**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI:**

# XÂY DỰNG HỆ THỐNG QUẢN LÝ CHUỖI CUNG ỨNG CHO SẢN PHẨM NÔNG NGHIỆP SẠCH DỰA TRÊN CÔNG NGHỆ BLOCKCHAIN

Người hướng dẫn: **THS. NGUYỄN CÔNG DANH**
Sinh viên thực hiện: **CHÂU CHÍ HIẾU**
Số thẻ sinh viên: **102210056**
Lớp**: 21TCLC_DT1**

**Đà Nẵng, 06/2026**

**ĐẠI HỌC ĐÀ NẴNG**
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
**KHOA CÔNG NGHỆ THÔNG TIN**

# ĐỒ ÁN TỐT NGHIỆP

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

**CHUYÊN NGÀNH: CÔNG NGHỆ PHẦN MỀM**

**ĐỀ TÀI:**

# XÂY DỰNG HỆ THỐNG QUẢN LÝ CHUỖI CUNG ỨNG CHO SẢN PHẨM NÔNG NGHIỆP SẠCH DỰA TRÊN CÔNG NGHỆ BLOCKCHAIN

Người hướng dẫn: **THS. NGUYỄN CÔNG DANH**
Sinh viên thực hiện: **CHÂU CHÍ HIẾU**
Số thẻ sinh viên: **102210056**
Lớp**: 21TCLC_DT1**

**Đà Nẵng, 06/2026**

# TÓM TẮT ĐỒ ÁN

Tên đề tài: **Xây dựng hệ thống quản lý chuỗi cung ứng cho sản phẩm nông nghiệp sạch dựa trên công nghệ blockchain.**

Sinh viên thực hiện: **Châu Chí Hiếu**

Số thẻ SV: **102210056** Lớp: **21TCLC_DT1**

Đồ án xây dựng AgriBid - hệ thống quản lý chuỗi cung ứng cho sản phẩm nông nghiệp sạch dựa trên công nghệ blockchain, hướng tới giải quyết bài toán minh bạch nguồn gốc và đa dạng hóa hình thức giao dịch (mua trực tiếp, đấu giá tăng dần, livestream bán hàng) cho nông sản theo lô hàng.

Về mặt kỹ thuật, hệ thống gồm bốn thành phần: (1) Backend xây dựng bằng NestJS/TypeORM/PostgreSQL/Redis/Bull/Socket.IO với 27 module nghiệp vụ và hơn 100 endpoint REST được tài liệu hóa bằng Swagger; (2) Frontend Client (Next.js 16/React 19) phục vụ Merchant và Farmer; (3) Frontend Admin (Next.js 14/MUI) phục vụ quản trị viên; (4) hai smart contract Ethereum (AgriBidEscrow.sol, AgriBidNFT.sol) xây dựng bằng Hardhat/Solidity/OpenZeppelin, triển khai trên Sepolia testnet, mint NFT (ERC - 721) làm chứng chỉ truy xuất nguồn gốc cho từng lô hàng đã được duyệt.

Hệ thống tích hợp thanh toán đa kênh (ví nội bộ, PayOS, VietQR), livestream qua LiveKit/WebRTC, và áp dụng các cơ chế bảo đảm an toàn giao dịch tài chính như database transaction kèm pessimistic locking và idempotency - key. Kết quả kiểm thử cho thấy 84/84 unit test Backend PASS và 20/32 test case smart contract PASS, cùng kiểm thử thủ công theo 8 use case chính xác nhận các luồng nghiệp vụ trọng yếu hoạt động đúng thiết kế.

*Từ khóa: chuỗi cung ứng, đấu giá nông sản, truy xuất nguồn gốc, NFT, blockchain Ethereum, NestJS, Next.js, PayOS, VietQR.*

ĐẠI HỌC ĐÀ NẴNG
**TRƯỜNG ĐẠI HỌC BÁCH KHOA**
KHOA CÔNG NGHỆ THÔNG TIN

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**
Độc lập - Tự do - Hạnh phúc

# NHIỆM VỤ ĐỒ ÁN TỐT NGHIỆP

Họ tên sinh viên: Châu Chí Hiếu Số thẻ sinh viên: 102210056
Lớp: 21TCLC_DT1 Ngành: Công nghệ thông tin Khoa: Công nghệ thông tin
*1. Tên đề tài đồ án:* Xây dựng hệ thống quản lý chuỗi cung ứng cho sản phẩm nông nghiệp sạch dựa trên công nghệ blockchain

*2. Đề tài thuộc diện: Có ký kết thỏa thuận sở hữu trí tuệ đối với kết quả thực hiện.*

*3. Các số liệu, dữ liệu ban đầu:*

Khảo sát hiện trạng các sàn thương mại điện tử và đấu giá trực tuyến (Shopee, Lazada, eBay, OpenSea); yêu cầu về quản lý lô hàng nông sản, đấu giá tăng dần, livestream bán hàng, ví điện tử nội bộ và tích hợp blockchain Ethereum để truy xuất nguồn gốc.

*4. Nội dung các phần thuyết minh và tính toán:*

- Chương 1: Tổng quan về đề tài.
- Chương 2: Phân tích và thiết kế hệ thống.
- Chương 3: Triển khai hệ thống và đánh giá kết quả.
- Kết luận, Tài liệu tham khảo và Phụ lục.

*5. Các bản vẽ, đồ thị:*

Sơ đồ Use case tổng quát, sơ đồ kiến trúc hệ thống, sơ đồ thực thể - quan hệ (ERD) rút gọn, sơ đồ tuần tự cho các use case chính (UC - 01, UC - 02).

6. Họ tên người hướng dẫn: ThS Nguyễn Công Danh

7. Ngày giao nhiệm vụ đồ án: ..... / ..... / 2026

8. Ngày hoàn thành đồ án: ..... / ..... / 2026

Đà Nẵng, ngày ..... tháng ..... năm 2026

| **Trưởng Bộ môn** | **Người hướng dẫn** |
|---|---|
| *(Ký và ghi rõ họ tên)* | *(Ký và ghi rõ họ tên)* |

# LỜI NÓI ĐẦU VÀ CẢM ƠN

Em xin chân thành cảm ơn quý Thầy Cô khoa Công nghệ thông tin, Trường Đại học Bách khoa - Đại học Đà Nẵng đã tận tình giảng dạy và truyền đạt kiến thức trong suốt quá trình học tập, tạo nền tảng để em hoàn thành đồ án tốt nghiệp này.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc đến Thầy Nguyễn Công Danh - giáo viên hướng dẫn, đã trực tiếp định hướng, góp ý và hỗ trợ em trong suốt quá trình nghiên cứu, phân tích, thiết kế và xây dựng hệ thống AgriBid.

Em cũng xin cảm ơn gia đình, bạn bè đã luôn đồng hành, động viên và tạo điều kiện thuận lợi để em tập trung hoàn thành đồ án.

Do thời gian và kinh nghiệm thực tế còn hạn chế, đồ án không tránh khỏi những thiếu sót. Em rất mong nhận được sự góp ý của quý Thầy Cô để đồ án được hoàn thiện hơn.

Em xin chân thành cảm ơn!

# LỜI CAM ĐOAN

Em xin cam đoan đồ án tốt nghiệp "Xây dựng hệ thống quản lý chuỗi cung ứng cho sản phẩm nông nghiệp sạch dựa trên công nghệ blockchain" là công trình nghiên cứu và xây dựng của riêng em, dưới sự hướng dẫn của Thầy Nguyễn Công Danh.

Các số liệu, kết quả khảo sát mã nguồn (số module, số endpoint, số trang giao diện, số test case...) được trình bày trong đồ án là kết quả khảo sát trực tiếp trên hệ thống do em xây dựng, không sao chép từ bất kỳ công trình nào khác. Các tài liệu tham khảo được trích dẫn rõ nguồn gốc tại mục Tài liệu tham khảo.

Em xin chịu hoàn toàn trách nhiệm về tính trung thực của nội dung đồ án.

Đà Nẵng, ngày ..... tháng ..... năm 2026

Sinh viên thực hiện

*(Ký và ghi rõ họ tên)*

**Châu Chí Hiếu**

# MỤC LỤC

**LỜI NÓI ĐẦU VÀ CẢM ƠN....1**
**LỜI CAM ĐOAN....2**
**MỤC LỤC....3**
**DANH MỤC BẢNG BIỂU....6**
**DANH MỤC HÌNH VẼ....8**
**DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT....9**
**MỞ ĐẦU....1**
**1. Lý do chọn đề tài....1**
**2. Mục tiêu đề tài....2**
**3. Đối tượng và phạm vi nghiên cứu....2**
***3.1. Đối tượng nghiên cứu....2***
***3.2. Phạm vi đề tài....3***
**4. Phương pháp nghiên cứu....4**
**5. Cấu trúc đồ án....4**
**Chương 1: TỔNG QUAN VỀ ĐỀ TÀI....5**
**1.1. Giới thiệu chung về hệ thống AgriBid....5**
**1.2. Khảo sát hiện trạng và các hệ thống liên quan....5**
***1.2.1. Mô hình tiêu thụ nông sản truyền thống....5***
***1.2.2. Các nền tảng thương mại điện tử và đấu giá hiện có....5***
***1.2.3. Bảng so sánh AgriBid với các nền tảng tương tự....6***
***1.2.4. Nhận xét và định hướng của AgriBid....8***
**1.3. Cơ sở lý thuyết và công nghệ sử dụng....8**
***1.3.1. Kiến trúc Client - Server và RESTful API....8***
***1.3.2. NestJS, TypeORM và PostgreSQL....8***
***1.3.3. Redis, Bull Queue và Socket.IO....9***
***1.3.4. Next.js, React và hệ sinh thái Frontend....9***
***1.3.5. LiveKit và WebRTC cho livestream....9***
***1.3.6. Blockchain Ethereum, Solidity và OpenZeppelin....10***
***1.3.7. Mô hình tin cậy và giới hạn của blockchain trong truy xuất nguồn gốc.. 10***
***1.3.8. Ví điện tử Web3 và đồng bộ on - chain/off - chain....11***
***1.3.9. Cổng thanh toán PayOS và VietQR....11***
***1.3.10. Bảo mật và xác thực....12***
***1.3.11. Công cụ và quy trình phát triển....12***
**Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG....16**
**2.1. Phân tích yêu cầu hệ thống....16**
***2.1.1. Tác nhân của hệ thống....16***
***2.1.2. Yêu cầu chức năng theo nhóm nghiệp vụ....16***

***2.1.3. Yêu cầu phi chức năng.....18***
**2.2. Đặc tả Use Case theo tác nhân.....19**
***2.2.1. Use case của FARMER.....19***
***2.2.2. Use case của MERCHANT.....20***
***2.2.3. Use case của ADMIN.....21***
***2.2.4. Đặc tả chi tiết một số use case quan trọng.....22***
**2.3. Kiến trúc tổng thể hệ thống.....33**
***2.3.1. Kiến trúc nội bộ Backend.....34***
***2.3.2. Sơ đồ luồng tích hợp.....34***
**2.4. Thiết kế cơ sở dữ liệu.....35**
***2.4.1. Mô hình quan hệ tổng quát (ERD rút gọn).....35***
***2.4.2. Mô tả chi tiết các bảng dữ liệu chính.....37***
***2.4.3. Các bảng dữ liệu bổ trợ.....42***
**2.5. Thiết kế các luồng nghiệp vụ trọng yếu.....47**
***2.5.1. Luồng đấu giá English Auction (VND).....47***
***2.5.2. Luồng đấu giá Blockchain (ETH).....49***
***2.5.3. Luồng ví & giao dịch (Wallet/Transaction).....50***
***2.5.4. Luồng thanh toán đơn hàng.....50***
***2.5.5. Luồng livestream bán hàng.....51***
***2.5.6. Cơ chế Idempotency Key.....51***
***2.5.7. Cơ chế đồng bộ Blockchain (BlockchainListener).....52***
***2.5.8. Sơ đồ tuần tự một số luồng tiêu biểu.....52***
**2.6. Thiết kế giao diện tổng quan.....53**
***2.6.1. Nhóm màn hình Marketplace & chi tiết sản phẩm (Client - Merchant)..64***
***2.6.2. Nhóm màn hình Đấu giá (Auction).....65***
***2.6.3. Nhóm màn hình Giỏ hàng & Thanh toán (Cart & Checkout).....65***
***2.6.4. Nhóm màn hình quản lý dành cho Farmer.....66***
***2.6.5. Nhóm màn hình Quản trị (Admin Dashboard).....66***
***2.6.6. Sơ đồ điều hướng (sitemap) chi tiết.....67***
**2.7. Thiết kế API hệ thống.....68**
***2.7.1. Xác thực & hồ sơ cá nhân (auth, users/media).....68***
***2.7.2. Quản lý người dùng & KYC (Admin).....69***
***2.7.3. Danh mục & sản phẩm tham chiếu (categories, products).....70***
**2.7.4. Địa chỉ giao hàng (addresses).....70**
***2.7.5. Lô hàng, media & truy xuất nguồn gốc (product - batches, batch - media, verify, metadata).....71***
***2.7.6. Listing, đánh giá & giỏ hàng (listings, listing - reviews, carts).....73***
***2.7.7. Đơn hàng (orders).....75***

*2.7.8. Đấu giá & đặt giá (auctions, bids)..... 77*
*2.7.9. Livestream (live - sessions)..... 79*
*2.7.10. Thanh toán & Ví (payments, wallets)..... 81*
*2.7.11. Payout, Voucher, Phí nền tảng & Giao dịch..... 82*
*2.7.12. Quản trị, giám sát hệ thống & Blockchain..... 84*
**Chương 3: TRIỂN KHAI HỆ THỐNG VÀ ĐÁNH GIÁ KẾT QUẢ..... 85**
**3.1. Môi trường và công nghệ cài đặt..... 85**
**3.2. Cấu trúc mã nguồn..... 86**
**3.3. Cấu hình môi trường (Environment Configuration)..... 87**
**3.4. Cài đặt môi trường phát triển..... 91**
**3.5. Tích hợp CI (Continuous Integration)..... 91**
**3.6. Một số đoạn mã cài đặt tiêu biểu..... 92**
*3.6.1. Lớp bảo vệ và phân quyền (Guard)..... 92*
*3.6.2. Chuẩn hóa response (Interceptor)..... 92*
*3.6.3. Cơ chế idempotency cho giao dịch tài chính..... 92*
*3.6.4. Đồng bộ trạng thái blockchain (BlockchainListener)..... 93*
**3.7. Kiểm thử hệ thống..... 93**
*3.7.1. Chiến lược kiểm thử..... 93*
*3.7.2. Kết quả kiểm thử Backend (Jest)..... 93*
*3.7.3. Kết quả kiểm thử Smart Contract (Hardhat)..... 95*
*3.7.4. Kết quả kiểm thử Frontend..... 99*
*3.7.5. Kiểm thử theo use case..... 100*
*3.7.6. Nhận xét, đánh giá..... 102*
**3.8. Đánh giá kết quả và so sánh hệ thống..... 102**
*3.8.1. Mục đích và phương pháp đánh giá..... 102*
*3.8.2. Kết quả đánh giá hiệu năng..... 103*
*3.8.3. Chi phí giao dịch trên smart contract..... 104*
*3.8.4. So sánh AgriBid với các nền tảng tham chiếu..... 105*
*3.8.5. Nhận xét chung..... 106*
**KẾT LUẬN..... 107**
**1. Kết quả đạt được..... 107**
**2. Hạn chế..... 110**
**3. Hướng phát triển..... 110**
**TÀI LIỆU THAM KHẢO..... 111**
**PHỤ LỤC..... 113**
**1. Bảng từ viết tắt và thuật ngữ..... 113**
**2. Hướng dẫn cài đặt và chạy hệ thống..... 115**
*2.1. Cài đặt và chạy Backend, Frontend..... 115*
*2.2. Triển khai smart contract (Sepolia testnet)..... 116*

***2.3. Kiểm tra hệ thống bằng smoke test (demo checklist)..... 117***
**3. Link Github..... 118**
***3.1. Backend..... 118***
***3.2. Frontend Client..... 118***
***3.3. Frontend Admin..... 118***
***3.4. Blockchain..... 118***

# DANH MỤC BẢNG BIỂU


| Số hiệu | Tên bảng |
|---|---|
| Bảng 1.1 | So sánh AgriBid với các nền tảng tham chiếu |
| Bảng 1.2 | Tổng hợp công nghệ sử dụng theo từng thành phần |
| Bảng 2.1 | Danh sách tác nhân của hệ thống |
| Bảng 2.2 | Yêu cầu chức năng theo nhóm nghiệp vụ |
| Bảng 2.3 | Use case của nông dân |
| Bảng 2.4 | Use case của người mua |
| Bảng 2.5 | Use case của quản trị viên |
| Bảng 2.6 | Đặc tả use case UC-01: Đặt giá trong phiên đấu giá (VND) |
| Bảng 2.7 | Đặc tả use case UC-02: Tạo phiên đấu giá |
| Bảng 2.8 | Đặc tả use case UC-03: Checkout giỏ hàng đa người bán |
| Bảng 2.9 | Đặc tả use case UC-04: Duyệt KYC nông dân |
| Bảng 2.10 | Đặc tả use case UC-05: Duyệt lô hàng và mint NFT truy xuất nguồn gốc |
| Bảng 2.11 | Đặc tả use case UC-06: Phát livestream bán hàng và xử lý heartbeat |
| Bảng 2.12 | Đặc tả use case UC-07: Nạp tiền và yêu cầu rút tiền ví VND |
| Bảng 2.13 | Đặc tả use case UC-08: Áp dụng mã giảm giá khi checkout |
| Bảng 2.14 | Các cơ chế tích hợp giữa các thành phần hệ thống |
| Bảng 2.15 | Bảng dữ liệu users |
| Bảng 2.16 | Bảng dữ liệu wallets và transactions |
| Bảng 2.17 | Bảng dữ liệu product_batches |
| Bảng 2.18 | Bảng dữ liệu listings và auctions |
| Bảng 2.19 | Bảng dữ liệu bids |
| Bảng 2.20 | Bảng dữ liệu orders và order_items |
| Bảng 2.21 | Bảng dữ liệu live_sessions và các bảng vận hành khác |
| Bảng 2.22 | Bảng dữ liệu categories và products |
| Bảng 2.23 | Bảng dữ liệu addresses |
| Bảng 2.24 | Bảng dữ liệu batch_media |
| Bảng 2.25 | Bảng dữ liệu carts và cart_items |
| Bảng 2.26 | Bảng dữ liệu vouchers và voucher_redemptions |
| Bảng 2.27 | Bảng dữ liệu listing_reviews |
| Bảng 2.28 | Bảng dữ liệu platform_fee_statements và system_settings |

| Số hiệu | Tên bảng |
|---|---|
| Bảng 2.29 | Nhóm màn hình chính của hệ thống |
| Bảng 2.30 | Sitemap Frontend Client (30 trang) và Frontend Admin (17 trang) |
| Bảng 2.31 | API xác thực và hồ sơ cá nhân |
| Bảng 2.32 | API quản lý người dùng và KYC |
| Bảng 2.33 | API danh mục và sản phẩm tham chiếu |
| Bảng 2.34 | API địa chỉ giao hàng |
| Bảng 2.35 | API lô hàng, media và truy xuất nguồn gốc |
| Bảng 2.36 | API listing, đánh giá và giỏ hàng |
| Bảng 2.37 | API đơn hàng |
| Bảng 2.38 | API đấu giá và đặt giá |
| Bảng 2.39 | API livestream |
| Bảng 2.40 | API thanh toán và ví |
| Bảng 2.41 | API payout, voucher, phí nền tảng và giao dịch |
| Bảng 2.42 | API quản trị, giám sát hệ thống và blockchain |
| Bảng 3.1 | Môi trường và công nghệ cài đặt |
| Bảng 3.2 | Cấu trúc mã nguồn |
| Bảng 3.3 | Biến môi trường (.env) theo từng package |
| Bảng 3.4 | Kết quả kiểm thử Backend theo module/service |
| Bảng 3.5 | Phân nhóm 32 test case smart contract theo file/describe block |
| Bảng 3.6 | Phân tích nhóm test case smart contract chưa PASS |
| Bảng 3.7 | Kết quả kiểm thử theo các use case chính |
| Bảng 3.8 | Thời gian phản hồi trung bình của một số API tiêu biểu (môi trường development) |
| Bảng 3.9 | Ước lượng chi phí gas của các giao dịch chính trên smart contract (Sepolia testnet) |
| Bảng 3.10 | So sánh AgriBid với các nền tảng tham chiếu theo tiêu chí chức năng |
| Bảng K.1 | Đối chiếu mục tiêu đề tài và kết quả đạt được |
| Bảng PL.1 | Bảng từ viết tắt và thuật ngữ |

# DANH MỤC HÌNH VẼ


| Số hiệu | Tên hình |
|---|---|
| Hình 2.1 | Biểu đồ Use case tổng quát hệ thống AgriBid |
| Hình 2.2 | Sơ đồ kiến trúc tổng thể hệ thống AgriBid |
| Hình 2.3 | Sơ đồ thực thể - quan hệ (ERD) rút gọn của hệ thống AgriBid |
| Hình 2.4 | Sơ đồ tuần tự UC-01: Đặt giá trong phiên đấu giá (VND) |
| Hình 2.5 | Sơ đồ tuần tự UC-02: Tạo phiên đấu giá (English Auction) |
| Hình 2.6 | Đăng nhập/Đăng ký (điểm bắt đầu mọi luồng) |
| Hình 2.7 | Marketplace & chi tiết sản phẩm/lô hàng (khám phá sản phẩm) |
| Hình 2.8 | Đấu giá realtime & Livestream (tương tác mua đặc thù) |
| Hình 2.9 | Giỏ hàng & Thanh toán (checkout, PayOS) |
| Hình 2.10 | Xác nhận chuyển khoản (upload + duyệt biên lai CK) |
| Hình 2.11 | Truy xuất nguồn gốc & Blockchain/NFT (sau khi mua, xem nguồn gốc sản phẩm) |
| Hình 2.12 | Hồ sơ & lịch sử ví (wallet, giao dịch on-chain) |
| Hình 2.13 | Phân chia doanh thu/% (góc nhìn farmer nhận tiền) |
| Hình 2.14 | Quản lý dành cho Farmer (lô hàng, listing, đơn bán, rút tiền) |
| Hình 2.15 | Admin Dashboard (góc nhìn quản trị, cuối luồng) |

# DANH MỤC CÁC KÝ HIỆU, CHỮ VIẾT TẮT

## KÝ HIỆU:

| Ký hiệu | Giải thích |
|---|---|
| ETH | Ether - đơn vị tiền điện tử của mạng Ethereum, dùng để thanh toán/đặt giá trong các smart contract của hệ thống. |
| VND | Đồng Việt Nam - đơn vị tiền pháp định dùng cho thanh toán qua ví nội bộ, PayOS, VietQR. |

## CHỮ VIẾT TẮT:

| Từ viết tắt / thuật ngữ | Giải thích |
|---|---|
| KYC | Know Your Customer - quy trình xác minh danh tính người dùng (Farmer/Merchant) trước khi cho phép bán hàng trên hệ thống. |
| NFT | Non - Fungible Token - token không thể thay thế, dùng làm chứng chỉ truy xuất nguồn gốc duy nhất cho từng lô hàng nông sản. |
| ERC - 721 | Chuẩn token không thể thay thế (NFT) trên Ethereum, được AgriBidNFT.sol triển khai để mint chứng chỉ lô hàng. |
| JWT | JSON Web Token - cơ chế xác thực dựa trên token, dùng cho access token/refresh token trong NestJS (Passport JWT strategy). |
| RBAC | Role - Based Access Control - kiểm soát truy cập theo vai trò (FARMER, MERCHANT, ADMIN), cài đặt qua RolesGuard. |
| REST API | Representational State Transfer - kiến trúc giao tiếp client - server qua HTTP theo tài nguyên (resource). |
| ABI | Application Binary Interface - định nghĩa giao diện gọi hàm của smart contract, dùng bởi ethers.js. |
| RPC | Remote Procedure Call - giao thức gửi yêu cầu tới node Ethereum (qua BLOCKCHAIN_RPC_URL). |

| Từ viết tắt / thuật ngữ | Giải thích |
|---|---|
| ACID | Atomicity, Consistency, Isolation, Durability - các tính chất của transaction cơ sở dữ liệu, đảm bảo toàn vẹn cho nghiệp vụ tài chính. |
| KYC/Idempotency/ERD... | Các thuật ngữ chuyên ngành khác được giải thích chi tiết tại Phụ lục PL.1 - Bảng từ viết tắt và thuật ngữ. |

# MỞ ĐẦU

## 1. Lý do chọn đề tài

Nông nghiệp giữ vai trò quan trọng trong nền kinh tế Việt Nam, tuy nhiên phần lớn nông sản hiện nay vẫn được tiêu thụ qua nhiều tầng trung gian (thương lái, chợ đầu mối) trước khi đến tay người mua. Điều này dẫn đến ba vấn đề lớn: (1) người nông dân không chủ động được giá bán và thường chịu thiệt khi giá thị trường biến động; (2) người mua (đặc biệt là các thương lái, doanh nghiệp thu mua số lượng lớn) khó tiếp cận trực tiếp nguồn hàng chất lượng với mức giá cạnh tranh; (3) thông tin về nguồn gốc, quy trình canh tác của nông sản gần như không được lưu trữ minh bạch, gây khó khăn cho việc truy xuất nguồn gốc khi có sự cố về chất lượng [1].

