ĐẠI HỌC BÁCH KHOA HÀ NỘI

ĐỒ ÁN TỐT NGHIỆP

Thiết kế xây dựng công nghệ thực tế ảo và ứng dụng

NGUYỄN VĂN A
nguyenvanabc@sis.hust.edu.vn

Chương trình đào tạo: Công nghệ thông tin Việt-Nhật

Giảng viên hướng dẫn: PGS. TS. Phạm Văn ABC

Khoa: Kỹ thuật máy tính

Trường: Công nghệ thông tin và Truyền thông

HÀ NỘI, 06/2022

LỜI CẢM ƠN

Lời cảm ơn của sinh viên (SV) tới người yêu, gia đình, bạn bè, thầy cô, và chính bản thân mình vì đã chăm chỉ và quyết tâm thực hiện ĐATN để đạt kết quả tốt nhất, nên viết phần cảm ơn ngắn gọn, tránh dùng các từ sáo rỗng, giới hạn trong khoảng 100-150 từ.

TÓM TẮT NỘI DUNG ĐỒ ÁN

Trong bối cảnh NFT trở thành một hình thức tài sản số được quan tâm rộng rãi, nhu cầu về một nền tảng giao dịch NFT thân thiện, dễ sử dụng và hỗ trợ quản lý hiệu quả ngày càng rõ rệt. Thực tế khảo sát cho thấy nhiều NFT marketplace hiện nay tuy đầy đủ chức năng nhưng trải nghiệm người dùng còn phức tạp, đặc biệt với người mới: thao tác mua/bán nhiều bước, thông tin hiển thị dày đặc, khó tập trung vào các yếu tố quan trọng để ra quyết định như giá, đặc điểm NFT, độ uy tín, lịch sử giao dịch và xu hướng biến động. Bên cạnh đó, người sáng tạo nội dung thường thiếu một khu vực tổng hợp giúp theo dõi hiệu quả bộ sưu tập và hoạt động giao dịch một cách trực quan. Từ những vấn đề đó, đồ án hướng tới xây dựng một NFT Marketplace theo định hướng lấy người dùng làm trung tâm, tối ưu trải nghiệm và đơn giản hoá quy trình, tập trung vào các hoạt động kinh doanh cốt lõi gồm tạo bộ sưu tập, phát hành NFT, niêm yết, mua/bán, quản lý tài sản và theo dõi thống kê. Hệ thống ưu tiên trình bày rõ ràng các thông tin phục vụ quyết định giao dịc, đồng thời cung cấp dashboard cơ bản giúp người dùng nắm nhanh tình hình tài sản và hiệu quả giao dịch. Kết quả kỳ vọng của đồ án là tạo ra một phiên bản marketplace vận hành được các luồng nghiệp vụ chính, mang lại trải nghiệm dễ tiếp cận hơn cho người dùng phổ thông và tạo nền tảng để phát triển thêm các tính năng nâng cao trong các giai đoạn tiếp theo.

Sinh viên thực hiện

(Ký và ghi rõ họ tên)

ABSTRACT

In the context where NFTs are gaining broader adoption as a form of digital asset, there is a clear demand for a marketplace that is user-friendly, easy to navigate, and supports effective asset management. A review of existing NFT marketplaces shows that while many platforms offer comprehensive features, the user experience can be overwhelming—especially for newcomers—due to complex buying/selling flows, information overload, and a lack of focus on what truly matters for decision-making such as pricing, key attributes, ownership, transaction history, and price trends. In addition, creators often lack a simple, consolidated view to track the performance of their collections and trading activity in a straightforward way. Based on these issues, this project aims to build an NFT Marketplace with a business-oriented, user-centered approach that streamlines the core lifecycle: creating collections, minting NFTs, listing, buying/selling, managing assets, and viewing essential analytics. The system prioritizes clear presentation of decision-critical information (price, attributes, ownership, history, and trends) and provides a basic dashboard that helps users quickly understand asset status and trading performance. The expected outcome is a functional marketplace that supports the main business flows end-to-end, lowers the entry barrier for general users, and serves as a solid foundation for future enhancements and scaling.

MỤC LỤC

DANH MỤC HÌNH VẼ

DANH MỤC BẢNG BIỂU

DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

API	Giao diện lập trình ứng dụng (Application Programming Interface)
EUD	Phát triển ứng dụng người dùng cuối(End-User Development)
GWT	Công cụ lập trình Javascript bằng Java của Google (Google Web Toolkit)
HTML	Ngôn ngữ đánh dấu siêu văn bản (HyperText Markup Language)
IaaS	Dịch vụ hạ tầng

CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

1.1 Đặt vấn đề

Trong vài năm trở lại đây, khái niệm tài sản số và công nghệ chuỗi khối (blockchain) không còn xa lạ với cộng đồng công nghệ cũng như người dùng phổ thông. Cùng với đó, các tài sản số không thể thay thế (Non-Fungible Token – NFT) xuất hiện như một cách mới để gắn danh tính và quyền sở hữu duy nhất cho từng đối tượng số, từ tranh vẽ, âm nhạc, vật phẩm trong game cho đến các bộ sưu tập sưu tầm. Thông qua NFT, người sáng tạo có thể phát hành tác phẩm của mình lên blockchain, còn nhà sưu tầm có thể mua, bán, trao đổi và theo dõi lịch sử sở hữu của từng tài sản.

Tuy nhiên, khi bắt đầu tiếp cận các sàn giao dịch NFT, đặc biệt là các nền tảng quốc tế, người dùng – nhất là người mới làm quen với Web3 – thường gặp nhiều rào cản:

l

- Giao diện và trải nghiệm người dùng (UX/UI) tương đối phức tạp, đòi hỏi người dùng phải hiểu rõ về ví điện tử, phí gas, chuẩn NFT và các khái niệm mới trong blockchain trước khi có thể thao tác một cách tự tin.
- Việc quản lý bộ sưu tập (collection) và từng NFT thường rời rạc, thiếu các công cụ trực quan để người sáng tạo theo dõi hiệu quả kinh doanh, mức độ quan tâm cũng như giá trị của tác phẩm theo thời gian.
- Các số liệu phân tích như lịch sử giá, thống kê doanh thu, xu hướng giao dịch hay xếp hạng (ranking) đôi khi bị phân tán hoặc trình bày khó hiểu, khiến cả nhà sưu tầm lẫn người sáng tạo gặp khó khăn khi đưa ra quyết định.
- Ở góc độ trải nghiệm, nhiều nền tảng hiện có còn tồn tại các rào cản như chi phí giao dịch cao, quy trình thao tác phức tạp, thiếu các tính năng hỗ trợ cộng đồng và tương tác, làm giảm động lực tham gia của người dùng mới.
- Hệ thống lưu trữ metadata và hình ảnh NFT thường dựa trên các giải pháp phân tán như IPFS hoặc những dịch vụ đòi hỏi cấu hình phức tạp, không thân thiện về mặt phát triển và có thể gây ra vấn đề về hiệu năng, dẫn đến trải nghiệm chưa tốt cho người dùng.

Từ những khó khăn nêu trên, trong khuôn khổ đồ án tốt nghiệp, em lựa chọn nghiên cứu và hiện thực một trang web NFT marketplace với giao diện gần gũi, tập trung hỗ trợ người dùng tạo, mua bán, quản lý và theo dõi giá trị NFT một cách trực quan. Hệ thống được thiết kế để vừa đáp ứng các yêu cầu kỹ thuật cốt lõi (tích

hợp smart contract, xử lý giao dịch on-chain) vừa hướng đến trải nghiệm sử dụng đơn giản, rõ ràng cho cả người sáng tạo nội dung và nhà sưu tầm.

1.2 Mục tiêu và phạm vi đề tài

Trên cơ sở tham khảo các hệ thống hiện có, các đồ án liên quan và nhu cầu thực tế, đồ án đặt ra các mục tiêu chính như sau:

- **Xây dựng một website NFT marketplace hoàn chỉnh**: cho phép người dùng kết nối ví điện tử (wallet), thực hiện quá trình tạo (mint) NFT, đưa NFT lên sàn (list), mua/bán NFT và xem lại lịch sử giao dịch của mình. Các giao dịch mua bán được thực hiện thông qua smart contract nhằm đảm bảo tính an toàn và minh bạch.
- **Quản lý bộ sưu tập (collection)**: hỗ trợ người sáng tạo khởi tạo collection, thêm NFT vào các collection hiện có, quản lý thông tin mô tả và trạng thái niêm yết của từng NFT trong từng collection. Mỗi collection có thể được theo dõi độc lập về hoạt động và hiệu quả.
- **Cung cấp công cụ tra cứu và phân tích**: hiển thị trang chi tiết cho từng NFT và collection, bao gồm giá hiện tại, lịch sử giá (price history), số lượng giao dịch, thống kê doanh thu theo mốc thời gian. Hệ thống cung cấp một dashboard để người dùng có cái nhìn tổng quan về hoạt động mua bán và sáng tạo của mình.
- **Quản lý tài khoản và hồ sơ người dùng**: cung cấp trang hồ sơ cho phép người dùng theo dõi thông tin cá nhân, danh sách NFT đang sở hữu, NFT đã tạo, NFT đang rao bán và lịch sử giao dịch cá nhân. Hệ thống áp dụng cơ chế phân quyền và xác thực rõ ràng để bảo vệ dữ liệu người dùng và tránh truy cập trái phép.
- **Đảm bảo an toàn và minh bạch giao dịch**: mọi hoạt động mua, bán, chuyển nhượng NFT đều được quản lý bởi smart contract, hạn chế tối đa việc phụ thuộc vào xử lý off-chain cho các bước quan trọng của giao dịch. Smart contract được thiết kế có tính đến các lỗ hổng phổ biến như tấn công **Reentrancy** và sử dụng thư viện OpenZeppelin để tăng độ tin cậy.
- **Tối ưu hóa quy trình upload và lưu trữ**: sử dụng cơ chế presigned URL để người dùng có thể upload hình ảnh NFT trực tiếp lên dịch vụ lưu trữ đám mây (AWS S3), giảm tải cho server backend, rút ngắn thời gian xử lý và giúp hệ thống vận hành ổn định hơn.
- **Tối ưu hóa hiệu năng với caching**: tích hợp Redis như một caching layer để lưu trữ tạm thời các dữ liệu được truy cập thường xuyên (danh sách NFT,

thông tin collection, thống kê người dùng, . . .), từ đó giảm số lượng truy vấn đến cơ sở dữ liệu và cải thiện đáng kể thời gian phản hồi của API.

Phạm vi thực hiện của đồ án tập trung vào:

- Thiết kế và xây dựng **phiên bản web chạy trên PC và hỗ trợ responsive trên thiết bị di động** cho hệ thống NFT marketplace với kiến trúc ba lớp: frontend (Next.js), backend (NestJS) và smart contract (Solidity).
- Hỗ trợ mạng blockchain **Binance Smart Chain (BSC) Testnet** làm môi trường triển khai và kiểm thử chính. Hệ thống được định hướng có thể nâng cấp để triển khai lên BSC Mainnet. Smart contract sử dụng chuẩn token ERC-721 cho NFT.
- Xây dựng các chức năng cốt lõi: xác thực người dùng thông qua ví điện tử **Metamask**, tạo và quản lý NFT/collection, mua/bán NFT thông qua smart contract, hiển thị các chỉ số thống kê và lịch sử giá. Hệ thống đồng bộ dữ liệu on-chain với cơ sở dữ liệu off-chain để phục vụ nhu cầu tìm kiếm, lọc và hiển thị nhanh.
- Lưu trữ metadata và hình ảnh NFT trên **AWS S3** thông qua cơ chế presigned URL, ưu tiên tính đơn giản, dễ triển khai và hiệu quả trong việc upload, phân phối nội dung.
- Các nội dung nằm ngoài phạm vi của phiên bản đầu tiên bao gồm: hỗ trợ đa chuỗi (multi-chain), tích hợp các cổng thanh toán fiat on-ramp, các cơ chế đấu giá phức tạp, hay các tính năng DeFi nâng cao như staking hoặc lending.

1.3 Định hướng giải pháp

Để đạt được các mục tiêu trên, hệ thống được định hướng xây dựng theo các hướng chính sau:

- **Tách bạch on-chain và off-chain**:
  - Phần on-chain (smart contract) chịu trách nhiệm các nghiệp vụ cốt lõi liên quan đến quyền sở hữu NFT, quá trình mint, tạo collection, niêm yết, mua/bán và thu phí nền tảng. Smart contract sử dụng các cơ chế bảo vệ như ReentrancyGuard và dựa trên thư viện OpenZeppelin nhằm hạn chế lỗi bảo mật.
  - Phần off-chain (backend server và cơ sở dữ liệu) tập trung quản lý dữ liệu phục vụ giao diện: metadata mở rộng, lịch sử giao dịch chi tiết, thống kê, thông tin người dùng, . . . Backend lắng nghe các sự kiện (event) phát sinh từ blockchain và cập nhật vào cơ sở dữ liệu, đồng thời vẫn có thể truy vấn

trực tiếp blockchain khi cần.

- **Thiết kế kiến trúc web hiện đại**:
  - **Frontend**: sử dụng Next.js 14 với App Router để tận dụng các kỹ thuật Server-Side Rendering (SSR) và Static Site Generation (SSG), giúp cải thiện hiệu năng tải trang và hỗ trợ SEO tốt hơn. Giao diện được xây dựng với Tailwind CSS kết hợp Ant Design nhằm đảm bảo tính thống nhất và khả năng responsive. Trạng thái ứng dụng và dữ liệu được quản lý bằng Redux Toolkit và React Query, giúp đồng bộ dữ liệu giữa UI và backend một cách hiệu quả.
  - **Backend**: sử dụng NestJS với mô hình module-based, tách biệt rõ ràng giữa controller, service và repository. Hệ thống API RESTful được mô tả bằng Swagger để thuận tiện trong kiểm thử và tích hợp. Backend phụ trách xử lý nghiệp vụ, xác thực người dùng (JWT), đồng bộ dữ liệu với blockchain, tương tác với PostgreSQL và Redis.
  - Kiến trúc được định hướng mở để có thể dễ dàng bổ sung các module thống kê nâng cao, xếp hạng hoặc gợi ý NFT trong tương lai.
- **Lưu trữ dữ liệu NFT và metadata hợp lý**:
  - Ảnh và các tệp dung lượng lớn liên quan đến NFT được lưu trữ trên **AWS S3** thông qua cơ chế presigned URL. Frontend trực tiếp upload tệp lên S3, giúp giảm tải cho backend và rút ngắn thời gian xử lý. Đường dẫn tới file trên S3 được lưu trong metadata của NFT và trong cơ sở dữ liệu.
  - Cơ sở dữ liệu quan hệ PostgreSQL lưu trữ các thông tin phục vụ tra cứu và thống kê: người dùng, collection, thông tin NFT, lịch sử giá, lịch sử giao dịch, . . . Bên cạnh đó, **Redis** được dùng làm caching layer cho các dữ liệu được truy cập nhiều như danh sách NFT phổ biến, thông tin collection hay một số thống kê tổng quan. Redis được cấu hình với thời gian sống (TTL) phù hợp và cơ chế làm mới (refresh) để cân bằng giữa hiệu năng và tính cập nhật của dữ liệu.
- **Tập trung vào trải nghiệm người dùng**:
  - Quy trình kết nối ví, tạo NFT, niêm yết và mua bán được chia thành các bước rõ ràng, có hướng dẫn trực quan để người mới có thể làm theo dễ dàng. Giao diện responsive giúp người dùng thao tác thuận tiện trên cả máy tính và thiết bị di động.
  - Hệ thống cung cấp các trang tổng quan (dashboard) cho người sáng tạo

và nhà sưu tầm, hiển thị các chỉ số như tổng tài sản, doanh thu, doanh thu theo mốc thời gian (24 giờ, 7 ngày, 30 ngày), các giao dịch gần nhất và xu hướng mua bán.

- **Chi phí sử dụng hợp lý**: trong khuôn khổ đồ án, hệ thống sử dụng **BSC** – một mạng blockchain có chi phí giao dịch thấp, tốc độ xử lý nhanh và tương thích với nhiều ví phổ biến. Điều này giúp quá trình thử nghiệm cũng như trải nghiệm của người dùng trở nên dễ tiếp cận hơn.
- **Đảm bảo tính minh bạch và khả năng truy vết**: lịch sử giao dịch của từng NFT được lưu trữ đầy đủ, bao gồm chủ sở hữu cũ, chủ sở hữu mới, giá giao dịch và mã giao dịch (transaction hash) trên blockchain. Người dùng có thể kiểm tra lại các thông tin này bất cứ lúc nào, góp phần tăng mức độ tin cậy đối với hệ thống.
- **Tối ưu hóa hiệu năng và khả năng mở rộng**: việc sử dụng Redis để cache các dữ liệu truy cập thường xuyên giúp giảm tải cho cơ sở dữ liệu PostgreSQL và cải thiện tốc độ đáp ứng khi số lượng người dùng cũng như số lượng NFT tăng lên. Cơ chế cập nhật và vô hiệu hóa cache (cache invalidation) được thiết kế để đảm bảo dữ liệu hiển thị luôn bám sát trạng thái thực tế của hệ thống.

1.4 Bố cục đồ án

Các nội dung còn lại của báo cáo đồ án tốt nghiệp được sắp xếp như sau:

- **Chương 2 – Khảo sát và phân tích yêu cầu**: trình bày tổng quan thị trường NFT trong và ngoài nước, phân tích một số nền tảng NFT marketplace tiêu biểu (như OpenSea, Rarible) và các đồ án liên quan. Từ đó, chương này tổng hợp các yêu cầu chức năng, phi chức năng cho hệ thống đề xuất và mô tả chúng thông qua biểu đồ use case, kịch bản sử dụng, biểu đồ hoạt động, đồng thời nêu các yêu cầu về hiệu năng, bảo mật và khả năng mở rộng.
- **Chương 3 – Công nghệ sử dụng**: giới thiệu các công nghệ, nền tảng và thư viện được áp dụng trong dự án như blockchain BSC, smart contract với Solidity và Hardhat, các công cụ phát triển frontend (Next.js, React, Tailwind CSS, Ant Design) và backend (NestJS, TypeORM, PostgreSQL), dịch vụ lưu trữ (AWS S3), hệ thống caching (Redis), cùng các thư viện hỗ trợ (ethers.js, wagmi, Redux Toolkit, React Query). Chương này cũng nêu lý do lựa chọn các công nghệ trên so với những phương án thay thế.
- **Chương 4 – Thiết kế, triển khai và đánh giá hệ thống**: mô tả kiến trúc tổng thể của hệ thống NFT marketplace (ba lớp: frontend, backend, smart contract), thiết kế cơ sở dữ liệu (schema và quan hệ giữa các bảng), thiết kế

smart contract (NFTMarketplace, NFTCollection), thiết kế API backend (các endpoint chính, cơ chế authentication/authorization) và giao diện người dùng (các màn hình, component và luồng tương tác). Chương này cũng trình bày quá trình hiện thực từng module, kèm theo hình ảnh minh họa và kết quả kiểm thử.

- **Chương 5 – Các giải pháp và đóng góp nổi bật**: tổng hợp các điểm nhấn của hệ thống so với những nền tảng tham khảo, chẳng hạn kiến trúc tách bạch on-chain/off-chain, quy trình upload thông qua presigned URL, cơ chế Redis caching để tối ưu hiệu năng, dashboard thống kê chi tiết và hệ thống lịch sử giao dịch rõ ràng, minh bạch. Với mỗi giải pháp, chương này nêu bài toán đặt ra, cách tiếp cận, công nghệ sử dụng và kết quả đạt được.
- **Chương 6 – Kết luận và hướng phát triển**: tóm tắt các kết quả chính của đồ án, đánh giá mức độ hoàn thành so với mục tiêu ban đầu, chỉ ra những hạn chế hiện tại (như chưa hỗ trợ đa chuỗi, chưa triển khai cơ chế đấu giá) và đề xuất các hướng mở rộng trong tương lai (đa chuỗi, bổ sung các loại tài sản số khác, tối ưu hiệu năng cho quy mô người dùng lớn, . . .).

Tóm lại, chương này giới thiệu bối cảnh, lý do lựa chọn đề tài, mục tiêu, phạm vi và định hướng giải pháp cho hệ thống website NFT marketplace. Các chương tiếp theo sẽ lần lượt đi sâu vào phần khảo sát, thiết kế, triển khai và đánh giá chi tiết hệ thống.

CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

Chương này trình bày quá trình khảo sát hiện trạng các hệ thống NFT marketplace, phân tích nhu cầu của người dùng và xác định các yêu cầu đối với hệ thống cần xây dựng. Trên cơ sở đó, em đề xuất tập các chức năng chính, mô hình hoá dưới dạng biểu đồ use case, đặc tả chi tiết một số use case quan trọng và đưa ra các yêu cầu phi chức năng. Những nội dung này là cơ sở để thiết kế kiến trúc, cơ sở dữ liệu, giao diện và các module trong các chương tiếp theo.

2.1 Khảo sát hiện trạng

2.1.1 Khảo sát UI/UX và mô hình phí của các NFT marketplace

Các hệ thống NFT marketplace phổ biến hiện nay (như OpenSea, Blur, Magic Eden, . . .) đều cung cấp tương đối đầy đủ các chức năng chính từ tạo NFT, quản lý bộ sưu tập, niêm yết, mua/bán cho đến thống kê giao dịch. Tuy nhiên, khi quan sát ở góc độ người dùng mới và so sánh trải nghiệm thực tế, có thể nhận thấy rằng bức tranh UI/UX của các nền tảng này khá phức tạp và đôi khi gây khó khăn cho người ít kinh nghiệm về Web3.

Trước hết, giao diện của nhiều nền tảng được thiết kế theo hướng dồn rất nhiều thông tin và thao tác lên cùng một màn hình. Trang chi tiết NFT thường hiển thị đồng thời các trường kỹ thuật như địa chỉ smart contract, token ID dạng số lớn, chuẩn token (ERC-721, ERC-1155), network, trạng thái contract đã được xác minh hay chưa, cùng với lịch sử giao dịch chi tiết. Khi thực hiện giao dịch mua/bán, người dùng thường phải trải qua nhiều bước xác nhận: phê duyệt (approve) token, ký nhiều giao dịch liên tiếp, lựa chọn gas price/gas limit, . . . Các thao tác này tuy quen thuộc với người dùng Web3 lâu năm, nhưng đối với người mới, việc phải xử lý nhiều khái niệm kỹ thuật liên tiếp dễ tạo cảm giác rối rắm và thiếu tự tin khi thao tác.

Bên cạnh đó, nhiều thông tin kỹ thuật khó hiểu xuất hiện ngay ở lớp giao diện đầu tiên. Địa chỉ ví và địa chỉ contract thường được hiển thị dưới dạng chuỗi hex dài; thông tin block, transaction hash, timestamp on-chain được trình bày chi tiết; các tham số gas (gas price, gas limit) và network fee được diễn đạt bằng nhiều đơn vị (Gwei, ETH, USD); các flag kỹ thuật như “contract not verified”, “token standard”, “on-chain metadata”, . . . cũng xuất hiện dày đặc. Về mặt chuyên môn, đây là những thông tin quan trọng, nhưng việc đưa chúng lên quá sớm và quá nhiều khiến người dùng phổ thông khó hình dung đâu là thông tin cần tập trung để ra quyết định (mua/bán, niêm yết) và đâu là thông tin chỉ mang tính tham khảo kỹ thuật.

