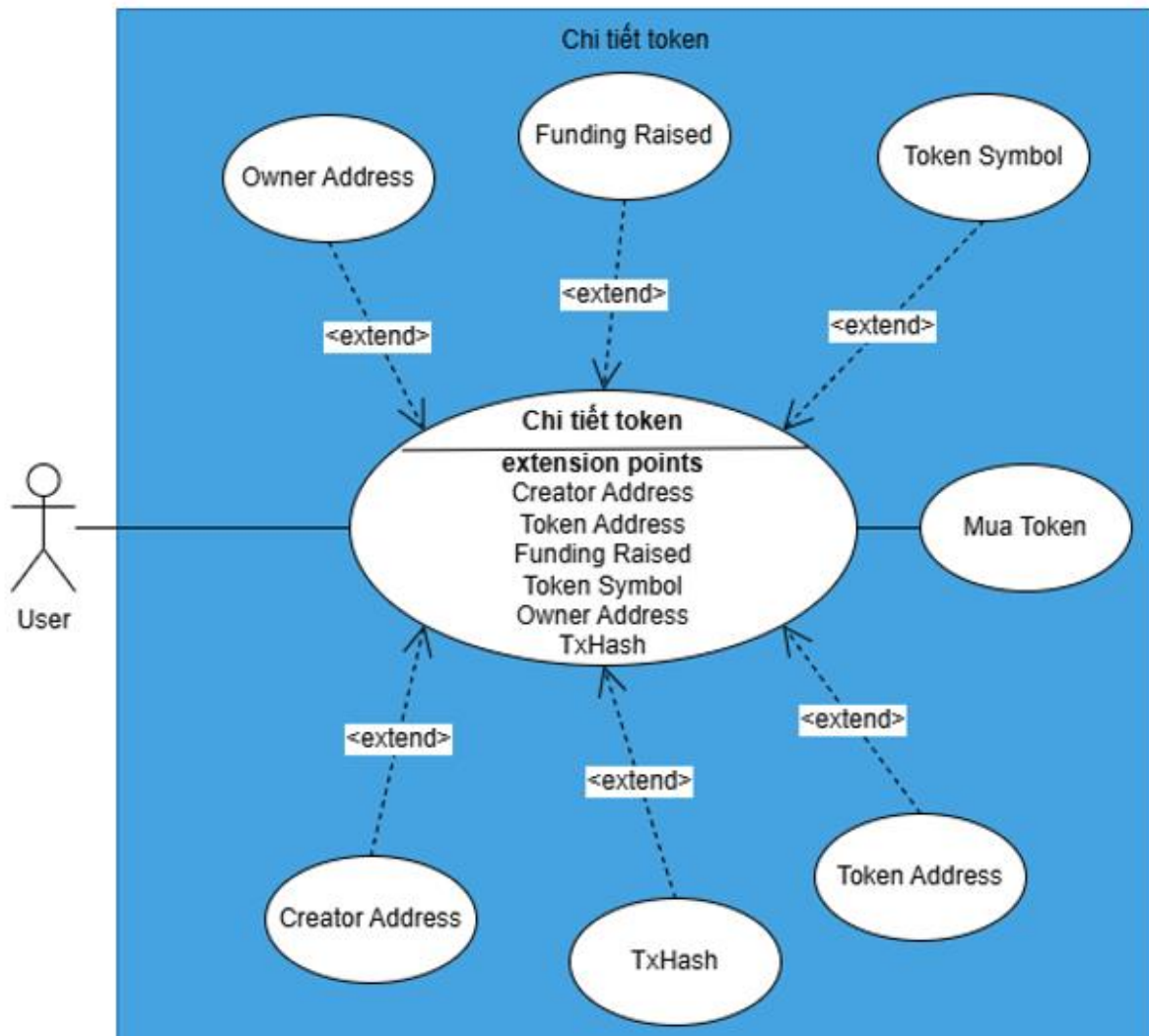
2.1.3. Biểu đồ usecase tổng quan



Hình 2.1: Biểu đồ usecase toàn hệ thống

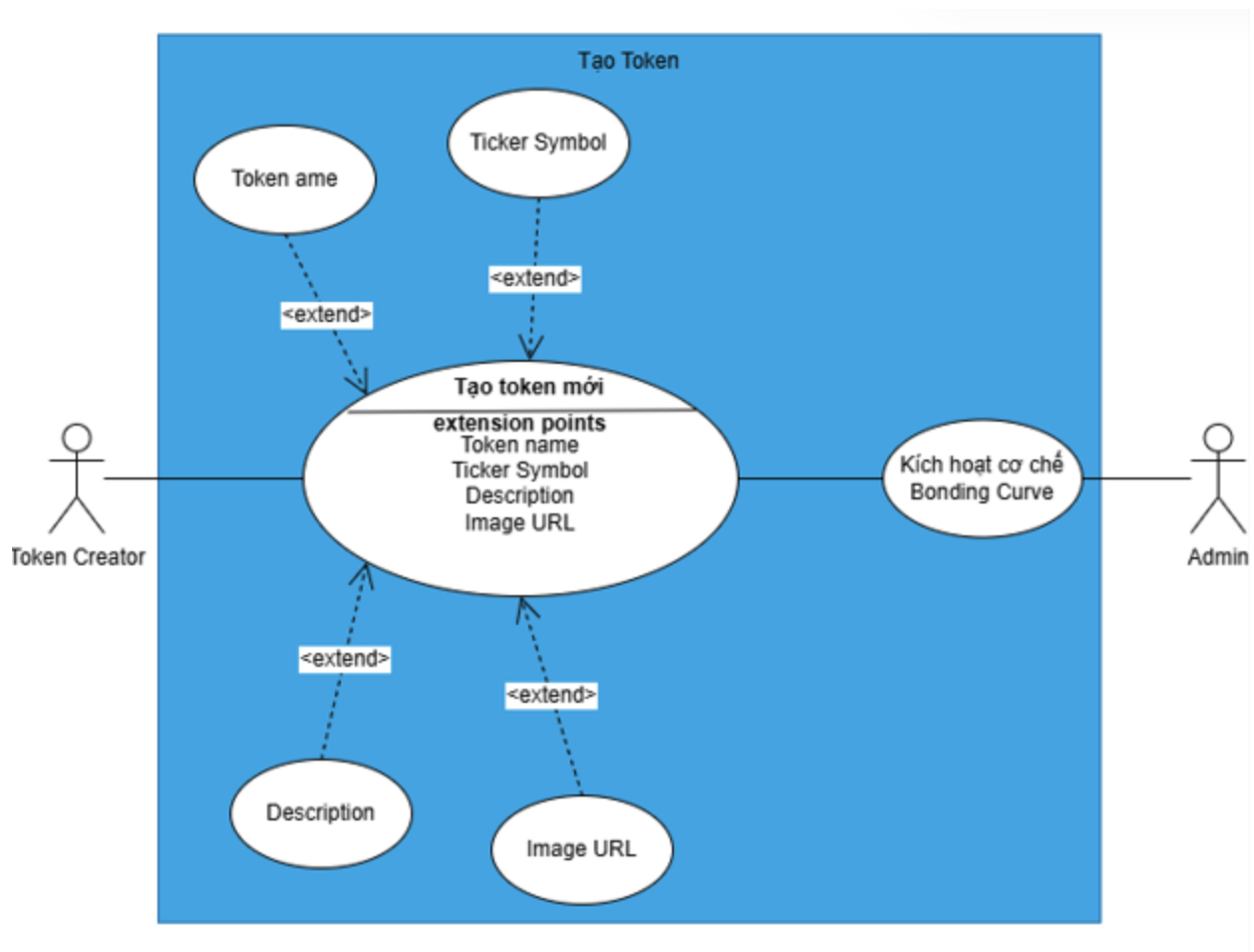
2.1.4. Biểu đồ usecase chi tiết

Chức năng chi tiết Token:



Hình 2.2: Biểu đồ phân rã usecase Chi tiết Token

Chức năng Tạo Token:



Hình 2.3: Biểu đồ phân rã usecase Tạo Token

2.2. Phân tích hệ thống

2.2.1. Srenario

Kịch bản chức năng Chi tiết Token

Bảng 2.3: Kịch bản chức năng Chi tiết Token

Tên use case	Chi tiết Token
Tác nhân chính	User

Tiền điều kiện	User truy cập thành công vào moondotfun
Đảm bảo thành công	User truy cập thành công và hiện ra các thông tin về một token.
Chuỗi sự kiện chính:	<ol style="list-style-type: none"> 1. User truy cập vào moondotfun. 2. Hệ thống trả về danh sách các token đã được tạo trên moondotfun (theo contract address). 3. User click vào một token mà User muốn truy cập. 4. Hệ thống hiển thị thông tin về token như: Token Address, Contract Address, Ticker,... 5. User có thể điền số lượng Token và click vào button Buy. 6. Hệ thống hiển thị pop-up MetaMask để hoàn thành giao dịch. 7. Hệ thống trả về txid cho User khi mua thành công.

Kịch bản chức năng Tạo Token

Bảng 2.4: Kịch bản chức năng Tạo Token

Tên use case	Tạo Token
Tác nhân chính	Token Creator