Song song đó, các xu hướng thương mại điện tử mới như bán hàng qua livestream và đấu giá trực tuyến đã chứng minh hiệu quả trong việc rút ngắn khoảng cách giữa người bán và người mua, tăng tính tương tác và minh bạch về giá [2], [3]. Tuy nhiên, tại Việt Nam hiện chưa có nhiều nền tảng kết hợp đồng thời ba hình thức giao dịch (mua bán cố định, đấu giá kiểu English Auction, livestream bán hàng) dành riêng cho lĩnh vực nông sản, cũng như chưa ứng dụng công nghệ blockchain để giải quyết bài toán minh bạch nguồn gốc và thanh toán.

Xuất phát từ thực tế trên, đề tài "Xây dựng hệ thống quản lý chuỗi cung ứng cho sản phẩm nông nghiệp sạch dựa trên công nghệ Blockchain" được lựa chọn nhằm xây dựng một hệ thống thương mại điện tử chuyên biệt cho nông sản, cho phép nông dân bán hàng trực tiếp tới thương lái/người mua thông qua ba kênh: Listing, Auction (đấu giá kiểu English Auction) và Live Session (livestream bán hàng), đồng thời hỗ trợ song song hai phương thức thanh toán: tiền pháp định Việt Nam (VND) qua ví nội bộ/PayOS/VietQR và tiền điện tử Ethereum (ETH) thông qua smart contract, kết hợp NFT để chứng minh nguồn gốc lô hàng. Các nghiên cứu về blockchain trong chuỗi cung ứng nông nghiệp cho thấy công nghệ này có tiềm năng hỗ trợ lưu vết dữ liệu, tăng tính minh bạch và hạn chế chỉnh sửa thông tin sau khi đã được ghi nhận [4]-[7], [9].

Trong hệ thống AgriBid, blockchain và NFT không được sử dụng để thay thế hoàn toàn cơ sở dữ liệu truyền thống, mà đóng vai trò bổ sung một lớp bằng chứng

công khai và khó bị chỉnh sửa cho thông tin truy xuất nguồn gốc của lô hàng [4]-[7], [9]. Tuy nhiên, blockchain không tự động bảo đảm rằng dữ liệu đầu vào là chính xác tuyệt đối; do đó, hệ thống vẫn cần kết hợp với các cơ chế kiểm soát off-chain như KYC Farmer, duyệt lô hàng, media minh chứng và trong tương lai có thể tích hợp thêm dữ liệu IoT hoặc kiểm định độc lập [6], [7].

## 2. Mục tiêu đề tài

Đề tài hướng đến các mục tiêu cụ thể sau:

- Xây dựng hệ thống backend (NestJS, TypeORM, PostgreSQL, Redis/Bull, Socket.IO) cung cấp đầy đủ API cho ba kênh giao dịch (Listing, Auction, Live Session), hệ thống ví điện tử nội bộ (VND), tích hợp cổng thanh toán PayOS và VietQR chuyển khoản trực tiếp.
- Xây dựng giao diện người dùng (Next.js 16/React 19) cho Farmer và Merchant: marketplace, trang chi tiết sản phẩm/lô hàng, trang đấu giá realtime, trang xem livestream, giỏ hàng, thanh toán, portal quản lý của nông dân.
- Xây dựng giao diện quản trị (Next.js 14 + MUI) cho Admin: duyệt KYC, duyệt lô hàng, quản lý danh mục, giám sát đơn hàng/đấu giá, duyệt yêu cầu rút tiền, tính phí nền tảng.
- Thiết kế và triển khai hai smart contract trên Ethereum (Hardhat, Solidity 0.8.28, OpenZeppelin): AgriBidEscrow.sol quản lý vòng đời đấu giá bằng ETH theo mô hình pull - payment, và AgriBidNFT.sol mint NFT (ERC - 721) làm chứng chỉ truy xuất nguồn gốc lô hàng.
- Đảm bảo an toàn giao dịch tài chính thông qua database transaction kèm pessimistic locking, cơ chế idempotency - key cho các API tạo tài nguyên quan trọng, và cơ chế đồng bộ dữ liệu hai chiều giữa backend và blockchain.

## 3. Đối tượng và phạm vi nghiên cứu

### *3.1. Đối tượng nghiên cứu*

- Quy trình nghiệp vụ mua bán nông sản theo lô hàng, bao gồm quy trình duyệt chất lượng, niêm yết giá, đấu giá tăng dần và bán hàng qua livestream.
- Mô hình ví điện tử nội bộ và các nghiệp vụ liên quan: nạp tiền, giữ tiền (hold/freeze), giải phóng tiền, thanh toán, rút tiền về ngân hàng.

- Công nghệ xây dựng ứng dụng web hiện đại theo kiến trúc client - server: NestJS, Next.js, PostgreSQL, Redis, Socket.IO, LiveKit (WebRTC).
- Công nghệ blockchain Ethereum: smart contract Solidity, thư viện OpenZeppelin, ethers.js, ví MetaMask, mô hình pull - payment và chuẩn token ERC - 721.

### *3.2. Phạm vi đề tài*

Phạm vi của đồ án tập trung vào việc xây dựng hệ thống hỗ trợ quản lý một số mắt xích cốt lõi trong chuỗi cung ứng nông sản sạch, bao gồm: quản lý lô hàng nông sản, xác minh và duyệt thông tin lô hàng, truy xuất nguồn gốc bằng NFT, tổ chức giao dịch giữa nông dân và người mua, hỗ trợ đấu giá, livestream bán hàng, thanh toán và quản lý ví nội bộ.

Cụ thể, đồ án bao gồm bốn thành phần chính: (1) Backend NestJS với 27 module nghiệp vụ như xác thực, người dùng, lô hàng, listing, auction, bid, giỏ hàng, đơn hàng, thanh toán, ví, giao dịch, payout, phí nền tảng, voucher, đánh giá, livestream, blockchain, dashboard quản trị, cấu hình hệ thống và giám sát hệ thống; (2) Frontend Client Next.js phục vụ Merchant và portal Farmer với các nhóm chức năng marketplace, chi tiết lô hàng, đấu giá, livestream, giỏ hàng, thanh toán và quản lý bán hàng; (3) Frontend Admin Next.js + MUI phục vụ vai trò quản trị, bao gồm duyệt KYC, duyệt lô hàng, quản lý danh mục, giám sát đơn hàng/đấu giá và xử lý payout; (4) hai smart contract AgriBidEscrow và AgriBidNFT triển khai bằng Hardhat, phục vụ đấu giá bằng ETH và phát hành NFT làm chứng chỉ truy xuất nguồn gốc cho từng lô hàng đã được duyệt.

Trong phạm vi "quản lý chuỗi cung ứng", đồ án không xây dựng đầy đủ một hệ thống quản trị chuỗi cung ứng end-to-end bao gồm quản lý kho, vận chuyển lạnh, logistics, nhà phân phối, điểm bán lẻ và tối ưu hóa tồn kho. Thay vào đó, hệ thống tập trung vào các khâu có ý nghĩa trực tiếp đối với bài toán minh bạch và giao dịch nông sản sạch: quản lý lô hàng từ phía Farmer, xác thực thông tin lô hàng bởi Admin, công khai nguồn gốc bằng blockchain/NFT, kết nối Farmer với Merchant, hỗ trợ mua bán/đấu giá/livestream và ghi nhận thanh toán. Các vấn đề triển khai production như

CI/CD, scaling, bảo mật hạ tầng nâng cao và tích hợp logistics thực tế nằm ngoài phạm vi của đồ án.

## 4. Phương pháp nghiên cứu

Phương pháp nghiên cứu tài liệu: tìm hiểu các nền tảng thương mại điện tử nông sản, đấu giá trực tuyến và các tài liệu kỹ thuật về NestJS, Next.js, blockchain Ethereum, LiveKit, PayOS/VietQR.

Phương pháp phân tích - thiết kế hướng đối tượng: xác định actor, use case, thiết kế cơ sở dữ liệu quan hệ (ERD), thiết kế kiến trúc hệ thống nhiều lớp (layered architecture).

Phương pháp thực nghiệm: xây dựng từng module theo hướng lặp (iterative), kiểm thử API bằng Swagger/Postman, kiểm thử smart contract bằng bộ test Hardhat (Mocha/Chai).

## 5. Cấu trúc đồ án

Ngoài phần Mở đầu, Kết luận, Tài liệu tham khảo và Phụ lục, nội dung chính của đồ án được trình bày trong các chương sau:

Chương 1 - Tổng quan về đề tài: trình bày bối cảnh, mục tiêu, phạm vi, khảo sát hệ thống liên quan và cơ sở lý thuyết/công nghệ sử dụng.

Chương 2 - Phân tích và thiết kế hệ thống: trình bày phân tích yêu cầu, đặc tả use case theo từng tác nhân, kiến trúc hệ thống tổng thể, thiết kế cơ sở dữ liệu và thiết kế các luồng nghiệp vụ trọng yếu.

Chương 3 - Cài đặt và triển khai hệ thống: trình bày môi trường, công nghệ, cấu trúc mã nguồn, cấu hình triển khai và một số đoạn mã cài đặt tiêu biểu.

# Chương 1: TỔNG QUAN VỀ ĐỀ TÀI

## 1.1. Giới thiệu chung về hệ thống AgriBid

AgriBid là một nền tảng thương mại điện tử chuyên biệt cho ngành nông nghiệp, cho phép người nông dân đăng bán nông sản theo từng lô hàng thu hoạch và lựa chọn một trong các kênh giao dịch: bán giá cố định, đấu giá trực tuyến kiểu English Auction, hoặc giới thiệu/bán trực tiếp qua livestream (Live Session). Người mua là các thương lái/doanh nghiệp thu mua có thể duyệt marketplace, đặt mua, tham gia đấu giá hoặc mua trực tiếp trong phiên live. Quản trị viên (ADMIN) chịu trách nhiệm duyệt hồ sơ KYC của nông dân, duyệt chất lượng lô hàng, quản lý danh mục sản phẩm, giám sát giao dịch và vận hành các chính sách phí, voucher, payout.

Điểm khác biệt nổi bật của AgriBid so với các sàn thương mại điện tử nông sản truyền thống là: (1) tích hợp đồng thời ba hình thức giao dịch trên cùng một lô hàng (mỗi lô chỉ thuộc một trong hai kênh Listing hoặc Auction, riêng livestream chỉ áp dụng cho listing đang ACTIVE); (2) hỗ trợ song song hai phương thức thanh toán - ví VND nội bộ/PayOS/VietQR và ví ETH qua smart contract; (3) ứng dụng NFT (ERC - 721) làm "chứng chỉ" gắn với lô hàng để chứng minh nguồn gốc, tạo niềm tin cho người mua [4]-[7], [9].

## 1.2. Khảo sát hiện trạng và các hệ thống liên quan

### *1.2.1. Mô hình tiêu thụ nông sản truyền thống*

Trong mô hình truyền thống, nông sản sau thu hoạch thường được bán cho thương lái tại vườn/ruộng với mức giá do thương lái quyết định, sau đó qua chợ đầu mối rồi mới tới các kênh phân phối (chợ lẻ, siêu thị, doanh nghiệp chế biến). Mỗi tầng trung gian làm tăng chi phí và thời gian, đồng thời người tiêu dùng cuối hầu như không có thông tin về nguồn gốc, quy trình canh tác. Người nông dân cũng không có công cụ để tổ chức đấu giá công khai nhằm bán được giá tốt nhất khi có nhiều bên cùng quan tâm tới một lô hàng chất lượng cao [1], [3].

### *1.2.2. Các nền tảng thương mại điện tử và đấu giá hiện có*

Các sàn thương mại điện tử phổ biến (Shopee, Lazada, Chợ Tốt...) tập trung vào mô hình mua ngay với giá niêm yết cố định, có hỗ trợ livestream bán hàng nhưng không

có cơ chế đấu giá tăng dần và không chuyên biệt cho nông sản theo lô (batch) gắn với mùa vụ. Các nền tảng đấu giá quốc tế (eBay) có cơ chế đấu giá nhưng không tích hợp thanh toán bằng tiền điện tử lẫn tiền pháp định trên cùng một hệ thống. Các nền tảng NFT marketplace (OpenSea) ứng dụng tốt smart contract và ví điện tử nhưng không có nghiệp vụ logistics, ví VND, KYC như một sàn thương mại điện tử nông sản cần có.

### *1.2.3. Bảng so sánh AgriBid với các nền tảng tương tự*

Để làm rõ hơn khoảng trống thị trường mà đề tài hướng tới, Bảng 1.1 so sánh AgriBid với ba nhóm nền tảng tham chiếu theo các tiêu chí: mô hình giao dịch, đối tượng phục vụ, phương thức thanh toán và khả năng truy xuất nguồn gốc. Các thông tin trong bảng được tổng hợp từ khảo sát chức năng công khai của các nền tảng tham chiếu tại thời điểm thực hiện đồ án và đặc tả chức năng của hệ thống AgriBid.

Bảng 1.1 - So sánh AgriBid với các nền tảng tham chiếu

| **Tiêu chí** | **Shopee/Lazada/TikTok Shop** | **eBay (đấu giá)** | **OpenSea (NFT)** | **AgriBid** |
|---|---|---|---|---|
| Mua bán giá cố định (Listing) | Có | Có (Buy It Now) | Có (mua trực tiếp NFT) | Có |
| Đấu giá kiểu English Auction | Không | Có | Có (dạng auction NFT) | Có, kèm đặt cọc & đóng băng số dư |
| Livestream bán hàng | Có (chuyên dụng) | Không | Không | Có, gắn với listing, ghim sản phẩm |
| Thanh toán tiền pháp định (VND) | Có | Có (theo khu vực) | Không | Có (ví nội bộ, PayOS, VietQR) |

| Tiêu chí | Shopee/Lazada/TikTok Shop | eBay (đấu giá) | OpenSea (NFT) | AgriBid |
|---|---|---|---|---|
| Thanh toán tiền điện tử (ETH) | Không | Không | Có | Có (smart contract Escrow) |
| Truy xuất nguồn gốc qua NFT | Không | Không | Có (cho tài sản số) | Có (NFT gắn với lô nông sản thực) |
| Chuyên biệt cho nông sản theo lô/mùa vụ | Không | Không | Không | Có |
| Cơ chế KYC nông dân | Có (định danh người bán nói chung) | Có | Không bắt buộc | Có, là điều kiện để bán hàng |
| Cập nhật giá/đặt giá realtime qua WebSocket | Một phần (thông báo) | Có (cập nhật trang) | Có (theo block) | Có (Socket.IO, đẩy ngay khi có bid mới) |
| Đóng băng số dư khi tham gia đấu giá | Không áp dụng | Không công khai cơ chế | Không (ký quỹ qua ví on - chain) | Có (frozenBalance quản lý ở tầng backend) |
| Idempotency cho API tạo phiên/giao dịch quan trọng | Không công khai | Không công khai | Tùy ứng dụng ví/sàn | Có (x - idempotency - key + băm SHA - 256 request) |

Bảng so sánh cho thấy: chưa có nền tảng nào hiện có cùng lúc đáp ứng đủ các tiêu chí trên cho riêng ngành nông sản. Đây chính là cơ sở để xác định các yêu cầu chức năng

cốt lõi cho AgriBid ở Chương 2, đồng thời khẳng định giá trị khác biệt (USP - Unique Selling Point) của đề tài: kết hợp ba kênh giao dịch, hai phương thức thanh toán và cơ chế truy xuất nguồn gốc bằng NFT trong cùng một hệ thống chuyên biệt cho nông sản.

#### *1.2.4. Nhận xét và định hướng của AgriBid*

Từ khảo sát trên, AgriBid được định hướng xây dựng như một nền tảng "lai" (hybrid): kế thừa trải nghiệm mua sắm/livestream của các sàn TMĐT phổ biến, bổ sung cơ chế đấu giá English Auction có quản lý đặt cọc và đóng băng số dư ví như các sàn đấu giá chuyên nghiệp, đồng thời ứng dụng smart contract Ethereum và NFT để giải quyết hai bài toán mà các nền tảng truyền thống chưa làm tốt: thanh toán/đấu giá phi tập trung và truy xuất nguồn gốc nông sản minh bạch, không thể chỉnh sửa [4]-[7], [9].

Định hướng thiết kế cũng tuân theo nguyên tắc "tách trách nhiệm theo nguồn dữ liệu tin cậy" (data authority separation): với các giao dịch VND, PostgreSQL là nguồn dữ liệu chính và mọi ràng buộc nghiệp vụ (số dư, đặt cọc, hoàn tiền) được kiểm soát chặt chẽ ở tầng backend bằng transaction và khóa hàng; với các giao dịch ETH, smart contract đóng vai trò "sổ cái" không thể thay đổi, backend chỉ đọc và phản chiếu trạng thái. Cách tiếp cận này giúp hệ thống vừa giữ được tốc độ xử lý và chi phí thấp của một hệ CSDL quan hệ truyền thống cho phần lớn giao dịch, vừa tận dụng được tính minh bạch, không thể giả mạo của blockchain cho các giao dịch có giá trị cao hoặc cần bằng chứng công khai [4], [7].

### 1.3. Cơ sở lý thuyết và công nghệ sử dụng

#### *1.3.1. Kiến trúc Client - Server và RESTful API*

Hệ thống được xây dựng theo kiến trúc client - server nhiều lớp: tầng giao diện (Frontend Client/Admin) giao tiếp với tầng dịch vụ (Backend) thông qua RESTful API (JSON) và kênh WebSocket cho dữ liệu thời gian thực (đấu giá, chat, chỉ số live). Backend đóng vai trò xử lý nghiệp vụ, xác thực, kiểm soát truy cập và là nguồn dữ liệu tin cậy (source of truth) cho phần lớn nghiệp vụ VND; với nghiệp vụ đấu giá ETH, blockchain Ethereum đóng vai trò nguồn dữ liệu tin cậy, backend chỉ đồng bộ (mirror) trạng thái về cơ sở dữ liệu quan hệ.

#### *1.3.2. NestJS, TypeORM và PostgreSQL*

NestJS (phiên bản 11) là framework Node.js theo kiến trúc module hóa (module - controller - service), hỗ trợ Dependency Injection, decorator và tích hợp tốt với TypeScript, giúp tổ chức 27 module nghiệp vụ (28 controller, tổng cộng khoảng 155 endpoint REST) một cách rõ ràng, dễ mở rộng. TypeORM (^0.3) được dùng làm ORM ánh xạ entity sang bảng PostgreSQL, hỗ trợ migration, quan hệ (OneToMany/ManyToOne/ManyToMany), và đặc biệt là cơ chế khóa hàng (pessimistic lock) thông qua QueryRunner - yếu tố then chốt để đảm bảo tính nhất quán khi nhiều người dùng cùng thao tác trên ví hoặc phiên đấu giá. PostgreSQL được chọn làm hệ quản trị cơ sở dữ liệu chính nhờ tính ổn định, hỗ trợ transaction ACID và kiểu dữ liệu phong phú (numeric, jsonb, enum) [10]-[12].