Ở khía cạnh quản lý tài sản, cách tổ chức giao diện để quản lý collection và NFT trên nhiều nền tảng cũng chưa thật sự thân thiện. Các thao tác xem, lọc, quản lý nhiều collection và nhiều NFT của một creator thường trải qua nhiều màn hình và nhiều tab, sử dụng nhiều bộ lọc và phân trang. Người sáng tạo nếu muốn nắm được hiệu quả của từng collection (doanh thu, số lượng đã bán, giá trung bình, ...) thường phải tự tổng hợp từ nhiều nơi, thay vì có một dashboard tập trung, đơn giản, hiển thị các chỉ số chính một cách trực quan.

Một vấn đề khác là mô hình phí nền tảng. Nhiều marketplace thu phí nền tảng trên mỗi giao dịch ở mức đáng kể (ví dụ khoảng 2–2.5% giá trị giao dịch), ngoài phần phí gas mà người dùng phải trả cho mạng blockchain. Với những creator nhỏ hoặc người sưu tầm giao dịch các NFT có giá trị không quá lớn, tổng chi phí phải trả (gas + platform fee) trở thành rào cản không nhỏ, khiến họ ngại thử nghiệm hoặc giao dịch với tần suất cao.

Trong bối cảnh đó, đề tài của em hướng đến một cách tiếp cận khác: giao diện đơn giản hơn, tập trung vào các bước nghiệp vụ chính nhưng vẫn cung cấp đầy đủ thông tin cần thiết cho quyết định của người dùng. Cụ thể, hệ thống tập trung vào các luồng cơ bản như kết nối ví, tạo NFT, tạo collection, niêm yết, mua, xem chi tiết và xem dashboard; các thông tin kỹ thuật phức tạp (địa chỉ contract, token ID, gas limit, nonce, ...) sẽ không được đưa vào dày đặc trong giao diện chính, mà được ẩn dưới dạng phần "chi tiết nâng cao" nếu người dùng có nhu cầu xem. Thay vào đó, các thông tin quan trọng cho quyết định giao dịch như giá hiện tại, lịch sử giá (price history), số lần giao dịch, chủ sở hữu hiện tại, mô tả và các thuộc tính chính của NFT, cùng với các chỉ số tổng quan trên dashboard (tổng doanh thu, số NFT đã bán, tổng số giao dịch, xu hướng tăng/giảm theo mốc thời gian) sẽ được ưu tiên hiển thị rõ ràng.

Về mô hình phí, trong phạm vi đồ án, hệ thống được thiết kế với định hướng phí nền tảng thấp hơn mặt bằng chung (ví dụ khoảng 1% mỗi giao dịch), nhằm khuyến khích người dùng thử nghiệm và giao dịch thường xuyên. Kiến trúc smart contract và backend được xây dựng theo hướng cho phép điều chỉnh mức phí này thông qua cấu hình, tạo sự linh hoạt nếu triển khai trong môi trường thực tế. Tóm lại, trọng tâm của hệ thống là đơn giản hoá UI/UX, trì hoãn hoặc ẩn bớt những thông tin kỹ thuật khó hiểu ở bước đầu, nhưng vẫn đảm bảo cung cấp đầy đủ dữ liệu quan trọng để người dùng, kể cả không chuyên, có thể nhanh chóng làm quen và ra quyết định trên marketplace.

2.1.2 So sánh và đánh giá hiện trạng

Dựa trên việc khảo sát một số nền tảng tiêu biểu, em tổng hợp một bảng so sánh khái quát một số đặc điểm quan trọng liên quan đến đối tượng phục vụ, cách thiết kế giao diện, mức độ phức tạp UI/UX và mô hình phí nền tảng như sau:

Tiêu chí	**OpenSea**	**Blur**	**Magic Eden**
Đối tượng chính	Nhà sưu tầm, nghệ sĩ, dự án NFT đa dạng	Trader, nhà giao dịch chuyên nghiệp, tập trung vào volume	Cộng đồng sưu tầm NFT theo từng hệ sinh thái (Solana, đa chuỗi)
Giao diện	Nhiều thông tin, nhiều tab; kết hợp thông tin kỹ thuật và thông tin giao dịch trên cùng màn hình	UI tối ưu cho giao dịch nhanh, bảng lệnh dày đặc, nhiều filter nâng cao	Giao diện tương đối trực quan nhưng vẫn chứa nhiều thông tin kỹ thuật trên trang chi tiết
Mức độ phức tạp UI/UX	Cao đối với người mới: phải hiểu contract, network, gas, . . .	Rất cao đối với người mới, nhiều khái niệm chuyên cho trader	Trung bình, thân thiện hơn nhưng vẫn yêu cầu hiểu một số khái niệm kỹ thuật
Thông tin kỹ thuật hiển thị	Địa chỉ contract, token ID, chuẩn token, network, transaction history chi tiết, . . .	Nhiều thông tin về lệnh, order book, nguồn dữ liệu tổng hợp từ nhiều sàn	Contract, chain, thuộc tính chi tiết; liên kết đến explorer, . . .
Dashboard cho người dùng	Có nhưng phân tán theo nhiều trang, nhiều tab	Tập trung vào chỉ số giao dịch (volume, PnL)	Có một số trang thống kê theo collection, chưa quá sâu cho từng người dùng
Phí nền tảng (platform fee)	Ở mức tương đối cao so với giao dịch nhỏ (thường khoảng vài phần trăm mỗi giao dịch, chưa tính gas)	Tối ưu cho trader, kết hợp nhiều nguồn, vẫn có phí nền tảng/aggregator	Tương tự, thu phí trên mỗi giao dịch, phụ thuộc từng collection và chính sách

Bảng 2.1: So sánh khái quát một số đặc điểm UI/UX và mô hình phí của các NFT marketplace

Từ bảng so sánh này có thể nhận thấy một số đặc điểm chung. Thứ nhất, các nền tảng lớn đều cố gắng phục vụ đồng thời nhiều nhóm người dùng (creator, collector,

trader chuyên nghiệp), dẫn đến việc giao diện phải gánh nhiều chức năng và nhiều loại thông tin trên cùng hệ thống. Điều này giúp đáp ứng được nhu cầu đa dạng, nhưng cũng làm tăng độ phức tạp, đặc biệt với người dùng mới.

Thứ hai, thông tin kỹ thuật xuất hiện khá dày đặc trên các trang quan trọng như trang chi tiết NFT hay trang giao dịch. Đối với người dùng đã quen thuộc với Web3, đây là điểm cộng vì họ có thể kiểm tra nhiều thông tin một cách nhanh chóng. Ngược lại, với người dùng phổ thông, việc phải đối diện với các khái niệm như contract address, network, gas, explorer, . . . ngay từ đầu lại trở thành rào cản tâm lý không nhỏ.

Thứ ba, mô hình phí nền tảng kết hợp với phí gas khiến tổng chi phí cho mỗi giao dịch không hề thấp, đặc biệt đối với các giao dịch có giá trị trung bình hoặc thấp. Mức phí nền tảng vài phần trăm giá trị giao dịch là có thể chấp nhận được với các dự án lớn hoặc nhà giao dịch chuyên nghiệp, nhưng đối với người dùng nhỏ lẻ, đây là yếu tố khiến họ cân nhắc kỹ trước khi tham gia.

2.1.3 Kết luận khảo sát hiện trạng

Từ các phân tích và so sánh ở trên, em rút ra một số định hướng chính cho hệ thống NFT marketplace trong phạm vi đồ án. Về mặt giao diện, hệ thống cần được thiết kế theo hướng đơn giản hoá, tối ưu cho người dùng không chuyên: các thông tin kỹ thuật khó hiểu như contract address chi tiết, gas limit, nonce, thông tin block, . . . nên được ẩn khỏi luồng thao tác chính và chỉ xuất hiện khi người dùng chủ động xem phần "chi tiết kỹ thuật". Thay vào đó, các thông tin quan trọng cho quyết định giao dịch như giá hiện tại, lịch sử giá, số lượng giao dịch, mô tả và các thuộc tính cốt lõi của NFT cần được hiển thị rõ ràng, dễ đọc.

Bên cạnh đó, hệ thống nên tập trung vào một số luồng nghiệp vụ chính, được trình bày mạch lạc: kết nối ví, tạo NFT, tạo collection, niêm yết, mua NFT, xem dashboard cá nhân và dashboard collection. Mỗi luồng được chia thành các bước rõ ràng, có hướng dẫn cụ thể, hạn chế việc người dùng phải chuyển qua lại quá nhiều tab hay màn hình.

Một điểm quan trọng khác là việc cung cấp các dashboard đơn giản nhưng đủ dùng. Người sáng tạo cần có khả năng nhanh chóng xem được tổng doanh thu, số NFT đã bán, hiệu quả của từng collection mà không phải tự tổng hợp dữ liệu. Người sưu tầm cần thấy được lịch sử mua/bán, tổng giá trị tài sản và xu hướng giá của các NFT đang sở hữu để có cơ sở đưa ra quyết định.

Cuối cùng, về chi phí, hệ thống được định hướng với mức phí nền tảng thấp hơn mặt bằng chung, ví dụ khoảng 1% mỗi giao dịch, nhằm giảm gánh nặng chi phí

cho người dùng, đặc biệt là các creator nhỏ và người sưu tầm giao dịch với giá trị không quá lớn nhưng tần suất có thể cao. Kiến trúc smart contract và backend sẽ được xây dựng sao cho mức phí này có thể cấu hình linh hoạt, tạo điều kiện thuận lợi nếu hệ thống được mở rộng trong tương lai.

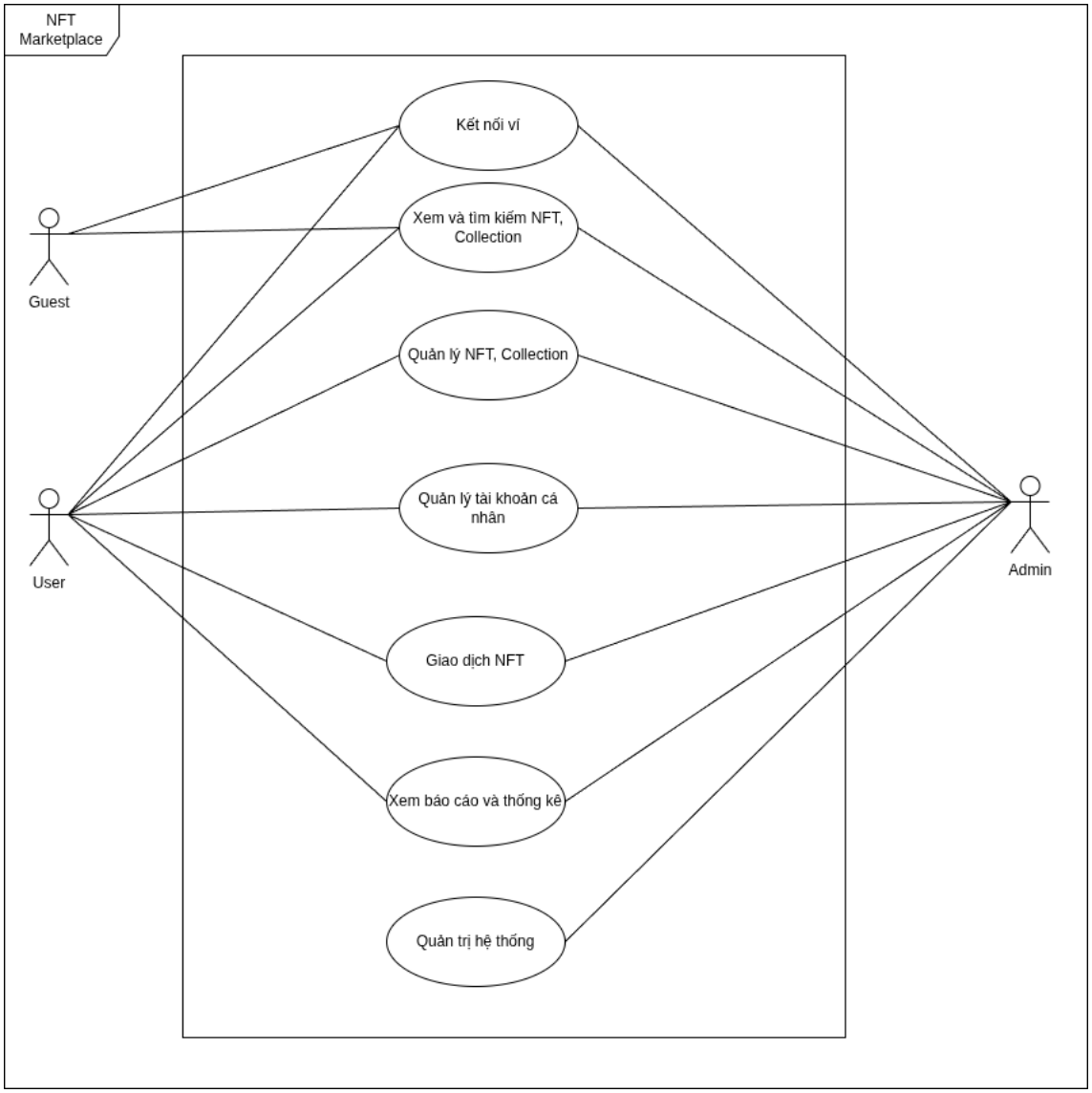
Những định hướng nêu trên sẽ là cơ sở để xác định tập chức năng và yêu cầu chi tiết của hệ thống trong các mục tiếp theo của chương này.

2.2 Tổng quan chức năng

Trong phần này, em trình bày tổng quan các chức năng chính của hệ thống dưới dạng biểu đồ use case tổng quát và các nhóm use case phân rã. Mục tiêu là cung cấp một cái nhìn toàn cảnh về những gì hệ thống cần hỗ trợ, trước khi đi vào đặc tả chi tiết từng use case cụ thể.

2.2.1 Biểu đồ usecase tổng quát

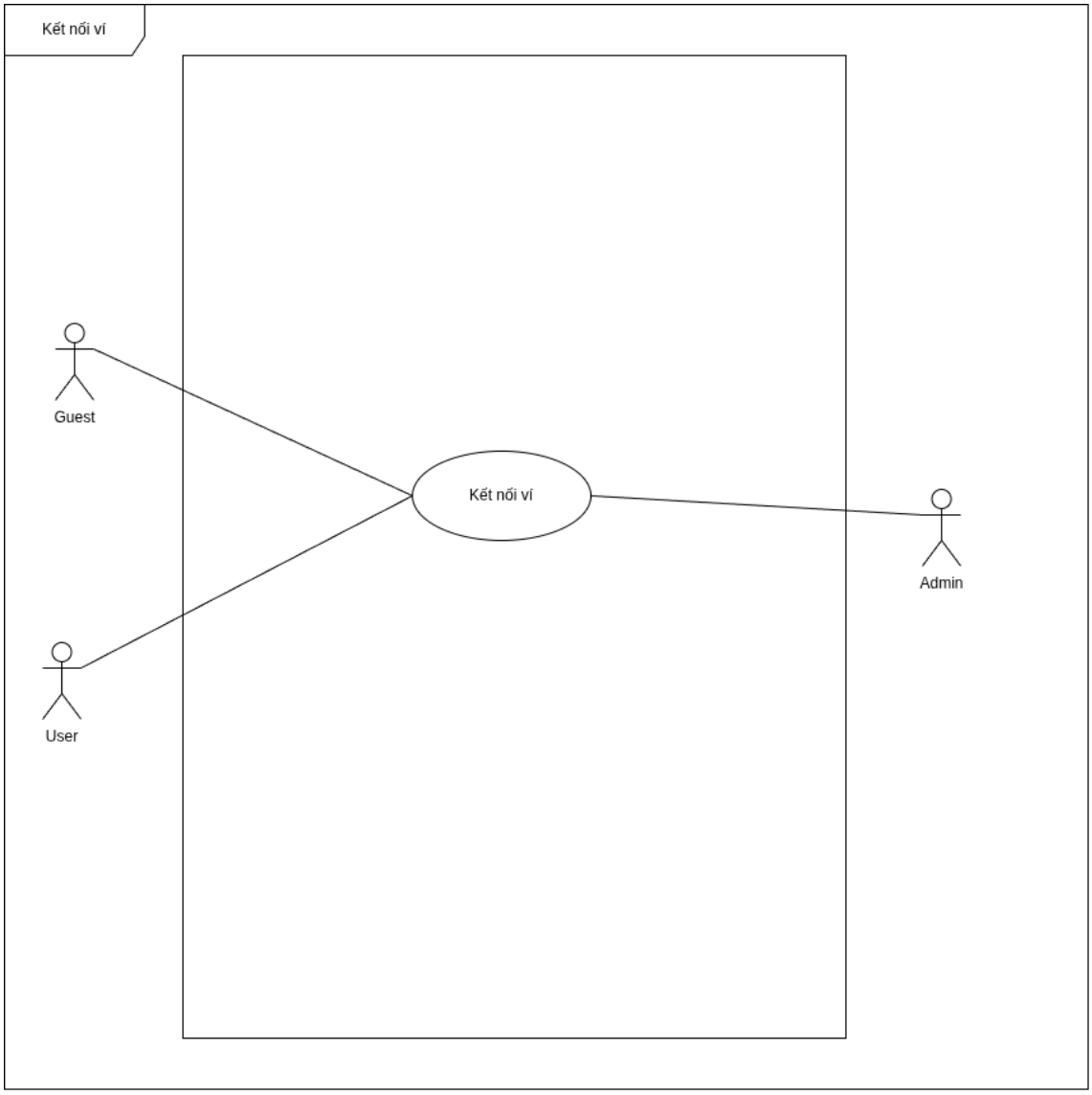
Hình 4.1 mô tả tổng quan các chức năng tổng quan của hệ thống. Các tác nhân gồm có Khách(Guest), User và Admin



Hình 2.1: Biểu đồ usecase tổng quan

2.2.2 Biểu đồ usecase phân rã Kết nối ví

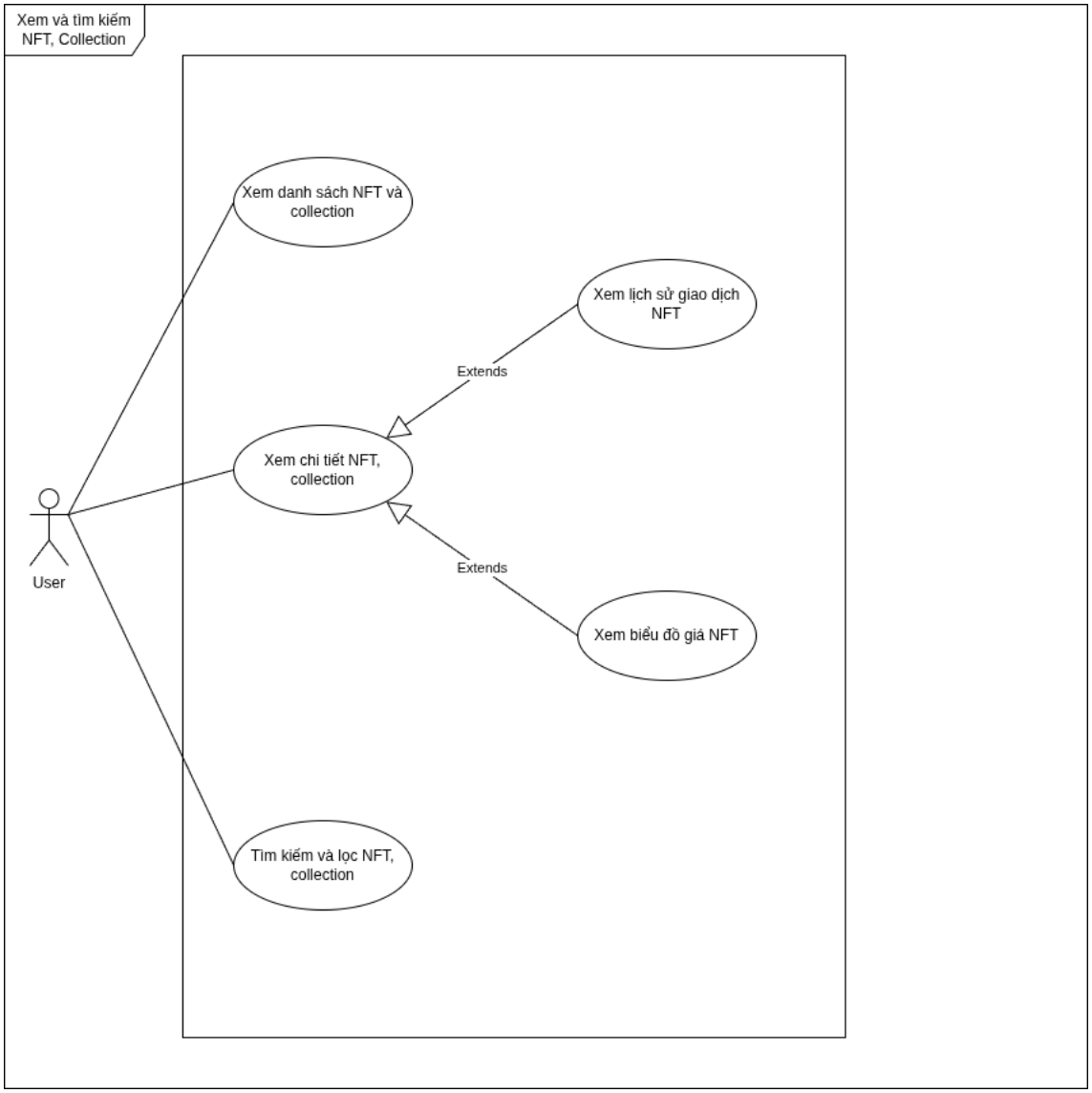
Hình 4.2 mô tả người dùng có thể kết nối đến ví Metamask