Tiền điều kiện	User đăng nhập thành công với role = Token Creator
Đảm bảo thành công	Token được đẩy lên Sepolia và có Txhash
Chuỗi sự kiện chính:	<ol style="list-style-type: none"> 1. Token Creator click vào button ‘create a new coin’. 2. Hệ thống hiển thị một list để người dùng nhập thông tin của token. Ví dụ như: Token name, Ticker Symbol, Image URL,...

	<ol style="list-style-type: none"> 3. Token Creator click vào button Create để tiến hành tạo token. 4. Hệ thống kết nối với ví Web3 và yêu cầu người dùng trả phí 0.0001 ETH cho lần tạo token. 5. Token Creator hoàn tất giao dịch. 6. Hệ thống trả về TxHash và báo thành công.
--	---

2.3. Thiết kế hệ thống.

2.3.1. Mô hình kiến trúc hệ thống.

Hệ thống được thiết kế với 3 thành phần chính:

Ứng dụng (ReactJS):

- Là giao diện người dùng chính, phát triển bằng ReactJS. ReactJS tương tác trực tiếp với người dùng, gửi yêu cầu và nhận dữ liệu từ Blockchain Sepolia qua Moralis API.
- ReactJS gửi các yêu cầu thông qua Ethereum JSON-RPC provider (Moralis) và trích xuất dữ liệu từ Sepolia.
- Dùng để kết nối tới ví người dùng, gửi giao dịch tới Sepolia, đọc dữ liệu từ Blockchain, tương tác với smart contract.

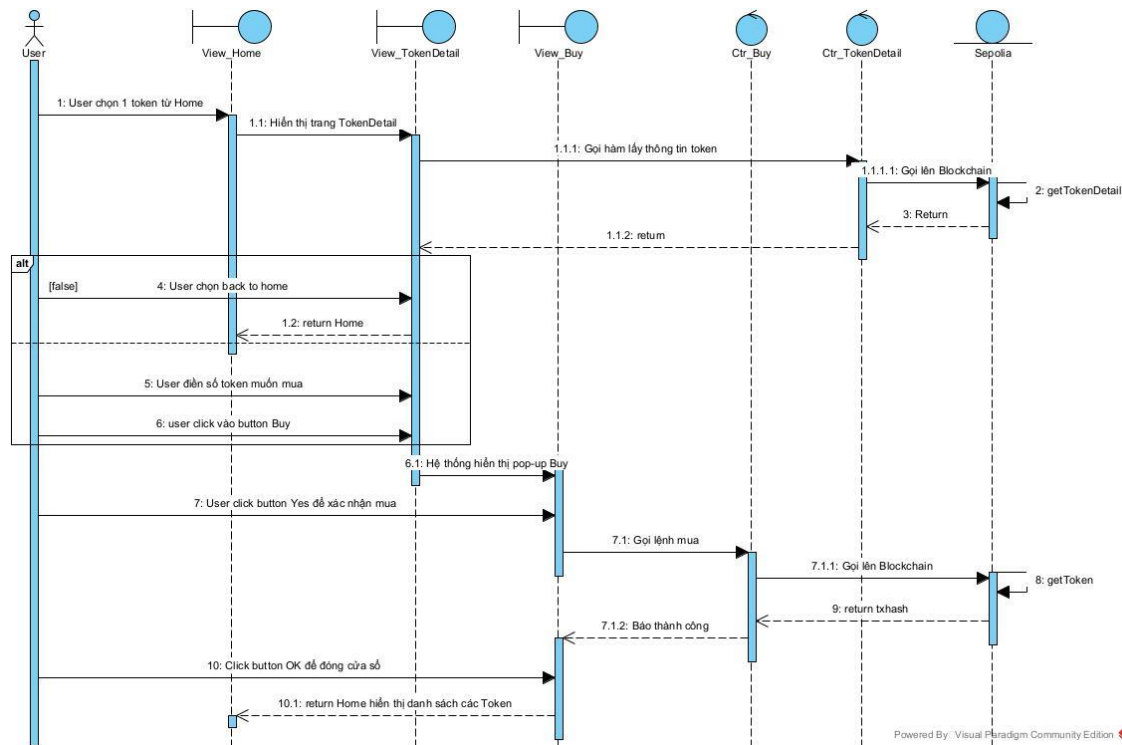
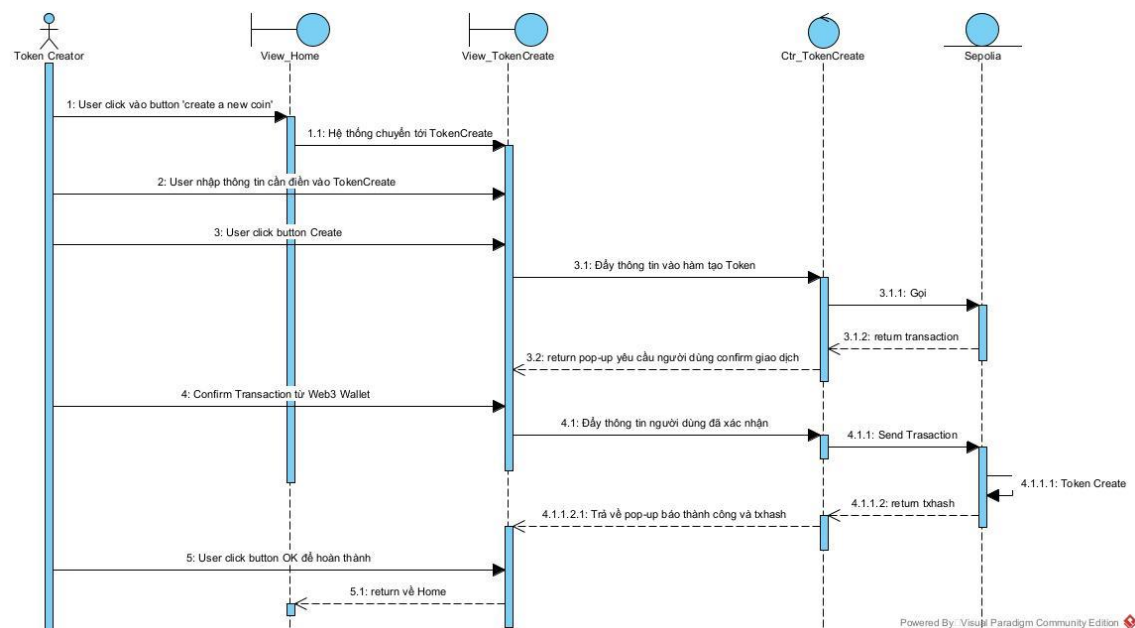
Blockchain Sepolia:

- Là trung tâm lưu trữ các smart contract và token address.
- **Ghi nhận giao dịch:** Mọi hành động như gửi ETH, gọi hàm contract, swap, lưu dữ liệu sẽ được ghi lại như một block không thể thay đổi.
- **Xác thực giao dịch:** Các node trên mạng Sepolia sẽ xác minh và ghi nhận các giao dịch gửi từ ReactJS thông qua Moralis.
- **Quản lý trạng thái:** Trạng thái các smart contract, ví, token balance, v.v... đều được duy trì tại đây.

Hardhat:

- Là trung tâm triển khai smart contract.
- Hardhat giúp người dùng viết smart contract và deploy trên mạng lưới Sepolia.
- Việc deploy contract chỉ diễn ra một lần khi chạy hardhat và sẽ trả về Contract Address để ReactJS tương tác với Blockchain Sepolia.

2.3.2. Biểu đồ tuần tự

Chức năng Chi tiết Token:**Hình 2.4: Sơ đồ tuần tự chức năng Chi tiết Token****Chức năng Tạo Token:****Hình 2.5: Sơ đồ tuần tự chức năng Tạo Token**

CHƯƠNG 3: CÀI ĐẶT THỬ NGHIỆM

3.1. Cài đặt hệ thống và triển khai hệ thống.

Yêu cầu công cụ:

Tải và cài đặt Visual Ttudio: code.visualstudio.com/download

Visual Studio Code (VS Code) là một trình soạn thảo mã nguồn miễn phí và mã nguồn mở, phát triển bởi Microsoft. Nó hỗ trợ nhiều ngôn ngữ lập trình, bao gồm JavaScript, Python, C++, và nhiều hơn nữa. VS Code có giao diện người dùng thân thiện, tính năng tự động hoàn thành mã, gỡ lỗi tích hợp, và khả năng mở rộng mạnh mẽ qua các tiện ích mở rộng (extensions). Đây là một công cụ phổ biến trong cộng đồng lập trình nhờ tính nhẹ, nhanh chóng và dễ sử dụng.

Tải và cài đặt NodeJs trên trang chủ của NodeJs:
nodejs.org/en/download/package-manager

- Node.js là một môi trường chạy JavaScript mã nguồn mở, đa nền tảng, được xây dựng trên công cụ V8 JavaScript Engine của Google. Nó cho phép thực thi JavaScript phía máy chủ, không chỉ giới hạn trong trình duyệt, giúp xây dựng các ứng dụng nhanh, hiệu quả và dễ mở rộng.
- Node.js sử dụng kiến trúc đơn luồng, bất đồng bộ và hướng sự kiện, giúp xử lý hàng nghìn kết nối đồng thời mà không bị chặn, tối ưu hóa hiệu suất. Với hệ thống quản lý gói mạnh mẽ npm (Node Package Manager), Node.js cung cấp hàng triệu thư viện hỗ trợ, từ đó giảm thiểu thời gian phát triển.
- Node.js được ứng dụng phổ biến trong các lĩnh vực như xây dựng ứng dụng web thời gian thực (chat, game), API backend (RESTful, GraphQL), xử lý luồng dữ liệu streaming (phát nhạc, video), và hệ thống microservices. Sự đơn giản, hiệu suất và khả năng mở rộng làm cho Node.js trở thành lựa chọn hàng đầu của các nhà phát triển hiện đại.

3.1.1. Cài đặt và triển khai phía BE

- Bước 1: Khởi tạo dự án Node: `npm init -y`
- Bước 2: Cài Hardhat vào project: `npm install --save-dev hardhat`
- Bước 3: Khởi tạo Hardhat: `npx hardhat`
- Bước 4: Chọn “Create a basic sample project”
- Bước 5: Test biên dịch contract: `npx hardhat compile`
- Bước 6: Chạy test để đảm bảo hoạt động trơn tru: `npx hardhat test`
- Bước 7: Cấu hình và xây các file TokenFactory.sol, Token.sol
- Bước 8: Cài đặt các thư viện cần thiết: Dotenv, uniswap v2, openzeppelin