### *1.3.3. Redis, Bull Queue và Socket.IO*

Redis được sử dụng cho hai mục đích: làm broker cho hàng đợi tác vụ nền Bull (lập lịch bắt đầu/kết thúc phiên đấu giá, tự động hủy đơn hàng quá hạn thanh toán, tự động hoàn tất yêu cầu rút tiền đã duyệt...) và làm adapter cho Socket.IO (@socket.io/redis - adapter) để đồng bộ sự kiện realtime giữa nhiều instance backend. Cơ chế Bull Queue cho phép lập lịch chính xác thời điểm bắt đầu/kết thúc của từng phiên đấu giá (job start - auction, end - auction) độc lập với việc có người dùng thao tác hay không, đồng thời hỗ trợ retry khi job thất bại [13].

### *1.3.4. Next.js, React và hệ sinh thái Frontend*

Frontend Client được xây dựng bằng Next.js 16 (App Router) và React 19, sử dụng TanStack Query để quản lý dữ liệu bất đồng bộ, Zustand cho state cục bộ, Tailwind CSS 4 cho giao diện, và next - intl cho đa ngôn ngữ (Tiếng Việt/Tiếng Anh). Frontend Admin sử dụng Next.js 14 với thư viện Material UI (MUI) theo template Materio, kết hợp TanStack Query và Zod để validate form. Việc tách hai ứng dụng frontend riêng biệt giúp tối ưu trải nghiệm cho từng nhóm người dùng: Client phục vụ Farmer/Merchant, Admin phục vụ đội vận hành [19].

### *1.3.5. LiveKit và WebRTC cho livestream*

Tính năng livestream bán hàng sử dụng LiveKit - hạ tầng WebRTC mã nguồn mở, cho phép nông dân phát trực tiếp từ trình duyệt và người xem tham gia phòng live theo thời gian thực. Backend sử dụng livekit - server - sdk để cấp token truy cập phòng

(room) cho host và viewer. Để đảm bảo không tồn tại phiên live "ma" khi kết nối của Farmer bị gián đoạn, hệ thống áp dụng cơ chế heartbeat: client gửi tín hiệu "còn sống" mỗi 30 giây, backend chạy bộ quét định kỳ (sweep) mỗi 30 giây và tự động chuyển trạng thái phiên sang FORCE_ENDED nếu quá 90 giây không nhận được heartbeat [14].

### *1.3.6. Blockchain Ethereum, Solidity và OpenZeppelin*

Phần đấu giá bằng tiền điện tử và chứng chỉ truy xuất nguồn gốc được xây dựng trên nền tảng Ethereum bằng ngôn ngữ Solidity (phiên bản 0.8.28), sử dụng các thư viện chuẩn của OpenZeppelin (AccessControl, Pausable, ReentrancyGuard, ERC - 721) để đảm bảo an toàn và tuân thủ chuẩn. Hai smart contract chính là AgriBidEscrow.sol (quản lý vòng đời phiên đấu giá ETH theo mô hình pull - payment: người dùng tự gọi lệnh rút tiền hoàn/khoản thắng thay vì hệ thống tự đẩy tiền, giúp hạn chế lỗ hổng reentrancy) và AgriBidNFT.sol (mint NFT chuẩn ERC - 721 làm chứng chỉ cho lô hàng đã được duyệt). Môi trường phát triển, biên dịch, kiểm thử (Mocha/Chai) và triển khai hợp đồng sử dụng Hardhat 3, thư viện ethers.js v6 dùng để giao tiếp giữa backend/frontend với mạng Ethereum [9], [15]-[17].

### *1.3.7. Mô hình tin cậy và giới hạn của blockchain trong truy xuất nguồn gốc*

Trong hệ thống AgriBid, blockchain và NFT không được sử dụng để thay thế hoàn toàn cơ sở dữ liệu truyền thống, mà đóng vai trò bổ sung một lớp bằng chứng công khai và khó bị chỉnh sửa cho thông tin truy xuất nguồn gốc của lô hàng. Nếu chỉ sử dụng QR code kết hợp với cơ sở dữ liệu tập trung, người dùng có thể truy cập thông tin lô hàng một cách thuận tiện, tuy nhiên toàn bộ dữ liệu vẫn phụ thuộc vào máy chủ của hệ thống. Trong trường hợp cơ sở dữ liệu bị can thiệp, chỉnh sửa hoặc mất tính toàn vẹn, người mua khó có cơ sở độc lập để kiểm chứng thông tin đã được công bố trước đó [4]-[7].

Với cách tiếp cận dựa trên blockchain, mỗi lô hàng sau khi được Admin phê duyệt sẽ được gắn với một NFT chuẩn ERC-721. NFT này đóng vai trò như một chứng chỉ số duy nhất đại diện cho lô hàng, chứa hoặc liên kết tới metadata mô tả các thông tin quan trọng như mã lô hàng, sản phẩm, Farmer, thời gian thu hoạch, quy trình

canh tác và các minh chứng liên quan. Sau khi NFT được mint và ghi nhận trên blockchain, thông tin định danh của chứng chỉ truy xuất nguồn gốc có thể được kiểm tra công khai, hạn chế khả năng sửa đổi hoặc thay thế dữ liệu sau khi lô hàng đã được phê duyệt [4]-[7], [9].

Tuy nhiên, blockchain không tự động bảo đảm rằng dữ liệu đầu vào là chính xác tuyệt đối. Nếu thông tin ban đầu do Farmer cung cấp không trung thực hoặc minh chứng không đầy đủ, việc ghi nhận dữ liệu đó lên blockchain chỉ giúp bảo toàn trạng thái đã được công bố, chứ không chứng minh bản thân dữ liệu là đúng. Vì vậy, mô hình tin cậy của AgriBid được xây dựng theo hướng kết hợp giữa kiểm soát off-chain và bằng chứng on-chain. Ở tầng off-chain, hệ thống áp dụng KYC đối với Farmer, quy trình Admin duyệt lô hàng, quản lý hình ảnh/video minh chứng và kiểm soát trạng thái lô hàng trước khi cho phép giao dịch. Ở tầng on-chain, NFT được dùng để tạo bằng chứng truy xuất nguồn gốc khó chỉnh sửa sau thời điểm phê duyệt [5]-[7].

Do đó, blockchain trong đồ án này giải quyết chủ yếu bài toán minh bạch, chống sửa đổi dữ liệu sau khi phê duyệt và tăng khả năng kiểm chứng công khai, chứ không thay thế hoàn toàn quy trình kiểm định chất lượng nông sản ngoài thực tế. Trong hướng phát triển tiếp theo, hệ thống có thể tích hợp thêm dữ liệu từ thiết bị IoT, cảm biến môi trường, đơn vị kiểm định độc lập hoặc cơ quan chứng nhận để nâng cao độ tin cậy của dữ liệu đầu vào trước khi ghi nhận lên blockchain [1], [5], [8].

### *1.3.8. Ví điện tử Web3 và đồng bộ on - chain/off - chain*

Ở phía frontend, thư viện Wagmi và Viem được dùng để kết nối ví MetaMask, ký và gửi giao dịch trực tiếp tới smart contract AgriBidEscrow (đặt giá, rút tiền hoàn, rút tiền thắng cuộc). Ở phía backend, một tiến trình BlockchainListener chạy nền (poll log theo chu kỳ, mặc định 15 giây, theo từng khoảng block và chờ đủ số block xác nhận) để lắng nghe các sự kiện trên smart contract (BidPlaced, AuctionEnded, AuctionCancelled...) và đồng bộ vào cơ sở dữ liệu quan hệ, đảm bảo dữ liệu hiển thị trên giao diện luôn phản ánh đúng trạng thái on - chain mà không cần người dùng tự refresh ví.

### *1.3.9. Cổng thanh toán PayOS và VietQR*

Đối với thanh toán bằng VND, hệ thống tích hợp PayOS - cổng thanh toán trực tuyến hỗ trợ tạo link thanh toán/QR code và xác nhận qua webhook (có kiểm tra chữ ký số) - dùng cho việc nạp tiền vào ví và thanh toán đơn hàng. Bên cạnh đó, hệ thống hỗ trợ hình thức "VietQR chuyển khoản thẳng" (VietQR Level 2): người mua quét mã QR chuyển khoản trực tiếp đến tài khoản ngân hàng của Farmer (không qua ví trung gian), sau đó tải lên biên lai chuyển khoản để Farmer/Admin xác nhận. Đây là giải pháp phù hợp với thực tế nhiều nông dân muốn nhận tiền trực tiếp vào tài khoản ngân hàng cá nhân [18].

### *1.3.10. Bảo mật và xác thực*

Hệ thống áp dụng JWT (JSON Web Token) cho xác thực, gồm access token thời hạn ngắn (15 phút) và refresh token thời hạn dài (7 ngày) lưu dưới dạng cookie httpOnly; mật khẩu người dùng được băm bằng bcrypt. Cơ chế RolesGuard kiểm soát truy cập theo vai trò (FARMER/MERCHANT/ADMIN) ở cấp controller. Để chống tấn công brute - force và lạm dụng API, hệ thống sử dụng ThrottlerGuard giới hạn tốc độ request. Các API tạo tài nguyên quan trọng (tạo listing, tạo phiên đấu giá) được bảo vệ bởi cơ chế idempotency - key kết hợp băm SHA - 256 nội dung request, đảm bảo việc client gửi lại request do lỗi mạng không tạo ra bản ghi trùng lặp [10].

Ngoài ra, dữ liệu nhạy cảm (mật khẩu, token nội bộ, thông tin tài khoản ngân hàng) không bao giờ được trả về nguyên vẹn cho client: TransformInterceptor đảm nhiệm việc lọc bỏ các trường nhạy cảm khỏi response trước khi đóng gói thành envelope chuẩn (status, data, message). Toàn bộ input của client đều được kiểm tra ràng buộc bằng class - validator (DTO) ở tầng controller trước khi vào tầng service, giúp ngăn chặn dữ liệu không hợp lệ làm sai lệch logic nghiệp vụ hoặc gây lỗi ở tầng cơ sở dữ liệu.

### *1.3.11. Công cụ và quy trình phát triển*

Mã nguồn được quản lý phiên bản bằng Git, tổ chức dưới dạng nhiều thư mục dự án độc lập (Backend, Client, Admin, Blockchain) trong cùng một repository tổng (monorepo theo kiểu thư mục, không dùng workspace npm chia sẻ). Backend cung cấp tài liệu API tự động bằng Swagger (truy cập tại endpoint /api), giúp việc kiểm thử thủ công và trao đổi giữa backend - frontend thuận tiện hơn trong quá trình phát triển. Việc

kiểm thử smart contract sử dụng bộ công cụ Hardhat kết hợp Mocha/Chai để viết test case cho từng hàm của AgriBidEscrow và AgriBidNFT (tạo phiên, đặt giá, kết thúc phiên, rút tiền hoàn/thắng, mint/transfer NFT), đảm bảo logic tài chính trên smart contract hoạt động đúng trước khi triển khai lên mạng thử nghiệm (testnet) [15].

Về môi trường vận hành, hệ thống cung cấp các endpoint health - check (kiểm tra kết nối PostgreSQL, Redis, RPC blockchain) và endpoint metrics theo định dạng Prometheus, phục vụ việc giám sát tình trạng hoạt động khi triển khai. Cấu hình môi trường (biến môi trường) được tách riêng cho từng package: Backend cấu hình PORT, các biến DB_*, JWT_SECRET, REDIS_*, BLOCKCHAIN_* và khóa PayOS; Client cấu hình NEXT_PUBLIC_API_URL, các biến NEXT_PUBLIC_BLOCKCHAIN_* và project ID của WalletConnect; Admin cấu hình NEXT_PUBLIC_API_URL trỏ tới cùng host với Backend (để cookie refresh hoạt động đúng trong môi trường phát triển).

Bảng 1.2 tổng hợp các công nghệ, thư viện chính và phiên bản cụ thể được sử dụng trong từng thành phần của hệ thống, trích xuất trực tiếp từ các tệp package.json của bốn package trong mã nguồn (Backend, Client, Admin, Blockchain) và đối chiếu với tài liệu chính thức của các công nghệ tương ứng [10]-[19]:

Bảng 1.2 - Tổng hợp công nghệ sử dụng theo từng thành phần

| Lớp | Công nghệ | Phiên bản | Vai trò |
|---|---|---|---|
| Backend | @nestjs/core | ^11.0.1 | Framework chính, kiến trúc module/dependency injection |
| Backend | typeorm | ^0.3.28 | ORM ánh xạ entity - bảng PostgreSQL, quản lý migration |
| Backend | pg | ^8.16.3 | Driver kết nối PostgreSQL |
| Backend | ioredis | ^5.8.2 | Kết nối Redis cho cache, rate - limit, hàng đợi |
| Backend | bull | ^4.16.5 | Hàng đợi job nền (sweep đơn hàng, đối soát phí, top - up PayOS) |

| Lớp | Công nghệ | Phiên bản | Vai trò |
|---|---|---|---|
| Backend | livekit - server - sdk | ^2.15.1 | Cấp token và quản lý phòng livestream WebRTC |
| Backend | ethers | ^6.16.0 | Tương tác với smart contract Ethereum (đọc sự kiện, gửi giao dịch) |
| Client | next | 16.2.2 | Framework React (App Router) cho Frontend Merchant/Farmer |
| Client | react | 19.2.4 | Thư viện dựng giao diện người dùng |
| Client | zustand | ^5.0.12 | Quản lý state phía client (giỏ hàng, phiên đấu giá...) |
| Client | @tanstack/react - query | ^5.96.1 | Quản lý dữ liệu bất đồng bộ, cache API |
| Client | tailwindcss | ^4 | Styling utility - first |
| Client | wagmi | ^3.6.8 | Kết nối ví MetaMask, gọi hợp đồng từ trình duyệt |
| Client | viem | ^2.48.4 | Thư viện tương tác blockchain mức thấp cho wagmi |
| Admin | next | ^14.2.3 | Framework React cho giao diện quản trị |
| Admin | react | ^18.3.1 | Thư viện dựng giao diện người dùng |
| Admin | @mui/material | ^5.15.19 | Thư viện component UI (Material UI) |

| Lớp | Công nghệ | Phiên bản | Vai trò |
|---|---|---|---|
| Admin | @tanstack/react - query | ^5.94.5 | Quản lý dữ liệu bất đồng bộ, cache API |
| Admin | tailwindcss | ^3.4.4 | Styling utility - first |
| Blockchain | ethers | ^6.16.0 | Thư viện triển khai/giao tiếp hợp đồng |
| Blockchain | hardhat | ^3.1.12 | Môi trường biên dịch, kiểm thử, triển khai smart contract |
| Blockchain | @openzeppelin/contracts | ^5.6.1 | Thư viện hợp đồng chuẩn (AccessControl, ERC - 721, Pausable...) |

Có thể thấy hệ sinh thái công nghệ của AgriBid trải dài từ backend (NestJS/TypeORM) đến hai frontend độc lập (Next.js, với các phiên bản khác nhau cho Client và Admin do yêu cầu giao diện khác nhau) và blockchain (Hardhat/Ethers/OpenZeppelin). Điều này thể hiện rõ tính chất full - stack và đa nền tảng của đề tài.

# Chương 2: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG

## 2.1. Phân tích yêu cầu hệ thống

### *2.1.1. Tác nhân của hệ thống*

Bảng 2.1 - Danh sách tác nhân của hệ thống

| Tác nhân | Mô tả |
|---|---|
| FARMER (Nông dân) | Người bán nông sản theo lô hàng; tạo listing/phiên đấu giá/livestream; quản lý đơn bán, ví, yêu cầu rút tiền; cần được ADMIN duyệt KYC trước khi đăng bán. |
| MERCHANT (Thương lái/Người mua) | Duyệt marketplace, mua ngay, tham gia đấu giá (đặt bid bằng VND hoặc ETH), xem livestream và mua trong phiên live, quản lý giỏ hàng/đơn mua, đánh giá sản phẩm. |
| ADMIN (Quản trị viên) | Duyệt KYC nông dân, duyệt/từ chối lô hàng (kích hoạt mint NFT), quản lý danh mục sản phẩm, voucher, cấu hình hệ thống, giám sát đơn hàng/đấu giá/blockchain, duyệt yêu cầu rút tiền, tạo báo cáo phí nền tảng. |

### *2.1.2. Yêu cầu chức năng theo nhóm nghiệp vụ*

Bảng 2.2 - Yêu cầu chức năng theo nhóm nghiệp vụ

| Nhóm chức năng | Mô tả yêu cầu |
|---|---|
| Xác thực & hồ sơ | Đăng ký/đăng nhập (JWT access + refresh), đổi mật khẩu, cập nhật hồ sơ, nộp hồ sơ KYC (ảnh giấy tờ), Admin duyệt/từ chối KYC. |
| Quản lý lô hàng (Product Batch) | Farmer tạo lô hàng (sản phẩm, danh mục, khối lượng/số lượng, ngày thu hoạch, quy trình kỹ thuật, hình ảnh/video); trạng thái PENDING → READY_FOR_SALE/REJECTED → SOLD; Admin duyệt và (nếu đạt) mint NFT gắn với lô. |

| Nhóm chức năng | Mô tả yêu cầu |
|---|---|
| Bán hàng cố định (Listing) | Farmer niêm yết giá/đơn vị và số lượng khả dụng cho một lô (1 batch - 1 listing); hỗ trợ thanh toán qua ví nội bộ, PayOS, VietQR chuyển khoản thẳng; áp dụng voucher giảm giá. |
| Đấu giá (Auction) | Farmer tạo phiên đấu giá English Auction (giá khởi điểm, bước giá, tiền đặt cọc, thời gian bắt đầu/kết thúc); hệ thống tự động bắt đầu/kết thúc theo lịch (Bull Queue); Merchant đặt giá realtime (WebSocket); hỗ trợ song song đấu giá VND (off - chain) và ETH (on - chain qua AgriBidEscrow). |
| Giỏ hàng & Đặt hàng | Thêm/xóa/chọn sản phẩm trong giỏ (hỗ trợ nhiều người bán); checkout từ giỏ hoặc mua ngay (buy - now); áp voucher; tạo đơn hàng với trạng thái theo dõi xuyên suốt vòng đời. |
| Thanh toán & Ví | Ví VND nội bộ (số dư khả dụng + số dư đang giữ); nạp tiền qua PayOS; thanh toán đơn hàng qua ví/PayOS/VietQR; ghi nhận mọi biến động vào sổ cái Transaction (DEPOSIT/WITHDRAW/HOLD/RELEASE/PAYMENT). |
| Rút tiền (Payout) | Farmer tạo yêu cầu rút tiền về tài khoản ngân hàng; Admin duyệt/hoàn tất/từ chối; job tự động hoàn tất các yêu cầu đã duyệt. |
| Livestream bán hàng | Farmer phát trực tiếp (LiveKit/WebRTC), ghim tối đa 24 listing đang bán; người xem mua ngay hoặc thêm giỏ không rời màn live; chat realtime; tự kết thúc phiên nếu mất kết nối > 90 giây. |
| Đánh giá sản phẩm | Merchant đánh giá (kèm media) sau khi đơn hàng hoàn tất; thống kê điểm đánh giá theo listing. |
| Voucher & Phí nền tảng | Admin tạo voucher (giảm theo số tiền cố định hoặc theo %, giới hạn lượt dùng/thời gian); job sinh báo cáo phí nền tảng hàng tháng theo từng Farmer dựa trên đơn hàng COMPLETED. |
| Blockchain & NFT | Đồng bộ hai chiều giữa backend và smart contract AgriBidEscrow/AgriBidNFT; mint NFT cho lô hàng sau khi được |

| Nhóm chức năng | Mô tả yêu cầu |
|---|---|
| | Admin phê duyệt; cho phép tra cứu công khai chứng chỉ truy xuất nguồn gốc theo NFT. Blockchain/NFT được sử dụng như lớp bằng chứng chống chỉnh sửa sau phê duyệt, trong khi dữ liệu đầu vào vẫn được kiểm soát thông qua KYC Farmer, quy trình duyệt lô hàng và media minh chứng. |
| Quản trị & Giám sát | Dashboard tổng hợp số liệu; cấu hình hệ thống (key - value); health - check và metrics (Prometheus) cho vận hành. |

### *2.1.3. Yêu cầu phi chức năng*

Tính nhất quán dữ liệu tài chính: mọi thao tác liên quan tới ví (đặt giá, hoàn tiền, thanh toán, nạp/rút tiền) phải được thực hiện trong database transaction kèm pessimistic lock (SELECT ... FOR UPDATE) trên dòng dữ liệu Wallet/Auction liên quan, tránh race - condition khi nhiều người dùng thao tác đồng thời.

Idempotency: API tạo listing và tạo phiên đấu giá phải hỗ trợ header x - idempotency - key kèm băm SHA - 256 nội dung request, đảm bảo retry do lỗi mạng không sinh dữ liệu trùng.

Realtime: cập nhật giá đấu giá mới nhất, tin nhắn chat, trạng thái phiên live phải được đẩy tới client trong thời gian thực qua WebSocket (Socket.IO, có Redis adapter để mở rộng nhiều instance).

Bảo mật: xác thực JWT (access 15 phút/refresh 7 ngày), mã hóa mật khẩu bcrypt, kiểm soát truy cập theo vai trò (RolesGuard), giới hạn tốc độ request (ThrottlerGuard), xác thực chữ ký webhook thanh toán.

Khả năng mở rộng & vận hành: kiến trúc module hóa theo NestJS, tách hàng đợi tác vụ nền (Bull/Redis) khỏi luồng request chính, có endpoint health - check/metrics phục vụ giám sát.

Đa ngôn ngữ: giao diện Client hỗ trợ song song Tiếng Việt và Tiếng Anh (next - intl), định tuyến theo locale.

## 2.2. Đặc tả Use Case theo tác nhân




Hình 2.1 - Biểu đồ Use case tổng quát hệ thống AgriBid

Hình 2.1 thể hiện ba tác nhân FARMER, MERCHANT, ADMIN và các nhóm use case tương ứng (riêng theo từng tác nhân và các use case dùng chung). Bảng dưới đây đặc tả các use case chính theo từng tác nhân, làm cơ sở để thiết kế API và giao diện ở các phần tiếp theo.