Hình 2.2: Biểu đồ usecase phân rã Kết nối ví

2.2.3 Biểu đồ usecase phân rã xem và tìm kiếm NFT, Collection

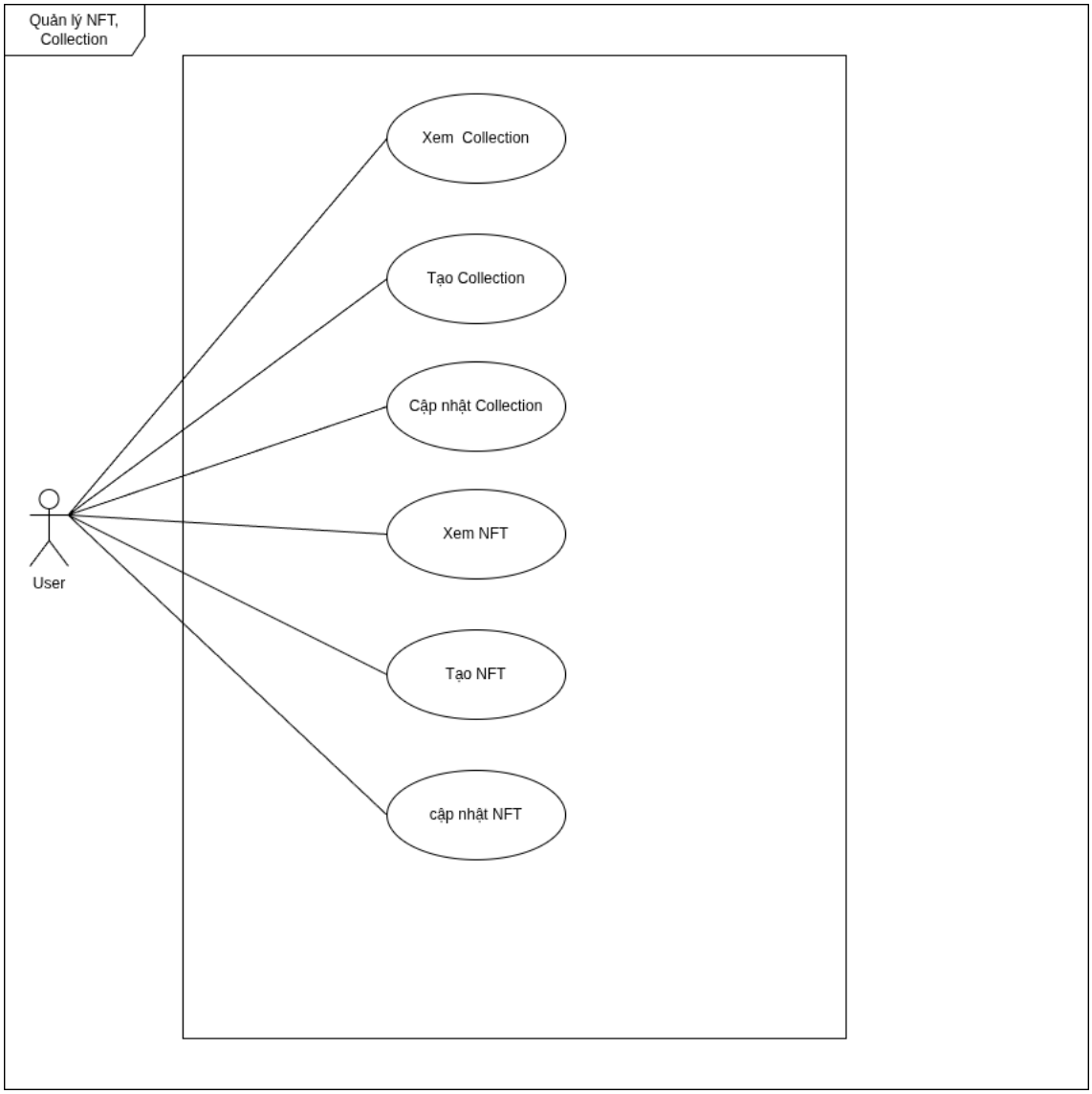
Hình 4.3 mô tả người dùng có thể tìm kiếm và xem thông tin của NFT và Collection



Hình 2.3: Biểu đồ usecase phân rã Xem và tìm kiếm NFT, Collection

2.2.4 Biểu đồ usecase phân rã quản lý NFT và Collection

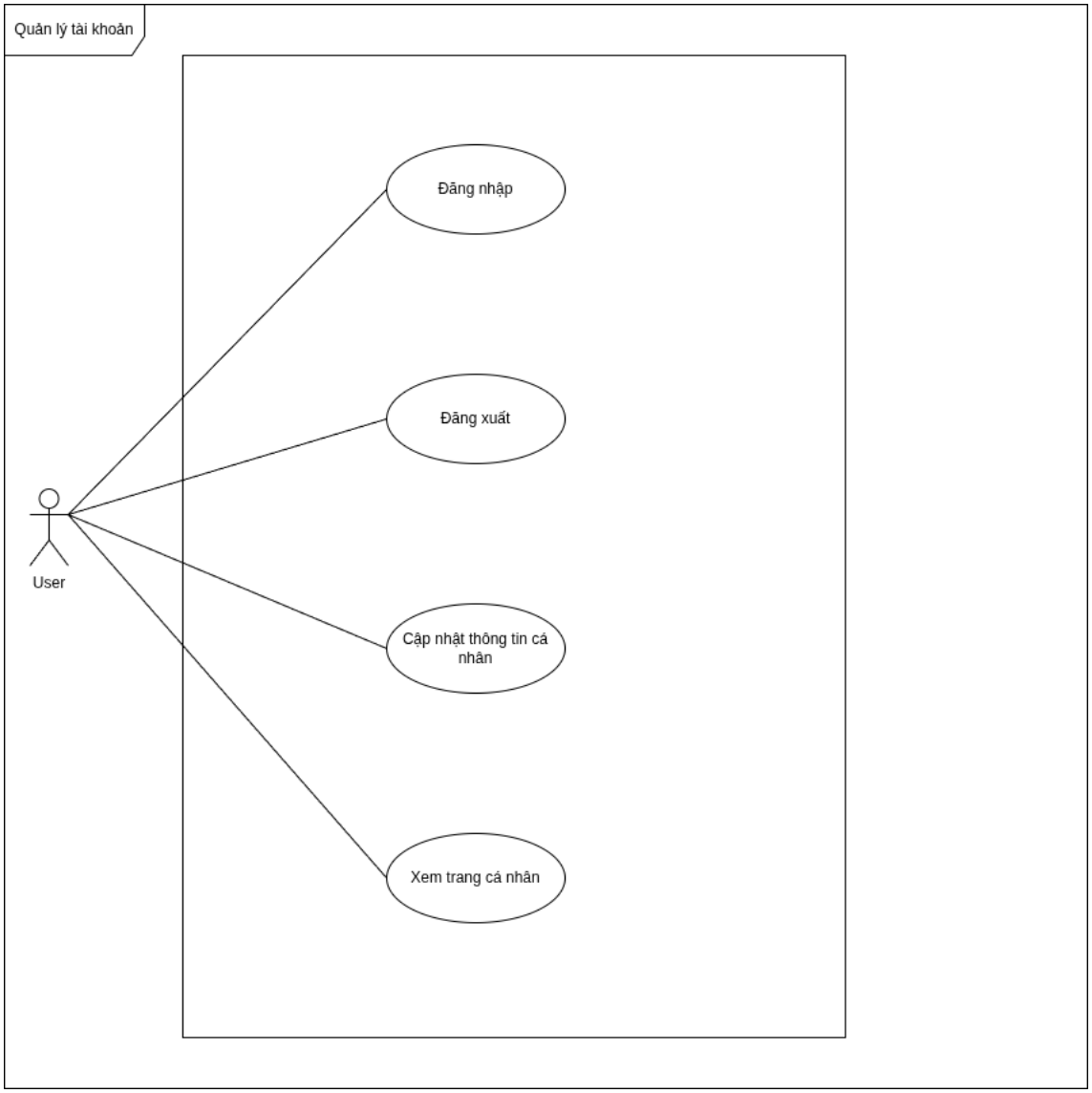
Hình 4.4 mô tả người dùng có thể quản lý thông tin và thay đổi NFT và Collection



Hình 2.4: Biểu đồ usecase quản lý NFT và Collection

2.2.5 Biểu đồ usecase phân rã quản lý tài khoản cá nhân

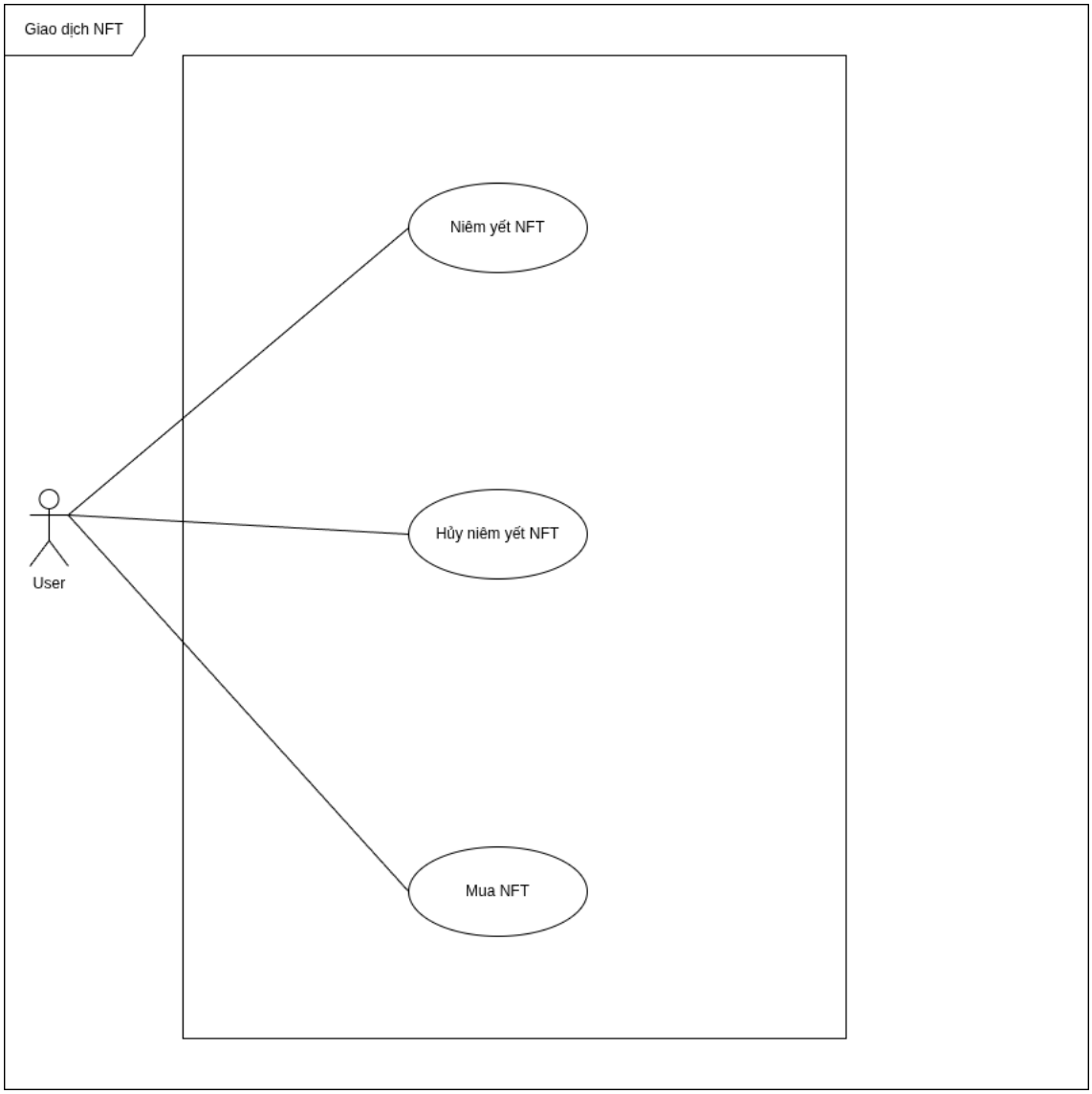
Hình 4.5 mô tả người dùng có thể quản lý thông tin cá nhân của mình



Hình 2.5: Biểu đồ usecase quản lý tài khoản cá nhân

2.2.6 Biểu đồ usecase phân rã giao dịch NFT

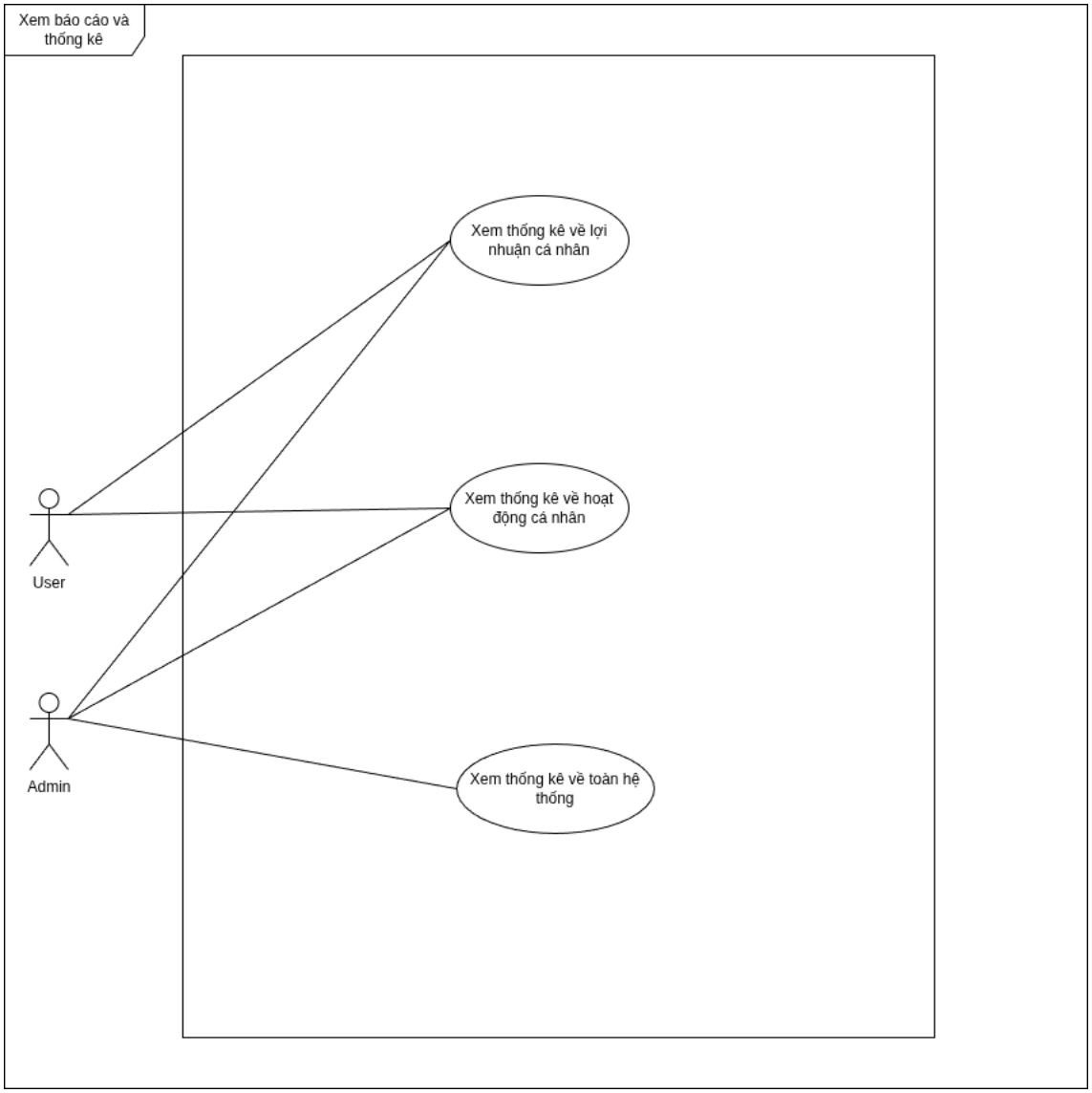
Hình 4.6 mô tả người dùng có thể thực hiện hành đông mua/bán NFT trên hệ thống



Hình 2.6: Biểu đồ usecase giao dịch NFT

2.2.7 Biểu đồ usecase phân rã xem báo cáo và thống kê

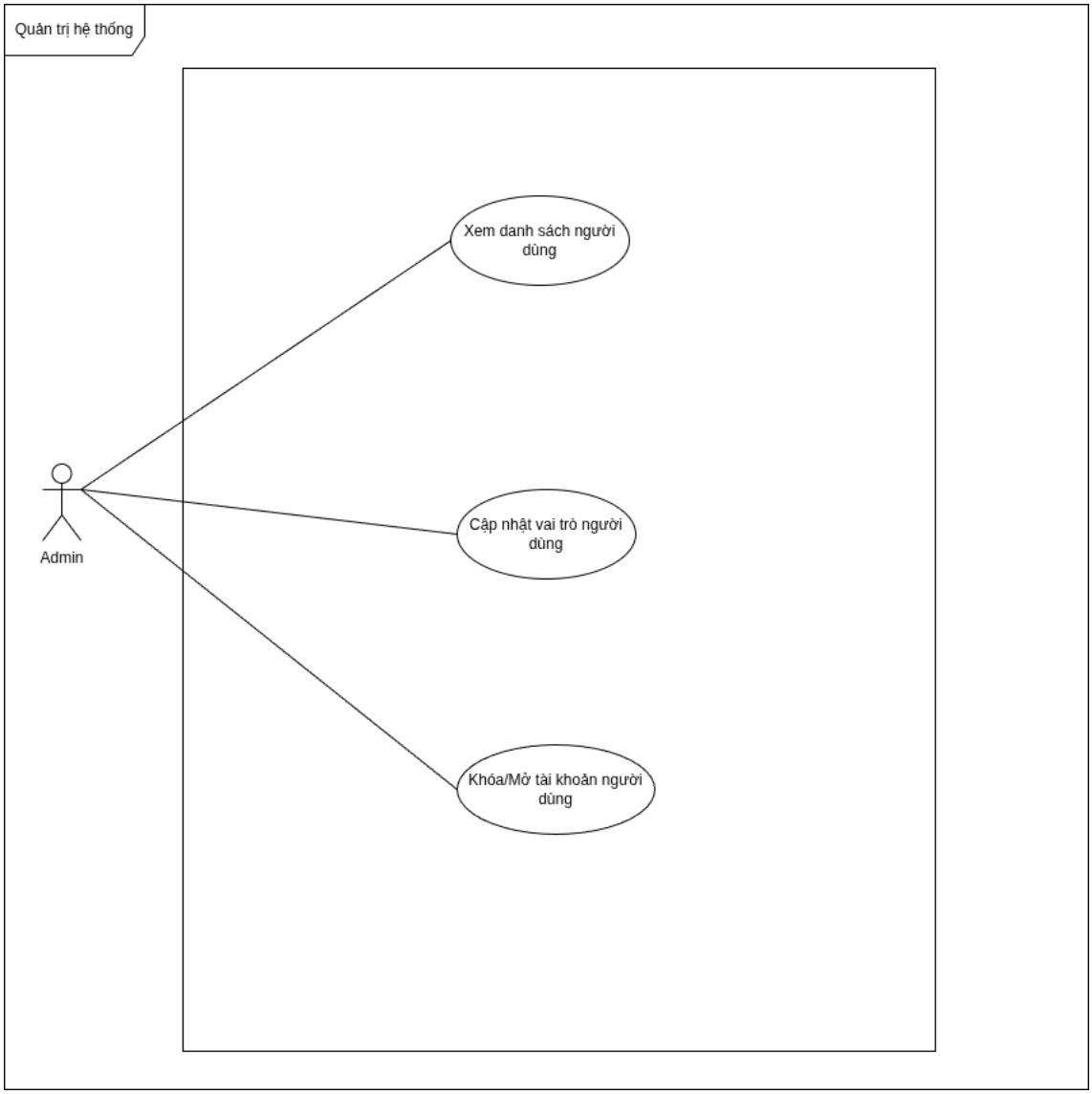
Hình 4.7 mô tả người dùng có thể xem các thống kê về các hoạt động trên hệ thống



Hình 2.7: Biểu đồ usecase xem báo cáo và thống kê

2.2.8 Biểu đồ usecase phân rã quản trị hệ thống

Hình 2.8 mô tả Admin có thể chỉnh sửa thông tin trên hệ thống



Hình 2.8: Biểu đồ usecase quản trị hệ thống

2.2.9 Quy trình nghiệp vụ

Bên cạnh các use case riêng lẻ, hệ thống còn có các quy trình nghiệp vụ kết hợp nhiều use case để thực hiện một nhiệm vụ hoàn chỉnh. Một số quy trình tiêu biểu:

- **Quy trình 1: Tạo và niêm yết NFT mới**
  1. Người dùng kết nối ví.
  2. Người dùng upload hình ảnh NFT thông qua presigned URL.
  3. Hệ thống lưu metadata off-chain và gọi smart contract để mint NFT.
  4. Sau khi mint thành công, người dùng cấu hình giá và niêm yết NFT.
  5. Hệ thống cập nhật trạng thái NFT sang "Đang niêm yết" và hiển thị trên marketplace.
- **Quy trình 2: Mua NFT**

1. Người dùng (người mua) kết nối ví và chọn một NFT đang được niêm yết.
2. Hệ thống hiển thị chi tiết NFT, giá bán, phí nền tảng (nếu có), tổng số tiền cần thanh toán.
3. Người dùng xác nhận mua, hệ thống gọi smart contract thực hiện giao dịch.
4. Sau khi giao dịch được xác nhận, chủ sở hữu mới được cập nhật on-chain.
5. Backend đồng bộ dữ liệu, ghi lại bản ghi lịch sử giao dịch và cập nhật dashboard của người mua và người bán.

Các quy trình này sẽ được mô tả bằng biểu đồ hoạt động (activity diagram) để minh hoạ luồng tương tác giữa người dùng, backend và smart contract.

2.3 Đặc tả chức năng

Trong phần này, em lựa chọn một số use case quan trọng để đặc tả chi tiết, bao gồm: Kết nối ví, Tạo NFT, Niêm yết NFT, Mua NFT.

2.3.1 Đặc tả usecase kết nối ví

Mã Usecase	Kết nối ví
Mô tả	Cho phép người dùng kết nối ví MetaMask để đăng nhập vào hệ thống.
Tác nhân	Khách (Guest)
Sự kiện kích hoạt	Người dùng bấm nút "Connect wallet" trên trang web.
Tiền điều kiện	1. Trình duyệt có ví tương thích (MetaMask). 2. Người dùng chưa đăng nhập.
Hậu điều kiện	Hệ thống lấy được public address, tạo hồ sơ người dùng, frontend nhận token và hiển thị trạng thái đã đăng nhập.
Luồng chính	1. Người dùng bấm nút "Connect wallet". 2. Frontend yêu cầu ví kết nối, nhận address. 3. Frontend gọi backend tạo nonce cho address. 4. Backend tạo nonce và trả message cần ký. 5. Người dùng ký message trên ví, frontend nhận signature. 6. Frontend gửi address, signature, nonce lên backend. 7. Backend xác thực chữ ký, kiểm tra hoặc tạo thông tin người dùng và trả lại token. 8. Frontend lưu token và cập nhật giao diện thành đã đăng nhập.
Luồng thay thế	A1: Người dùng đã tồn tại, backend chỉ xác nhận và cấp token mới, không tạo thông tin người dùng. A2: Ví MetaMask đang sai network, frontend yêu cầu chuyển network trước khi giao dịch.
Luồng ngoại lệ	E1: Người dùng chưa cài đặt ví, hiển thị yêu cầu cài đặt ví. E2: Người dùng từ chối kết nối hoặc ký, giữ trạng thái khách và thông báo kết nối bị hủy. E3: Signature hoặc nonce hết hạn, báo lỗi và yêu cầu thực hiện lại.