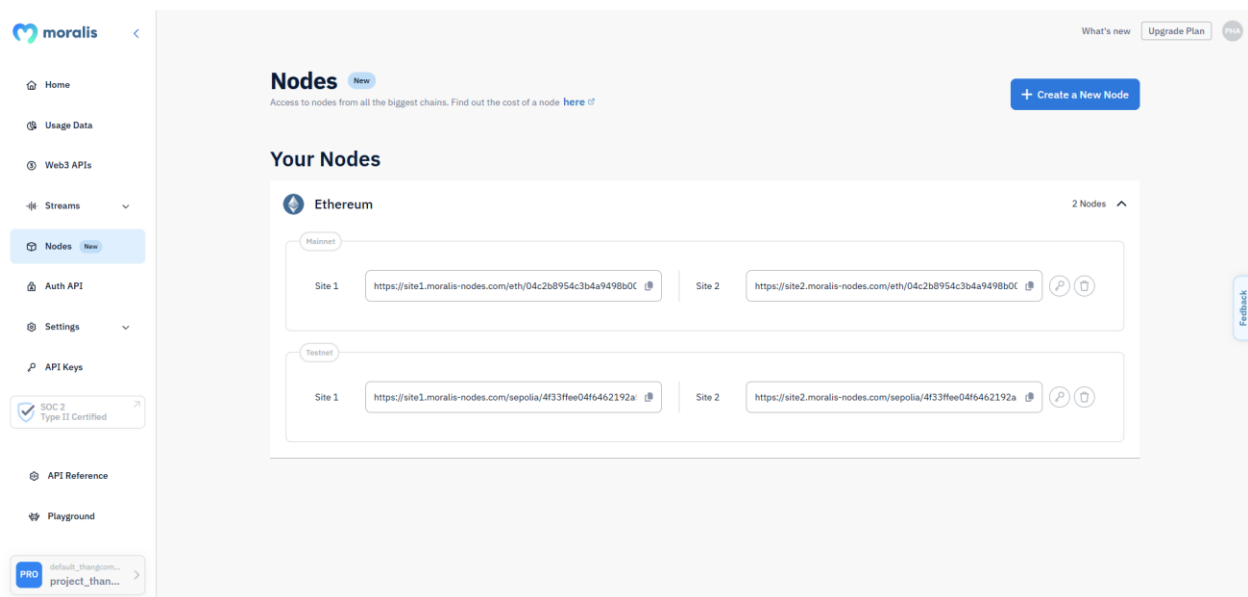
- Bước 9: Cấu hình file .env lưu private key, RPC_URL, Etherscan_API_KEY.

3.1.2. Cài đặt và triển khai phía FE

- Bước 1: Khởi tạo dự án mới: `npx create-react-app`
- Bước 2: Cấu hình routers, pages, providers cho website.
- Bước 3: Cài đặt các thư viện cần thiết.
- Bước 4: Cấu hình file .env để lưu API và Contract Address
- Bước 5: Tích hợp các thư viện, dịch vụ cần thiết.

3.1.3. Cài đặt Moralis API,URL

- Bước 1: Truy cập <https://admin.moralis.com/login> và sử dụng tài khoản để đăng nhập/đăng kí.
- Bước 2: Click vào giao diện Nodes và Create a New Node để tạo một Node trên mạng lưới Sepolia



Start creating your node

[Support](#)


Select Protocol *

Ethereum

Select Network *

Sepolia

Node information



Site 1


https://site1.moralis-nodes.com/sepolia/{Key to be added on creation}

Site 2

https://site2.moralis-nodes.com/sepolia/{Key to be added on creation}

Create Node

- Bước 3: Bấm chọn mục API Keys để Copy API Key của Moralis



Home
Usage Data
Web3 APIs
Streams
Nodes New
Auth API
Settings
API Keys
SOC 2 Type II Certified
API Reference
Playground

What's new
Upgrade Plan
PRO

API Keys

Unique Keys for secure access and communication between applications.

Name	API Key	Web3 API	Auth API	Streams	Created
default	*** **	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Read	<input checked="" type="checkbox"/> Read <input checked="" type="checkbox"/> Write	05/25/2025

Create New API Key

Feedback

3.1.4 Cài đặt code để tạo và lấy Token từ phía Back-end

Bước 1: Cài đặt các trọng số

- memeToken: Cấu trúc của 1 token khi khởi tạo:
 - name: Tên của Token.
 - symbol: Kí hiệu của Token.
 - description: Mô tả của Token.
 - tokenImageUrl: Đường dẫn tới ảnh của Token.
 - fundingRaised: Số ETH mà Token đã kêu gọi được.
 - tokenAddress: Địa chỉ của token.
 - creatorAddress: Địa chỉ của người tạo ra Token.
- DECIMALS: Phần thập phân của token.
- MAX_SUPPLY: Tổng cung tối đa mà một token có thể mint được.
- INIT_SUPPLY: Cung ban đầu, tương ứng 20% cung tối đa.
- MEMETOKEN_CREATION_FEE: Chi phí Token Creator phải trả cho web là 0.0001 ETH.
- MEMECOIN_FUNDING_GOAL: Số ETH mà một token tối đa có thể raise được.

```
// Cấu trúc của 1 token
struct memeToken{
    string name;
    string symbol;
    string description;
    string tokenImageUrl;
    uint fundingRaised;
    address tokenAddress;
    address creatorAddress;
}

address[] public memeTokenAddresses;

uint constant DECIMALS = 10 ** 18;
uint constant MAX_SUPPLY = 1000000 * DECIMALS; //Cung tối đa 1 triệu token
uint constant INIT_SUPPLY = 20 * MAX_SUPPLY / 100; //Cung ban đầu là 20% của cung tối đa

uint constant MEMETOKEN_CREATION_FEE = 0.0001 ether; //Phí tạo token

uint constant MEMECOIN_FUNDING_GOAL = 24 ether;
```

Bước 2: Cài đặt hàm tạo Token

- memeTokenCt: Tạo một token từ component Token
- memeTokenAddress: Lấy địa chỉ của memeTokenCt
- newlyCreatedToken: Thêm thông tin cho token


```
function createMemeToken(string memory name, string memory symbol, string memory description,
string memory imageUrl) public payable returns(address) {

    require(msg.value >= MEMETOKEN_CREATION_FEE, "Invalid token creation fee");
    Token memeTokenCt = new Token(name,symbol,INIT_SUPPLY);
    address memeTokenAddress = address(memeTokenCt);
    memeToken memory newlyCreatedToken = memeToken(name,symbol,description,imageUrl,0,memeTokenAddress,msg.sender);
    memeTokenAddresses.push(memeTokenAddress);
    addressToMemeTokenMapping[memeTokenAddress] = newlyCreatedToken;
    console.log(memeTokenAddress);
    return memeTokenAddress;
}
```

- Đây token vừa tạo vào một mảng để lưu các token đã tạo để liệt kê các danh sách token.
- Trả về địa chỉ token vừa tạo.