### *2.2.1. Use case của FARMER*

Bảng 2.3 - Use case của nông dân

| Use case | Mô tả tóm tắt |
|---|---|
| Quản lý hồ sơ & KYC | Cập nhật thông tin cá nhân, tài khoản ngân hàng (phục vụ VietQR), nộp ảnh giấy tờ để Admin xác minh KYC. |

| Use case | Mô tả tóm tắt |
|---|---|
| Quản lý lô hàng | Tạo/sửa/xóa lô hàng; tải lên hình ảnh/video minh chứng; theo dõi trạng thái duyệt (PENDING/READY_FOR_SALE/REJECTED/SOLD). |
| Tạo & quản lý Listing | Niêm yết giá bán cố định cho lô hàng đã duyệt; cập nhật giá/số lượng; hủy listing. |
| Tạo & quản lý phiên đấu giá | Thiết lập giá khởi điểm, bước giá, tiền đặt cọc, thời gian bắt đầu/kết thúc; theo dõi diễn biến đấu giá realtime; xem lịch sử bid. |
| Phát livestream bán hàng | Khởi tạo phiên live, ghim/bỏ ghim listing đang giới thiệu, gửi heartbeat, kết thúc phiên, trò chuyện với người xem. |
| Quản lý đơn bán | Xem danh sách đơn có sản phẩm của mình, cập nhật trạng thái vận chuyển, duyệt biên lai chuyển khoản VietQR của người mua. |
| Quản lý ví & rút tiền | Xem số dư/lịch sử giao dịch ví; tạo yêu cầu rút tiền (payout) về tài khoản ngân hàng. |

### *2.2.2. Use case của MERCHANT*

Bảng 2.4 - Use case của người mua

| Use case | Mô tả tóm tắt |
|---|---|
| Duyệt marketplace | Tìm kiếm, lọc theo danh mục, xem chi tiết sản phẩm/lô hàng, xem thông tin truy xuất nguồn gốc (NFT). |
| Mua hàng (Listing) | Thêm vào giỏ hoặc mua ngay; áp dụng voucher; thanh toán qua ví nội bộ/PayOS/VietQR. |
| Tham gia đấu giá | Xem giá hiện tại, đặt giá mới (≥ giá hiện tại + bước giá); theo dõi số dư bị đóng băng; nhận thông báo khi bị trả giá vượt; (tuỳ chọn) đặt giá bằng ETH qua MetaMask. |

| Use case | Mô tả tóm tắt |
|---|---|
| Xem & mua trong livestream | Tham gia phòng live, xem danh sách listing được ghim, mua ngay hoặc thêm giỏ không rời màn live, chat với người bán. |
| Quản lý giỏ hàng & đơn mua | Quản lý sản phẩm trong giỏ (nhiều người bán); theo dõi trạng thái đơn hàng; xác nhận đã nhận hàng; tải lên biên lai chuyển khoản VietQR. |
| Đánh giá sản phẩm | Viết đánh giá kèm hình ảnh sau khi đơn hàng hoàn tất. |
| Quản lý ví | Nạp tiền qua PayOS; xem lịch sử giao dịch và số dư khả dụng/đang giữ. |

### *2.2.3. Use case của ADMIN*

Bảng 2.5 - Use case của quản trị viên

| Use case | Mô tả tóm tắt |
|---|---|
| Duyệt KYC | Xem danh sách hồ sơ KYC chờ duyệt, phê duyệt hoặc từ chối kèm lý do. |
| Duyệt lô hàng | Xem chi tiết lô hàng chờ duyệt, phê duyệt (kích hoạt mint NFT) hoặc từ chối kèm lý do. |
| Quản lý danh mục & voucher | CRUD danh mục sản phẩm, sản phẩm gốc, voucher giảm giá. |
| Giám sát giao dịch | Xem/can thiệp đơn hàng, duyệt biên lai chuyển khoản VietQR khi cần, theo dõi phiên đấu giá, thử đồng bộ lại on - chain (retry - chain - sync) khi lỗi. |
| Quản lý Payout | Duyệt, hoàn tất hoặc từ chối yêu cầu rút tiền của Farmer. |
| Báo cáo phí nền tảng | Sinh báo cáo phí hàng tháng theo Farmer dựa trên đơn hàng COMPLETED, xem tổng hợp. |
| Cấu hình & giám sát hệ thống | Cập nhật cấu hình hệ thống (key - value), xem dashboard tổng hợp, kiểm tra health - check/metrics, sức khỏe kết nối blockchain. |

### *2.2.4. Đặc tả chi tiết một số use case quan trọng*

Phần này đặc tả chi tiết theo mẫu chuẩn (mã use case, tác nhân, mô tả, điều kiện tiên quyết, luồng sự kiện chính, luồng rẽ nhánh/ngoại lệ, hậu điều kiện) cho tám use case có độ phức tạp nghiệp vụ cao nhất của hệ thống - đây cũng là các luồng được hiện thực chi tiết ở mục 2.5.

**Đặc tả use case UC - 01: Đặt giá trong phiên đấu giá (VND)**

Bảng 2.6 - Đặc tả use case UC-01: Đặt giá trong phiên đấu giá (VND)

| | |
|---|---|
| **Tác nhân** | MERCHANT |
| **Mô tả** | Người mua đặt một mức giá mới cho phiên đấu giá đang diễn ra; hệ thống xác thực, đóng băng số dư ví tương ứng và cập nhật giá hiện tại của phiên theo thời gian thực. |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập với vai trò MERCHANT và đã xác minh KYC.<br>Phiên đấu giá tồn tại, có paymentMode = TRADITIONAL (VND) và đang ở trạng thái HAPPENING (giữa startTime và endTime).<br>Người dùng đã thiết lập địa chỉ giao hàng mặc định.<br>Người dùng không phải là chủ phiên đấu giá (Farmer không tự đấu giá lô của mình). |
| **Luồng sự kiện chính** | 1. Merchant nhập mức giá đặt (bidAmount) trên giao diện chi tiết phiên đấu giá.<br>2. Hệ thống kiểm tra: bidAmount ≥ giá hiện tại + bước giá (stepPrice), hoặc ≥ giá khởi điểm nếu chưa có lượt đặt nào.<br>3. Hệ thống mở database transaction, khóa pessimistic_write trên bản ghi Auction và Wallet của người đặt giá.<br>4. Tính số tiền cần đóng băng thêm: nếu là lượt đặt đầu tiên của người dùng trong phiên, đóng băng (bidAmount + depositReq); nếu đã từng đặt trước đó, chỉ đóng băng phần chênh lệch so với mức đặt cao nhất trước đó của chính họ. |

| | |
|---|---|
| | 5. Kiểm tra số dư khả dụng (balance) ≥ số tiền cần đóng băng thêm; nếu đủ, trừ vào balance, cộng vào frozenBalance, ghi một bản ghi Transaction loại HOLD (giá trị âm).<br>6. Lưu bản ghi Bid mới; cập nhật currentPriceMinor của Auction = bidAmount.<br>7. Commit transaction; phát sự kiện realtime (Socket.IO) cập nhật giá mới và lịch sử bid tới toàn bộ client đang xem phiên. |
| **Luồng rẽ nhánh / ngoại lệ** | 3a. Phiên không ở trạng thái HAPPENING hoặc đã hết endTime → trả lỗi 400 "Phiên đấu giá không khả dụng".<br>5a. Số dư khả dụng không đủ → rollback transaction, trả lỗi 400 "Số dư ví không đủ để đặt giá".<br>2a. bidAmount không thỏa điều kiện bước giá tối thiểu → trả lỗi 400 kèm thông tin giá hợp lệ tối thiểu. |
| **Hậu điều kiện** | Phiên đấu giá có currentPriceMinor và bidder dẫn đầu mới; số dư ví của người đặt giá được cập nhật (giảm balance, tăng frozenBalance); sổ cái Transaction ghi nhận khoản HOLD tương ứng. |

**Đặc tả use case UC - 02: Tạo phiên đấu giá**

Bảng 2.7 - Đặc tả use case UC-02: Tạo phiên đấu giá

| | |
|---|---|
| **Tác nhân** | FARMER |
| **Mô tả** | Nông dân thiết lập một phiên đấu giá English Auction cho một lô hàng đã được duyệt (READY_FOR_SALE), gồm giá khởi điểm, bước giá, tiền đặt cọc và thời gian bắt đầu/kết thúc. |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập với vai trò FARMER, hồ sơ KYC ở trạng thái VERIFIED.<br>Lô hàng (product batch) thuộc sở hữu của Farmer, có status = READY_FOR_SALE và chưa gắn với bất kỳ listing/auction nào khác. |

| | |
|---|---|
| | Nếu paymentMode = TRADITIONAL, Farmer đã cập nhật đầy đủ thông tin tài khoản ngân hàng (phục vụ thanh toán VietQR cho người thắng).<br>Client gửi kèm header x - idempotency - key. |
| **Luồng sự kiện chính** | 1. Farmer nhập thông tin phiên: giá khởi điểm, bước giá, tiền đặt cọc, thời gian bắt đầu/kết thúc, paymentMode (TRADITIONAL/BLOCKCHAIN).<br>2. Hệ thống kiểm tra idempotency - key qua IdempotencyService.reserve(); nếu key đã COMPLETED, trả lại response cũ (replay) và kết thúc.<br>3. Hệ thống kiểm tra batch hợp lệ (thuộc Farmer, đúng trạng thái, chưa được dùng cho kênh khác) và kiểm tra ràng buộc dữ liệu (giá khởi điểm > 0, endTime > startTime > thời điểm hiện tại, bước giá > 0).<br>4. Tạo bản ghi Auction với status = WAITING, onChainSyncStatus = NOT_REQUIRED (nếu TRADITIONAL) hoặc PENDING (nếu BLOCKCHAIN).<br>5. Đẩy hai job vào Bull Queue 'auction - scheduler': start - auction (chạy lúc startTime) và end - auction (chạy lúc endTime).<br>6. Đánh dấu idempotency record là COMPLETED kèm response trả về. |
| **Luồng rẽ nhánh / ngoại lệ** | 2a. Idempotency - key đã tồn tại nhưng requestHash khác (payload bị đổi) → trả lỗi 400.<br>2b. Idempotency - key đang ở trạng thái PENDING (request trùng đang xử lý) → trả lỗi 409.<br>3a. KYC chưa VERIFIED hoặc thiếu thông tin ngân hàng (với TRADITIONAL) → trả lỗi 403.<br>3b. Lô hàng không hợp lệ (sai trạng thái/đã gắn kênh khác) → trả lỗi 400. |

| | |
|---|---|
| **Hậu điều kiện** | Một bản ghi Auction mới ở trạng thái WAITING được tạo; hai job lập lịch được đăng ký trong Redis/Bull; lô hàng được đánh dấu đang sử dụng cho kênh Auction. |

**Đặc tả use case UC - 03: Checkout giỏ hàng đa người bán**

Bảng 2.8 - Đặc tả use case UC-03: Checkout giỏ hàng đa người bán

| | |
|---|---|
| Tác nhân | MERCHANT |
| **Mô tả** | Người mua thanh toán cùng lúc các sản phẩm trong giỏ hàng (có thể thuộc nhiều Farmer khác nhau); hệ thống tách thành nhiều đơn hàng riêng biệt theo từng Farmer và áp dụng nguyên tắc tất cả hoặc không. |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập với vai trò MERCHANT, giỏ hàng có ít nhất một sản phẩm còn khả dụng (listing đang ACTIVE, đủ số lượng).<br>Người dùng đã chọn địa chỉ giao hàng và kênh thanh toán (INTERNAL_WALLET / PAYOS / BANK_TRANSFER_VIETQR). |
| **Luồng sự kiện chính** | 1. Merchant chọn các cart_item muốn thanh toán và (tuỳ chọn) nhập mã voucher.<br>2. Hệ thống nhóm các cart_item theo Farmer sở hữu listing tương ứng; với mỗi nhóm, kiểm tra lại giá và số lượng khả dụng tại thời điểm checkout.<br>3. Hệ thống mở transaction tổng; với mỗi nhóm, tính finalAmount (sau khi áp voucher nếu hợp lệ cho nhóm đó) và tạo một bản ghi Order kèm các OrderItem tương ứng.<br>4. Nếu kênh thanh toán là INTERNAL_WALLET: với mỗi Order, khóa pessimistic_write ví người mua, kiểm tra và trừ trực tiếp số dư, ghi Transaction loại PAYMENT, đặt order.status = CONFIRMED_PAID. |

| | |
|---|---|
| | 5. Nếu kênh thanh toán là PAYOS: tạo một link thanh toán PayOS cho tổng giá trị các đơn, đặt order.status = PENDING_CONFIRM cho đến khi nhận webhook xác nhận.<br>6. Nếu kênh thanh toán là BANK_TRANSFER_VIETQR: với mỗi Order, sinh externalOrderCode và thông tin chuyển khoản tới tài khoản ngân hàng của Farmer tương ứng, đặt order.status = PENDING_BANK_TRANSFER.<br>7. Giảm availableQuantity của các listing tương ứng, xóa các cart_item đã thanh toán, commit transaction. |
| **Luồng rẽ nhánh / ngoại lệ** | 2a. Một listing trong giỏ không còn đủ số lượng hoặc đã CANCELLED → toàn bộ thao tác checkout bị hủy (rollback), trả lỗi 409 kèm danh sách sản phẩm bị ảnh hưởng.<br>4a. Số dư ví không đủ cho một trong các Order → rollback toàn bộ transaction (không tạo Order nào).<br>1a. Voucher hết lượt sử dụng/hết hạn/không áp dụng cho sản phẩm trong nhóm → bỏ qua giảm giá cho nhóm đó và thông báo cho người dùng. |
| **Hậu điều kiện** | Một hoặc nhiều Order (mỗi Order tương ứng một Farmer) được tạo với trạng thái phù hợp kênh thanh toán; giỏ hàng được cập nhật (loại bỏ các item đã mua); nếu thanh toán ví, số dư và sổ cái Transaction được cập nhật ngay. |

**Đặc tả use case UC - 04: Duyệt KYC nông dân**

Bảng 2.9 - Đặc tả use case UC-04: Duyệt KYC nông dân

| | |
|---|---|
| **Tác nhân** | ADMIN |
| **Mô tả** | Quản trị viên xem xét hồ sơ định danh (ảnh CCCD/CMND, thông tin cá nhân) do Farmer nộp và quyết định phê duyệt hoặc từ chối, làm điều kiện để Farmer được phép bán hàng. |

| | |
|---|---|
| **Điều kiện tiên quyết** | Người dùng đăng nhập với vai trò ADMIN.<br>Tồn tại ít nhất một hồ sơ Farmer có kycStatus = PENDING. |
| **Luồng sự kiện chính** | 1. Admin mở danh sách hồ sơ KYC đang chờ duyệt (lọc theo kycStatus = PENDING).<br>2. Admin chọn một hồ sơ, xem ảnh giấy tờ và thông tin cá nhân đã kê khai.<br>3a. Nếu hồ sơ hợp lệ: Admin chọn "Phê duyệt" → hệ thống cập nhật kycStatus = VERIFIED.<br>3b. Nếu hồ sơ không hợp lệ: Admin nhập lý do từ chối và chọn "Từ chối" → hệ thống cập nhật kycStatus = REJECTED và lưu rejectionReason.<br>4. Hệ thống ghi log thao tác và (tuỳ cấu hình) gửi thông báo cho Farmer về kết quả duyệt. |
| **Luồng rẽ nhánh / ngoại lệ** | 2a. Hồ sơ đã được Admin khác xử lý trước đó (không còn ở trạng thái PENDING) → hệ thống báo hồ sơ không còn hợp lệ để duyệt và làm mới danh sách. |
| **Hậu điều kiện** | Trường kycStatus của user được cập nhật thành VERIFIED hoặc REJECTED; nếu VERIFIED, Farmer đủ điều kiện tạo lô hàng/listing/auction/livestream; nếu REJECTED, Farmer có thể chỉnh sửa và nộp lại hồ sơ. |

**Đặc tả use case UC - 05: Duyệt lô hàng và mint NFT truy xuất nguồn gốc**

Bảng 2.10 - Đặc tả use case UC-05: Duyệt lô hàng và mint NFT truy xuất nguồn gốc

| | |
|---|---|
| **Tác nhân** | ADMIN |
| **Mô tả** | Quản trị viên kiểm tra thông tin và minh chứng (hình ảnh/video, quy trình kỹ thuật) của một lô hàng do Farmer khai báo; nếu đạt yêu cầu, hệ thống tự động mint một NFT (ERC - 721) gắn với lô hàng làm chứng chỉ truy xuất nguồn gốc. |

| | |
|---|---|
| **Điều kiện tiên quyết** | Người dùng đăng nhập với vai trò ADMIN.<br>Lô hàng có status = PENDING, đã được Farmer đính kèm đầy đủ batch_media (hình ảnh/video) và mô tả quy trình kỹ thuật.<br>Farmer sở hữu lô hàng đã khai báo địa chỉ ví (walletAddress) hợp lệ. |
| **Luồng sự kiện chính** | 1. Admin mở danh sách lô hàng PENDING, xem chi tiết: thông tin sản phẩm, khối lượng, ngày thu hoạch, quy trình kỹ thuật và media đính kèm.<br>2. Admin chọn "Phê duyệt".<br>3. Hệ thống gọi BlockchainService.mintNFT(), gửi giao dịch tới smart contract AgriBidNFT để mint một token mới về địa chỉ ví của Farmer, kèm metadata tham chiếu tới batchId.<br>4. Khi giao dịch on - chain được xác nhận, hệ thống cập nhật batch.status = READY_FOR_SALE, lưu nftTokenId và txHash vào bản ghi product_batches.<br>5. Farmer nhận thông báo lô hàng đã được duyệt và có thể tạo Listing hoặc Auction. |
| **Luồng rẽ nhánh / ngoại lệ** | 2a. Admin chọn "Từ chối": nhập rejectionReason → hệ thống cập nhật batch.status = REJECTED, không thực hiện mint NFT; Farmer có thể chỉnh sửa và nộp lại.<br>3a. Giao dịch mint NFT thất bại (lỗi mạng/gas) → batch giữ nguyên status = PENDING, hệ thống ghi log lỗi để Admin thử lại. |
| **Hậu điều kiện** | Lô hàng chuyển sang READY_FOR_SALE kèm nftTokenId hợp lệ (trường hợp duyệt) hoặc REJECTED kèm lý do (trường hợp từ chối); NFT tương ứng tồn tại trên smart contract AgriBidNFT và có thể tra cứu công khai qua trang truy xuất nguồn gốc. |

**Đặc tả use case UC - 06: Phát livestream bán hàng và xử lý heartbeat**

Bảng 2.11 - Đặc tả use case UC-06: Phát livestream bán hàng và xử lý heartbeat

| | |
|---|---|
| **Tác nhân** | FARMER (host), MERCHANT (viewer) |
| **Mô tả** | Farmer khởi tạo phiên livestream để giới thiệu các listing đang bán; người xem có thể mua ngay hoặc thêm giỏ mà không rời màn hình; hệ thống tự động kết thúc phiên nếu mất kết nối kéo dài. |
| **Điều kiện tiên quyết** | Farmer đã KYC VERIFIED và có ít nhất một listing đang ACTIVE.<br>Trình duyệt của Farmer hỗ trợ WebRTC (camera/microphone). |
| **Luồng sự kiện chính** | 1. Farmer gọi API bắt đầu phiên live (loại LISTING); hệ thống tạo phòng trên LiveKit và bản ghi live_sessions với status = LIVE, lưu providerRoomId.<br>2. Backend cấp token host (livekit - token.service.ts) cho Farmer để kết nối WebRTC và phát hình.<br>3. Farmer ghim (pin) tối đa 24 listing muốn giới thiệu; danh sách pin được lưu vào live_session_pins và phát realtime cho người xem.<br>4. Người xem (Merchant) tham gia phòng, nhận token viewer, xem video trực tiếp và danh sách sản phẩm đang ghim; có thể mua ngay (chuyển hướng tới luồng VietQR Level 2) hoặc thêm vào giỏ ngay trên giao diện live.<br>5. Client của Farmer gửi heartbeat (PATCH .../heartbeat) mỗi 30 giây để cập nhật lastHeartbeatAt.<br>6. Bộ quét nội bộ của LiveSessionsService chạy mỗi 30 giây, kiểm tra các phiên đang LIVE.<br>7. Khi kết thúc chủ động, Farmer gọi API end → hệ thống cập nhật status = ENDED, đóng phòng LiveKit. |
| **Luồng rẽ nhánh / ngoại lệ** | 6a. Nếu một phiên LIVE không nhận heartbeat quá 90 giây, bộ quét tự động cập nhật status = FORCE_ENDED, đóng phòng |

| | |
|---|---|
| | LiveKit và gửi thông báo cho người xem rằng phiên đã kết thúc do mất kết nối.<br>3a. Số lượng pin đã đạt 24 → yêu cầu ghim thêm bị từ chối cho đến khi Farmer bỏ ghim một sản phẩm khác. |
| **Hậu điều kiện** | Bản ghi live_sessions phản ánh đúng trạng thái cuối (ENDED hoặc FORCE_ENDED); danh sách pin và lịch sử chat được lưu lại; các đơn hàng phát sinh trong phiên (nếu có) được tạo theo luồng UC - 03. |