Bảng 2.2: Đặc tả use case kết nối ví

2.3.2 Đặc tả usecase tạo NFT

Mã Usecase	Tạo NFT
Mô tả	Tạo NFT mới từ ảnh.
Tác nhân	Người dùng (User)
Sự kiện kích hoạt	Người dùng bấm nút tạo NFT trên trang web và xác nhận giao dịch trên ví.
Tiền điều kiện	Người dùng đã đăng nhập, có đủ số dư trả phí gas.
Hậu điều kiện	NFT được tạo trên blockchain, database lưu trữ bản ghi về thông tin NFT.
Luồng chính	1. Người dùng bấm nút tạo NFT và nhập thông tin NFT. 2. Người dùng chọn file ảnh. 3. Frontend gọi backend lấy presigned URL. 4. Frontend upload ảnh trực tiếp lên S3 bằng presigned URL. 5. Frontend gửi imageUrl và metadata lên backend để lưu metadata off-chain. 6. Backend trả metadataUri cho frontend. 7. Frontend gọi ví gửi transaction mint lên smart contract. 8. Transaction xác nhận thành công, cập nhật event mint hoặc đồng bộ theo txHash vào backend. 9. Backend tạo bản ghi NFT trong database.
Luồng thay thế	Không có.
Luồng ngoại lệ	E1: Tải ảnh lên S3 thất bại, thông báo lỗi và cho tải ảnh lại. E2: Người dùng từ chối ký transaction trên ví, dừng luồng lại. E3: Transaction bị lỗi, hiển thị lỗi và dừng luồng lại.

Bảng 2.3: Đặc tả usecase tạo NFT

t

2.3.3 Đặc tả usecase niêm yết NFT

Mã Usecase	Niêm yết NFT
Mô tả	Chủ sở hữu đưa NFT lên marketplace với giá cố định để người khác mua.
Tác nhân	Người dùng (User)
Sự kiện kích hoạt	Người dùng bấm nút niêm yết NFT trên trang web và xác nhận giao dịch trên ví.
Tiền điều kiện	Người dùng đã đăng nhập, có đủ số dư trả phí gas, NFT chưa được niêm yết và người dùng là chủ sở hữu của NFT.
Hậu điều kiện	NFT được niêm yết trên blockchain và NFT hiển thị trạng thái đã niêm yết trên trang web.
Luồng chính	1. Người dùng ấn vào NFT. 2. Người dùng ấn nút niêm yết NFT. 3. Người dùng xác nhận niêm yết, ký và gửi transaction. 4. Transaction xác nhận thành công. 5. Backend cập nhật database cho NFT và hiển thị lên marketplace.
Luồng thay thế	Người dùng nhập giá mới trước khi ấn niêm yết.
Luồng ngoại lệ	E1: Giá không hợp lệ, chặn niêm yết và hiển thị lỗi. E2: Người dùng từ chối ký transaction trên ví, dừng luồng lại. E3: Transaction bị lỗi, hiển thị lỗi và dừng luồng lại.

Bảng 2.4: Đặc tả usecase niêm yết NFT

2.3.4 Đặc tả usecase mua NFT

Mã Usecase	Mua NFT
Mô tả	Người mua thanh toán để mua NFT; smart contract chuyển quyền sở hữu; hệ thống ghi lịch sử giao dịch và lịch sử giá.
Tác nhân	Người dùng (User)
Sự kiện kích hoạt	Người dùng bấm nút mua NFT trên trang web và xác nhận giao dịch trên ví.
Tiền điều kiện	Người dùng đã đăng nhập, có đủ số dư trả phí gas và giá, NFT đã được niêm yết và người dùng không phải là chủ sở hữu của NFT.
Hậu điều kiện	Chủ sở hữu đổi từ người bán sang người mua; niêm yết kết thúc; database có transaction record; NFT chuyển về chưa niêm yết.
Luồng chính	1. Người dùng bấm nút mua NFT. 2. UI hiển thị số tiền cần thanh toán. 3. Người dùng xác nhận, ví ký và gửi transaction mua. 4. Smart contract thực hiện thanh toán, trừ fee theo cấu hình và chuyển NFT cho người mua. 5. Transaction xác nhận thành công. 6. Backend cập nhật thông tin mới về NFT và transaction vào database. 7. Frontend thông báo mua thành công và hiển thị chủ sở hữu mới cho NFT.
Luồng thay thế	A1: Trong lúc người mua thao tác, NFT bị người khác mua trước, hiển thị thông báo mua lỗi.
Luồng ngoại lệ	E1: Người mua từ chối ký transaction trên ví, dừng luồng lại. E2: Transaction bị lỗi, hiển thị lỗi và dừng luồng lại.

Bảng 2.5: Đặc tả usecase mua NFT

2.4 Yêu cầu phi chức năng

Ngoài các yêu cầu chức năng đã được xác định thông qua quá trình khảo sát và phân tích, hệ thống NFT Marketplace cũng cần đáp ứng các yêu cầu phi chức năng nhằm đảm bảo chất lượng vận hành, mức độ an toàn và trải nghiệm sử dụng trong điều kiện thực tế. Nhóm yêu cầu này đóng vai trò như các tiêu chí ràng buộc về chất lượng, giúp hệ thống không chỉ đáp ứng đúng chức năng mà còn hoạt động ổn định, hiệu quả và thuận tiện cho việc bảo trì, mở rộng về sau.

2.4.1 Yêu cầu về hiệu năng

Hệ thống cần đảm bảo thời gian phản hồi hợp lý đối với các chức năng được sử dụng thường xuyên, đặc biệt là các API phục vụ hiển thị danh sách NFT, thông tin chi tiết NFT và thông tin collection. Trong phạm vi đồ án, thời gian phản hồi trung

bình của các API này cần nằm trong một ngưỡng chấp nhận được là dưới 1s, đồng thời không tính thời gian chờ xác nhận giao dịch on-chain do phụ thuộc vào tốc độ xử lý của mạng blockchain. Để tối ưu hiệu năng, hệ thống áp dụng cơ chế cache bằng Redis cho các dữ liệu có tần suất đọc cao nhưng ít thay đổi như danh sách NFT phổ biến, thông tin collection và một số thống kê tổng quan. Bên cạnh đó, hệ thống cần có khả năng phục vụ đồng thời nhiều người dùng truy cập mà không xảy ra suy giảm hiệu năng đáng kể trong phạm vi triển khai và kiểm thử của đồ án.

2.4.2 Yêu cầu về bảo mật

Do đặc thù của NFT marketplace liên quan đến tài sản số và giao dịch, hệ thống phải đảm bảo cơ chế xác thực và kiểm soát truy cập chặt chẽ. Người dùng được xác thực dựa trên ví điện tử thông qua chữ ký (signature), kết hợp với cơ chế token (JWT) ở phía backend nhằm bảo vệ các API và duy trì phiên làm việc. Đồng thời, hệ thống phân quyền rõ ràng giữa người dùng thường và admin; người dùng chỉ được phép truy cập và thao tác trên dữ liệu thuộc về mình (ví dụ: NFT đang sở hữu, thông tin hồ sơ cá nhân, lịch sử giao dịch của bản thân). Ngoài ra, các thông tin nhạy cảm như access token, presigned URL, cấu hình hệ thống, khóa bí mật và thông tin kết nối dịch vụ cần được quản lý thông qua biến môi trường và cấu hình an toàn để hạn chế rủi ro rò rỉ. Ở lớp blockchain, smart contract cũng cần được thiết kế theo các nguyên tắc an toàn cơ bản nhằm hạn chế các lỗ hổng phổ biến như reentrancy, cũng như các sai sót liên quan đến tràn số (overflow/underflow) hoặc kiểm tra điều kiện không đầy đủ.

2.4.3 Yêu cầu về khả dụng và trải nghiệm người dùng

Hệ thống cần hướng tới giao diện đơn giản, nhất quán và dễ sử dụng, đồng thời hỗ trợ responsive trên nhiều độ phân giải màn hình như desktop, tablet và mobile. Các thao tác quan trọng như kết nối ví, mint NFT, list NFT lên marketplace và thực hiện mua (buy) cần có bước xác nhận rõ ràng, kèm theo trạng thái xử lý và kết quả hiển thị minh bạch để người dùng theo dõi tiến trình thao tác. Khi xảy ra sự cố, hệ thống phải cung cấp thông báo lỗi rõ ràng, dễ hiểu, giúp người dùng nhận biết các nguyên nhân phổ biến như giao dịch thất bại, không kết nối được ví, lỗi mạng hoặc dữ liệu đầu vào không hợp lệ. Những yếu tố này góp phần nâng cao mức độ tin cậy và giảm khó khăn cho người dùng, đặc biệt đối với các thao tác gắn với giao dịch on-chain.

2.4.4 Yêu cầu về bảo trì và mở rộng

Do hệ thống NFT Marketplace có khả năng phát triển thêm nhiều tính năng theo thời gian, kiến trúc hệ thống cần được thiết kế theo hướng module hoá để thuận tiện mở rộng và thay đổi. Backend cần tổ chức mã nguồn theo chuẩn, tách bạch các lớp

như controller, service, repository nhằm tăng khả năng bảo trì và hỗ trợ kiểm thử. Đồng thời, hệ thống cần có tài liệu mô tả API và các ghi chú cần thiết để thuận lợi cho việc tiếp cận và phát triển tiếp theo. Ngoài ra, một số tham số quan trọng như phí giao dịch (fee), địa chỉ contract, endpoint node blockchain cần được cấu hình linh hoạt thông qua file cấu hình hoặc biến môi trường, hạn chế việc phải chỉnh sửa mã nguồn khi thay đổi môi trường triển khai.

2.4.5 Ràng buộc kỹ thuật

Trong phạm vi đồ án, hệ thống được định hướng triển khai với frontend sử dụng Next.js và backend sử dụng NestJS. Cơ sở dữ liệu chính dùng PostgreSQL để lưu trữ dữ liệu nghiệp vụ, Redis được sử dụng cho cơ chế cache nhằm tối ưu tốc độ truy vấn, và AWS S3 đảm nhiệm lưu trữ file (hình ảnh NFT). Smart contract được triển khai trên Binance Smart Chain Testnet. Bên cạnh đó, các công cụ và thư viện được lựa chọn cần có tính phổ biến, tài liệu rõ ràng và cộng đồng hỗ trợ tốt nhằm đảm bảo quá trình phát triển thuận lợi, đồng thời giảm rủi ro khi bảo trì hoặc mở rộng về sau.

Tóm lại, nhóm yêu cầu phi chức năng nêu trên là cơ sở để đánh giá chất lượng hệ thống NFT Marketplace, đồng thời định hướng cho các quyết định lựa chọn công nghệ và thiết kế kiến trúc hệ thống trong các chương tới.

CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Chương này trình bày các công nghệ và nền tảng được lựa chọn trong đồ án. Nội dung chương được tổ chức theo từng lớp của hệ thống. Với mỗi công nghệ, phần trình bày tập trung vào ba ý chính: công nghệ là gì và thường được dùng để giải quyết vấn đề nào; công nghệ đó đáp ứng các yêu cầu đã nêu ở Chương 2 ra sao; và trong số các hướng tiếp cận phổ biến, vì sao đồ án chọn phương án hiện tại.

3.1 Kiến trúc hệ thống

3.1.1 Monolith theo mô hình client–server

Trong phát triển hệ thống web, mô hình client–server là cách tổ chức quen thuộc: phía client chạy trên trình duyệt, chịu trách nhiệm hiển thị giao diện và tiếp nhận thao tác; phía server cung cấp API, xử lý nghiệp vụ và quản lý dữ liệu. Trên nền tảng đó, kiến trúc monolith được hiểu là backend được triển khai như một ứng dụng thống nhất (một codebase và một đơn vị triển khai), nhưng bên trong vẫn có thể tổ chức theo các module nghiệp vụ để tách bạch trách nhiệm. Ưu điểm lớn của monolith là tính đơn giản và trực tiếp: luồng xử lý ít tầng trung gian, dễ kiểm thử end-to-end, triển khai gọn, và thuận tiện khi cần đảm bảo tính nhất quán dữ liệu vì mọi thao tác nghiệp vụ tập trung trong một hệ thống.

Các yêu cầu ở Chương 2 cho thấy đồ án cần ưu tiên hoàn thiện các luồng nghiệp vụ chính và đảm bảo trải nghiệm người dùng. Hệ thống không chỉ có các thao tác thông thường như tạo và quản lý dữ liệu, mà còn phải tương tác với blockchain, cập nhật trạng thái giao dịch và phản hồi rõ ràng khi giao dịch thành công hoặc thất bại. Với phạm vi như vậy, monolith giúp giảm đáng kể chi phí thiết kế và vận hành so với mô hình phân tán: không phải giải quyết quá nhiều vấn đề giao tiếp giữa các dịch vụ, không phải xây dựng cơ chế đồng bộ dữ liệu phức tạp giữa nhiều hệ thống, và việc kiểm thử luồng từ giao diện đến cơ sở dữ liệu cũng trực tiếp hơn. Điều này đặc biệt phù hợp khi đồ án cần ổn định và hạn chế rủi ro trong tích hợp.

Microservices là hướng tiếp cận phổ biến khác, trong đó backend được tách thành nhiều dịch vụ nhỏ giao tiếp qua API hoặc hàng đợi tin nhắn. Cách làm này có lợi khi hệ thống ở quy mô lớn, nhiều nhóm phát triển song song, hoặc cần mở rộng độc lập theo từng chức năng. Tuy nhiên, đổi lại là độ phức tạp của một hệ phân tán: cấu hình, giám sát, truy vết luồng request, xử lý lỗi và đảm bảo nhất quán dữ liệu đều khó hơn và tốn công hơn. Trong bối cảnh đồ án, khi mục tiêu là xây dựng một marketplace chạy mạch lạc và tích hợp blockchain rõ ràng, lựa chọn monolith là hợp lý. Đồng thời, để tránh việc hệ thống trở nên cồng kềnh, backend vẫn được tổ chức theo các module nghiệp vụ, tạo nền tảng cho khả năng tách dần về sau nếu

cần mở rộng.

3.1.2 Nguyên tắc phân tách on-chain và off-chain

NFT marketplace khác với các ứng dụng web thông thường ở chỗ tài sản và giao dịch được ghi nhận trên blockchain, nên quyền sở hữu và trạng thái mua bán cần có khả năng kiểm chứng công khai. Vì vậy, trong đồ án em lựa chọn cách tách rõ phần on-chain và off-chain: các thao tác làm thay đổi quyền sở hữu hoặc trạng thái giao dịch được thực hiện trong smart contract để đảm bảo tính tin cậy; còn các dữ liệu phục vụ hiển thị và có nhu cầu chỉnh sửa linh hoạt được lưu off-chain để thuận tiện thao tác và tránh phát sinh chi phí không cần thiết.

Cụ thể, hình ảnh NFT/collection được lưu trên AWS S3; các thông tin như tên và mô tả được lưu trong cơ sở dữ liệu để người dùng có thể cập nhật trực tiếp mà không phải ký giao dịch hay chờ xác nhận blockchain. Ngược lại, các thao tác list, unlist và transfer được xử lý on-chain nhằm đảm bảo trạng thái niêm yết và quyền sở hữu luôn minh bạch, tránh trường hợp dữ liệu off-chain không phản ánh đúng trạng thái thực tế trên blockchain. Cách tiếp cận này giúp tăng hiệu suất của hệ thống, đồng thời vẫn giữ đúng bản chất "giao dịch có ràng buộc" của một NFT marketplace.

3.2 Ngôn ngữ và nền tảng phát triển

3.2.1 TypeScript trong phát triển ứng dụng web

TypeScript là ngôn ngữ mở rộng từ JavaScript, bổ sung hệ kiểu tĩnh và các cơ chế hỗ trợ xây dựng phần mềm quy mô vừa và lớn như interface, generics, decorator và kiểm tra kiểu ở thời điểm biên dịch **typescript**. Trên thực tế, nhiều hệ thống web chọn TypeScript không phải vì hiệu năng cú pháp, mà vì cần kiểm soát rủi ro: dữ liệu qua API, dữ liệu UI state, và dữ liệu nghiệp vụ thường biến đổi liên tục trong quá trình phát triển; nếu chỉ dựa vào JavaScript thuần, lỗi kiểu dữ liệu thường xuất hiện và khó truy vết.

Trong đồ án, lượng mô hình dữ liệu không hề nhỏ. TypeScript giúp mô tả rõ cấu trúc request/response của API, giảm sai lệch giữa frontend và backend, đồng thời tăng độ an toàn khi xử lý các trường dễ nhầm như địa chỉ ví, chain id, giá, phí và trạng thái giao dịch. Đặc biệt với UI hiển thị lịch sử giá và thống kê, dữ liệu thường là mảng thời gian và phép tổng hợp; việc có kiểu dữ liệu rõ ràng giúp tránh lỗi không mong muốn và giúp quá trình phát triển thống nhất, đồng bộ hơn.

JavaScript thuần là một lựa chọn thay thế hợp lệ, nhất là khi muốn viết nhanh giai đoạn đầu. Tuy nhiên, trong bối cảnh dự án cần sự ổn định, TypeScript làm cho code dễ đọc hơn khi nhiều file cùng tham gia một luồng nghiệp vụ. Vì vậy,

TypeScript được chọn làm ngôn ngữ thống nhất cho cả frontend và backend.

3.2.2 Node.js và mô hình I/O bất đồng bộ

Node.js là môi trường chạy JavaScript phía server dựa trên event loop, tối ưu cho các ứng dụng có nhiều tác vụ I/O như gọi cơ sở dữ liệu, gọi API, đọc/ghi file, hoặc giao tiếp mạng. Thế mạnh của Node.js không nằm ở xử lý tính toán nặng, mà ở khả năng xử lý đồng thời nhiều kết nối và nhiều request trong khi phần lớn thời gian là chờ I/O.

Đồ án có đặc điểm tương đối phù hợp với Node.js: số lượng API đọc nhiều (danh sách, chi tiết, search, lịch sử giao dịch, dashboard), còn các tác vụ ghi chủ yếu là cập nhật dữ liệu off-chain sau các hành động của người dùng. Ngoài ra, hệ sinh thái Web3 hỗ trợ Node.js rất tốt; việc tích hợp RPC, ký message, đọc event, và tương tác EVM trở nên thuận lợi nhờ các thư viện phổ biến (chẳng hạn ethers.js). Đây là lý do thực tiễn khiến Node.js phù hợp hơn trong đồ án so với việc dùng công nghệ khác.

Tất nhiên, backend có thể được viết bằng Java (Spring Boot) hoặc .NET; những nền tảng này rất mạnh trong doanh nghiệp và có hệ sinh thái lớn. Tuy nhiên, chọn chúng trong đồ án sẽ làm công nghệ sử dụng trở nên không đồng nhất, tăng thời gian tích hợp, trong khi mục tiêu là hoàn thiện một marketplace có luồng nghiệp vụ rõ ràng. Vì vậy, TypeScript/Node.js là lựa chọn hợp lý về cả kỹ thuật lẫn tiến độ.

3.3 Công nghệ xây dựng giao diện người dùng

3.3.1 React và Next.js

React là thư viện xây dựng giao diện theo hướng thành phần (component), cho phép chia UI thành các khối độc lập, tái sử dụng và quản lý trạng thái theo cách có cấu trúc. Next.js là framework dựa trên React, cung cấp cơ chế routing theo file, tối ưu build, và hỗ trợ nhiều chế độ render (SSR/SSG/CSR) để cân bằng giữa tốc độ tải trang và tính tương tác. Trong nhiều dự án web hiện đại, Next.js được chọn vì nó cung cấp sẵn các tính năng sẵn có: cách tổ chức trang, cách build, cách tối ưu tài nguyên, và cách mở rộng.

Định hướng UI/UX ở Chương 2 yêu cầu giao diện đơn giản hơn các marketplace lớn. Với Next.js/React, giao diện dễ được tổ chức theo component. Việc tách UI theo component giúp duy trì sự nhất quán giữa các trang, ví dụ danh sách và chi tiết có cùng cách trình bày trạng thái của NFT, cùng kiểu hiển thị giá và phí, và cùng chuẩn thông báo lỗi.

Nếu dùng React SPA thuần (chẳng hạn Vite), dự án vẫn có thể hoàn thành, nhưng phải tự quyết nhiều thứ hơn về routing, tổ chức trang và tối ưu build. Angular là

một lựa chọn khác, cung cấp framework đầy đủ với DI, routing, form; tuy nhiên Angular thường nặng hơn và không phổ biến bằng React trong hệ sinh thái Web3. Vue/Nuxt cũng là phương án đáng cân nhắc, nhưng trong phạm vi đồ án, Next.js có lợi thế về mức độ phổ biến và tài liệu, giúp giảm rủi ro khi triển khai và khi viết báo cáo tham khảo.

3.3.2 Tương tác ví và trải nghiệm Web3 trên giao diện

Một trang web thông thường chỉ cần tương tác với trang web chính, còn Web3 thì khác: kết nối ví, ký message để đăng nhập, ký giao dịch để mint, ký giao dịch để list/unlist, rồi chờ mạng xác nhận. Nếu UI/UX không đươc thiết kế tốt, người dùng mới sẽ cảm thấy khó sử dụng và không hiểu luồng nó hoạt động như thế nào. Vì vậy, ngoài việc gọi RPC, giao diện phải thể hiện rõ trạng thái: đang yêu cầu ký, đang chờ xác nhận, đã thành công, hay thất bại và cần thử lại. Đây là phần trực tiếp đáp ứng yêu cầu trải nghiệm ở Chương 2: thao tác quan trọng phải có xác nhận rõ ràng và thông báo minh bạch.

Về mặt kỹ thuật, các thao tác này thường được thực hiện thông qua provider của ví và thư viện tương tác EVM (ví dụ ethers.js). Những thư viện này không tự làm UI; chúng chỉ giúp gọi giao dịch đúng cách. Vì vậy, lựa chọn Next.js/React không chỉ để phát triển giao diện, mà còn để tổ chức các trạng thái UI quanh những điểm cốt lõi để người dùng hiểu rõ hơn: phí gas, thời gian chờ, và các trường hợp người dùng từ chối ký.

3.4 Công nghệ xây dựng backend và API

3.4.1 NestJS và cách tổ chức backend theo module

NestJS là framework Node.js hướng kiến trúc, khuyến khích tổ chức dự án theo module và tách lớp rõ ràng giữa controller, service và data access, đồng thời cung cấp dependency injection để quản lý phụ thuộc. Trong thực tế, NestJS thường được chọn khi đội phát triển muốn tổ chức mã nguồn theo hướng nghiệp vụ dễ mở rộng: thay vì một tập hợp route handler rời rạc, hệ thống được tổ chức theo miền nghiệp vụ và có chuẩn chung cho validation, error handling, auth guard và logging.

Trong đồ án, backend đóng vai trò trung tâm để xử lý nghiệp vụ và điều phối dữ liệu giữa các thành phần của hệ thống. Cụ thể, backend quản lý dữ liệu off-chain trong PostgreSQL, sử dụng Redis để giảm tải cho các truy vấn đọc nhiều, làm việc với S3 để lưu trữ hình ảnh, đồng thời theo dõi và đồng bộ các thay đổi quan trọng từ smart contract lên cơ sở dữ liệu nhằm phục vụ hiển thị và thống kê.