Bước 3: Cài đặt hàm lấy Token

- Duyệt các memeToken có trong mảng memeTokenAddresses và trả về một mảng memeToken

```
function getAllMemeTokens() public view returns(memeToken[] memory) {
    memeToken[] memory allTokens = new memeToken[](memeTokenAddresses.length);
    for (uint i = 0; i < memeTokenAddresses.length; i++) {
        allTokens[i] = addressToMemeTokenMapping[memeTokenAddresses[i]];
    }
    return allTokens;
}
```

3.1.5. Cài đặt thuật toán Exponential Bonding Curve

- calculateCost: Tính toán chi phí cần thiết để mua tokensToBuy khi currentSupply đã có.
- exponent1 và exponent2: Tính toán $k(x+n)$ và $k.x$
- exp1 và exp2: Gọi hàm exp để tính giá trị gần đúng của e^x
- cost: Tính chi phí với công thức của Exponential Bonding Curve và trả về chi phí.
- Hàm exp: Tính gần đúng e^x và trả về kết quả.

```
// Tính toán số lượng "tokensToBuy" từ "currentSupply"
function calculateCost(uint256 currentSupply, uint256 tokensToBuy) public pure returns (uint256) {

    // Exponential Bonding Curve

    // Calculate the exponent parts scaled to avoid precision loss
    uint256 exponent1 = (K * (currentSupply + tokensToBuy)) / 10**18;
    uint256 exponent2 = (K * currentSupply) / 10**18;

    // Calculate e^(kx) using the exp function
    uint256 exp1 = exp(exponent1);
    uint256 exp2 = exp(exponent2);

    // Công thức Cost: (P0 / k) * (e^(k * (currentSupply + tokensToBuy)) - e^(k * currentSupply))
    // We use (P0 * 10^18) / k to keep the division safe from zero
    uint256 cost = (INITIAL_PRICE * 10**18 * (exp1 - exp2)) / K; // Adjust for k scaling without dividing by zero
    return cost;
}

// Improved helper function to calculate e^x for larger x using a Taylor series approximation
function exp(uint256 x) internal pure returns (uint256) {
    uint256 sum = 10**18; // Start with 1 * 10^18 for precision
    uint256 term = 10**18; // Initial term = 1 * 10^18
    uint256 xPower = x; // Initial power of x

    for (uint256 i = 1; i <= 20; i++) { // Increase iterations for better accuracy
        term = (term * xPower) / (i * 10**18); // x^i / i!
        sum += term;

        // Prevent overflow and unnecessary calculations
        if (term < 1) break;
    }

    return sum;
}
```

3.1.6. Cài đặt hàm mua Token

- **Bước 1: Cài đặt các trọng số**
 - memeTokenAddress: Địa chỉ của Token muốn mua.
 - purchaseQty: Số lượng token muốn mua
 - listedToken: Gọi đến Token muốn mua
 - tokenCt: Gọi hàm token tới biến memeTokenAddress
 - currentSupply: Tổng cung hiện tại.
 - availableSupply: Số token khả dùng hiện tại có thể mua.
 - availableSupplyScaled: Lấy về đơn vị của user thay vì ERC20.
 - purchaseScaled: Đổi số lượng user muốn mua thành đơn vị ERC20.
 - currentSupplyScaled: Số token khả dụng hiện tại theo đơn vị user.
 - requiredEth: Số lượng ETH cần để mua 1 lượng token.

Bước 2: Cài đặt lệnh mint Token và kiểm tra Funding Goal

- Yêu cầu token chưa đạt Funding Raised.
- Gọi hàm mint: tokenCt.mint(purchaseQtyScaled,msg.sender) với số lượng muốn mua và gửi request tới Token Creator theo cơ chế Bonding Curve.

- Kiểm tra xem lượng Funding Raised đã đạt Funding Goal chưa, nếu đạt thì chuyển token đã tạo và ETH lên uniswap để tạo thanh khoản cặp Token/ETH và Burn lượng token ở trong Liquid.

```
function buyMemeToken(address memeTokenAddress, uint purchaseQty) public payable returns(uint) {
    require(addressToMemeTokenMapping[memeTokenAddress].tokenAddress != address(0), "Token is not exist");

    memeToken storage listedToken = addressToMemeTokenMapping[memeTokenAddress];

    // Yêu cầu token chưa đạt FundingRasie
    require(listedToken.fundingRaised <= MEMECOIN_FUNDING_GOAL, "Funding has already been raised");

    Token tokenCt = Token(memeTokenAddress);

    uint currentSupply = tokenCt.totalSupply();
    uint availableSupply = MAX_SUPPLY - currentSupply;

    uint availableSupplyScaled = availableSupply / DECIMALS;
    uint purchaseQtyScaled = purchaseQty * DECIMALS;

    require(purchaseQty <= availableSupplyScaled, "Not enough supply");

    // tính toán chi phí và số token nhận được (Bonding Curve)
    uint currentSupplyScaled = (currentSupply - INIT_SUPPLY) / DECIMALS;
    uint requiredEth = calculateCost(currentSupplyScaled, purchaseQty);

    console.log("Required eth for purchase is ", requiredEth);

    require(msg.value >= requiredEth, "Incorrect value of ETH sent");

    listedToken.fundingRaised += msg.value;

    tokenCt.mint(purchaseQtyScaled, msg.sender);

    console.log("User token balance is ", tokenCt.balanceOf(msg.sender));

    if (listedToken.fundingRaised >= MEMECOIN_FUNDING_GOAL) {
        // tạo pool trên Uniswap
        address pool = _createLiquidityPool(memeTokenAddress);
        console.log("pool created address is ", pool);

        // cung cấp thanh khoản vào cổng pool
        uint ethAmount = listedToken.fundingRaised;
        uint liquidity = _provideLiquidity(memeTokenAddress, INIT_SUPPLY, ethAmount);
        console.log("Liquidity added to pool ", liquidity);
        // đốt lượng token ở liquid trong liquidity position

        _burnlpTokens(pool, liquidity);
    }

    return requiredEth;
}
```

3.1.7. Cài đặt tạo, thêm và burn thanh khoản trên Uniswap

Bước 1: Cài đặt hàm tạo Liquid trên Uniswap

- Gọi 2 hàm IUniswapV2Factory và IUniswapV2Router01 của thư viện Uniswap để add thanh khoản cho cặp memeTokenAddress/WETH.
- Pair dùng để lưu địa chỉ của cặp thanh khoản và trả về chúng.