**Đặc tả use case UC - 07: Nạp tiền và yêu cầu rút tiền ví VND**

Bảng 2.12 - Đặc tả use case UC-07: Nạp tiền và yêu cầu rút tiền ví VND

| | |
|---|---|
| **Tác nhân** | MERCHANT/FARMER (nạp tiền), FARMER (rút tiền), ADMIN (duyệt rút tiền) |
| **Mô tả** | Người dùng nạp tiền vào ví nội bộ qua PayOS/VietQR để có số dư giao dịch; Farmer có thể yêu cầu rút số dư khả dụng về tài khoản ngân hàng, yêu cầu này cần Admin duyệt trước khi tiền thực sự được chuyển. |
| **Điều kiện tiên quyết** | Người dùng đã đăng nhập và có ví (wallets) được khởi tạo tự động khi đăng ký.<br>Đối với rút tiền: Farmer đã liên kết thông tin tài khoản ngân hàng nhận tiền. |
| **Luồng sự kiện chính** | 1. (Nạp tiền) Người dùng chọn số tiền muốn nạp, hệ thống gọi PayOS tạo link thanh toán VietQR và trả về cho client.<br>2. Người dùng quét mã/chuyển khoản; PayOS gửi webhook POST /payments/payos/webhook xác nhận giao dịch thành công kèm chữ ký xác thực.<br>3. Backend kiểm tra chữ ký webhook, đối chiếu mã giao dịch (idempotent theo orderCode của PayOS) và ghi bản ghi transactions loại DEPOSIT, cộng vào balance của ví tương ứng. |

| | |
|---|---|
| | 4. (Rút tiền) Farmer gửi yêu cầu rút tiền với số tiền ≤ availableBalance (= balance − frozenBalance); hệ thống tạo bản ghi yêu cầu rút (withdrawal request) ở trạng thái PENDING và ghi transactions loại HOLD, chuyển phần tương ứng từ balance sang frozenBalance.<br>5. Admin xem danh sách yêu cầu rút tiền đang chờ, kiểm tra thông tin và thực hiện chuyển khoản thủ công ra ngoài hệ thống.<br>6. Admin xác nhận đã chuyển khoản → hệ thống ghi transactions loại WITHDRAW, trừ frozenBalance và đóng yêu cầu ở trạng thái COMPLETED. |
| **Luồng rẽ nhánh / ngoại lệ** | 3a. Webhook PayOS gửi trùng lặp (đã xử lý trước đó) → hệ thống nhận diện qua orderCode đã tồn tại trong transactions và bỏ qua, không cộng tiền hai lần.<br>4a. Số tiền yêu cầu rút vượt quá availableBalance → hệ thống từ chối ngay tại API, không tạo bản ghi yêu cầu.<br>6a. Admin từ chối yêu cầu rút tiền → hệ thống ghi transactions loại RELEASE để hoàn frozenBalance về balance, chuyển trạng thái yêu cầu sang REJECTED kèm lý do. |
| **Hậu điều kiện** | Số dư ví (balance/frozenBalance) và lịch sử transactions phản ánh chính xác trạng thái nạp/rút; mọi thay đổi số dư đều có bản ghi giao dịch tương ứng để đối soát. |

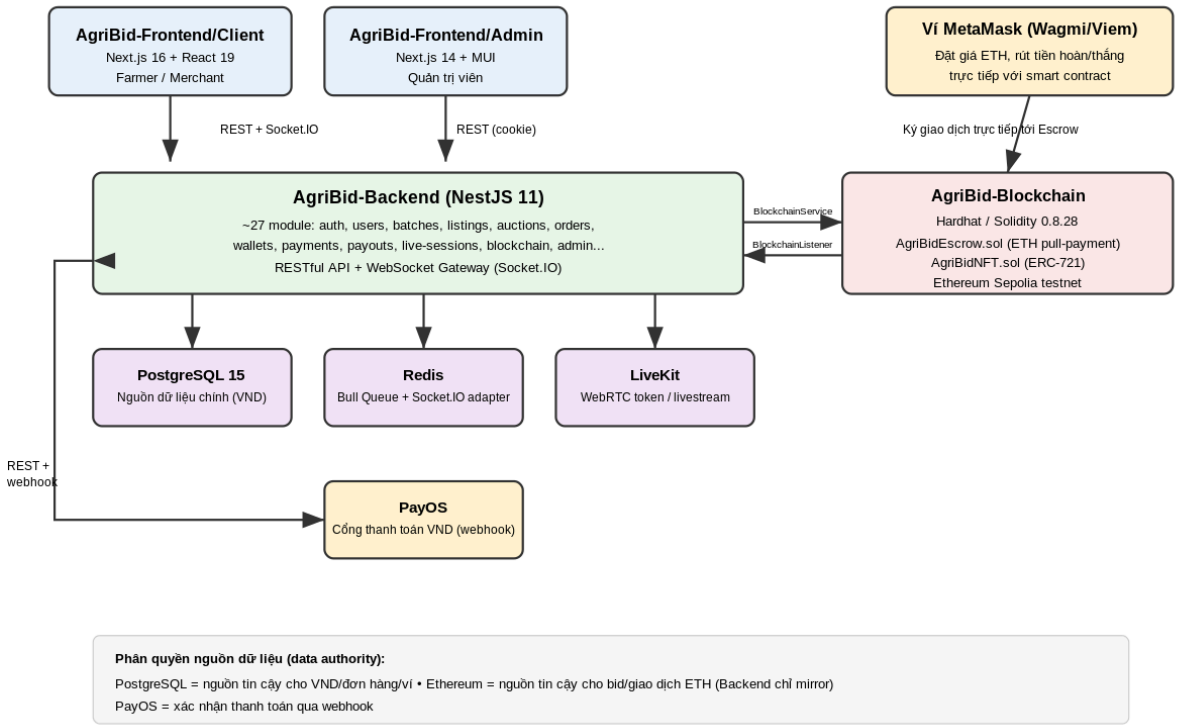
**Đặc tả use case UC - 08: Áp dụng mã giảm giá (voucher) khi checkout**

Bảng 2.13 - Đặc tả use case UC-08: Áp dụng mã giảm giá khi checkout

| | |
|---|---|
| **Tác nhân** | MERCHANT |
| **Mô tả** | Merchant nhập mã voucher trong quá trình checkout để được giảm giá trên tổng giá trị đơn hàng, hệ thống kiểm tra điều kiện áp dụng và ghi nhận lượt sử dụng để tránh dùng vượt giới hạn. |

| | |
|---|---|
| **Điều kiện tiên quyết** | Voucher tồn tại, đang trong thời gian hiệu lực (validFrom ≤ hiện tại ≤ validTo) và còn lượt sử dụng (usageLimit chưa đạt).<br>Giỏ hàng/đơn hàng đáp ứng điều kiện tối thiểu của voucher (ví dụ giá trị đơn hàng tối thiểu). |
| **Luồng sự kiện chính** | 1. Tại trang checkout, Merchant nhập mã voucher và gọi API áp dụng voucher cho đơn hàng dự kiến.<br>2. Hệ thống tra bảng vouchers theo mã, kiểm tra thời hạn hiệu lực, điều kiện giá trị đơn hàng tối thiểu và số lượt sử dụng còn lại.<br>3. Nếu hợp lệ, hệ thống tính số tiền giảm (theo phần trăm hoặc số tiền cố định, có thể giới hạn mức giảm tối đa) và trả về tổng tiền sau giảm để hiển thị cho Merchant.<br>4. Khi Merchant xác nhận đặt hàng, hệ thống tạo bản ghi voucher_redemptions liên kết voucher với order, đồng thời tăng biến đếm lượt sử dụng của voucher trong cùng giao dịch CSDL với việc tạo đơn hàng. |
| **Luồng rẽ nhánh / ngoại lệ** | 2a. Mã voucher không tồn tại, hết hạn, hoặc đã hết lượt sử dụng → trả lỗi 400 kèm thông báo cụ thể, không cho áp dụng.<br>2b. Giá trị đơn hàng chưa đạt mức tối thiểu yêu cầu của voucher → trả lỗi kèm thông tin mức tối thiểu cần đạt.<br>4a. Hai yêu cầu checkout đồng thời cùng sử dụng voucher khi chỉ còn 1 lượt → ràng buộc tăng biến đếm trong transaction CSDL đảm bảo chỉ một yêu cầu thành công, yêu cầu còn lại nhận lỗi hết lượt sử dụng. |
| **Hậu điều kiện** | Đơn hàng được tạo với giá trị đã trừ khuyến mãi; bản ghi voucher_redemptions và biến đếm lượt sử dụng của voucher được cập nhật nhất quán. |

## 2.3. Kiến trúc tổng thể hệ thống




Hình 2.2 - Sơ đồ kiến trúc tổng thể hệ thống AgriBid

Hệ thống AgriBid được tổ chức thành bốn thành phần độc lập, giao tiếp với nhau qua các giao thức chuẩn (Hình 2.2):

- AgriBid - Backend (NestJS): trung tâm xử lý nghiệp vụ, expose RESTful API và WebSocket gateway, kết nối PostgreSQL (lưu trữ chính), Redis (Bull Queue + Socket.IO adapter), LiveKit (token livestream) và mạng Ethereum (qua ethers.js).
- AgriBid - Frontend/Client (Next.js 16): giao diện cho Merchant (marketplace, đấu giá, livestream, giỏ hàng, thanh toán) và portal cho Farmer (quản lý lô hàng, listing, đấu giá, livestream, đơn bán, ví); kết nối Backend qua REST + Socket.IO, kết nối ví MetaMask qua Wagmi/Viem để tương tác trực tiếp với smart contract Escrow khi đấu giá ETH.
- AgriBid - Frontend/Admin (Next.js 14 + MUI): giao diện vận hành cho Admin, kết nối Backend qua REST (cookie refresh).

- AgriBid - Blockchain (Hardhat/Solidity): hai smart contract AgriBidEscrow (đấu giá ETH, pull - payment) và AgriBidNFT (NFT truy xuất nguồn gốc), được Backend gọi (BlockchainService) và lắng nghe sự kiện (BlockchainListener) để đồng bộ về PostgreSQL.

Về phân quyền dữ liệu nguồn (data authority): PostgreSQL là nguồn dữ liệu tin cậy cho người dùng, đơn hàng, ví và các phiên đấu giá theo đường VND; Ethereum là nguồn dữ liệu tin cậy cho các bid/giao dịch ETH, Backend chỉ "mirror" trạng thái thông qua BlockchainListener; PayOS là nguồn xác nhận thanh toán bên ngoài thông qua webhook.

### *2.3.1. Kiến trúc nội bộ Backend*

Mã nguồn Backend được tổ chức theo chuẩn NestJS: thư mục common chứa guard, interceptor, enum, DTO phân trang dùng chung; database chứa cấu hình TypeORM và migration; modules chứa 27 module nghiệp vụ, mỗi module theo mô hình controller → service → entity. Một số module phức tạp (ví dụ orders) được tách thành nhiều service chuyên biệt (order - creation, order - payment, order - bank - transfer, order - cart - checkout, order - settlement, order - voucher, order - pending - payment - scheduler...) đứng sau một service "facade" (orders.service.ts) để giữ controller gọn nhẹ. Các thành phần xuyên suốt (cross - cutting concerns) gồm: TransformInterceptor (chuẩn hóa envelope phản hồi API và loại bỏ trường nhạy cảm), HttpExceptionFilter (chuẩn hóa envelope lỗi), ThrottlerGuard (giới hạn tần suất), JwtAuthGuard kết hợp RolesGuard (xác thực và phân quyền).

### *2.3.2. Sơ đồ luồng tích hợp*

Bảng 2.14 - Các cơ chế tích hợp giữa các thành phần hệ thống

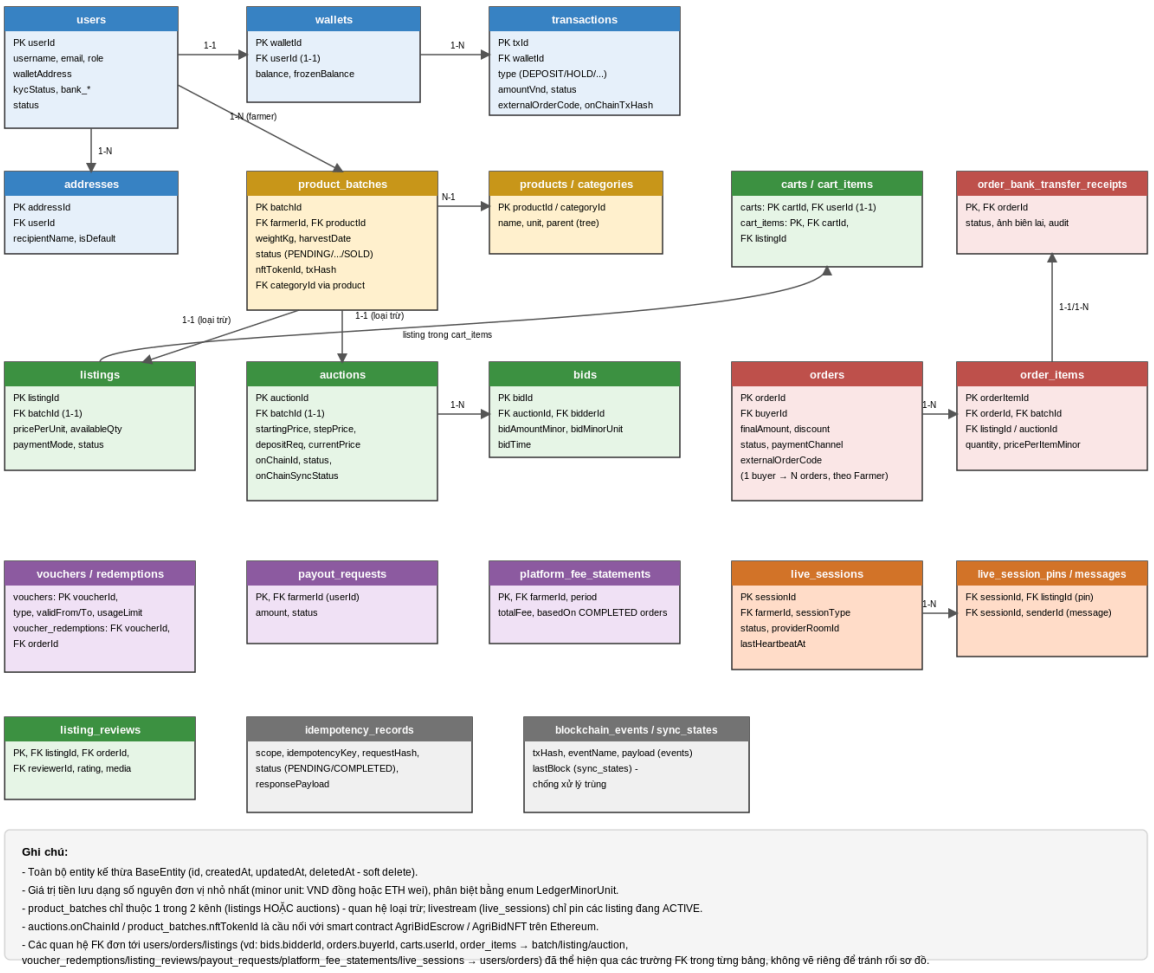
| Tích hợp | Cơ chế |
|---|---|
| Client/Admin → Backend | RESTful API (NEXT_PUBLIC_API_URL), xác thực Bearer JWT / cookie refresh |
| Client → Backend (realtime) | WebSocket (Socket.IO) cho cập nhật đấu giá, chat live |

| Tích hợp | Cơ chế |
|---|---|
| Client → LiveKit | Token cấp phát từ API live - sessions, kết nối WebRTC trực tiếp |
| Client → Blockchain | Wagmi/Viem + GET /blockchain/client - config, ký giao dịch qua MetaMask tới AgriBidEscrow |
| Backend → Blockchain | BlockchainService (gửi giao dịch hệ thống) và BlockchainListener (lắng nghe sự kiện, ethers.js) |
| Backend → PayOS | REST API tạo link thanh toán + endpoint webhook công khai POST /payments/payos/webhook |

### 2.4. Thiết kế cơ sở dữ liệu

Cơ sở dữ liệu được thiết kế theo mô hình quan hệ trên PostgreSQL, quản lý bằng TypeORM. Các bảng được nhóm theo bốn miền nghiệp vụ chính: (1) Người dùng & ví (users, wallets, transactions, addresses); (2) Sản phẩm & lô hàng (categories, products, product_batches, batch_media); (3) Giao dịch (listings, auctions, bids, auction_messages, carts, cart_items, orders, order_items, order_bank_transfer_receipts, vouchers, voucher_redemptions, listing_reviews); (4) Vận hành & Blockchain (live_sessions, live_session_pins, live_session_messages, payout_requests, platform_fee_statements, idempotency_records, system_settings, blockchain_events, blockchain_sync_states). Toàn bộ entity kế thừa BaseEntity với cơ chế xóa mềm (deletedAt). Các giá trị tiền được lưu dưới dạng số nguyên đơn vị nhỏ nhất (minor unit - đồng VND hoặc wei ETH), phân biệt bằng cờ enum LedgerMinorUnit, giúp thống nhất cách lưu trữ cho cả hai loại tiền tệ.

### *2.4.1. Mô hình quan hệ tổng quát (ERD rút gọn)*




Hình 2.3 - Sơ đồ thực thể - quan hệ (ERD) rút gọn của hệ thống AgriBid

Hình 2.3 minh họa quan hệ chính giữa các thực thể trọng tâm:

- users (1) - (1) wallets; users (1) - (N) addresses, product_batches (với vai trò farmer), bids, orders (với vai trò buyer).
- wallets (1) - (N) transactions.
- product_batches (1) - (1) listings HOẶC (1) - (1) auctions (loại trừ nhau - một lô chỉ thuộc một kênh).
- auctions (1) - (N) bids; auctions.on_chain_id tương ứng id phiên trên smart contract AgriBidEscrow.
- orders (1) - (N) order_items; mỗi order_item tham chiếu tới product_batches và (tuỳ kênh) listings hoặc auctions.

- product_batches.nft_token_id tương ứng tokenId trên smart contract AgriBidNFT.
- live_sessions (1) - (N) live_session_pins, live_session_messages; live_sessions.primary_listing tham chiếu listings.

### *2.4.2. Mô tả chi tiết các bảng dữ liệu chính*

Bảng 2.15 - Bảng dữ liệu users

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| userId | PK | Khóa chính |
| username, email | varchar (unique) | Định danh đăng nhập |
| passwordHash | varchar | Mật khẩu đã băm bcrypt |
| role | enum UserRole | ADMIN / FARMER / MERCHANT |
| walletAddress | varchar(42, unique) | Địa chỉ ví MetaMask |
| bank_* | varchar | Tên thụ hưởng, ngân hàng, số tài khoản, mã ngân hàng - phục vụ VietQR |
| kycStatus | enum | PENDING / VERIFIED / REJECTED + ảnh KYC, lý do từ chối |
| status | enum | ACTIVE / BLOCKED |

Bảng 2.16 - Bảng dữ liệu wallets và transactions

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| wallets.balance | int (VND) | Số dư khả dụng |
| wallets.frozenBalance | int (VND) | Số dư đang bị giữ (đặt giá đấu, đặt cọc...) |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| transactions.type | enum TransactionType | DEPOSIT / WITHDRAW / HOLD / RELEASE / PAYMENT |
| transactions.status | enum PaymentStatus | PENDING / SUCCESS / FAILED / CANCELLED |
| transactions.amountVnd | int (±) | Số tiền thay đổi (âm/dương) |
| transactions.externalOrderCode | varchar (unique) | Mã tham chiếu giao dịch ngoài (PayOS...) |
| transactions.onChainTxHash | varchar | Hash giao dịch on - chain liên quan (nếu có) |

Bảng 2.17 - Bảng dữ liệu product_batches

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| batchId | PK | Khóa chính |
| farmer | FK → users | Chủ lô hàng |
| product | FK → products | Loại nông sản |
| weightKg / quantityValue, quantityUnit | decimal / enum | Khối lượng và đơn vị (KG/ITEM) |
| harvestDate, techProcess, farmerDescription | date / text | Thông tin thu hoạch & quy trình kỹ thuật |
| status | enum BatchStatus | PENDING / READY_FOR_SALE / REJECTED / SOLD |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| nftTokenId | int (unique, nullable) | Token ID NFT (gán khi Admin duyệt) |
| txHash, rejectionReason | varchar / text | Hash giao dịch mint NFT; lý do từ chối |

Bảng 2.18 - Bảng dữ liệu listings và auctions

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| listings.pricePerUnit, availableQuantity | decimal | Giá/đơn vị và số lượng còn bán |
| listings.paymentMode | enum PaymentMode | TRADITIONAL / BLOCKCHAIN |
| listings.status | enum ListingStatus | ACTIVE / SOLD / CANCELLED |
| auctions.auctionType | enum AuctionType | ENGLISH (chính) / DUTCH (dự phòng) |
| auctions.onChainId | int (unique, nullable) | ID phiên trên AgriBidEscrow |
| auctions.startingPriceMinor / stepPriceMinor / depositReqMinor / currentPriceMinor | bigint | Giá khởi điểm / bước giá / đặt cọc / giá hiện tại (đơn vị nhỏ nhất) |
| auctions.startTime, endTime | timestamp | Thời gian bắt đầu/kết thúc, dùng cho Bull job |
| auctions.status | enum AuctionStatus | WAITING / HAPPENING / ENDED / CANCELLED |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| auctions.onChainSyncStatus | enum AuctionChainSyncStatus | NOT_REQUIRED / PENDING / SYNCED / FAILED |

Bảng 2.19 - Bảng dữ liệu bids

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| bidId | PK (bigint) | Khóa chính |
| auction | FK → auctions | Phiên đấu giá |
| bidder | FK → users | Người đặt giá |
| bidMinorUnit | enum LedgerMinorUnit | VND / ETH_WEI |
| bidAmountMinor | bigint | Giá trị đặt giá (đơn vị nhỏ nhất) |
| bidTime | timestamp | Thời điểm đặt giá |
| (index/unique) | | Index (auction, bidAmountMinor); unique (auction, bidder, bidAmountMinor) |

Bảng 2.20 - Bảng dữ liệu orders và order_items

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| orders.buyer | FK → users | Người mua |
| orders.finalAmountMinor / discountMinor | bigint | Tổng tiền sau giảm giá / số tiền giảm |
| orders.status | enum OrderStatus | PENDING_BANK_TRANSFER / BUYER_MARKED_TRANSFERRED / |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| | | CONFIRMED_PAID / PENDING_CONFIRM / SHIPPING / COMPLETED / CANCELLED |
| orders.paymentChannel | enum OrderPaymentChannel | INTERNAL_WALLET / PAYOS / BANK_TRANSFER_VIETQR / BLOCKCHAIN |
| orders.externalOrderCode | varchar (unique) | Mã đơn dùng cho thanh toán/biên lai |
| order_items.order / listing / auction / productBatch | FK | Liên kết đơn hàng tới lô hàng và kênh bán (listing hoặc auction, nullable) |
| order_items.quantity, pricePerItemMinor | decimal / bigint | Số lượng và đơn giá tại thời điểm đặt hàng |
| order_bank_transfer_receipts.status | enum BankTransferReceiptStatus | SUBMITTED / APPROVED / REJECTED, kèm ảnh biên lai và audit (IP, user - agent, traceId) |

Bảng 2.21 - Bảng dữ liệu live_sessions và các bảng vận hành khác

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| live_sessions.sessionType | enum LiveSessionType | LISTING (chính) / AUCTION (legacy) |
| live_sessions.status | enum LiveSessionStatus | SCHEDULED / LIVE / ENDED / FORCE_ENDED |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| live_sessions.provider, providerRoomId | varchar | Nhà cung cấp & ID phòng LiveKit |
| live_sessions.lastHeartbeatAt | timestamp | Thời điểm heartbeat gần nhất - dùng cho cơ chế FORCE_ENDED |
| live_session_pins | FK → live_sessions, listings | Sản phẩm được ghim trong phiên (tối đa 24) |
| payout_requests.status | enum PayoutRequestStatus | PENDING_APPROVAL / APPROVED / COMPLETED / REJECTED / CANCELLED |
| idempotency_records | scope, userId, idempotencyKey, requestHash, status, responsePayload | Lưu trạng thái xử lý theo idempotency - key (PENDING/COMPLETED) |
| blockchain_events / blockchain_sync_states | txHash, eventName... / lastBlock | Log sự kiện đã xử lý và con trỏ block của BlockchainListener (chống xử lý trùng) |

### *2.4.3. Các bảng dữ liệu bổ trợ*

Bên cạnh các bảng nghiệp vụ cốt lõi đã trình bày ở mục 2.4.2, hệ thống còn sử dụng một nhóm bảng bổ trợ phục vụ danh mục sản phẩm, địa chỉ giao hàng, giỏ hàng, đánh giá, voucher, biên lai chuyển khoản và cấu hình hệ thống. Các bảng này tuy không phải trung tâm của các luồng tài chính nhưng là điều kiện cần để các luồng chính (Listing, Order, Live Session) vận hành đầy đủ.