Các luồng nghiệp vụ chính như tạo NFT, niêm yết hay mua bán đều cần backend phối hợp nhiều bước liên tiếp: vừa đảm bảo dữ liệu hiển thị đầy đủ, vừa đảm bảo

trạng thái phản ánh đúng kết quả giao dịch on-chain. Vì vậy, nếu mã nguồn không được tổ chức tốt, các phần xử lý dễ bị làm theo cách phức tạp và khó bảo trì khi mở rộng chức năng hoặc xử lý lỗi. NestJS được lựa chọn nhằm hỗ trợ cấu trúc dự án theo hướng module hoá, giúp tách bạch rõ các phần xử lý và phù hợp với yêu cầu bảo trì, mở rộng.

Express hoặc Fastify là lựa chọn thay thế phổ biến, nhẹ và linh hoạt. Tuy nhiên, dùng nền tảng thuần đồng nghĩa nhóm phải tự đặt chuẩn kiến trúc, tự tổ chức module, tự thống nhất cách trả lỗi và validation; gây ra tốn thời gian và không nhất quán khi dự án lớn dần. Spring Boot cũng là một đối thủ mạnh; nhưng với phạm vi của đồ án và nhu cầu tích hợp Web3 nhanh, NestJS giúp giảm thời gian phát triển và giữ công nghệ sử dụng được thống nhất.

3.4.2 RESTful API và tài liệu hoá API

REST là một phong cách thiết kế API theo hướng tài nguyên, trong đó mỗi loại dữ liệu chính của hệ thống được biểu diễn bằng một nhóm endpoint và được thao tác thông qua các phương thức HTTP quen thuộc; dữ liệu trao đổi thường ở dạng JSON. Cách tiếp cận này được sử dụng rộng rãi vì đơn giản, dễ kiểm thử và phù hợp với ứng dụng web. Trong đồ án, các nhóm chức năng được tổ chức thành các cụm API tương ứng với người dùng, NFT, collection, niêm yết và giao dịch. Phần lớn request từ phía giao diện là các truy vấn đọc để hiển thị danh sách, trang chi tiết, lịch sử giá, lịch sử giao dịch và các số liệu thống kê; bên cạnh đó là các API ghi phục vụ các thao tác off-chain như cập nhật hồ sơ, tạo và chỉnh sửa thông tin NFT/collection, cũng như các luồng đồng bộ dữ liệu sau khi trạng thái on-chain thay đổi.

Do hệ thống có nhiều màn hình và nhiều luồng nghiệp vụ, việc chuẩn hoá và tài liệu hoá API giúp giảm thời gian tích hợp giữa frontend và backend, đồng thời làm cho báo cáo có tính kiểm chứng cao hơn. OpenAPI/Swagger là chuẩn mô tả API phổ biến, cho phép sinh tài liệu và giao diện thử trực tiếp các endpoint. Trong phạm vi đồ án, phần tài liệu này giúp người đọc dễ đối chiếu mô tả chức năng của từng API có trong hệ thống.

Một hướng tiếp cận khác là GraphQL, thường phù hợp khi giao diện cần truy vấn dữ liệu linh hoạt và muốn tự chọn các trường dữ liệu cần lấy. Tuy nhiên, GraphQL đòi hỏi thiết kế schema và cơ chế kiểm soát truy vấn chặt chẽ để tránh ảnh hưởng hiệu năng, đặc biệt với các màn hình danh sách lớn có lọc, sắp xếp và phân trang. Với mục tiêu trình bày rõ ràng và triển khai trong đồ án, REST là lựa chọn phù hợp hơn.

3.5 Cơ sở dữ liệu và quản lý dữ liệu off-chain

3.5.1 PostgreSQL cho dữ liệu nghiệp vụ

PostgreSQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở, được phát triển lâu đời và được sử dụng rộng rãi trong các hệ thống web hiện đại **postgresql**. Điểm mạnh của PostgreSQL nằm ở khả năng đảm bảo tính đúng đắn của dữ liệu thông qua giao dịch và các ràng buộc toàn vẹn, đồng thời hỗ trợ SQL tương đối đầy đủ và linh hoạt. Hệ thống cho phép mô hình hoá dữ liệu theo dạng bảng với khoá chính, khoá ngoại, ràng buộc duy nhất và các quy tắc kiểm tra, nhờ đó giảm đáng kể rủi ro sinh dữ liệu sai lệch khi nhiều chức năng cùng đọc/ghi. PostgreSQL cũng nổi bật ở khả năng xử lý đồng thời tốt nhờ cơ chế MVCC, nghĩa là việc đọc và ghi có thể diễn ra song song mà hạn chế tình trạng "chặn" nhau như các cơ chế khoá truyền thống, phù hợp với các ứng dụng có nhiều lượt truy cập.

Về hiệu năng, PostgreSQL cung cấp hệ thống chỉ mục phong phú và nhiều kỹ thuật tối ưu truy vấn. Ngoài chỉ mục B-tree phổ biến cho các truy vấn so sánh và sắp xếp, PostgreSQL còn hỗ trợ các loại chỉ mục khác như GIN/GiST, hữu ích trong những trường hợp tìm kiếm theo tập hợp, dữ liệu bán cấu trúc hoặc truy vấn theo điều kiện phức tạp. Bên cạnh mô hình quan hệ, PostgreSQL cũng hỗ trợ kiểu dữ liệu JSON/JSONB, giúp lưu trữ một số trường dữ liệu linh hoạt khi cần mà vẫn có thể truy vấn và lập chỉ mục ở mức nhất định. Đây là lý do PostgreSQL thường được lựa chọn trong các dự án cần vừa chặt chẽ về quan hệ dữ liệu, vừa có một phần dữ liệu có thể thay đổi theo thời gian.

Trong đồ án, PostgreSQL được sử dụng làm kho dữ liệu chính cho phần off-chain, phục vụ lưu trữ thông tin nghiệp vụ và cung cấp dữ liệu cho các màn hình hiển thị. Nhờ cơ chế truy vấn và tổng hợp tốt, hệ thống có thể trích xuất lịch sử giao dịch theo thời gian để xây dựng lịch sử giá và các thống kê dashboard. Ngoài ra, việc lưu metadata như tên và mô tả ở cơ sở dữ liệu giúp thao tác cập nhật nội dung diễn ra thuận tiện hơn, trong khi các phần ràng buộc sở hữu và trạng thái giao dịch vẫn được đảm bảo bởi smart contract.

MySQL là phương án thay thế trực tiếp và cũng rất phổ biến trong các ứng dụng web. Với các thao tác CRUD cơ bản, cả MySQL và PostgreSQL đều đáp ứng tốt. Tuy nhiên, trong phạm vi đồ án, PostgreSQL được ưu tiên vì thuận lợi hơn khi xây dựng các truy vấn tổng hợp và thống kê, đặc biệt ở các chức năng như dashboard và lịch sử giá theo thời gian.

Một lựa chọn khác là MongoDB, phù hợp khi dữ liệu cần lưu dưới dạng document và cấu trúc thay đổi linh hoạt. Dù vậy, bài toán marketplace thường có nhiều quan hệ rõ ràng giữa các thực thể và nhu cầu tổng hợp dữ liệu theo nhiều chiều xuất hiện

thường xuyên, nên mô hình quan hệ giúp quản lý ràng buộc chặt chẽ hơn và việc truy vấn theo nghiệp vụ cũng dễ kiểm soát hơn.

3.6 Cache và tối ưu hiệu năng

3.6.1 Redis

Redis là một hệ thống lưu trữ dữ liệu dạng key–value hoạt động chủ yếu trên bộ nhớ, vì vậy tốc độ đọc/ghi thường rất nhanh **redis**. Redis được dùng phổ biến trong vai trò cache cho ứng dụng web nhờ cơ chế lưu trữ đơn giản, hỗ trợ đặt thời gian hết hạn cho dữ liệu và có thể giảm đáng kể số lần truy vấn về cơ sở dữ liệu chính. Ngoài cache, Redis cũng hay được tận dụng cho các dữ liệu "ngắn hạn" như nonce phục vụ xác thực, giới hạn tần suất truy cập hoặc các trạng thái tạm trong một khoảng thời gian nhất định, tuỳ theo cách thiết kế hệ thống.

Trong đồ án, Redis được sử dụng nhằm hỗ trợ yêu cầu hiệu năng đã nêu ở Chương 2, đặc biệt với các luồng đọc như danh sách NFT/collection, trang chi tiết và các số liệu thống kê. Đặc trưng của marketplace là số lượt đọc thường lớn hơn nhiều so với số lượt ghi; nếu mỗi lần tải trang đều phải truy vấn và tổng hợp lại toàn bộ từ PostgreSQL thì độ trễ sẽ tăng và hệ thống dễ bị quá tải khi lượng truy cập lớn. Khi có Redis, một số dữ liệu được truy cập lặp lại có thể được lưu tạm trong cache và phục vụ nhanh hơn; dữ liệu sẽ được làm mới theo thời gian hết hạn hoặc theo cơ chế xoá cache khi có cập nhật. Nhờ đó, thời gian phản hồi ổn định hơn và trải nghiệm người dùng cũng mượt hơn.

Về hướng tiếp cận thay thế, hệ thống có thể cache trực tiếp trong bộ nhớ của ứng dụng. Cách này đơn giản nếu chỉ chạy một instance, nhưng khi cần mở rộng nhiều instance thì cache bị phân tán và dễ không nhất quán. Memcached cũng là một lựa chọn phổ biến cho cache thuần, tuy nhiên Redis thường linh hoạt hơn và được hỗ trợ tốt trong hệ sinh thái Node.js, đồng thời tài liệu và cộng đồng cũng đông đảo giúp cho phần sửa lỗi trong quá trình phát triển dễ dàng hơn.

3.7 Lưu trữ nội dung số

3.7.1 Amazon S3 và cơ chế presigned URL

Amazon S3 là dịch vụ lưu trữ đối tượng của AWS, được thiết kế để lưu trữ dữ liệu dạng file với khả năng mở rộng lớn và độ bền cao **s3**. S3 phù hợp với các nội dung tĩnh như hình ảnh vì cho phép tách phần lưu file ra khỏi tầng dữ liệu nghiệp vụ. Nếu lưu ảnh trực tiếp trong cơ sở dữ liệu, dung lượng và chi phí vận hành sẽ tăng, việc sao lưu hoặc khôi phục cũng nặng hơn, trong khi dữ liệu ảnh vốn không cần được xử lý như dữ liệu quan hệ. Với một marketplace, hình ảnh là yếu tố hiển thị quan trọng, nhưng về bản chất nên được đặt ở một lớp lưu trữ chuyên biệt để hệ thống gọn và dễ vận hành hơn.

Trong đồ án, S3 được dùng để lưu ảnh của NFT và collection. Quy trình upload được triển khai theo hướng phổ biến là sử dụng presigned URL: backend tạo một đường dẫn tải lên có thời hạn, frontend dùng đường dẫn này để upload trực tiếp lên S3, sau đó hệ thống lưu lại object key hoặc đường dẫn ảnh trong cơ sở dữ liệu. Cách làm này giúp backend tránh phải xử lý file dung lượng lớn, đồng thời hạn chế rủi ro bảo mật vì phía client không cần nắm giữ thông tin truy cập AWS. Nhìn tổng thể, đây là lựa chọn phù hợp với yêu cầu về hiệu năng và bảo mật đã đặt ra trong Chương 2.

Về phương án thay thế, IPFS thường được nhắc đến trong các hệ thống Web3 vì lưu trữ theo hướng phi tập trung và tham chiếu nội dung bằng mã băm. Tuy nhiên, để đảm bảo dữ liệu luôn sẵn sàng, IPFS thường cần thêm cơ chế pinning và việc vận hành có thể phức tạp hơn trong phạm vi đồ án. Một số dịch vụ như Cloudinary hỗ trợ xử lý ảnh rất tiện, nhưng sẽ làm hệ thống phụ thuộc vào nhà cung cấp bên ngoài. Với định hướng triển khai đơn giản trên AWS và mục tiêu tập trung vào nghiệp vụ marketplace, S3 là phương án cân bằng và dễ triển khai.

3.8 Blockchain và Smart Contract

3.8.1 Solidity, EVM và chuẩn NFT

Solidity là ngôn ngữ được dùng phổ biến để phát triển smart contract trên các blockchain tương thích EVM. EVM có thể hiểu như một môi trường chạy chương trình được chuẩn hoá: cùng một đoạn mã hợp đồng sẽ được thực thi theo cùng một quy tắc trên mọi nút mạng, và kết quả được toàn mạng xác nhận. Nhờ đó, các quy tắc về quyền sở hữu và điều kiện giao dịch không còn phụ thuộc vào một máy chủ trung tâm, mà được bảo đảm bởi cơ chế đồng thuận của blockchain. Đây cũng là lý do các ứng dụng gắn với tài sản số thường đặt phần logic cốt lõi lên smart contract, vì mọi thao tác đều để lại dấu vết công khai và có thể kiểm chứng.

Trong bài toán NFT, các chuẩn token như ERC-721 đóng vai trò như một bộ quy ước chung để mô tả cách mint, transfer và truy vấn chủ sở hữu của từng token. Khi tuân theo chuẩn, hợp đồng sẽ tương thích tốt hơn với hệ sinh thái xung quanh như ví, trình duyệt blockchain, cũng như các thư viện kết nối từ ứng dụng web. Việc chuẩn hoá này giúp giảm đáng kể chi phí tích hợp, vì frontend và backend có thể dựa vào các hàm và sự kiện đã được quy ước sẵn và có thể áp dụng cho nhiều dự án khác nhau.

Trong đồ án, smart contract được triển khai trên Binance Smart Chain Testnet, phù hợp cho mục đích thử nghiệm và trình diễn do chi phí thấp và dễ triển khai. Từ góc nhìn thiết kế, điểm quan trọng là những hành động mang tính ràng buộc tài sản cần có khả năng kiểm chứng công khai. Vì vậy, các thao tác như list, unlist và

transfer được đặt trên smart contract để trạng thái niêm yết và quyền sở hữu luôn bám theo logic on-chain, giảm phụ thuộc vào việc server tự gán trạng thái. Cách làm này giúp mô hình giao dịch rõ ràng hơn: khi người dùng nhìn thấy một NFT đang được niêm yết, trạng thái đó phản ánh đúng điều kiện trong hợp đồng; khi giao dịch mua bán hoàn tất, quyền sở hữu được cập nhật trực tiếp trên blockchain và có thể đối chiếu lại bằng các công cụ của mạng.

Trong thực tế, nhiều marketplace còn áp dụng cơ chế tối ưu hơn như off-chain order hoặc lazy listing, tức là người bán chỉ ký một lệnh bán và chỉ tạo giao dịch on-chain khi có người mua khớp lệnh. Cách tiếp cận này giảm số lần phải trả phí cho list hoặc unlist và hỗ trợ giao dịch nhanh, nhưng đi kèm độ phức tạp cao hơn: cần thiết kế chữ ký, chống replay, quản lý danh sách lệnh và xử lý các tình huống cạnh tranh khi nhiều người mua cùng nhắm vào một NFT. Với phạm vi đồ án hướng đến luồng nghiệp vụ mạch lạc và dễ trình bày, lựa chọn đưa list và unlist lên on-chain giúp hệ thống đơn giản hơn, đồng thời vẫn giữ đúng tính chất giao dịch có ràng buộc của marketplace.

3.9 Xác thực và bảo mật

3.9.1 SIWE (EIP-4361) cho đăng nhập bằng ví

SIWE (Sign-In With Ethereum) là một chuẩn đăng nhập bằng ví dựa trên cơ chế ký số, được đặc tả trong EIP-4361. Thay vì tạo tài khoản bằng mật khẩu, người dùng chứng minh mình thực sự sở hữu một địa chỉ ví bằng cách ký một thông điệp đăng nhập. Thông điệp này thường chứa các thông tin như tên miền của ứng dụng, địa chỉ ví, một giá trị nonce và một số trường mô tả phiên làm việc. Sau khi nhận chữ ký, phía server kiểm tra tính hợp lệ của chữ ký để xác nhận người đang đăng nhập chính là chủ sở hữu của địa chỉ ví đó. Nonce đóng vai trò quan trọng để tránh việc chữ ký cũ bị kẻ khác lấy lại và dùng lặp lại cho một lần đăng nhập khác.

Trong đồ án NFT marketplace, đăng nhập bằng ví là nền tảng vì hầu hết chức năng gắn với danh tính theo địa chỉ ví, chẳng hạn tạo NFT/collection, quản lý tài sản và xem dashboard cá nhân. SIWE được lựa chọn vì nó đưa ra một khuôn dạng thông điệp thống nhất, giúp quá trình xác thực rõ ràng và giảm rủi ro sai sót so với việc tự đặt ra format ký tuỳ ý. Ở mức triển khai, nonce có thể được lưu tạm để đảm bảo mỗi nonce chỉ sử dụng một lần, từ đó đáp ứng yêu cầu bảo mật đã nêu ở Chương 2. Đồng thời, cách đăng nhập này cũng phù hợp với trải nghiệm Web3: người dùng chỉ cần kết nối và ký xác nhận bằng ví, không phải tạo thêm mật khẩu hay ghi nhớ thông tin đăng nhập riêng.

Nếu dùng OAuth2 như đăng nhập bằng Google hoặc Facebook, người dùng có thể thấy quen thuộc hơn, nhưng cơ chế này không chứng minh được quyền sở hữu

địa chỉ ví, trong khi đây lại là yếu tố cốt lõi của marketplace. Một hướng khác là tự thiết kế cơ chế ký message, tuy nhiên dễ phát sinh sai sót ở những chi tiết quan trọng như nonce, ràng buộc theo domain hoặc cách chuẩn hoá nội dung thông điệp. Vì vậy, SIWE là lựa chọn phù hợp để vừa đúng bản chất xác thực bằng ví, vừa dễ trình bày và kiểm chứng.

3.9.2 JWT (RFC 7519) để bảo vệ API

JWT (JSON Web Token) là một chuẩn token được mô tả trong RFC 7519. Token có cấu trúc gọn, mang theo một số thông tin định danh dưới dạng JSON và đi kèm chữ ký số để phía server có thể kiểm tra xem dữ liệu có bị sửa trong quá trình truyền hay không. Một ưu điểm thường được nhắc đến của JWT là tính stateless: thay vì phải lưu thông tin phiên cho từng người dùng trên server, hệ thống có thể dựa vào token mà client gửi kèm để xác thực và phân quyền cho mỗi request. Cách làm này phù hợp với mô hình API hiện đại vì đơn giản hoá tầng backend và dễ mở rộng khi lượng truy cập tăng.

Trong đồ án, SIWE được dùng như bước xác thực ban đầu dựa trên chữ ký ví. Khi chữ ký hợp lệ, backend phát hành JWT để frontend sử dụng trong các request tiếp theo tới những API cần quyền truy cập. Luồng kết hợp này khá tự nhiên: người dùng chỉ cần ký một lần để đăng nhập, sau đó có thể thực hiện các thao tác off-chain như cập nhật hồ sơ, chỉnh sửa metadata, xem dashboard mà không phải ký lại cho từng thao tác nhỏ. Ngược lại, các hành động tạo giao dịch thật trên blockchain như mint, list hoặc buy vẫn yêu cầu người dùng xác nhận trong ví. Nhờ vậy, hệ thống vừa giữ đúng nguyên tắc an toàn cho các thao tác ràng buộc tài sản, vừa tránh làm trải nghiệm bị nặng nề vì phải ký quá nhiều lần.

Một phương án khác trong các hệ thống web truyền thống là xác thực theo session: server lưu session và client giữ cookie. Session có lợi thế ở chỗ dễ thu hồi và quản lý tập trung, nhưng sẽ phát sinh thêm phần quản lý trạng thái phía server, đặc biệt khi triển khai nhiều instance hoặc cần cơ chế chia sẻ session. Với định hướng triển khai gọn và mô hình API của đồ án, JWT là lựa chọn phù hợp vì nhẹ, phổ biến và dễ tích hợp với NestJS.

3.10 Triển khai và vận hành(cái này cần update lại, hiện tại đang chưa làm)

3.10.1 Docker và định hướng triển khai trên AWS

Docker là nền tảng container hoá, cho phép đóng gói ứng dụng và phụ thuộc thành image để chạy nhất quán ở nhiều môi trường. Trong thực tế phát triển, Docker giúp giảm các lỗi do lệch phiên bản runtime, thư viện và cấu hình. Với đồ án, việc tái lập môi trường nhất quán rất quan trọng: triển khai demo cần ổn định và ít phụ thuộc "máy cài thế nào".

Trong định hướng triển khai, hệ thống có thể đặt frontend và backend lên một compute service (EC2 hoặc dịch vụ container), cơ sở dữ liệu PostgreSQL có thể dùng dịch vụ quản lý như RDS để giảm công vận hành, Redis dùng ElastiCache để ổn định và dễ cấu hình, còn ảnh được lưu trên S3. Đây là mô hình triển khai khá điển hình và dễ giải thích, phù hợp yêu cầu "cấu hình có thể thay đổi mà không cần sửa mã nguồn" ở Chương 2 thông qua biến môi trường và cấu hình triển khai. Kubernetes là một lựa chọn mạnh khi cần điều phối cụm container và tự động mở rộng, nhưng vượt nhu cầu của đồ án và làm tăng độ phức tạp. Vì vậy, Docker được xem như công cụ đóng gói và tái lập môi trường, còn triển khai AWS theo hướng tối giản để đảm bảo hệ thống chạy ổn định.

3.11 Kết luận chương

Chương này đã làm rõ hướng tiếp cận công nghệ của đồ án và cách các lựa chọn kỹ thuật bám sát các yêu cầu đã nêu ở Chương 2. Trọng tâm của hệ thống là cân bằng giữa tính tin cậy của các thao tác gắn với blockchain và tính linh hoạt, hiệu năng của phần dữ liệu và trải nghiệm trên web. Từ nền tảng đó, chương tiếp theo sẽ trình bày về kết quả thực nghiệm của hệ thống.