Bước 2: Cài đặt hàm thêm Liquid trên Uniswap

- memeTokenCt: Tạo biến lấy Token của memeTokenAddress
- router: Lấy Router của Uniswapv2
- Gọi hàm addLiquidityETH để thêm liquid vào cho pair

Bước 3: Cài đặt hàm Burn Liquid trên Uniswap

- uniswapv2pairct: Gọi đến Pool của pair.
- Sử dụng lệnh transfer để chuyển lượng liquid đó đến địa chỉ ví 0 (là địa chỉ ví không tồn tại) => Burn

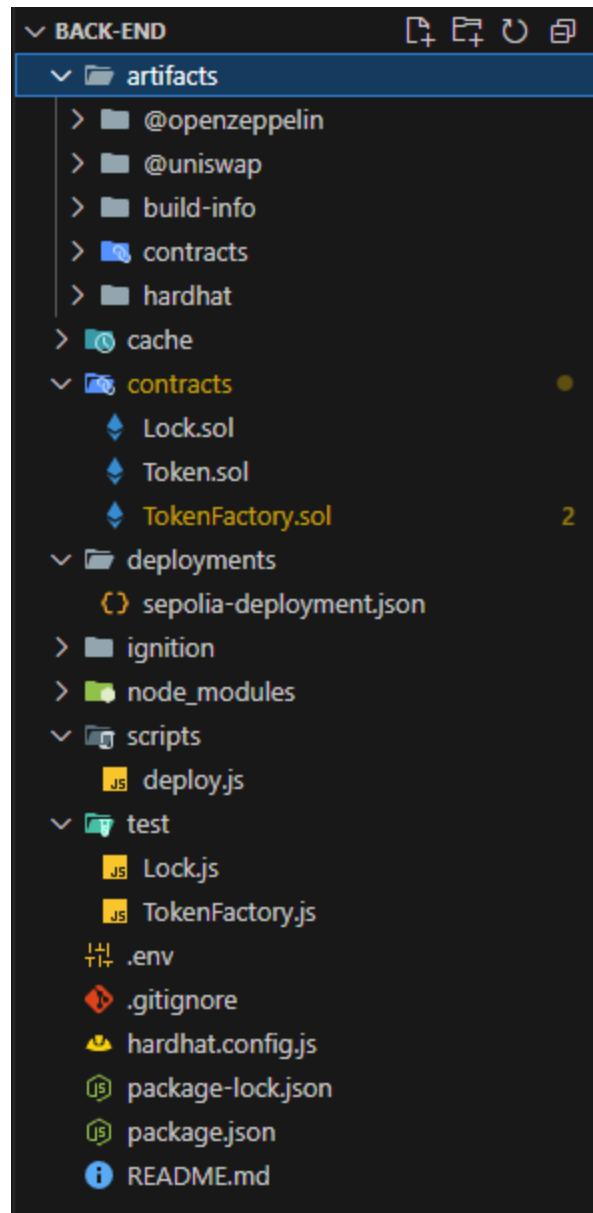
```
function _createLiquidityPool(address memeTokenAddress) internal returns(address) {
    IUniswapV2Factory factory = IUniswapV2Factory(UNISWAP_V2_FACTORY_ADDRESS);
    IUniswapV2Router01 router = IUniswapV2Router01(UNISWAP_V2_ROUTER_ADDRESS);
    address pair = factory.createPair(memeTokenAddress, router.WETH());
    return pair;
}

function _provideLiquidity(address memeTokenAddress, uint tokenAmount, uint ethAmount) internal returns(uint) {
    Token memeTokenCt = Token(memeTokenAddress);
    memeTokenCt.approve(UNISWAP_V2_ROUTER_ADDRESS, tokenAmount);
    IUniswapV2Router01 router = IUniswapV2Router01(UNISWAP_V2_ROUTER_ADDRESS);
    (uint amountToken, uint amountETH, uint liquidity) = router.addLiquidityETH(
        value: ethAmount
    )(memeTokenAddress, tokenAmount, tokenAmount, ethAmount, address(this), block.timestamp);
    return liquidity;
}

function _burnLPTokens(address pool, uint liquidity) internal returns(uint){
    IUniswapV2Pair uniswapv2pairct = IUniswapV2Pair(pool);
    uniswapv2pairct.transfer(address(0), liquidity);
    console.log("LP Tokens burnt ", liquidity);
    return 1;
}
```

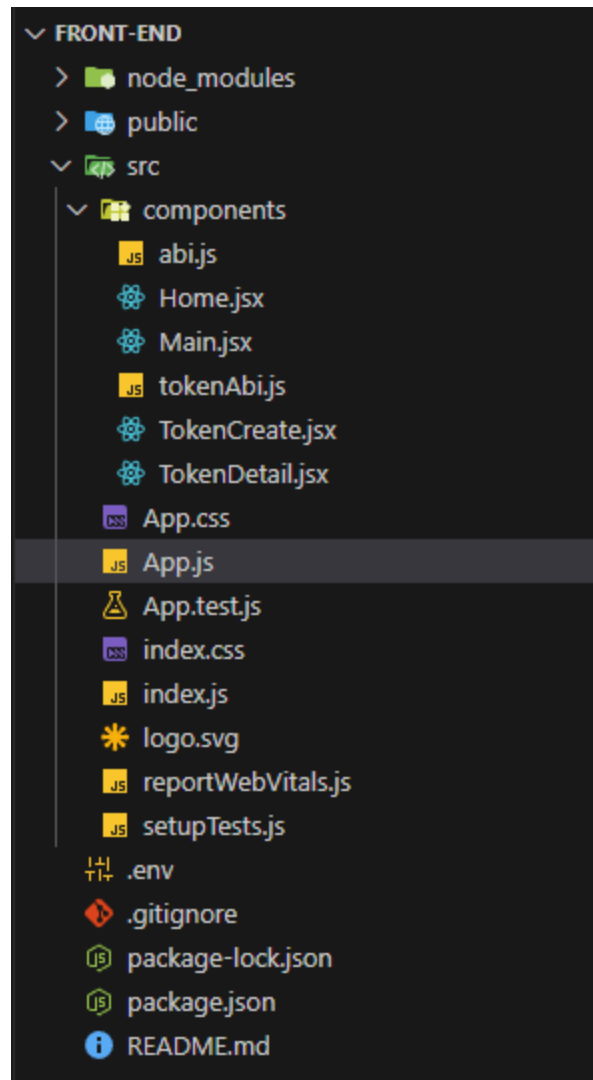
3.2. Kết quả cài đặt**3.2.1. Triển khai phía Back-end**

Sau khi cài đặt, khai báo các contracts ta có cấu trúc thư mục của ứng dụng phía back-end quản lý như sau:



Hình 3.1: Cấu trúc thư mục của Back-end

3.2.2. Triển khai phía Front-end



Hình 3.2: Cấu trúc thư mục phía Front-end

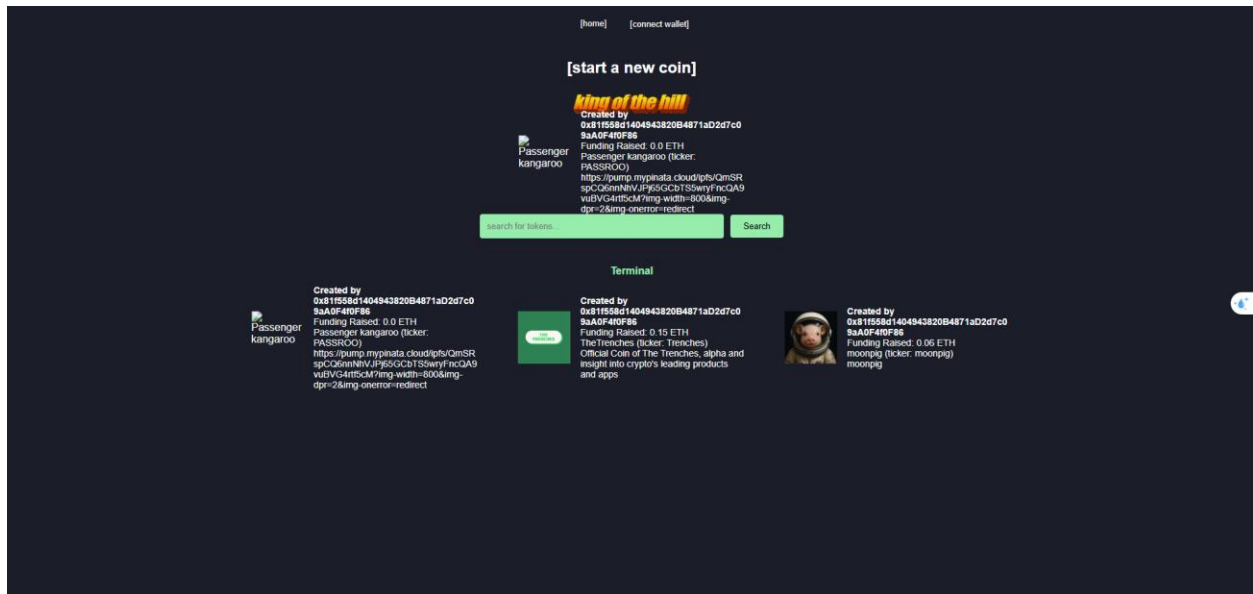
3.3. Kết luận

3.3.1. Kết quả đạt được

- Qua quá trình thực hiện, trang web giúp người dùng tự tạo token trên mạng lưới Sepolia đã hoàn thành với các chức năng cơ bản, bao gồm tìm kiếm, trang chủ, chi tiết token, tạo token. Ngoài ra, hệ thống còn tích hợp được các công nghệ hiện đại như ReactJS, Hardhat, Moralis giúp tối ưu hóa hiệu suất và bảo mật.
- **Contract Address** đã deploy trên mạng Sepolia:
0x4B7Bc58EBD3f1BAD60DCb0baC23080BECC50F61c

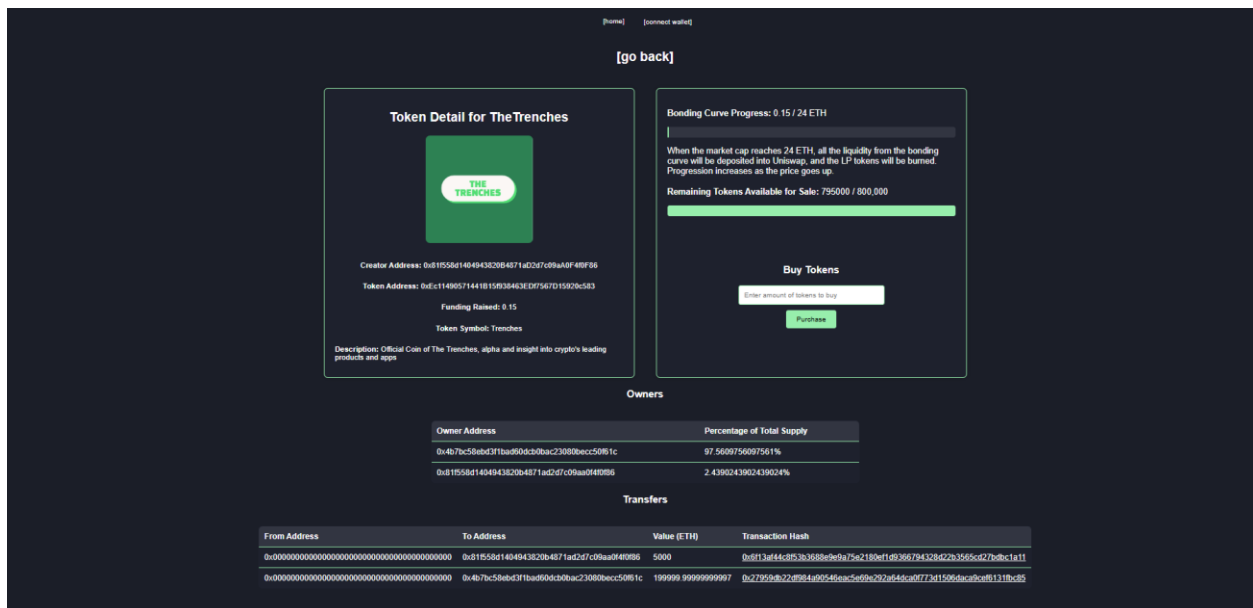
Các giao diện chính:

Giao diện trang chủ:



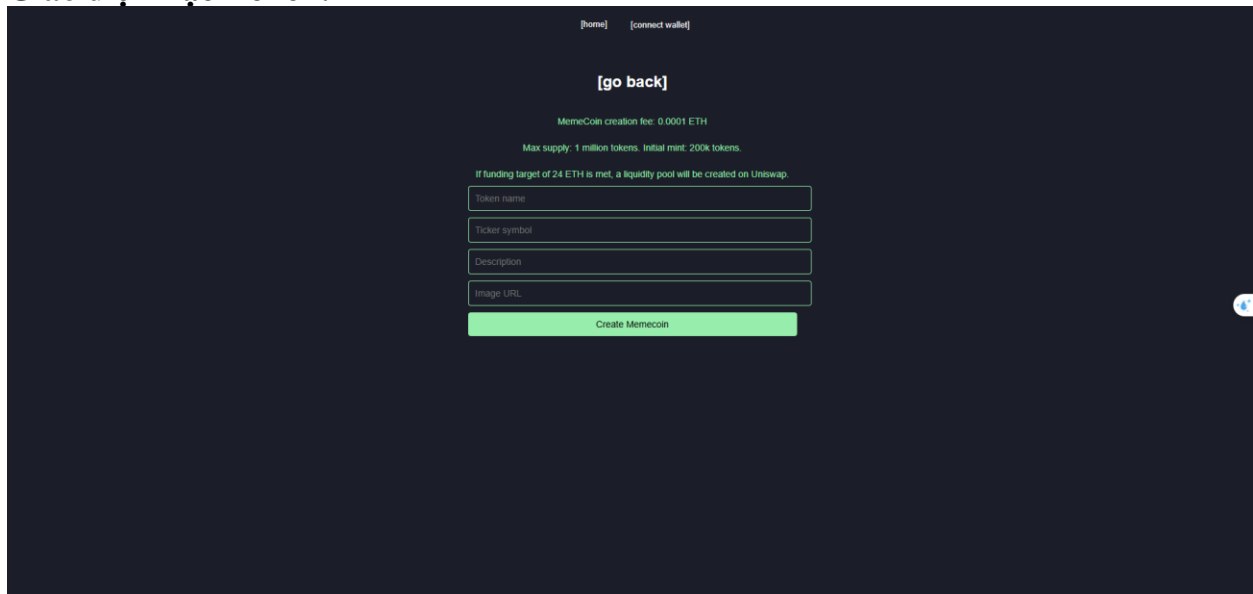
Hình 3.3: Hình ảnh giao diện trang chủ

Giao diện Token Detail:



Hình 3.4: Hình ảnh giao diện Token Detail

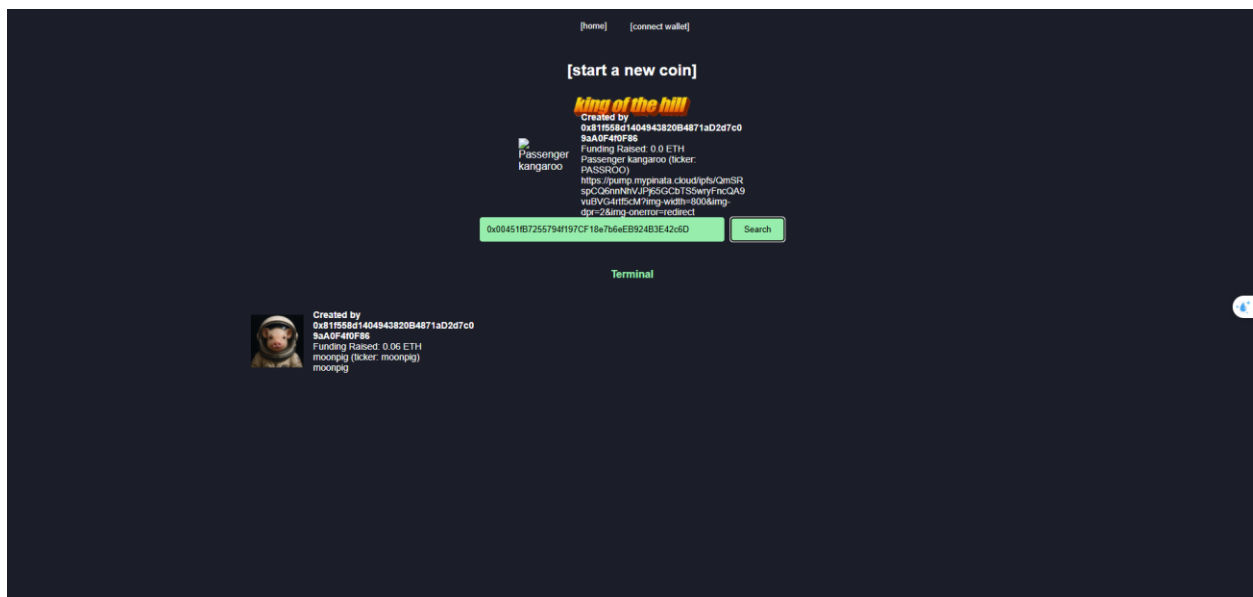
Giao diện Tạo Token:



The screenshot shows a web interface for creating a MemeCoin. At the top, there are links for [home] and [connect wallet]. Below them is a [go back] button. The interface displays the MemeCoin creation fee as 0.0001 ETH and the maximum supply as 1 million tokens, with an initial mint of 200k tokens. A note states that if the funding target of 24 ETH is met, a liquidity pool will be created on Uniswap. There are four input fields: Token name, Ticker symbol, Description, and Image URL. A green button labeled 'Create MemeCoin' is at the bottom.

Hình 3.5: Hình ảnh giao diện Tạo Token

Giao diện tìm kiếm Token:



The screenshot shows a web interface for searching MemeCoins. At the top, there are links for [home] and [connect wallet]. Below them is a [start a new coin] button. The interface displays a search result for a coin named 'King of the hill'. The coin was created by 0x811956ed1404943820B4871aD2d7c09aA0F40F86. The funding raised is 0.6 ETH. The ticker is 'PASSROO'. The image URL is https://img.nypeneta.cloud/9fuQmSR5gC20nnNivJp5G2nTSwYFnCQA9vuBVG4rtf5cM/img-width=800&img-opr=Z&img-onerror=redirect. A green button labeled 'Search' is at the bottom. Below the search result, there is a 'Terminal' section showing a list of coins created by the same address, including 'moonpig' and 'moonpig'.

Hình 3.6: Hình ảnh giao diện tìm kiếm Token

3.3.2. Hạn chế

- Mặc dù hệ thống moondotfun đã đạt được những kết quả tích cực, nhưng vẫn còn một số điểm hạn chế cần cải thiện.

- Thứ nhất, người dùng chưa thể tự update image từ máy, người dùng vẫn phải tải ảnh hoặc lấy source từ một bên thứ 3 để thêm vào mục imageURL khi tạo token.
- Thứ hai, người dùng hiện tại chỉ có thể mua token từ web nhưng chưa thể bán được token, điều này dẫn đến việc ETH của người dùng bị web nắm giữ.

3.3.3. Định hướng phát triển

- Định hướng phát triển của Moondotfun trong tương lai sẽ tập trung vào việc hoàn thiện việc bán token và cho người dùng tự do đăng hình ảnh lên từ máy tính.
- Cải thiện giao diện người dùng. Tối ưu hóa việc giao dịch của người dùng cho trơn tru mượt mà.
- Cải thiện thêm nhiều khả năng đăng nhập, người dùng có thể đăng nhập bằng các bên web2 như Gmail, Twitter(X), hệ thống sẽ tạo cho người dùng một ví và người dùng có thể trích xuất private key của ví đó.
- Tăng cường bảo mật trên Blockchain, cải thiện Smart Contract để tránh Hacker xâm nhập tạo lỗ hổng và khai thác ETH trên đó.
- Đưa Moondotfun vào Ethereum Mainnet sau khi hoàn thành những tính năng giúp người dùng trải nghiệm tốt nhất và tạo ra doanh thu.

3.4. TÀI LIỆU THAM KHẢO

Hardhat – “Hardhat Docs”, hardhat.org/hardhat-runner/docs/getting-started#overview

Binance Academy – “What is a Bonding Curve in Crypto”, academy.binance.com/en/articles/what-is-a-bonding-curve-in-crypto

Moralis – Moralis Web3 Docs, docs.moralis.com/