Bảng 2.22 - Bảng dữ liệu categories và products

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| categories.categoryId / parent | PK / FK → categories | Danh mục nông sản, hỗ trợ cây phân cấp (parent–children) để phân loại theo nhóm/loại |
| categories.name, slug | varchar | Tên hiển thị và định danh dùng cho URL/lọc tìm kiếm |
| products.productId | PK | Sản phẩm gốc (loại nông sản), ví dụ: "Xoài cát Hòa Lộc", "Cà phê Robusta" |
| products.category | FK → categories | Danh mục mà sản phẩm thuộc về |
| products.unit | enum (KG/ITEM) | Đơn vị tính mặc định của sản phẩm, dùng làm gợi ý khi Farmer tạo lô hàng |
| products.description, imageUrl | text / varchar | Mô tả chung và hình ảnh minh họa của loại sản phẩm |

Bảng 2.23 - Bảng dữ liệu addresses

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| addressId | PK | Khóa chính |
| user | FK → users | Chủ sở hữu địa chỉ |
| recipientName, phone | varchar | Tên và số điện thoại người nhận hàng |
| addressLine, ward, district, province | varchar | Chi tiết địa chỉ giao hàng theo đơn vị hành chính |
| isDefault | boolean | Đánh dấu địa chỉ mặc định - là điều kiện tiên quyết để Merchant đặt giá đấu giá hoặc đặt hàng (UC - 01, UC - 03) |

Bảng 2.24 - Bảng dữ liệu batch_media

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| mediaId | PK | Khóa chính |
| batch | FK → product_batches | Lô hàng mà media minh chứng thuộc về |
| mediaType | enum (IMAGE/VIDEO) | Loại tệp minh chứng |
| url | varchar | Đường dẫn lưu trữ tệp (ảnh/video thu hoạch, quy trình canh tác) |
| sortOrder | int | Thứ tự hiển thị trong gallery của trang chi tiết lô hàng |

Bảng 2.25 - Bảng dữ liệu carts và cart_items

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| carts.cartId | PK | Mỗi user (MERCHANT) có duy nhất một giỏ hàng (1–1 với users) |
| cart_items.cartItemId | PK | Khóa chính |
| cart_items.cart | FK → carts | Giỏ hàng chứa item |
| cart_items.listing | FK → listings | Listing được thêm vào giỏ (chỉ áp dụng cho kênh Listing, không áp dụng cho Auction) |
| cart_items.quantity | decimal | Số lượng người dùng muốn mua, được kiểm tra lại so với availableQuantity tại thời điểm checkout (UC - 03) |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| cart_items.selected | boolean | Đánh dấu item có được chọn để thanh toán trong lượt checkout hiện tại hay không |

Bảng 2.26 - Bảng dữ liệu vouchers và voucher_redemptions

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| vouchers.code | varchar (unique) | Mã giảm giá do Admin tạo, người mua nhập khi checkout |
| vouchers.discountType | enum (FIXED_AMOUNT / PERCENTAGE) | Hình thức giảm: số tiền cố định (VND) hoặc theo phần trăm |
| vouchers.discountValue | int | Giá trị giảm tương ứng với discountType |
| vouchers.maxDiscountAmount | int (nullable) | Trần giảm tối đa khi discountType = PERCENTAGE |
| vouchers.usageLimit, usedCount | int | Giới hạn tổng số lượt dùng và số lượt đã sử dụng |
| vouchers.startDate, endDate | timestamp | Khoảng thời gian hiệu lực của voucher |
| voucher_redemptions | FK → vouchers, orders, users | Lịch sử áp dụng voucher cho từng đơn hàng - dùng để chống áp dụng vượt quá usageLimit hoặc dùng lại trên cùng một đơn |

Bảng 2.27 - Bảng dữ liệu listing_reviews

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| reviewId | PK | Khóa chính |
| listing | FK → listings | Listing được đánh giá |
| order | FK → orders | Đơn hàng làm căn cứ cho phép đánh giá (chỉ khi order.status = COMPLETED) |
| reviewer | FK → users | Người mua viết đánh giá |
| rating | int (1–5) | Điểm đánh giá |
| comment, mediaUrls | text / jsonb | Nội dung nhận xét và danh sách hình ảnh đính kèm (nếu có) |

Bảng 2.28 - Bảng dữ liệu platform_fee_statements và system_settings

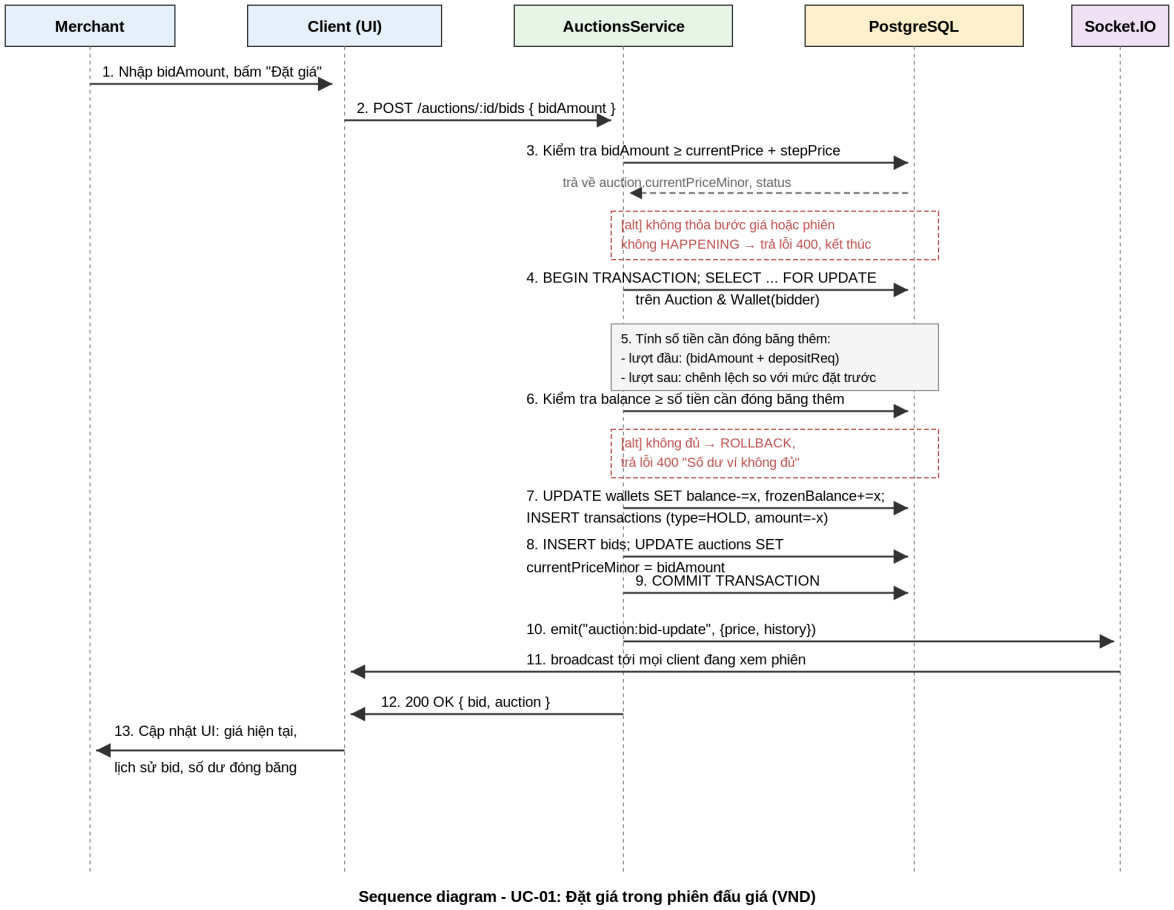
| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| platform_fee_statements.farmer | FK → users | Farmer được tính phí trong kỳ |
| platform_fee_statements.period | varchar (YYYY - MM) | Kỳ tính phí (theo tháng) |
| platform_fee_statements.totalRevenueMinor | bigint | Tổng doanh thu từ các Order có status = COMPLETED của Farmer trong kỳ |
| platform_fee_statements.feeRate, feeAmountMinor | decimal / bigint | Tỉ lệ phí áp dụng (mặc định 3%) và số tiền phí tương ứng |

| Trường | Kiểu dữ liệu | Mô tả |
|---|---|---|
| system_settings.key, value | varchar / jsonb | Cấu hình hệ thống dạng key–value (ví dụ: feeRate mặc định, ngưỡng số block xác nhận, thời gian timeout heartbeat...) |

Bảng order_bank_transfer_receipts đã được giới thiệu ở Bảng 2.20 (gắn với orders); bảng idempotency_records, blockchain_events và blockchain_sync_states đã được mô tả ở Bảng 2.21. Như vậy, tổng thể lược đồ CSDL của AgriBid gồm khoảng 30 bảng (entity), được tổ chức thành bốn miền nghiệp vụ như đã trình bày ở đầu mục 2.4, đảm bảo mỗi miền có thể được phát triển, kiểm thử và mở rộng độc lập trong khi vẫn duy trì tính toàn vẹn tham chiếu (referential integrity) thông qua khóa ngoại.

## 2.5. Thiết kế các luồng nghiệp vụ trọng yếu

### *2.5.1. Luồng đấu giá English Auction (VND)*




**Sequence diagram - UC-01: Đặt giá trong phiên đấu giá (VND)**

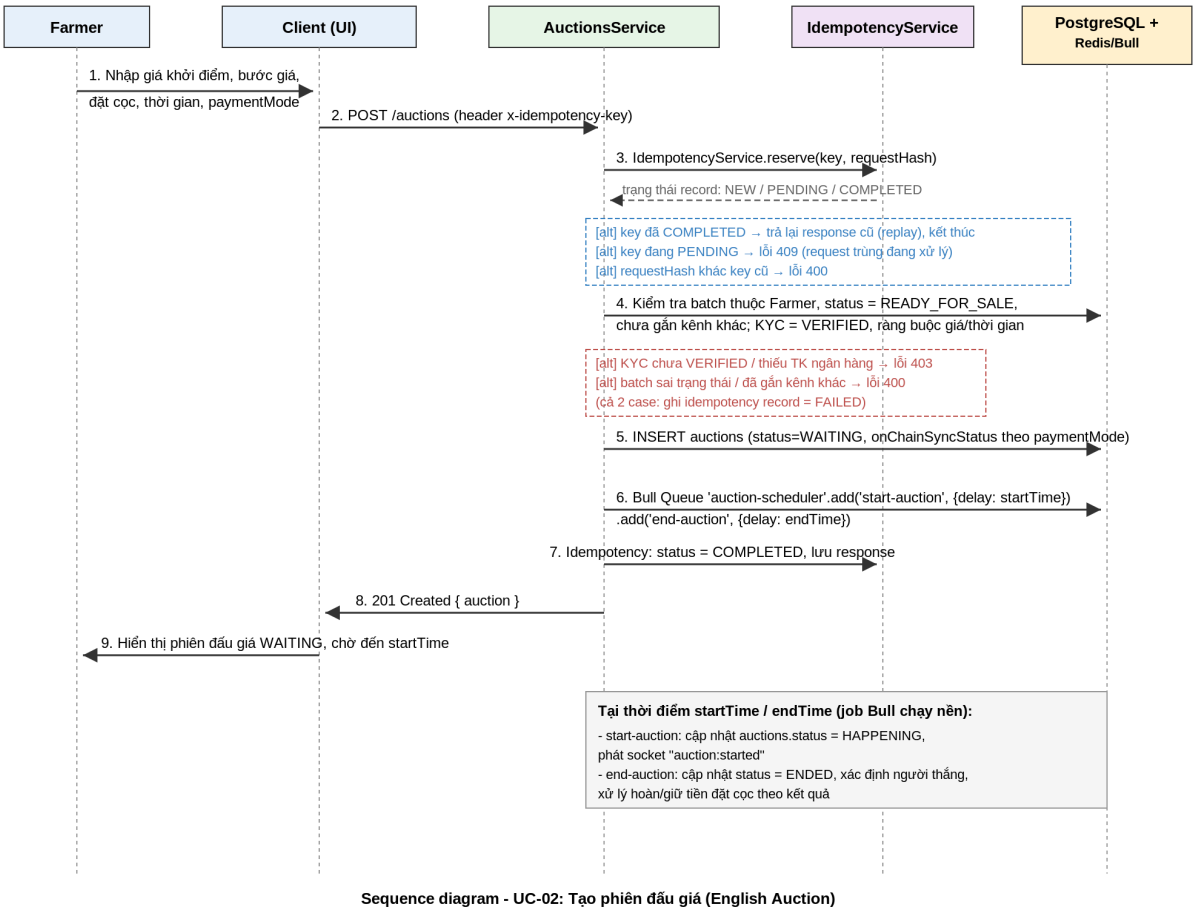
Hình 2.4 - Sơ đồ tuần tự UC - 01: Đặt giá trong phiên đấu giá (VND)

Hình 2.4 minh họa các bước chính trong code (BidsService, AuctionsService, AuctionProcessor):

- Farmer tạo phiên đấu giá (kiểm tra KYC đã VERIFIED và - nếu thanh toán TRADITIONAL - đã có thông tin ngân hàng để nhận VietQR); phiên ở trạng thái WAITING.
- Bull Queue (queue 'auction - scheduler') đến đúng startTime sẽ chạy job start - auction → chuyển trạng thái sang HAPPENING.
- Khi Merchant đặt giá: hệ thống mở transaction, khóa pessimistic_write trên dòng Auction và Wallet liên quan; kiểm tra phiên đang HAPPENING, đúng loại ENGLISH, chưa hết hạn, không tự đấu giá của chính mình, có địa chỉ giao hàng

mặc định, và giá đặt ≥ giá hiện tại + bước giá (hoặc ≥ giá khởi điểm nếu chưa có ai đặt).

- Đóng băng số dư ví: nếu là lượt đặt đầu tiên của người dùng trong phiên, đóng băng (giá đặt + tiền đặt cọc); nếu đã từng đặt trước đó, chỉ đóng băng thêm phần chênh lệch so với mức đặt cao nhất trước đó của chính họ. Mỗi lần đóng băng được ghi một bản ghi Transaction loại HOLD (giá trị âm).
- Lưu Bid mới, cập nhật currentPriceMinor của phiên, phát sự kiện realtime tới các client đang xem qua AuctionGateway (Socket.IO).
- Đến endTime, Bull job end - auction xác định bid cao nhất là người thắng, tạo bản ghi AuctionSettlementJob, gọi syncWinnerFromWinningBid và OrdersService.createFromAuction để tạo đơn hàng từ phiên đấu giá; số dư đã giữ của người thắng được dùng để thanh toán, số dư đã giữ của những người thua được giải phóng (RELEASE) toàn bộ.
- Hệ thống broadcast sự kiện kết thúc phiên (broadcastAuctionEnded) tới tất cả client đang theo dõi.




Hình 2.5 - Sơ đồ tuần tự UC - 02: Tạo phiên đấu giá (English Auction)

Hình 2.5 mô tả chi tiết bước tạo phiên đấu giá (UC - 02) ở đầu luồng trên, bao gồm kiểm tra idempotency key, xác thực lô hàng và lập lịch hai job Bull (start - auction, end - auction).

### *2.5.2. Luồng đấu giá Blockchain (ETH)*

- Khi listing/auction có paymentMode = BLOCKCHAIN, việc đặt giá thực hiện trực tiếp trên smart contract AgriBidEscrow thông qua ví MetaMask của người dùng (không qua API đặt giá VND).
- BlockchainListener poll log định kỳ (mặc định 15 giây, theo từng khoảng block, chờ đủ số block xác nhận - mặc định 6 ở production), khi phát hiện sự kiện BidPlaced sẽ gọi BidsService.handleOnChainBid để ánh xạ địa chỉ ví (so khớp walletAddress, không phân biệt hoa/thường) và cập nhật currentPriceMinor (dùng GREATEST để tránh ghi đè bởi sự kiện đến muộn).

- Khi kết thúc phiên (job end - auction), nếu paymentMode = BLOCKCHAIN và trạng thái on - chain vẫn OPEN, Backend gọi BlockchainService.settleOnChainAuction để chốt phiên trên smart contract, sau đó transferNFT chuyển NFT của lô hàng từ ví Farmer sang ví người thắng (có kiểm tra chủ sở hữu hiện tại để tránh gọi lại gây revert nếu đã chuyển).
- Trạng thái đồng bộ được lưu vào auctions.onChainSyncStatus (PENDING/SYNCED/FAILED); hệ thống cung cấp endpoint cho Admin gọi lại retry - chain - sync khi đồng bộ thất bại. Phiên BLOCKCHAIN không có lượt đặt giá nào sẽ tự đánh dấu SYNCED vì không có ETH bị khóa.
- Người thắng và người thua tự gọi hàm rút tiền (withdrawPayout/withdrawRefund) trên smart contract theo mô hình pull - payment, hạn chế rủi ro reentrancy.

### *2.5.3. Luồng ví & giao dịch (Wallet/Transaction)*

- Mỗi thao tác làm thay đổi số dư ví (đặt giá, hoàn tiền, thanh toán, nạp/rút) đều mở transaction PostgreSQL, khóa pessimistic_write trên bản ghi Wallet, cập nhật balance/frozenBalance và ghi đồng thời một bản ghi Transaction tương ứng (DEPOSIT/WITHDRAW/HOLD/RELEASE/PAYMENT) - đảm bảo sổ cái luôn khớp với số dư hiện tại.
- API getMyWalletActivity trả về lịch sử giao dịch có phân trang; API trace/:externalOrderCode cho phép tra cứu một giao dịch theo mã đơn ngoài (phục vụ đối soát PayOS/VietQR).
- Trong môi trường phát triển, depositMock cho phép nạp tiền giả lập; adminDeposit cho phép Admin điều chỉnh số dư thủ công, đều được ghi log qua Transaction để đảm bảo có thể truy vết.

### *2.5.4. Luồng thanh toán đơn hàng*

Đơn hàng có thể thanh toán theo một trong ba kênh (OrderPaymentChannel):

- INTERNAL_WALLET: trừ trực tiếp số dư ví tại thời điểm tạo đơn (atomic cùng việc tạo Order/OrderItem).

- PAYOS: tạo link thanh toán (POST /payments/payos/create - link), người dùng quét QR/thanh toán qua app ngân hàng; PayOS gọi webhook công khai (có xác thực chữ ký) để xác nhận, cập nhật paymentStatus của đơn.
- BANK_TRANSFER_VIETQR (VietQR Level 2): hệ thống sinh thông tin chuyển khoản tới tài khoản ngân hàng của Farmer với nội dung chuẩn hóa (AGRIBID ORDER_<orderId>); đơn ở trạng thái PENDING_BANK_TRANSFER → buyer tải lên biên lai (BUYER_MARKED_TRANSFERRED) → Farmer/Admin xác nhận biên lai (lưu kèm IP, user - agent, traceId trong order_bank_transfer_receipts) → đơn chuyển CONFIRMED_PAID.

Việc checkout từ giỏ hàng (đa người bán) được xử lý theo nguyên tắc tất - cả - hoặc - không (all - or - nothing): mỗi Farmer trong giỏ sẽ tương ứng với một Order riêng; nếu một phần thất bại (ví dụ không đủ số dư ví), toàn bộ thao tác được rollback. Một job nền (order - pending - payment - scheduler) định kỳ tự động hủy các đơn quá hạn thanh toán.

### *2.5.5. Luồng livestream bán hàng*

- Farmer gọi API listing/start để khởi tạo phiên (LiveSessionType.LISTING), backend tạo phòng LiveKit và trả token host (livekit - token.service.ts).
- Farmer ghim (pin) các listing muốn giới thiệu (tối đa 24 pin đang hoạt động - ràng buộc trong live - session - pins.policy.ts); người xem nhận danh sách pin theo thời gian thực.
- Người xem tham gia phòng live (token viewer), xem danh sách sản phẩm đang ghim, có thể mua ngay (chuyển hướng VietQR Level 2) hoặc thêm vào giỏ mà không rời màn hình live; chat realtime qua WebSocket.
- Client gửi heartbeat (PATCH :id/heartbeat) mỗi 30 giây để cập nhật lastHeartbeatAt; bộ quét nội bộ của LiveSessionsService chạy mỗi 30 giây, nếu một phiên đang LIVE không nhận heartbeat quá 90 giây sẽ tự động chuyển trạng thái sang FORCE_ENDED và thông báo cho người xem.
- Farmer chủ động kết thúc phiên qua API end, chuyển trạng thái sang ENDED.

### *2.5.6. Cơ chế Idempotency Key*

- Client gửi header x - idempotency - key kèm yêu cầu tạo listing/tạo phiên đấu giá. IdempotencyService.reserve(scope, userId, key, requestHash) kiểm tra theo bộ khóa (scope, userId, idempotencyKey).
- Nếu đã tồn tại bản ghi với requestHash khác → trả lỗi 400 (key bị tái sử dụng cho payload khác); nếu bản ghi đang ở trạng thái PENDING → trả lỗi 409 (request trùng đang được xử lý); nếu COMPLETED → trả lại nguyên response đã lưu (replay), không tạo dữ liệu mới.
- Nếu chưa tồn tại, hệ thống tạo bản ghi PENDING (xử lý cả tình huống hai request song song cùng tạo bản ghi nhờ unique constraint); sau khi xử lý xong gọi complete() để lưu response và chuyển COMPLETED, hoặc release() để xóa bản ghi nếu cần rollback.