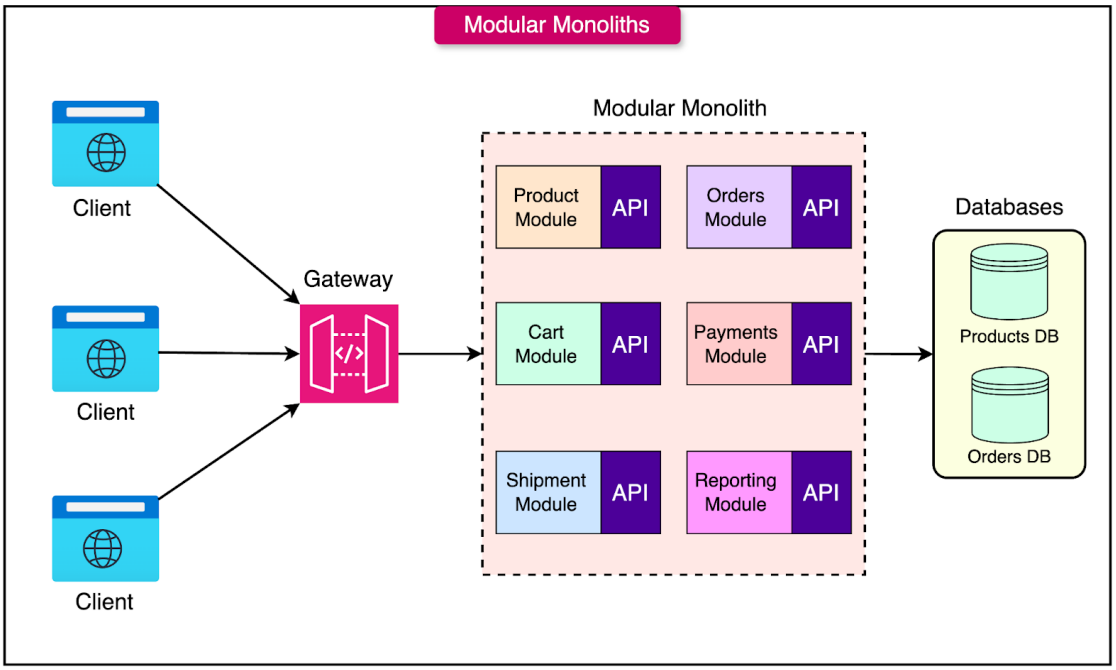
CHƯƠNG 4. THIẾT KẾ, TRIỂN KHAI VÀ ĐÁNH GIÁ HỆ THỐNG

4.1 Thiết kế kiến trúc

4.1.1 Lựa chọn kiến trúc phần mềm

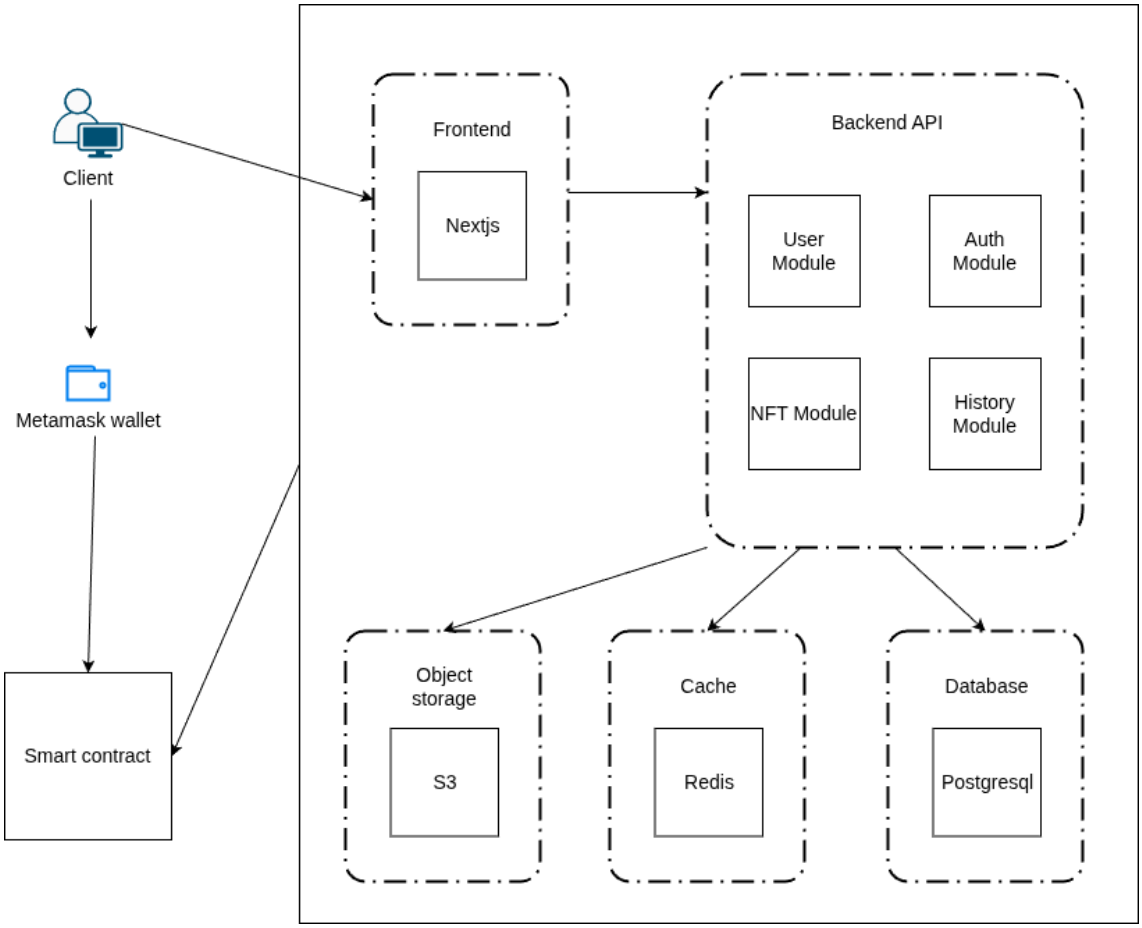
Kiến trúc phần mềm có thể được hiểu là cách tổ chức tổng thể của một hệ thống, bao gồm các thành phần chính, mối quan hệ giữa các thành phần và các nguyên tắc chi phối việc thiết kế và triển khai. Việc lựa chọn kiến trúc phù hợp giúp hệ thống có cấu trúc rõ ràng, dễ phát triển, dễ bảo trì và thuận lợi cho việc mở rộng trong tương lai. Trong thực tế, có nhiều phong cách kiến trúc được áp dụng tùy theo mục tiêu và bối cảnh như MVC/MVP trong ứng dụng hướng giao diện, kiến trúc phân lớp, Clean Architecture tập trung tách biệt business logic với chi tiết triển khai, hoặc Microservice hướng đến triển khai độc lập và khả năng mở rộng ở quy mô lớn.

Trong phạm vi đồ án này, em lựa chọn kiến trúc Modular Monolith. Đây là hướng tiếp cận trong đó hệ thống được triển khai như một ứng dụng thống nhất (monolith) nhằm đơn giản hóa quá trình triển khai và vận hành, đồng thời được chia thành các module theo nghiệp vụ (modular) để đảm bảo tính tổ chức, cô lập và dễ bảo trì. Mỗi module đóng vai trò quản lý tập trung các thành phần liên quan đến nghiệp vụ đó, nhờ vậy hạn chế tình trạng phụ thuộc chồng chéo và giúp mở rộng tính năng theo từng phần mà không ảnh hưởng đến các phần tính năng khác. Có thể xem Modular Monolith là sự cân bằng giữa monolith truyền thống và microservice. Không những phù hợp với điều kiện thời gian, nguồn lực và mục tiêu hoàn thiện các luồng nghiệp vụ của đồ án mà còn dễ dàng mở rộng nâng cấp khi hệ thống phát triển trong tương lai.



Hình 4.1: Ví dụ kiến trúc Modular monolith

Dựa trên kiến trúc đã lựa chọn, hệ thống NFT Marketplace được thiết kế theo mô hình client–server kết hợp blockchain được biểu thị bằng hình vẽ dưới đây.



Hình 4.2: Kiến trúc của hệ thống NFT marketplace

Ở phía client, người dùng thao tác trên giao diện web (Next.js/React) và tương tác với ví (Wallet Extension) để ký và gửi giao dịch blockchain. Các nghiệp vụ quan trọng như tạo, mua/bán NFT được thực hiện on-chain thông qua smart contract triển khai trên blockchain. Smart contract là thành phần nằm trên mạng blockchain, có nhiệm vụ thực thi logic giao dịch và phát sinh các sự kiện (events/logs) làm căn cứ xác minh giao dịch.

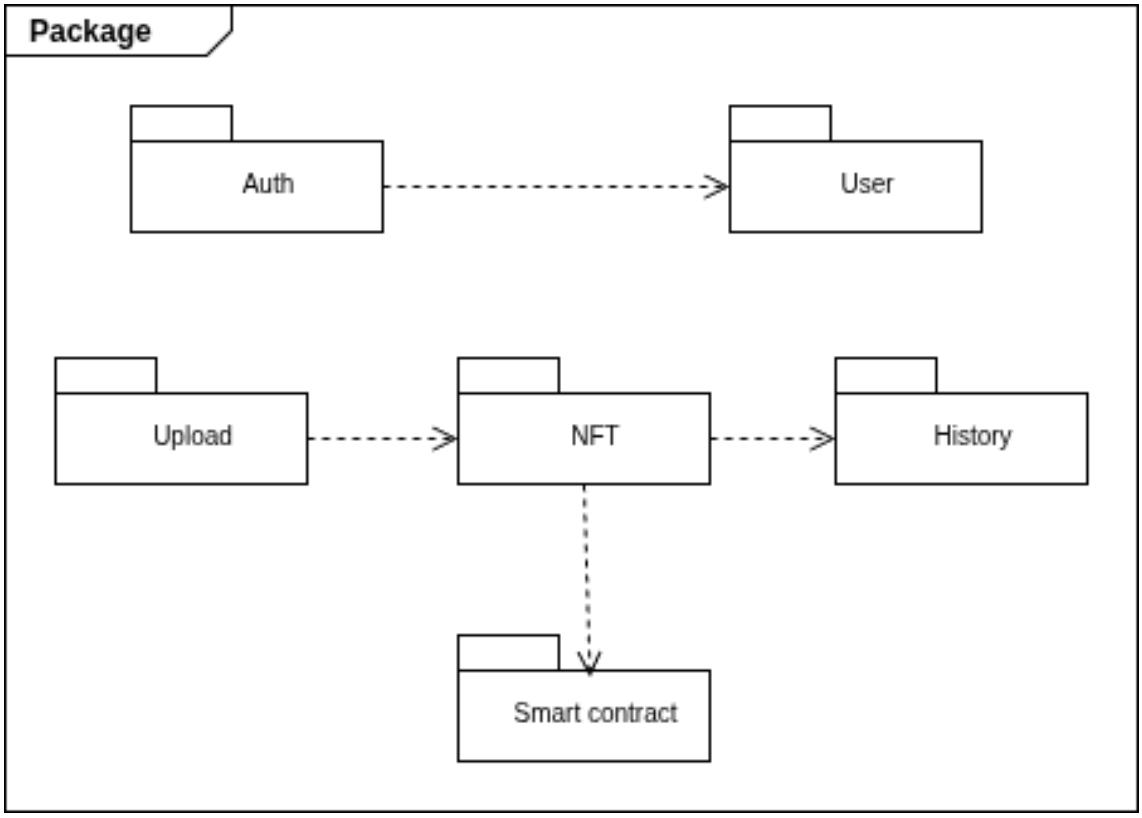
Ở phía server, backend được triển khai như một ứng dụng monolith và được chia thành các module nghiệp vụ chính gồm: Auth, User, NFT và History. Trong đó, Auth phụ trách xác thực và luồng kết nối ví; User quản lý thông tin hồ sơ người dùng; NFT xử lý nghiệp vụ liên quan NFT và trạng thái hiển thị; History ghi nhận các sự kiện và lịch sử giao dịch phục vụ hiển thị lịch sử và thống kê cơ bản. Với cách chia này, mỗi module có phạm vi trách nhiệm rõ ràng, dễ phát triển theo từng nhóm chức năng và thuận lợi cho việc bảo trì, mở rộng.

Về lưu trữ dữ liệu, hệ thống sử dụng PostgreSQL làm cơ sở dữ liệu chính cho dữ liệu nghiệp vụ, kết hợp Redis làm lớp cache để tối ưu các truy vấn đọc nhiều như danh sách NFT/collection, chi tiết NFT/collection và một số dữ liệu thống kê. Tài nguyên ảnh và metadata được lưu trên Amazon S3, đồng thời hệ thống có module upload hỗ trợ tải ảnh lên S3 để giảm tải cho backend và tối ưu tốc độ upload.

Một điểm thiết kế quan trọng của đồ án là cơ chế đồng bộ giữa dữ liệu on-chain và off-chain. Cụ thể, client thực hiện giao dịch blockchain qua ví và nhận txHash. Sau đó, client gọi API lên backend để lưu giữ liệu. Backend sẽ truy vấn blockchain thông qua RPC provider để kiểm tra giao dịch đã thành công hay chưa, có đúng smart contract và dữ liệu trong event có khớp với yêu cầu nghiệp vụ hay không. Chỉ khi xác minh hợp lệ, backend mới cập nhật dữ liệu vào PostgreSQL, ghi nhận lịch sử vào History module và cập nhật cache Redis khi cần. Cách làm này giúp giảm sai lệch trạng thái hiển thị off-chain so với trạng thái thực tế on-chain, đồng thời nâng cao tính tin cậy của hệ thống.

4.1.2 Thiết kế tổng quan

Với thiết kế theo hướng tách thành các module, mỗi module sẽ là một package riêng.



Hình 4.3: Biểu đồ tổng quan gói

Dưới đây là mô tả sơ lược về từng package:

Auth: Quản lý xác thực người dùng dựa trên ví. Package này chịu trách nhiệm kiểm tra chữ ký, tạo/duy trì token phiên đăng nhập.

User: Quản lý thông tin người dùng trong hệ thống. Package này lưu trữ và cập nhật hồ sơ người dùng.

NFT: Chịu trách nhiệm tạo, quản lý trạng thái và mua bán NFT.

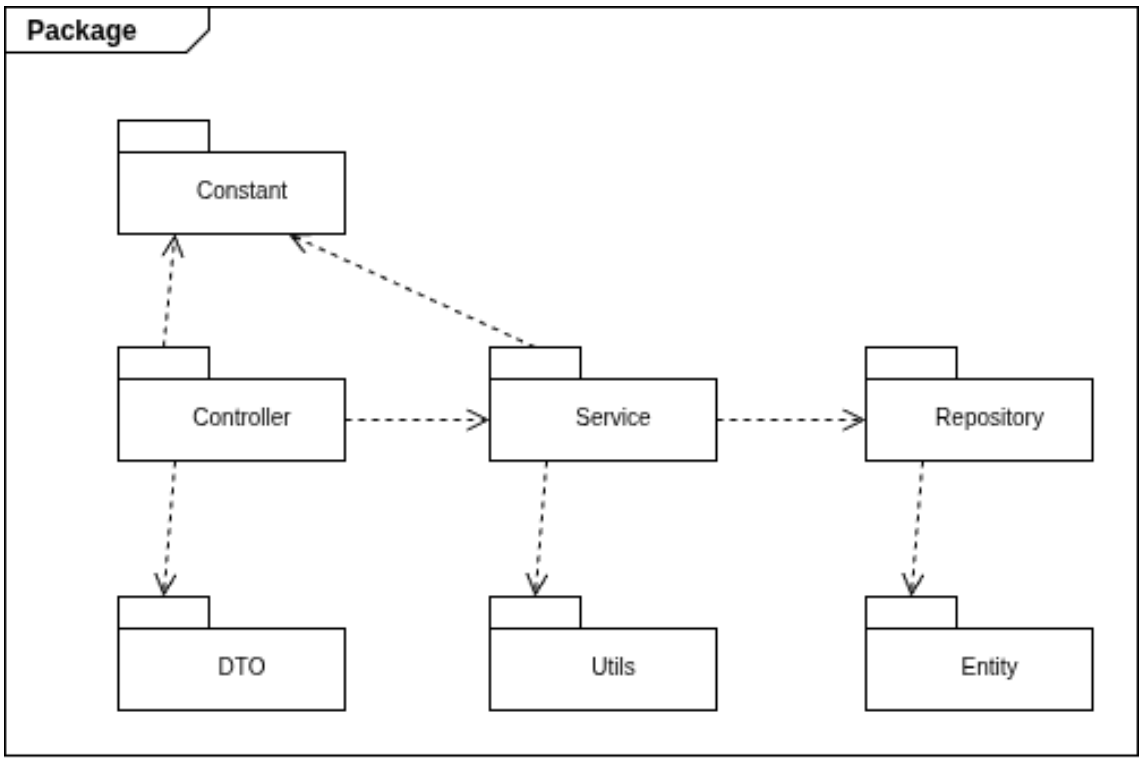
Upload: Quản lý upload và lưu trữ tài nguyên đa phương tiện.

History: Ghi nhận và truy vấn lịch sử hoạt động/giao dịch. Package này lưu các sự kiện khi phát sinh giao dịch trên smart contract, phục vụ màn hình lịch sử, thống kê cơ bản theo thời gian.

SmartContract: Giúp hệ thống tương tác với blockchain và smart contract.

4.1.3 Thiết kế chi tiết gói

Các package được tổ chức theo hướng module nên kiến trúc mỗi gói là tương tự nhau, dưới đây là chi tiết về package



Hình 4.4: Biểu đồ chi tiết gói

Controller: Đảm nhận vai trò tiếp nhận và xử lý các yêu cầu từ phía client thông qua các API. Nhiệm vụ chính của gói này là định tuyến yêu cầu đến đúng chức năng, kiểm tra/chuẩn hoá dữ liệu đầu vào và trả về phản hồi theo định dạng thống nhất. Controller không tập trung triển khai nghiệp vụ, mà đóng vai trò kết nối giữa client và hệ thống.

Service: Là nơi triển khai các nghiệp vụ cốt lõi của từng module. Gói này chịu trách nhiệm điều phối luồng xử lý: gọi các lớp truy cập dữ liệu, kết hợp các bước xử lý nghiệp vụ, áp dụng các quy tắc/điều kiện, và phối hợp với các module liên quan khi cần. Có thể xem service là tầng xử lý chính giúp đảm bảo nghiệp vụ được thực thi đúng và nhất quán.

Repository: Phụ trách tương tác với nguồn dữ liệu và các thành phần hạ tầng lưu trữ. Mục đích chính của gói này là đóng gói các thao tác đọc/ghi dữ liệu, giúp tầng service không phụ thuộc trực tiếp vào chi tiết triển khai như ORM, câu truy vấn SQL hay cơ chế lưu trữ cụ thể. Nhờ đó, hệ thống dễ bảo trì và dễ thay đổi công nghệ lưu trữ khi cần.

Entity: Mục đích của gói này là chuẩn hoá cấu trúc dữ liệu, quan hệ giữa các đối tượng, và làm nền tảng để các tầng phía trên thao tác một cách thống nhất, tránh việc xử lý dữ liệu rời rạc, thiếu nhất quán.

Utils: Chứa các thành phần hỗ trợ dùng chung trong phạm vi module. Mục đích

của utils là giảm trùng lặp mã nguồn, tăng khả năng tái sử dụng và giữ cho service tập trung vào nghiệp vụ chính.

DTO: Định nghĩa các cấu trúc dữ liệu dùng để trao đổi giữa client và server. Mục đích chính là đảm bảo cấu trúc dữ liệu rõ ràng, hỗ trợ kiểm tra tính hợp lệ của dữ liệu đầu vào và chuẩn hoá dữ liệu đầu ra. Việc tách DTO khỏi entity giúp tránh lộ cấu trúc nội bộ và tăng tính ổn định của API.

Constant: Tập hợp các hằng số và giá trị cấu hình logic được sử dụng xuyên suốt module. Mục đích của gói này là hạn chế việc cấu hình mã nguồn rải rác trong hệ thống, đảm bảo tính nhất quán và giúp việc thay đổi/điều chỉnh trở nên thuận tiện, an toàn hơn.

4.2 Thiết kế chi tiết

4.2.1 Thiết kế giao diện

Phần giao diện của hệ thống NFT Marketplace được thiết kế theo hướng responsive nhằm đảm bảo khả năng sử dụng tốt trên cả máy tính (desktop/laptop) và thiết bị di động (mobile). Mục tiêu của thiết kế là tạo trải nghiệm nhất quán, dễ thao tác, đồng thời hỗ trợ người dùng thực hiện các chức năng chính như connect wallet, tạo NFT, mua/bán NFT trên nhiều kích thước màn hình khác nhau.

Thông tin về màn hình mà ứng dụng hướng tới

Hệ thống hướng đến các thiết bị phổ biến trên thị trường hiện nay, bao gồm máy tính và điện thoại thông minh. Các độ phân giải dưới đây được lựa chọn do có tần suất sử dụng cao và đại diện tốt cho phần lớn thiết bị người dùng.

STT	Loại thiết bị	Kích thước phổ biến	Độ phân giải mục tiêu
1	Máy tính để bàn / Laptop	13” – 15.6”, 21” trở lên	1366×768; 1920×1080; 2560×14
2	Điện thoại thông minh	5” – 6.9”	1280×720; 1920×1080

Bảng 4.1: Bảng mô tả thông tin về màn hình mà hệ thống hướng tới

Chuẩn hoá thiết kế giao diện

Để đảm bảo tính thống nhất giữa các màn hình và giảm sai lệch trong quá trình triển khai, đồ án đưa ra các quy tắc chuẩn hoá giao diện được thiết kế theo bố cục linh hoạt, thay đổi theo độ rộng màn hình (breakpoint). Các breakpoint được sử dụng theo nhóm phổ biến để đảm bảo tương thích tốt:

Nhóm màn hình	Độ rộng (tham khảo)	Nguyên tắc bố cục
Mobile	$\leq$ 576px	Bố cục một cột, ưu tiên nội dung chính, nút hành động rõ ràng, thao tác chạm thuận tiện
Tablet nhỏ	577–768px	Hai cột đơn giản, giảm mật độ thông tin
Tablet lớn / Laptop	769–1024px	Ba cột, bắt đầu hiển thị thêm khu vực lọc/sắp xếp khi phù hợp
Desktop	1025–1440px	Nhiều cột, danh sách NFT hiển thị dạng lưới (grid) để tăng khả năng quan sát
Desktop lớn	$>$ 1440px	Tăng khoảng trắng, giữ chiều rộng nội dung hợp lý, tránh dàn trải quá rộng

Bảng 4.2: Các breakpoint và nguyên tắc bố cục responsive

Nguyên tắc tổng quát:

Trên desktop/laptop, danh sách NFT và collection hiển thị theo dạng lưới (grid) để tăng khả năng quan sát.

Trên mobile, giao diện chuyển sang một cột theo chiều dọc; thẻ NFT được thiết kế đủ lớn để dễ thao tác.

Các thông tin trọng tâm như hình ảnh NFT, tên NFT/collection, giá, trạng thái và nút hành động được ưu tiên đặt ở vùng dễ nhìn nhằm giảm số thao tác cuộn.

Chuẩn hoá phối màu và hiển thị chữ

Giao diện sử dụng tông màu xám và trắng làm nền chủ đạo nhằm tạo cảm giác hiện đại, tối giản và làm nổi bật nội dung hình ảnh. Màu xanh nước biển được sử dụng làm màu nhấn cho các hành động quan trọng và các thành phần cần thu hút sự chú ý.

Quy ước màu chữ:

Chữ màu xám cho nội dung thông thường và mô tả phụ.

Chữ màu xanh nước biển cho nội dung cần nhấn mạnh như liên kết, trạng thái, hoặc điểm nổi bật.