### *2.5.7. Cơ chế đồng bộ Blockchain (BlockchainListener)*

- BlockchainListener khởi chạy khi module được load (OnModuleInit), poll log on - chain theo chu kỳ cấu hình (mặc định 15 giây), chia nhỏ phạm vi quét theo block (chunk 100–2000 block) và chỉ xử lý các block đã đạt đủ số xác nhận (1 ở môi trường dev, 6 ở production) để tránh xử lý sự kiện trên nhánh bị đảo (reorg).
- Vị trí quét gần nhất được lưu trong blockchain_sync_states; mỗi sự kiện đã xử lý được ghi vào blockchain_events để đảm bảo không xử lý trùng khi worker khởi động lại.
- Sự kiện BidPlaced → đồng bộ vào bids/auctions; AuctionEnded/ AuctionCancelled → cập nhật trạng thái auctions tương ứng. Hệ thống cung cấp lệnh replayRange() để Admin/Dev có thể quét lại thủ công một khoảng block khi cần.

### *2.5.8. Sơ đồ tuần tự một số luồng tiêu biểu*

Phần này mô tả bằng lời trình tự tương tác giữa các thành phần (Client, Backend, CSDL, dịch vụ ngoài) cho ba luồng nghiệp vụ tiêu biểu. Luồng (1) đã được minh họa chi tiết bằng sơ đồ tuần tự ở Hình 2.4 (đặt giá) và Hình 2.5 (tạo phiên đấu giá - bước khởi đầu của luồng); hai luồng còn lại được mô tả bằng lời:

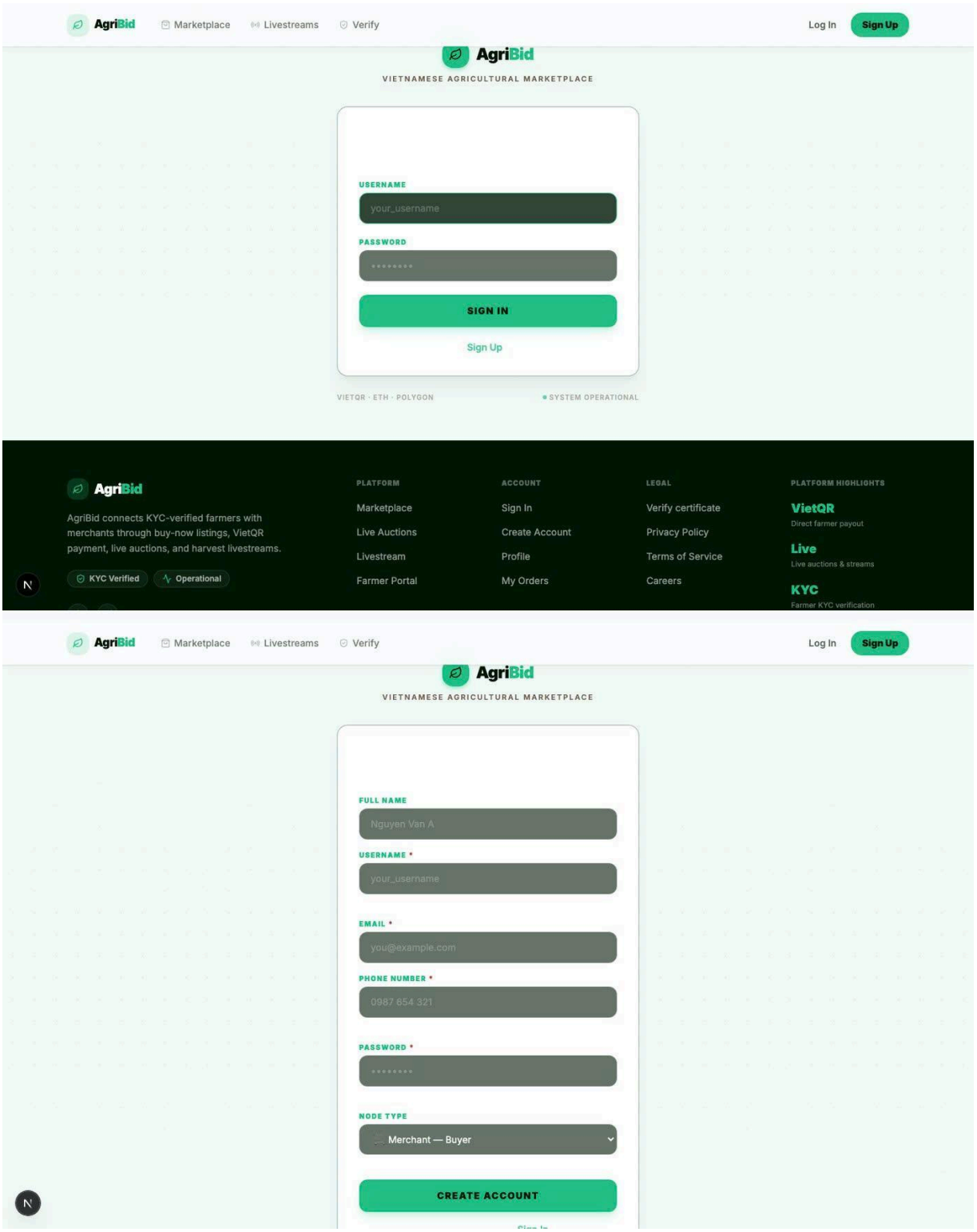
(1) Luồng đặt giá đấu giá VND: Client → Backend: POST /auctions/:id/bids (kèm x - idempotency - key) → Backend: IdempotencyService kiểm tra khóa → Backend: mở transaction, SELECT ... FOR UPDATE bản ghi auctions để khóa pessimistic → Backend: kiểm tra số tiền đặt giá > currentPrice + minIncrement và availableBalance của ví đủ → Backend: ghi bản ghi bids, cập nhật currentPrice/currentWinnerId của auctions, ghi transactions loại HOLD để đóng băng phần chênh lệch so với bid trước đó (nếu có, RELEASE phần đã hold của người giữ giá cũ) → Backend: commit transaction → Backend: AuctionsGateway phát sự kiện WebSocket "bid:new" tới phòng tương ứng → Client (mọi người xem): nhận sự kiện, cập nhật giao diện realtime → Backend: trả response 201 cho Client gửi yêu cầu.

(2) Luồng kết thúc phiên đấu giá tự động: Bull Queue (auction - scheduler) → Backend: tới thời điểm endTime, worker nhận job "end - auction" → Backend: mở transaction, đọc bản ghi auctions và bid thắng cuộc → Backend: nếu có người thắng, chuyển trạng thái auctions sang ENDED, ghi transactions loại RELEASE cho các bid thua (hoàn tiền đã hold) và chuẩn bị PAYMENT cho người thắng → Backend: tạo order tương ứng cho người thắng ở trạng thái chờ thanh toán/đã thanh toán bằng số dư đã hold → Backend: commit transaction → Backend: AuctionsGateway phát sự kiện "auction:ended" → Client: cập nhật giao diện hiển thị kết quả đấu giá.

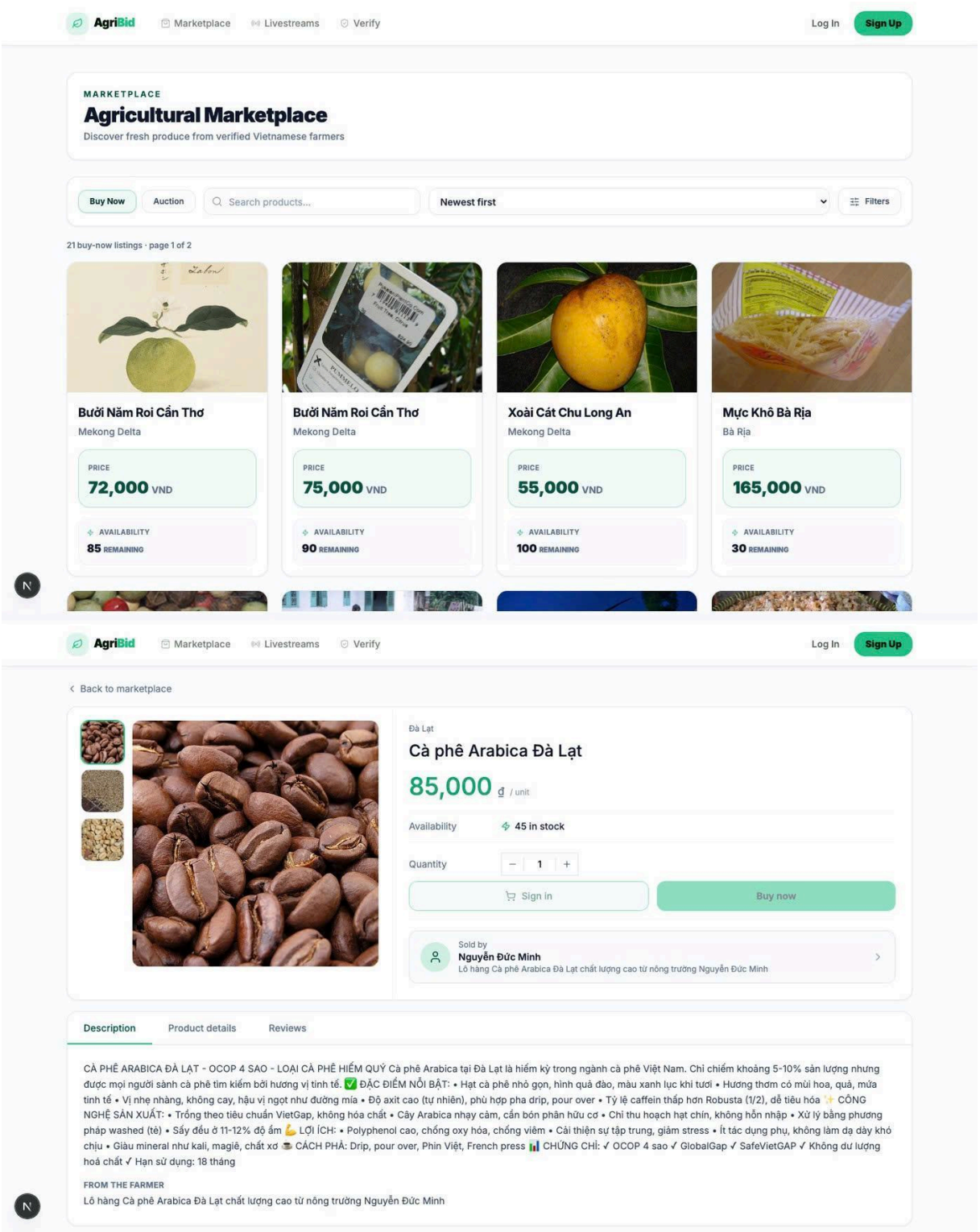
(3) Luồng webhook thanh toán PayOS: PayOS → Backend: POST /payments/payos/webhook (kèm chữ ký) → Backend: PaymentsController xác thực chữ ký bằng secret key → Backend: tra cứu orderCode trong transactions để kiểm tra trùng lặp (idempotency theo orderCode) → Backend: nếu hợp lệ và chưa xử lý, mở transaction cập nhật trạng thái order liên quan sang PAID, ghi transactions loại PAYMENT/DEPOSIT tương ứng, cập nhật balance của ví → Backend: commit transaction → Backend: trả HTTP 200 cho PayOS để xác nhận đã nhận webhook (PayOS sẽ retry nếu không nhận được 200).

### 2.6. Thiết kế giao diện tổng quan

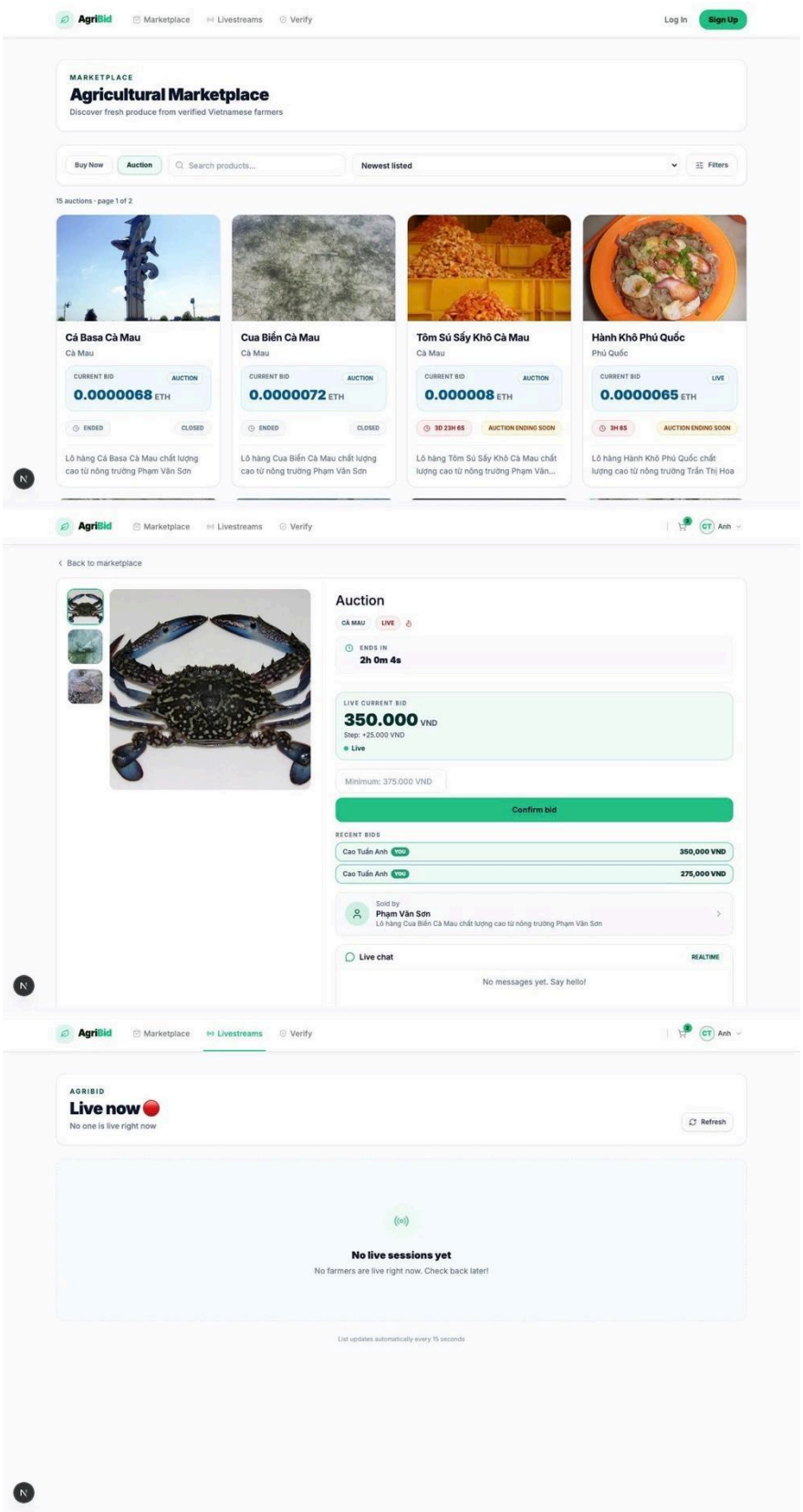
Các hình từ Hình 2.6 đến Hình 2.16 sẽ trình bày một số giao diện tiêu biểu của hệ thống, bao gồm giao diện Marketplace, giao diện đấu giá và giao diện quản trị. Các nhóm màn hình chính của hệ thống được mô tả như sau:



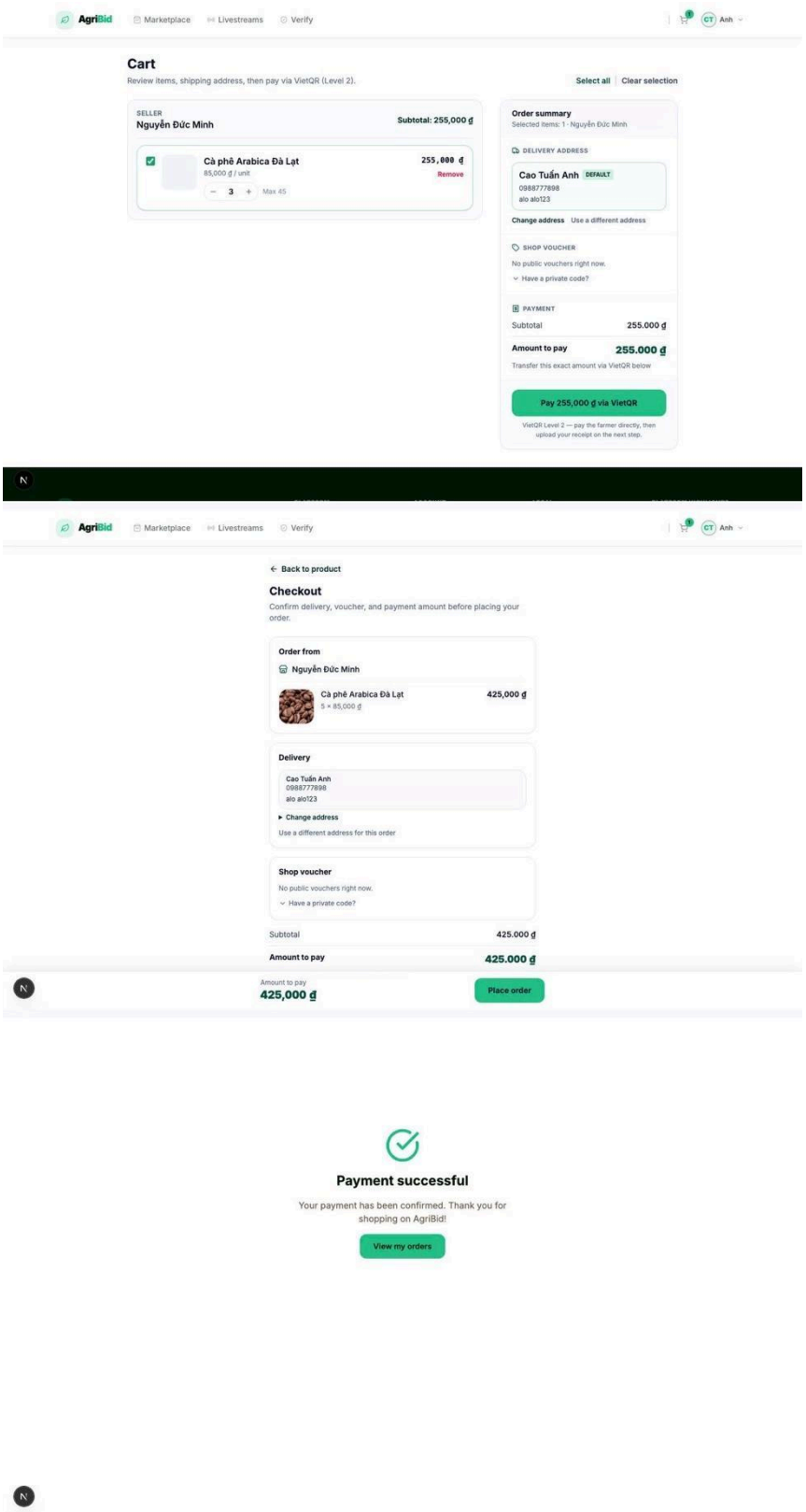

Hình 2.6 – Đăng nhập/Đăng ký (điểm bắt đầu mọi luồng)



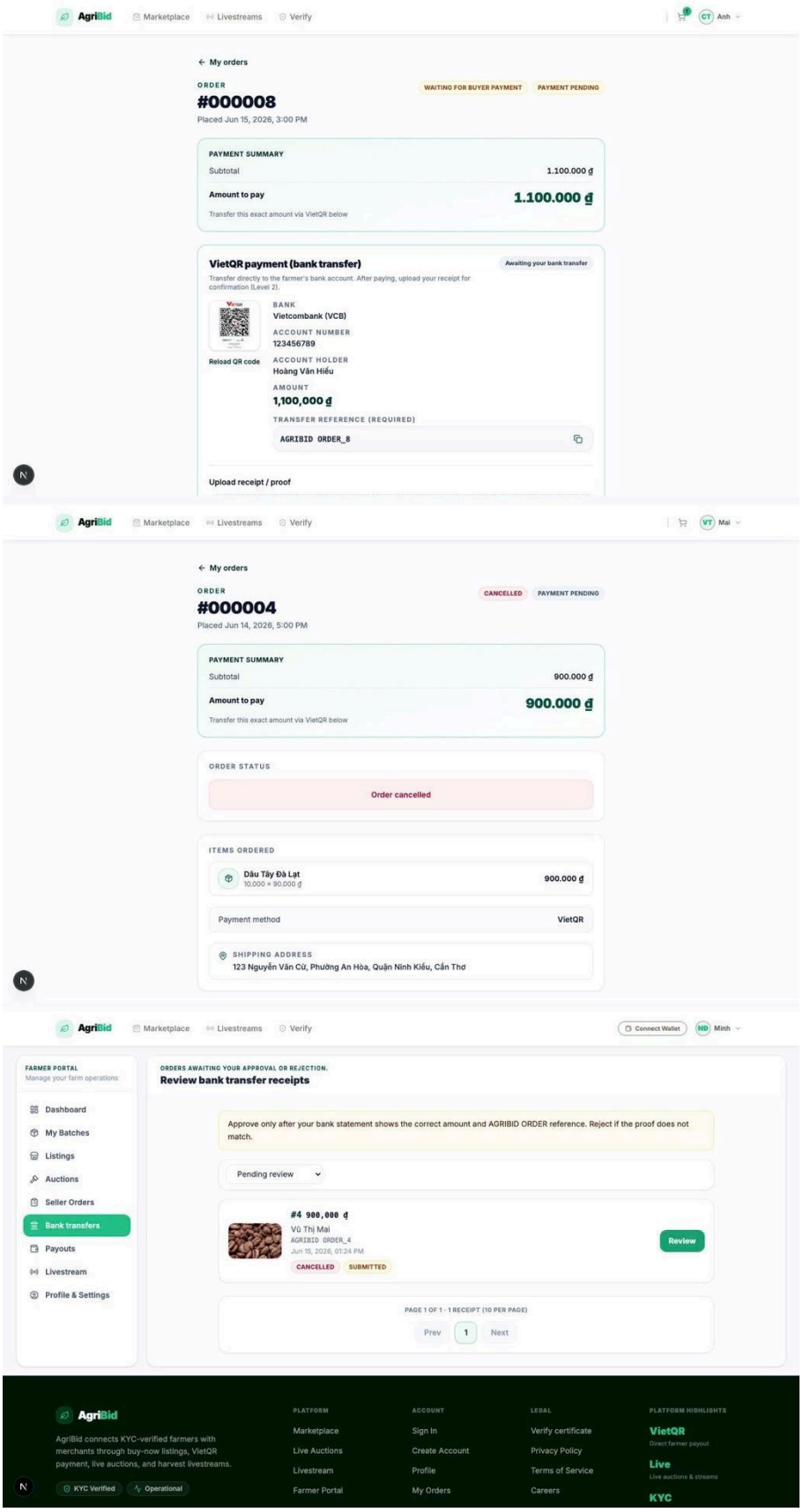

Hình 2.7 – Marketplace & chi tiết sản phẩm/lô hàng (khám phá sản phẩm)