Chữ màu trắng dùng trên nền tối hoặc trên nút hành động để đảm bảo độ tương phản.

Chuẩn hoá nút bấm và điều khiển

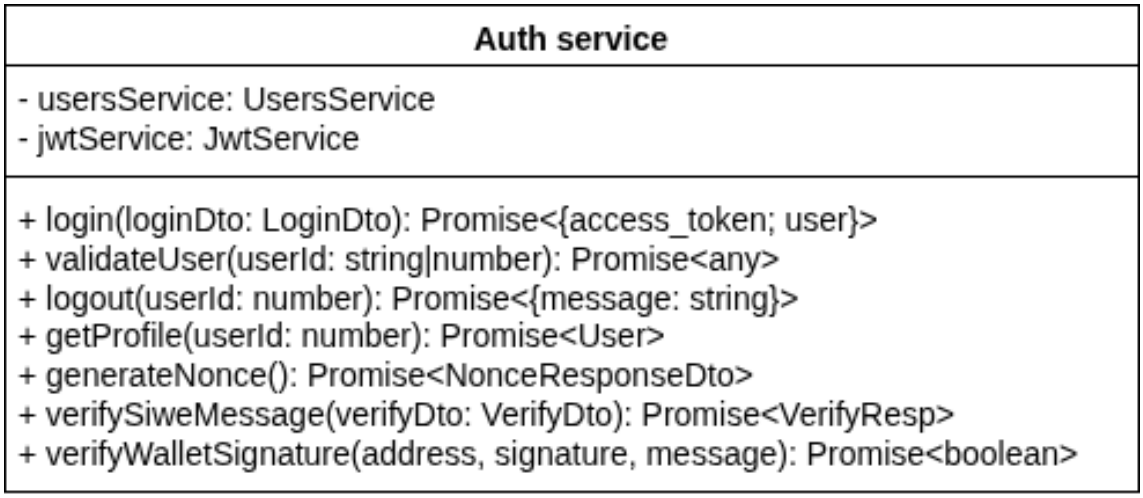
Hệ thống chuẩn hoá 2 loại nút chính:

Primary Button dùng cho hành động quan trọng như: *Connect Wallet*, *Create NFT*. **(2) Secondary Button** dùng cho hành động phụ/hỗ trợ như: *Cancel*.

Các nút được thiết kế thống nhất về kiểu dáng và trạng thái hiển thị: (1) Trạng thái bình thường: hiển thị rõ nội dung, dễ nhận biết. (2) Trạng thái hover/active (desktop): thay đổi sắc độ/viền để phản hồi thao tác. (3) Trạng thái disabled: giảm độ nổi bật để biểu thị không thể thao tác. (4) Trạng thái loading: hiển thị tiến trình khi hệ thống đang xử lý các thao tác như xác nhận giao dịch hoặc tải dữ liệu.

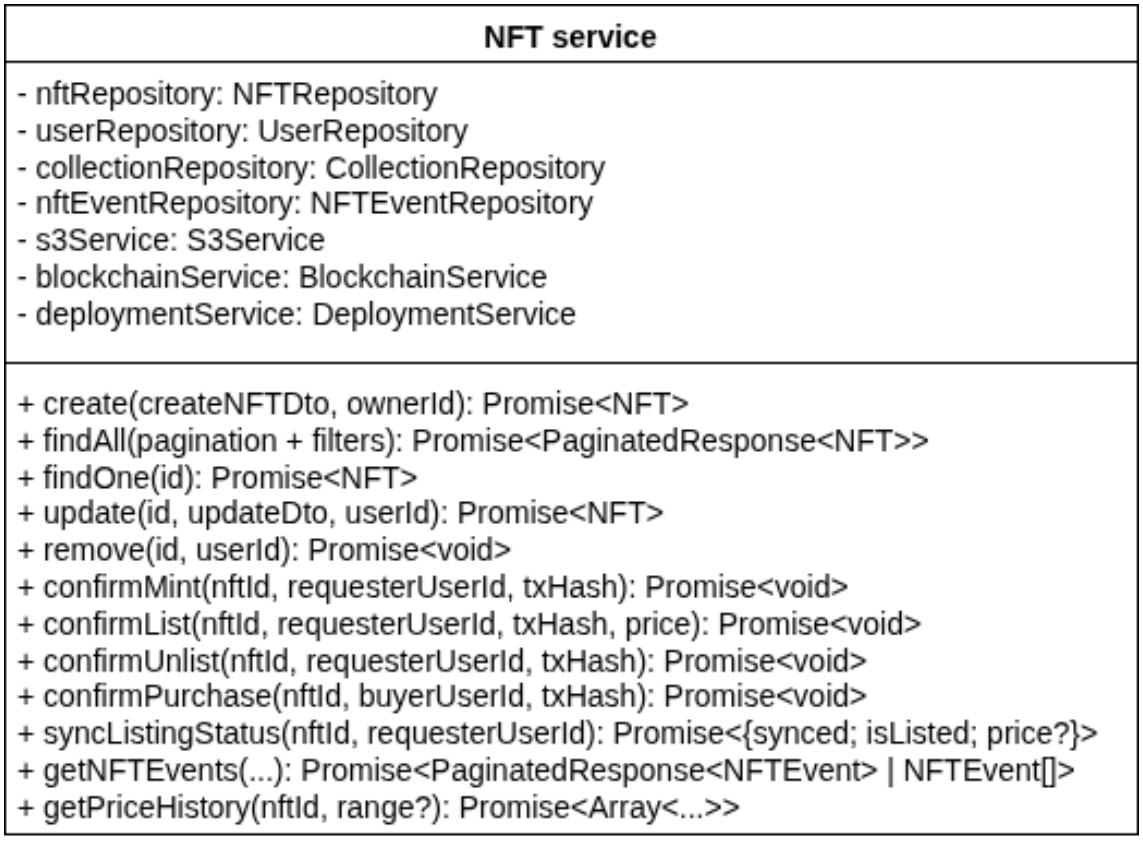
Minh họa thiết kế giao diện(đoạn này cần thêm ảnh vào, sau đọc lại nhớ bổ sung)

4.2.2 Thiết kế lớp



Hình 4.5: Biểu đồ Lớp Auth service

Lớp AuthService được sử dụng để thực hiện xác thực người dùng dựa trên địa chỉ ví và chữ ký (signature). Khi người dùng gửi yêu cầu đăng nhập, hệ thống chuẩn hoá địa chỉ ví để đảm bảo tính nhất quán dữ liệu. Nếu request có kèm chữ ký và thông điệp ký, backend sẽ tiến hành xác minh chữ ký nhằm đảm bảo người dùng thực sự sở hữu ví đó. Sau khi xác thực hợp lệ, hệ thống tìm hoặc tự động tạo mới tài khoản theo địa chỉ ví, sau đó phát hành JWT để người dùng sử dụng trong các request tiếp theo. Lớp cũng hỗ trợ xác thực theo chuẩn SIWE và cung cấp các chức năng phụ trợ như tạo nonce và truy vấn hồ sơ người dùng.



Hình 4.6: Biểu đồ Lớp NFT service

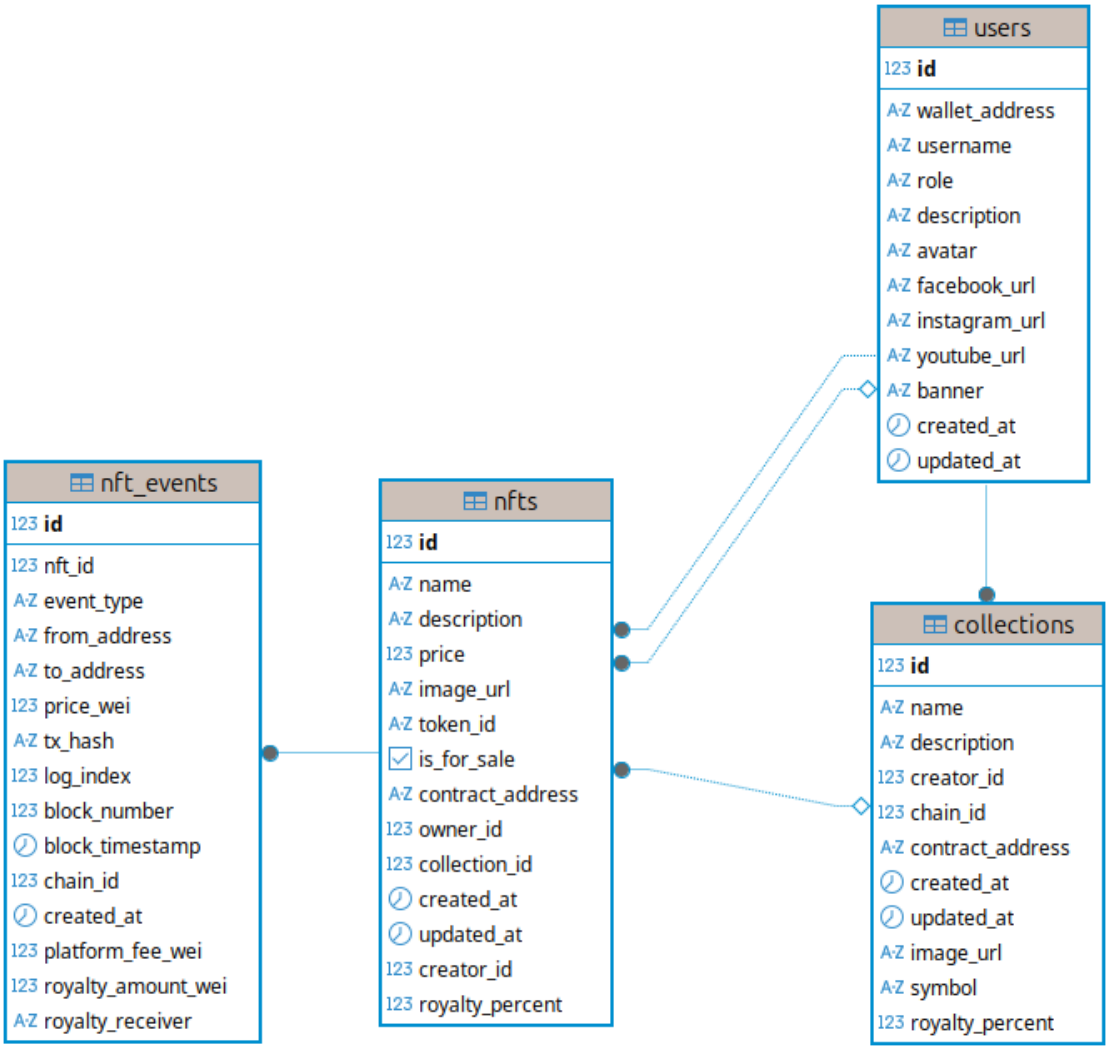
Lớp NFTService là lớp nghiệp vụ trung tâm của module NFT. Lớp này chịu trách nhiệm quản lý vòng đời của NFT trong hệ thống off-chain (tạo NFT, truy vấn, cập nhật, xoá), đồng thời xử lý các luồng nghiệp vụ gắn với blockchain như mint, list/unlist và purchase. Điểm quan trọng trong thiết kế là hệ thống không ghi nhận kết quả giao dịch chỉ dựa trên dữ liệu từ client, mà sử dụng txHash để truy vấn transaction và xác minh event phát sinh trên blockchain. Khi dữ liệu xác minh hợp lệ, backend mới cập nhật trạng thái NFT trong cơ sở dữ liệu và ghi nhận lịch sử giao dịch (event) để phục vụ các chức năng của history. Cách tiếp cận này giúp dữ liệu off-chain đồng bộ và có cơ sở tin cậy từ dữ liệu on-chain.

Biểu đồ trinh tự kết nối ví

Biểu đồ trinh mua NFT

4.2.3 Thiết kế cơ sở dữ liệu

Thiết kế sơ đồ thực thể liên kết:



Hình 4.7: Sơ đồ thực thể liên kết

Đặc tả bảng User:

Trường	**Kiểu dữ liệu**	**Mô tả**
id	BIGINT	Khóa chính (PK) định danh người dùng.
wallet_address	VARCHAR(42)	Địa chỉ ví (0x...), dùng để đăng nhập/định danh on-chain.
username	VARCHAR(100)	Tên hiển thị của người dùng trên hệ thống.
role	VARCHAR(30)	Vai trò người dùng (ví dụ: user/admin).
description	TEXT	Mô tả/bio của người dùng.
avatar	TEXT	URL ảnh đại diện.
facebook_url	TEXT	Liên kết Facebook (nếu có).
instagram_url	TEXT	Liên kết Instagram (nếu có).
youtube_url	TEXT	Liên kết Youtube (nếu có).
banner	TEXT	URL ảnh bìa (banner).

Trường	Kiểu dữ liệu	Mô tả
created_at	TIMESTAMP	Thời điểm tạo bản ghi.
updated_at	TIMESTAMP	Thời điểm cập nhật gần nhất.

Bảng 4.3: Chi tiết bảng người dùng

Đặc tả bảng Collection:

Trường	Kiểu dữ liệu	Mô tả
id	BIGINT	Khóa chính (PK) định danh collection.
name	VARCHAR(150)	Tên collection.
description	TEXT	Mô tả collection.
creator_id	BIGINT	Khóa ngoại (FK) → users.id, người tạo collection.
chain_id	INT	Mã mạng blockchain (ví dụ: 1, 56, 97...).
contract_address	VARCHAR(42)	Địa chỉ contract của collection trên blockchain.
created_at	TIMESTAMP	Thời điểm tạo bản ghi.
updated_at	TIMESTAMP	Thời điểm cập nhật gần nhất.
image_url	TEXT	URL ảnh đại diện collection.
symbol	VARCHAR(30)	Ký hiệu (symbol) của collection (ví dụ: BAYC).
royalty_percent	NUMERIC(5,2)	Phần trăm royalty áp dụng cho collection (ví dụ: 2.50).

Bảng 4.4: Chi tiết bảng Collection

Đặc tả bảng NFT:

Trường	Kiểu dữ liệu	Mô tả
id	BIGINT	Khóa chính (PK) định danh NFT trong DB.
name	VARCHAR(150)	Tên NFT.
description	TEXT	Mô tả NFT.
price	NUMERIC(38,0)	Giá hiện tại (có thể lưu theo wei hoặc theo đơn vị chuẩn tùy thiết kế).
image_url	TEXT	URL ảnh NFT (S3/IPFS/HTTP...).
token_id	BIGINT	TokenId on-chain của NFT.
is_for_sale	BOOLEAN	Trạng thái đang được niêm yết bán hay không.
contract_address	VARCHAR(42)	Địa chỉ contract chứa NFT.
owner_id	BIGINT	FK → users.id, owner hiện tại.
collection_id	BIGINT	FK → collections.id, NFT thuộc collection nào.

Trường	Kiểu dữ liệu	Mô tả
created_at	TIMESTAMP	Thời điểm tạo bản ghi.
updated_at	TIMESTAMP	Thời điểm cập nhật gần nhất.
creator_id	BIGINT	FK → users.id, người tạo/mint NFT.
royalty_percent	NUMERIC(5,2)	Royalty cho NFT.

Bảng 4.5: Chi tiết bảng NFT

Đặc tả bảng NFT event:

Trường	Kiểu dữ liệu	Mô tả
id	BIGINT	Khóa chính (PK) định danh sự kiện NFT.
nft_id	BIGINT	FK → nfts.id, NFT liên quan tới sự kiện.
event_type	VARCHAR(50)	Loại sự kiện (mint/list/buy/transfer/cancel...).
from_address	VARCHAR(42)	Địa chỉ ví nguồn (người gửi/seller).
to_address	VARCHAR(42)	Địa chỉ ví đích (người nhận/buyer).
price_wei	NUMERIC(38,0)	Giá giao dịch theo wei (nếu có).
tx_hash	VARCHAR(66)	Transaction hash trên blockchain.
log_index	INT	Log index trong transaction (phục vụ đối chiếu event log).
block_number	BIGINT	Số block chứa transaction.
block_timestamp	TIMESTAMP	Thời điểm block được ghi nhận.
chain_id	INT	Mã mạng blockchain phát sinh sự kiện.
created_at	TIMESTAMP	Thời điểm hệ thống lưu sự kiện vào DB.
platform_fee_wei	NUMERIC(38,0)	Phí nền tảng thu được (tính theo wei).
royalty_amount_wei	NUMERIC(38,0)	Số tiền royalty trả cho creator (theo wei).
royalty_receiver	VARCHAR(42)	Địa chỉ nhận royalty.

Bảng 4.6: Chi tiết bảng NFT event

4.3 Xây dựng ứng dụng

4.3.1 Thư viện và công cụ sử dụng

Các công cụ được sử dụng trong quá trình phát triển sản phẩm được liệt kê dưới bảng 4.7:

STT	Mục đích	Tên công cụ	Phiên bản	URL
1	Ngôn ngữ + runtime backend	Node.js (LTS) + TypeScript	24.12.0; 5.9.3	https://nodejs.org/; https://www.typescriptlang.org/
2	Framework backend API	NestJS	11.1.10	https://nestjs.com/
3	Framework frontend web	Next.js	16.1.1	https://nextjs.org/
4	Hệ quản trị CSDL (SQL)	PostgreSQL	18.1	https://www.postgresql.org/
5	Cache / tăng tốc truy vấn	Redis (Open Source)	8.4.0	https://redis.io/
6	IDE phát triển	Visual Studio Code	1.107.1	https://code.visualstudio.com/
7	Triển khai/vận hành trên AWS	AWS CLI (v2)	2.32.24	https://aws.amazon.com/cli/

Bảng 4.7: Các công cụ chính sử dụng trong quá trình phát triển hệ thống

4.3.2 Kết quả đạt được(cần cập nhật lại bảng chỉ số thống kê)

Trong quá trình thực hiện đồ án, em đã xây dựng được một hệ thống NFT Marketplace có thể vận hành và trình diễn đầy đủ các luồng chức năng chính. Kết quả đạt được không chỉ dừng ở việc hoàn thiện mã nguồn, mà còn được đóng gói thành các sản phẩm cụ thể để thuận tiện cho việc triển khai, kiểm thử và demo. Các sản phẩm đóng gói bao gồm: ứng dụng giao diện người dùng (frontend) giúp người dùng thao tác tạo/bán/mua và khám phá NFT; dịch vụ backend API xử lý nghiệp vụ, quản lý dữ liệu và đồng bộ trạng thái sau giao dịch; smart contract triển khai các chức năng on-chain liên quan đến NFT; ngoài ra kèm theo các script cấu hình/migration phục vụ khởi tạo dữ liệu. Việc đóng gói theo từng thành phần giúp hệ thống có cấu trúc rõ ràng, dễ quản lý và có thể mở rộng trong tương lai.

Chỉ số thống kê	Frontend	Backend	Smart contract	Tổng
Số dòng code (LOC)	...	...	...	...
Số thư mục/module chính	...	...	...	...
Số file mã nguồn	...	...	...	...
Số lớp (class) / thành phần chính	...	...	...	...
Dung lượng toàn bộ mã nguồn (MB)	...	...	...	...
Dung lượng sản phẩm sau build (MB)	...	...	...	...

Bảng 4.8: Thống kê quy mô mã nguồn và sản phẩm đóng gói

4.3.3 Minh họa các chức năng chính

Sinh viên lựa chọn và đưa ra màn hình cho các chức năng chính, quan trọng, và thú vị nhất. Mỗi giao diện cần phải có lời giải thích ngắn gọn. Khi giải thích, sinh viên có thể kết hợp với các chú thích ở trong hình ảnh giao diện.

4.4 Kiểm thử

4.4.1 Mục tiêu và phạm vi kiểm thử

Phần này tập trung thiết kế và thực hiện kiểm thử cho các chức năng quan trọng nhất của hệ thống NFT Marketplace: đăng nhập kết nối ví; tạo NFT; niêm yết và mua/bán NFT.

4.4.2 Kỹ thuật kiểm thử đã sử dụng

Trong quá trình thiết kế test case, em sử dụng kết hợp các kỹ thuật sau:

- **Kiểm thử hộp đen (Black-box testing):** thiết kế test case dựa trên đầu vào/đầu ra và hành vi mong đợi theo yêu cầu.
- **Phân lớp tương đương (Equivalence Partitioning):** chia dữ liệu đầu vào thành nhóm hợp lệ/không hợp lệ để giảm số lượng test case nhưng vẫn đảm bảo bao phủ.
- **Giá trị biên (Boundary Value Analysis):** kiểm thử tại các ngưỡng quan trọng (ví dụ giá = 0, giá rất nhỏ, độ dài chuỗi tối đa,...).
- **Kiểm thử theo luồng (Scenario/Flow testing):** kiểm thử theo chuỗi hành động thực tế (login → tạo NFT → list → buy → cập nhật).
- **Kiểm thử API/kiểm thử tích hợp (API & Integration testing):** gọi API và đối chiếu dữ liệu phát sinh trong cơ sở dữ liệu.

4.4.3 Thiết kế các trường hợp kiểm thử

a, Chức năng 1: Đăng nhập bằng ví (SIWE)

Người dùng lấy nonce, ký thông điệp SIWE bằng ví, backend xác thực chữ ký và cấp access token.

Mã TC	Mục tiêu	Dữ liệu vào	Các bước thực hiện	Kết quả mong đợi
TC-LOGIN-01	Đăng nhập thành công	wallet hợp lệ, message hợp lệ, signature hợp lệ	(1) Lấy nonce (2) Ký SIWE (3) Gửi verify/login	Trả token; user được tạo mới(nếu chưa tồn tại trong hệ thống).
TC-LOGIN-02	Sai chữ ký	signature không khớp message	Gửi verify/login với signature sai	Trả lỗi và không cấp token.
TC-LOGIN-03	Nonce đã dùng/hết hạn	nonce cũ hoặc bị dùng lại	Dùng lại nonce và gửi verify/login	Bị từ chối và yêu cầu lấy nonce mới.

Mã TC	Mục tiêu	Dữ liệu vào	Các bước thực hiện	Kết quả mong đợi
TC-LOGIN-04	Sai chain/domain	chainId hoặc domain không đúng cấu hình	Gửi verify/login với chain/domain sai	Bị từ chối xác thực và trả lỗi.
TC-LOGIN-05	Wallet sai định dạng	wallet address sai chuẩn	Gửi verify/login với địa chỉ ví sai	Validate thất bại và trả lỗi.

Bảng 4.9: Thiết kế test case cho chức năng đăng nhập bằng ví (SIWE)

b, Chức năng 2: Tạo NFT

Người dùng nhập thông tin NFT (tên, mô tả, ảnh/metadata), backend lưu dữ liệu NFT và liên kết vào collection.

Mã TC	Mục tiêu	Dữ liệu vào	Các bước thực hiện	Kết quả mong đợi
TC-NFT-01	Tạo NFT hợp lệ	dữ liệu NFT hợp lệ	Gọi POST /nfts	Tạo thành công, trả về dữ liệu đúng, DB có bản ghi mới.
TC-NFT-02	Thiếu trường bắt buộc	name rỗng hoặc thiếu	Gọi POST /nfts	Trả lỗi, không tạo bản ghi.
TC-NFT-03	Collection không tồn tại	collectionId sai	Gọi POST /nfts	Trả lỗi, không tạo NFT.
TC-NFT-04	Ảnh/URL sai định dạng	imageUrl không hợp lệ	Gọi POST /nfts	Trả lỗi, thông báo rõ trường sai.
TC-NFT-05	Giá trị biên (tên)	name quá dài hoặc vượt giới hạn	Gọi POST /nfts	Trả lỗi validation và yêu cầu user nhập lại.

Bảng 4.10: Thiết kế test case cho chức năng tạo NFT

c, Chức năng 3: Niêm yết bán và cập nhật trạng thái sau giao dịch

User niêm yết NFT với giá bán; sau giao dịch mua trên ví, frontend gọi API để backend cập nhật dữ liệu dựa theo transaction event.

Mã TC	Mục tiêu	Dữ liệu vào	Các bước thực hiện	Kết quả mong đợi
TC-LIST-01	List NFT thành công	nft thuộc owner hiện tại, price $>$ 0	Gọi POST /listings	Chuyển trạng thái NFT listing sang active và cập nhật giá bán.
TC-LIST-02	Không phải chủ sở hữu	nft không thuộc user hiện tại	Gọi POST /listings	Trả lỗi và không tạo listing.
TC-LIST-03	Giá bán không hợp lệ (biên)	price = 0 hoặc price $<$ 0	Gọi POST /listings	Trả lỗi và không tạo listing.

Mã TC	Mục tiêu	Dữ liệu vào	Các bước thực hiện	Kết quả mong đợi
TC-BUY-01	Xác nhận mua thành công	txHash hợp lệ, buyer/seller hợp lệ	Gọi POST /transactions/confirm	Cập nhật thành công owner sang buyer, tạo bản ghi history/event tương ứng.
TC-BUY-02	Trùng txHash (idempotent)	txHash đã tồn tại trong DB	Gọi lại confirm với cùng txHash	Không tạo trùng; trả kết quả an toàn khi retry (idempotent).
TC-CANCEL-01	Hủy listing	listing đang active	Gọi POST /listings/cancel	Chuyển trạng thái listing sang unactive thành công.

Bảng 4.11: Thiết kế test case cho chức năng niêm yết và cập nhật trạng thái sau giao dịch

4.5 Triển khai(khi nào deploy có cấu hình server thì cập nhật vào đây)

CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

Chương 5 tổng hợp các giải pháp và đóng góp mà em cho là quan trọng và nổi bật nhất trong quá trình xây dựng hệ thống NFT marketplace. Khác với các chương trước chủ yếu mô tả yêu cầu, thiết kế, chương này tập trung vào cách em phân tích để xác định các vấn đề gặp phải, cách lựa chọn giải pháp phù hợp và kết quả đạt được.

5.1 Đồng bộ dữ liệu từ blockchain về cơ sở dữ liệu

5.1.1 Giới thiệu

Trong NFT marketplace, các thao tác cốt lõi như tạo NFT, niêm yết, mua/bán đều phát sinh giao dịch trên blockchain. Trong khi đó, hệ thống off-chain (backend và cơ sở dữ liệu) lại cần lưu trữ và tổ chức dữ liệu để phục vụ tìm kiếm, thống kê, hiển thị lịch sử và tối ưu trải nghiệm người dùng. Vì vậy, một bài toán trọng tâm là làm sao để trạng thái hiển thị off-chain bám sát trạng thái thực tế on-chain, đồng thời vẫn giữ được trải nghiệm sử dụng mượt mà và dễ kiểm soát.

5.1.2 Giải pháp

Hệ thống chỉ thực hiện cập nhật dữ liệu phía backend sau khi giao dịch trên blockchain đã có kết quả xác định. Cụ thể, phía frontend thực hiện giao dịch thông qua ví, chờ giao dịch được ghi nhận thành công hoặc tối thiểu nhận được transaction hash hợp lệ. Sau đó, frontend gọi các API để yêu cầu backend xử lý và cập nhật dữ liệu off-chain.

Backend không sử dụng trực tiếp toàn bộ dữ liệu do client gửi lên để cập nhật trạng thái, mà lấy transaction hash làm căn cứ kiểm tra lại giao dịch trên blockchain. Backend truy vấn transaction receipt để xác định trạng thái thực thi của giao dịch (thành công hoặc thất bại). Trên cơ sở đó, backend trích xuất các thông tin cần thiết phục vụ cập nhật hệ thống như tokenId, địa chỉ người mua hoặc người bán, giá trị giao dịch và địa chỉ hợp đồng từ dữ liệu receipt hoặc từ các log cơ bản của giao dịch.

Quá trình cập nhật dữ liệu được tổ chức theo nguyên tắc on-chain đóng vai trò nguồn dữ liệu chuẩn cho các thông tin liên quan đến quyền sở hữu và giao dịch. Dữ liệu off-chain được cập nhật nhằm mục đích tối ưu truy vấn và hiển thị cho người dùng. Trong trường hợp phát sinh sai lệch, hệ thống ưu tiên đối chiếu và hiệu chỉnh theo trạng thái on-chain để đảm bảo tính nhất quán của thông tin quan trọng.

5.1.3 Kết quả đạt được

Với cách làm này, backend sẽ luôn được đồng bộ dữ liệu từ blockchain, giúp hiệu năng và trải nghiệm người dùng tốt hơn thay vì phải truy vấn trực tiếp trên blockchain(tốc độ rất chậm). Đồng thời khi làm chủ dữ liệu, hệ thống có thể dễ dàng phát triển và mở rộng tính năng mới dựa vào lượng data sẵn có.

5.2 Cơ chế xác minh giao dịch để tránh cập nhật sai hoặc bị giả mạo

5.2.1 Giới thiệu

Trong hệ thống NFT marketplace, dữ liệu off-chain trên backend được sử dụng để phục vụ truy vấn, hiển thị và quản lý trạng thái nghiệp vụ. Nếu backend chấp nhận yêu cầu cập nhật trạng thái chỉ dựa trên dữ liệu do client gửi lên như tokenId, giá hoặc địa chỉ người mua, hệ thống có thể phát sinh cập nhật sai lệch. Nguyên nhân có thể đến từ lỗi phía frontend, thao tác không mong muốn của người dùng, hoặc các yêu cầu giả mạo được gửi trực tiếp đến API.

Khi trạng thái quan trọng bị cập nhật sai như trạng thái đã mua, đã niêm yết hoặc đã mint, dữ liệu hiển thị sẽ không còn phản ánh đúng trạng thái thực tế trên blockchain. Điều này ảnh hưởng trực tiếp đến tính tin cậy của hệ thống và trải nghiệm người dùng. Vì vậy, yêu cầu đặt ra là mỗi lần cập nhật off-chain cần có căn cứ từ giao dịch on-chain và hạn chế tối đa việc client tự khai báo trạng thái.

5.2.2 Giải pháp

Giải pháp của hệ thống được xây dựng dựa trên nguyên tắc sử dụng transaction hash như bằng chứng giao dịch và bắt buộc xác minh trước khi cập nhật dữ liệu off-chain. Cụ thể, sau khi người dùng thực hiện giao dịch thông qua ví trên frontend và gửi yêu cầu xác nhận lên backend, hệ thống sẽ truy vấn transaction receipt để kiểm tra giao dịch có tồn tại trên blockchain và đã thực thi thành công hay chưa. Đồng thời, backend đối chiếu địa chỉ ví tham gia giao dịch với danh tính người dùng đang đăng nhập nhằm đảm bảo giao dịch được tạo bởi đúng ví và đúng ngữ cảnh nghiệp vụ. Tiếp theo, backend kiểm tra giao dịch có tương tác đúng với hợp đồng thông minh và phương thức tương ứng của chức năng (ví dụ: mint/list/buy/cancel), tránh trường hợp client cung cấp nhầm một transaction hash không liên quan. Bên cạnh đó, các API thay đổi trạng thái được thiết kế theo hướng hạn chế quyền cập nhật trực tiếp: thay vì cho phép client gửi trạng thái tuỳ ý để backend ghi nhận ngay, các endpoint quan trọng chỉ chấp nhận transaction hash và một số thông tin tối thiểu cần thiết cho việc đối chiếu, qua đó giảm rủi ro cập nhật sai hoặc bị thao túng dữ liệu off-chain.

5.2.3 Kết quả đạt được

Cơ chế xác minh theo transaction hash giúp dữ liệu off-chain không phụ thuộc hoàn toàn vào dữ liệu do client cung cấp, đồng thời tăng độ tin cậy cho các luồng nghiệp vụ chính như mint, niêm yết và mua NFT. Trong quá trình kiểm thử, hệ thống có thể phát hiện và từ chối cập nhật đối với các trường hợp transaction hash không hợp lệ, giao dịch không liên quan đến chức năng hoặc giao dịch thực thi thất bại. Nhờ đó, hệ thống hạn chế được sai lệch giữa dữ liệu hiển thị và trạng thái thực tế trên blockchain.

5.3 Thiết kế cập nhật dữ liệu sau giao dịch theo hướng idempotent và có trạng thái trung gian

thiệu

5.3.1 Giới thiệu

Trong môi trường thực tế, người dùng có thể gặp tình trạng mạng chập chờn hoặc thao tác lặp lại (refresh trang, bấm lại nút xác nhận), dẫn tới việc FE gọi API xác nhận nhiều lần cho cùng một giao dịch. Nếu backend mỗi lần nhận request đều cập nhật thêm một lần nữa, hệ thống có thể phát sinh lỗi như tạo bản ghi trùng, cộng thống kê sai, hoặc đẩy trạng thái NFT sang sai trạng thái.

5.3.2 Giải pháp

Dùng idempotency theo txHash. Mỗi giao dịch sau khi được xác nhận sẽ tương ứng với một bản ghi xử lý ở database. Khi nhận request xác nhận, backend kiểm tra xem txHash đã được xử lý thành công trước đó hay chưa; nếu rồi thì trả về kết quả hiện tại thay vì cập nhật lại. Nhờ vậy, việc FE gọi lại API không làm thay đổi kết quả cuối.

5.3.3 Kết quả đạt được

Thiết kế này giúp hệ thống ổn định hơn khi người dùng thao tác lặp hoặc khi mạng không ổn định. Đồng thời, việc dùng txHash làm khoá kiểm soát giúp tránh trùng dữ liệu và tránh cập nhật sai thống kê trong các luồng mua/bán.

5.4 Tối ưu các API đọc nhiều bằng cache và tổ chức truy vấn phục vụ UI

5.4.1 Giới thiệu

NFT marketplace có nhiều màn hình đọc nhiều: danh sách NFT, chi tiết NFT, trang collection, dashboard thống kê cơ bản, lịch sử giao dịch/giá. Nếu mỗi lần tải trang đều thực hiện truy vấn nặng hoặc tính toán lặp lại, thời gian phản hồi sẽ giảm khi dữ liệu tăng.

5.4.2 Giải pháp

Em tập trung vào hai hướng.

Thứ nhất là cache các dữ liệu đọc nhiều và ít thay đổi trong một khoảng thời gian (thống kê tổng quan, thông tin người dùng, NFT,...). Cache được đặt TTL hợp lý để cân bằng giữa độ "mới" của dữ liệu và hiệu năng.

Thứ hai là tổ chức truy vấn theo đúng nhu cầu hiển thị của UI. Các endpoint danh sách đều có phân trang và chỉ trả về trường dữ liệu cần thiết cho danh sách, tránh trả về toàn bộ dữ liệu như trang chi tiết. Với những dữ liệu có thể phát sinh nhiều bản ghi (lịch sử), em ưu tiên truy vấn theo trang và theo thứ tự thời gian, giúp tải dần và giảm tải cho hệ thống.

5.4.3 Kết quả đạt được

Giải pháp giúp các màn hình đọc chính đạt thời gian phản hồi ổn định hơn trong quá trình demo và kiểm thử. Đồng thời, cache giúp giảm số truy vấn lặp đối với các dữ liệu phổ biến.

5.5 Chiến lược migration an toàn để giảm rủi ro gián đoạn

5.5.1 Giới thiệu

Trong quá trình phát triển, việc thay đổi yêu cầu kéo theo thay đổi cấu trúc dữ liệu. Nếu thay đổi trực tiếp trên cột đang được sử dụng (ví dụ đổi kiểu dữ liệu), rủi ro phát sinh lỗi hoặc khoá bảng là điều cần cân nhắc, đặc biệt khi hệ thống phát triển và dữ liệu tăng.

5.5.2 Giải pháp

Em áp dụng chiến lược migration theo hướng tạo cột mới (shadow column): thay vì sửa trực tiếp cột cũ, em tạo cột mới với kiểu dữ liệu/ý nghĩa đúng, thực hiện backfill dữ liệu theo lô, cập nhật ứng dụng để chuyển dần sang đọc/ghi theo cột mới, và chỉ loại bỏ cột cũ khi đã ổn định. Cách tiếp cận này giúp giảm rủi ro và cho phép quay lui dễ hơn nếu phát sinh sai sót.

5.5.3 Kết quả đạt được

Chiến lược này giúp quá trình thay đổi schema an toàn hơn, hạn chế tác động tới các chức năng đang hoạt động, đồng thời làm rõ cách em tiếp cận bài toán theo hướng có kiểm soát rủi ro.

5.6 Kết luận chương

Các đóng góp trong chương tập trung vào những vấn đề em trực tiếp giải quyết trong phạm vi triển khai hiện tại: đồng bộ dữ liệu on-chain/off-chain theo mô hình FE giao dịch và backend xác minh bằng *txHash*; tăng độ tin cậy của cập nhật dữ

liệu bằng cơ chế đối chiếu giao dịch; đảm bảo ổn định thông qua idempotency và trạng thái trung gian; tối ưu trải nghiệm bằng chiến lược tối ưu API đọc qua cache; và cuối cùng là cách tiếp cận migration an toàn để hệ thống dễ thay đổi trong giai đoạn phát triển.

CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

6.1 Kết luận

Sau quá trình thực hiện, đồ án đã xây dựng được một hệ thống NFT Marketplace với quy trình tổng thể tương đối hoàn chỉnh, giúp người dùng có thể tiếp cận và sử dụng các chức năng cốt lõi của nền tảng một cách thuận tiện.

Trong quá trình phát triển, em không chỉ tập trung vào việc hoàn thiện giao diện hay triển khai từng chức năng riêng lẻ, mà chú trọng tổ chức hệ thống theo hướng có cấu trúc rõ ràng và có khả năng mở rộng. Em đã thiết kế kiến trúc tổng quan, phân tách các module theo từng nhóm nghiệp vụ, xây dựng các API phục vụ các luồng xử lý chính, đồng thời chuẩn hoá cơ chế tiếp nhận và xử lý dữ liệu phát sinh từ giao dịch blockchain. Nhờ cách tổ chức này, hệ thống có nền tảng tốt để mở rộng thêm chức năng và cải thiện hiệu năng trong tương lai. Bên cạnh đó, em đặc biệt quan tâm đến tính đúng đắn của dữ liệu: hệ thống chỉ ghi nhận và cập nhật trạng thái nghiệp vụ khi giao dịch on-chain được xác nhận thành công. Điều này giúp dữ liệu hiển thị có cơ sở rõ ràng, hạn chế tình trạng sai lệch trạng thái khi người dùng thao tác qua ví hoặc khi giao dịch không thành công.

Ngoài phần nghiệp vụ, đồ án cũng thể hiện định hướng triển khai theo hướng thực tiễn trong việc tổ chức dữ liệu và trải nghiệm sử dụng. Người dùng có thể quản lý tài sản của mình, theo dõi trạng thái, xem thông tin chi tiết và lịch sử liên quan một cách trực quan. Dữ liệu được lưu trữ theo hướng tách bạch giữa on-chain và off-chain, từ đó hỗ trợ linh hoạt cho việc hiển thị, truy vấn và mở rộng thống kê theo nhu cầu. Nhìn chung, đồ án đã tạo ra một sản phẩm có thể vận hành ổn định theo luồng nghiệp vụ cơ bản của marketplace, thể hiện được tư duy thiết kế hệ thống và khả năng triển khai end-to-end.

Khi so sánh với các nền tảng marketplace phổ biến như OpenSea, hệ thống trong đồ án có cùng định hướng về bản chất nghiệp vụ nhưng phạm vi triển khai nhỏ hơn và phù hợp với mục tiêu học thuật. Các sản phẩm thương mại thường hỗ trợ đa chuỗi, đa hình thức giao dịch và sở hữu hạ tầng vận hành mạnh, kèm theo các cơ chế giám sát, tối ưu và chuẩn hoá vận hành ở quy mô lớn. Trong khi đó, đồ án của em tập trung làm rõ kiến trúc, đảm bảo luồng xử lý xuyên suốt và xây dựng một phiên bản marketplace có thể vận hành ổn định trong phạm vi triển khai của đồ án. Qua đó, kết quả đạt được vừa đáp ứng yêu cầu học thuật, vừa tạo nền tảng để tiếp tục mở rộng theo hướng sản phẩm thực tế.

6.2 Hướng phát triển

Trong thời gian tới, để hệ thống có thể tiến gần hơn tới một sản phẩm triển khai thực tế, phần hạ tầng vận hành cần được đầu tư bài bản hơn. Hệ thống nên hướng tới khả năng hoạt động ổn định khi tải tăng cao, hạn chế gián đoạn dịch vụ và có cơ chế phục hồi khi xảy ra sự cố. Đồng thời, việc bổ sung các thành phần quan sát như log tập trung, theo dõi chỉ số vận hành, cảnh báo lỗi và truy vết luồng xử lý sẽ giúp việc vận hành, kiểm tra và tối ưu hiệu năng trở nên chủ động và hiệu quả hơn.

Về mặt mở rộng, hệ thống có thể phát triển theo hướng hỗ trợ đa chuỗi (multi-chain) để người dùng có thể giao dịch trên nhiều mạng blockchain khác nhau. Khi mở rộng theo hướng này, cần chuẩn hoá cách cấu hình theo từng chain và thống nhất cách xử lý giao dịch, nhằm đảm bảo trải nghiệm sử dụng không bị phân mảnh giữa các môi trường.

Một hướng phát triển có tính ứng dụng cao là tích hợp các cổng thanh toán fiat on-ramp, giúp người dùng nạp tiền bằng phương thức thanh toán truyền thống thay vì phải chuẩn bị sẵn crypto. Điều này giúp giảm rào cản tiếp cận, từ đó mở rộng nhóm người dùng và tăng khả năng triển khai thực tế của sản phẩm.

Bên cạnh đó, marketplace có thể được nâng cấp thêm các cơ chế giao dịch nâng cao, đặc biệt là các hình thức đấu giá. Những cơ chế này sẽ làm tăng tính linh hoạt của nền tảng và giúp hệ thống tiệm cận hơn với các sàn giao dịch lớn.

Ngoài ra, nếu định hướng dài hạn là xây dựng một hệ sinh thái sâu hơn thay vì chỉ dừng ở mua bán NFT, hệ thống có thể mở rộng sang các tính năng DeFi như staking hoặc lending. Khi đó, NFT không chỉ là tài sản để giao dịch mà còn có thể tham gia vào các mô hình tạo lợi nhuận hoặc vay/mượn, qua đó tăng thêm giá trị sử dụng cho người dùng.

MỘT SỐ LƯU Ý VỀ TÀI LIỆU THAM KHẢO

Lưu ý: Sinh viên không được đưa bài giảng/slide, các trang Wikipedia, hoặc các trang web thông thường làm tài liệu tham khảo.

Một trang web được phép dùng làm tài liệu tham khảo **chỉ khi** nó là công bố chính thống của cá nhân hoặc tổ chức nào đó. Ví dụ, trang web đặc tả ngôn ngữ XML của tổ chức W3C `https://www.w3.org/TR/2008/REC-xml-20081126/` là TLTK hợp lệ.

Có năm loại tài liệu tham khảo mà sinh viên phải tuân thủ đúng quy định về cách thức liệt kê thông tin như sau. Lưu ý: các phần văn bản trong cặp dấu < > dưới đây chỉ là hướng dẫn khai báo cho từng loại tài liệu tham khảo; sinh viên cần xóa các phần văn bản này trong ĐATN của mình.

<Bài báo đăng trên tạp chí khoa học: Tên tác giả, tên bài báo, tên tạp chí, volume, từ trang đến trang (nếu có), nhà xuất bản, năm xuất bản >

[1] E. H. Hovy, “Automated discourse generation using discourse structure relations," *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993

<Sách: Tên tác giả, tên sách, volume (nếu có), lần tái bản (nếu có), nhà xuất bản, năm xuất bản>

[2] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[3] N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.

<Tập san Báo cáo Hội nghị Khoa học: Tên tác giả, tên báo cáo, tên hội nghị, ngày (nếu có), địa điểm hội nghị, năm xuất bản>

[4] M. Poesio and B. Di Eugenio, “Discourse structure and anaphoric accessibility," in *ESSLLI workshop on information structure, discourse structure and discourse semantics*, Copenhagen, Denmark, 2001, pp. 129–143.

<Đồ án tốt nghiệp, Luận văn Thạc sĩ, Tiến sĩ: Tên tác giả, tên đồ án/luận văn, loại đồ án/luận văn, tên trường, địa điểm, năm xuất bản>

[5] A. Knott, “A data-driven methodology for motivating a set of coherence relations," Ph.D. dissertation, The University of Edinburgh, UK, 1996.

<Tài liệu tham khảo từ Internet: Tên tác giả (nếu có), tựa đề, cơ quan (nếu có), địa chỉ trang web, thời gian lần cuối truy cập trang web>

[6] T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. [Online]. Available:

ftp:/info.cern.ch/pub/www/doc/http-spec.txt.Z (visited on 09/30/2010).

[7] Princeton University, *Wordnet*. [Online]. Available: http://www.cogsci.princeton.edu/~wn/index.shtml (visited on 09/30/2010).

TÀI LIỆU THAM KHẢO

[1] E. H. Hovy, "Automated discourse generation using discourse structure relations," *Artificial intelligence*, vol. 63, no. 1-2, pp. 341–385, 1993.

[2] L. L. Peterson and B. S. Davie, *Computer networks: a systems approach*. Elsevier, 2007.

[3] N. T. Hải, *Mạng máy tính và các hệ thống mở*. Nhà xuất bản giáo dục, 1999.

[4] M. Poesio and B. Di Eugenio, "Discourse structure and anaphoric accessibility," in *ESSLLI workshop on information structure, discourse structure and discourse semantics, Copenhagen, Denmark*, 2001, pp. 129–143.

[5] A. Knott, "A data-driven methodology for motivating a set of coherence relations," Ph.D. dissertation, The University of Edinburgh, UK, 1996.

[6] T. Berners-Lee, *Hypertext transfer protocol (HTTP)*. Accessed: Sep. 30, 2010. [Online]. Available: `ftp:/info.cern.ch/pub/www/doc/http-spec.txt.Z`

[7] Princeton University, *Wordnet*. Accessed: Sep. 30, 2010. [Online]. Available: `http://www.cogsci.princeton.edu/~wn/index.shtml`

PHỤ LỤC