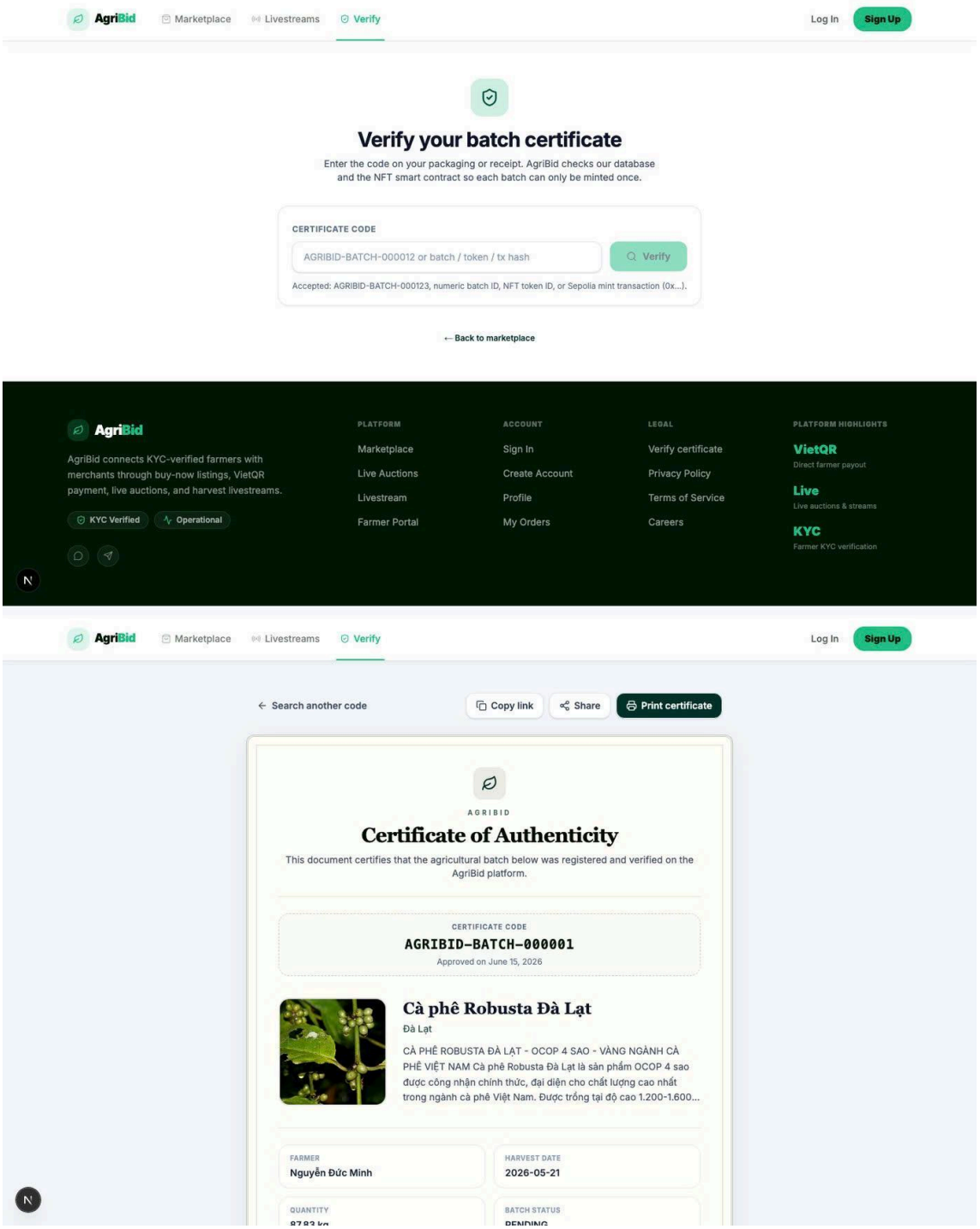

Hình 2.8 – Đấu giá realtime & Livestream (tương tác mua đặc thù)



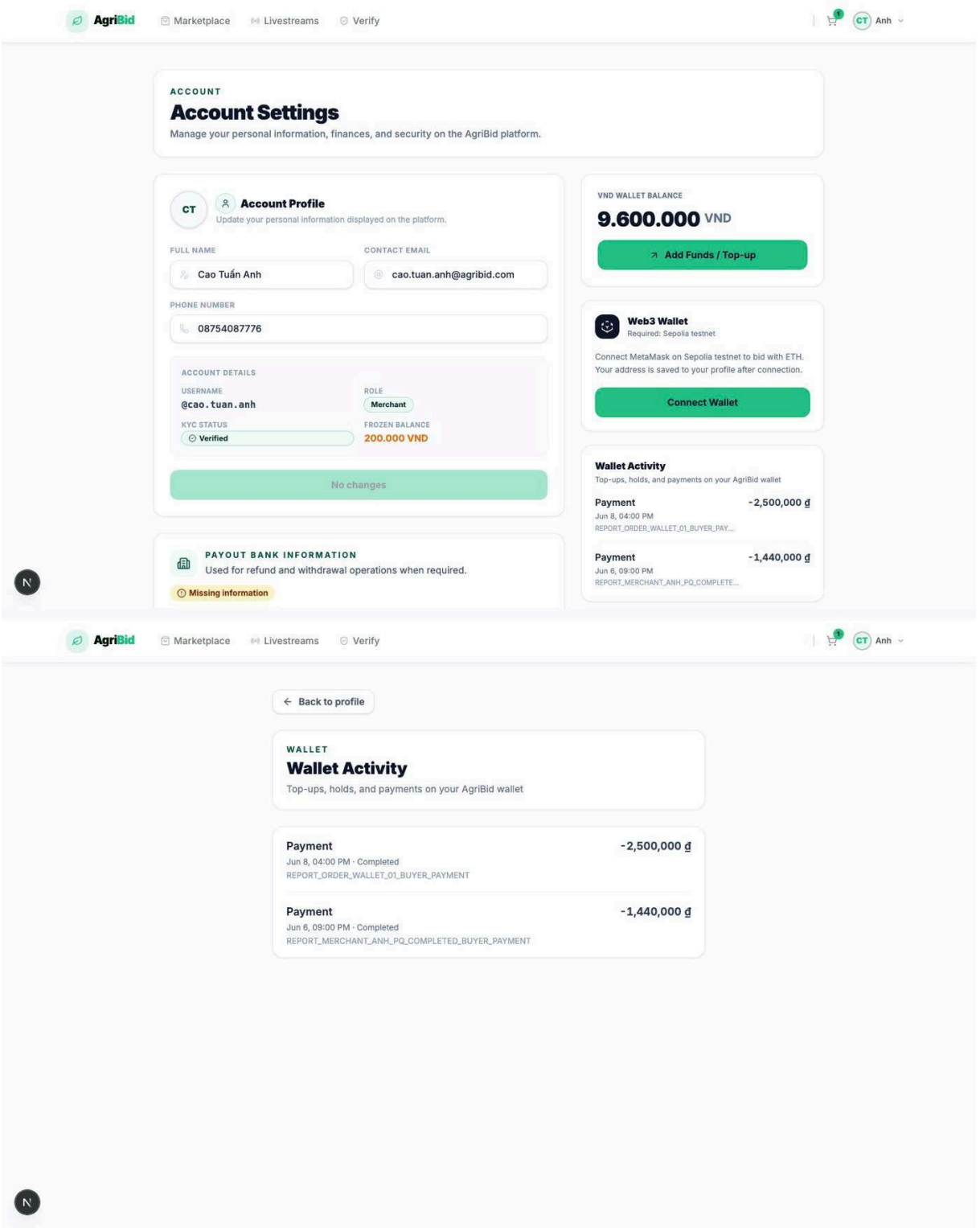

Hình 2.9 – Giỏ hàng & Thanh toán (checkout, PayOS)



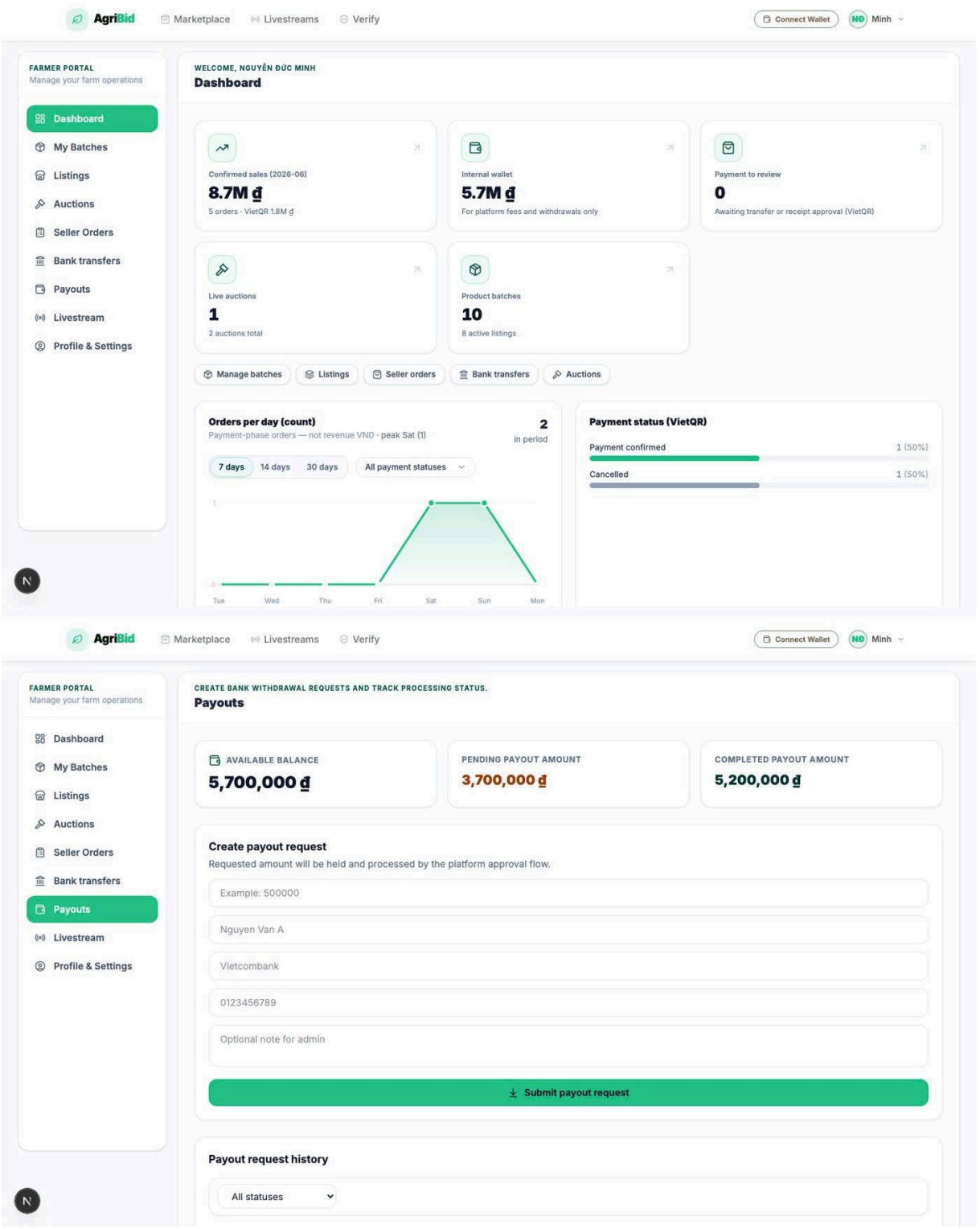

Hình 2.10 – Xác nhận chuyển khoản (upload + duyệt biên lai CK)



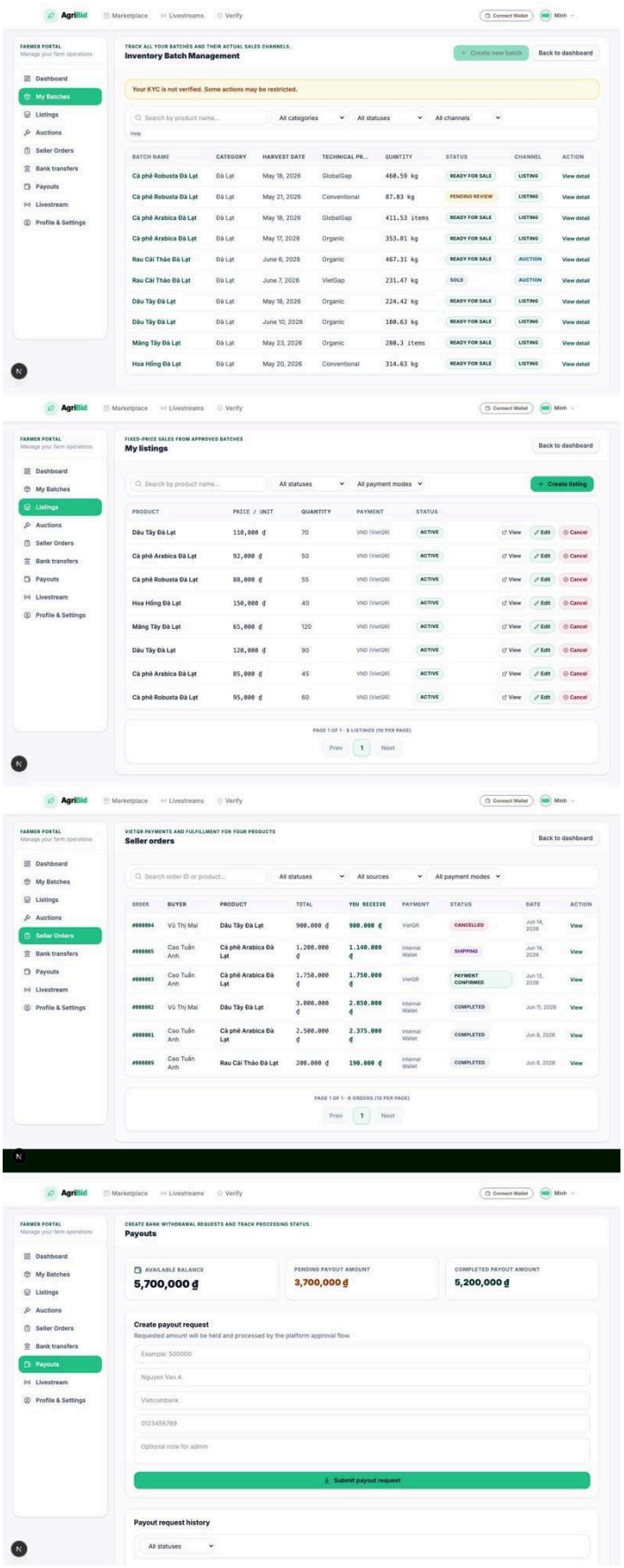

Hình 2.11 – Truy xuất nguồn gốc & Blockchain/NFT (sau khi mua, xem nguồn gốc sản phẩm)



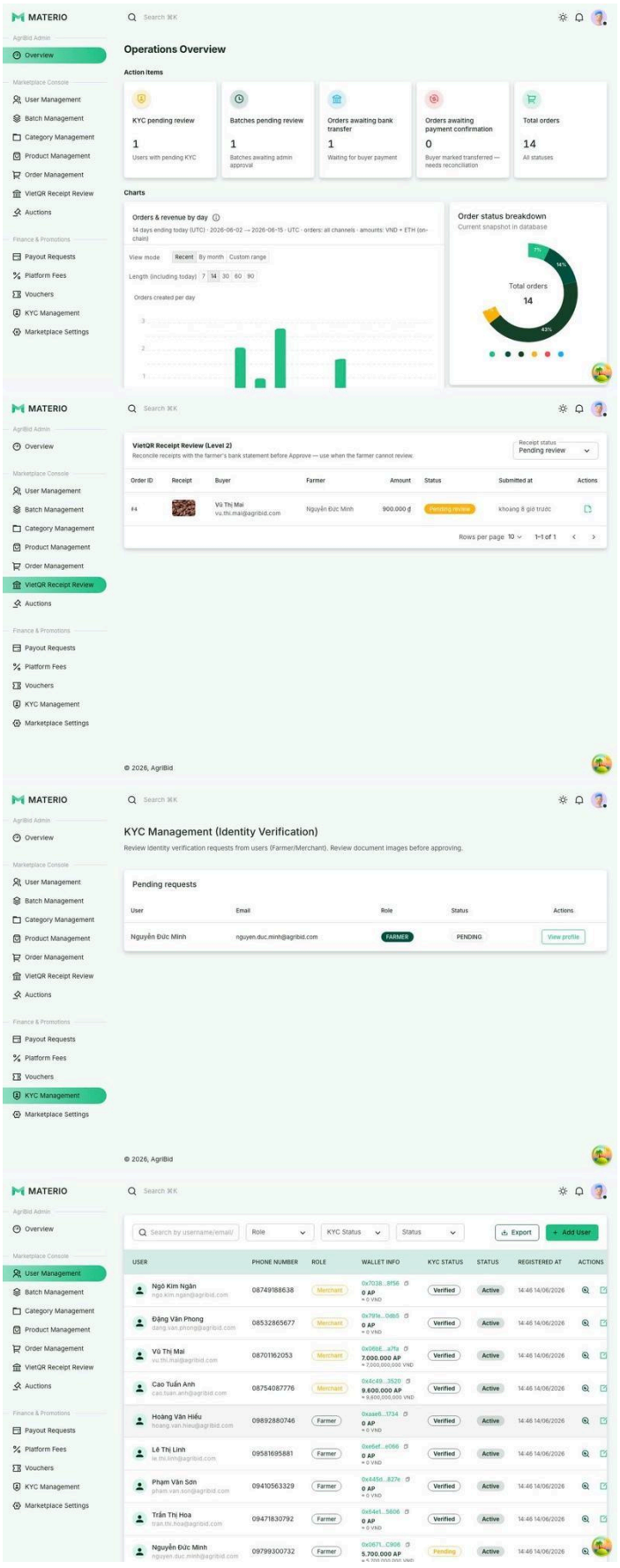

Hình 2.12 – Hồ sơ & lịch sử ví (wallet, giao dịch on-chain)




Hình 2.13 – Phân chia doanh thu/% (góc nhìn farmer nhận tiền)




Hình 2.14 – Quản lý dành cho Farmer (lô hàng, listing, đơn bán, rút tiền)




Hình 2.15 – Admin Dashboard (góc nhìn quản trị, cuối luồng)

Bảng 2.29 - Nhóm màn hình chính của hệ thống

| **Ứng dụng / Khu vực** | **Các màn hình chính** |
|---|---|
| Client - (shop) [Merchant] | Marketplace, chi tiết sản phẩm/lô hàng, danh sách & chi tiết phiên đấu giá, giỏ hàng, checkout (mua ngay & theo lô), theo dõi đơn hàng, xem livestream, trang truy xuất nguồn gốc (verify), hồ sơ & lịch sử ví. |
| Client - (farmer) [Farmer] | Quản lý lô hàng, quản lý listing, quản lý phiên đấu giá, livestream, quản lý đơn bán, quản lý biên lai chuyển khoản, yêu cầu rút tiền, hồ sơ & lịch sử ví. |
| Client - auth/payment | Đăng nhập, đăng ký, quên mật khẩu, trang kết quả thanh toán PayOS (success/cancel). |
| Admin - dashboard | Tổng quan số liệu, quản lý người dùng & KYC, quản lý lô hàng/sản phẩm/danh mục, giám sát đấu giá, quản lý đơn hàng, payout, phí nền tảng, voucher, cấu hình hệ thống. |

### *2.6.1. Nhóm màn hình Marketplace & chi tiết sản phẩm (Client - Merchant)*

Trang Marketplace (trang chủ mua sắm) sử dụng bố cục lưới (grid layout) responsive, gồm thanh tìm kiếm/lọc theo danh mục ở phía trên, thanh điều hướng (navbar) hiển thị trạng thái đăng nhập và số dư ví rút gọn, và khu vực lưới sản phẩm hiển thị card cho từng listing/lô hàng (ảnh đại diện, tên sản phẩm, giá, trạng thái - đang bán/đang đấu giá/đang livestream). Bộ lọc bên trái (sidebar filter) cho phép lọc theo danh mục (categories), khoảng giá, khu vực địa lý của nông trại, và loại kênh giao dịch (Listing/Auction/Live).

Trang chi tiết sản phẩm/lô hàng hiển thị thông tin đầy đủ về batch (mã lô, ngày thu hoạch, vùng trồng, chứng nhận VietGAP/OCOP nếu có), ảnh/video từ batch_media, thông tin người bán (farmer profile), giá và số lượng còn lại. Nếu batch đã được mint NFT truy xuất nguồn gốc, trang hiển thị thêm khối "Truy xuất nguồn gốc" với liên kết đến trang verify công khai và mã token NFT/giao dịch on - chain. Các nút hành động chính gồm "Thêm vào giỏ", "Mua ngay" (chuyển sang luồng

checkout), và nếu sản phẩm thuộc một phiên đấu giá đang diễn ra thì hiển thị thêm khối "Tham gia đấu giá" với giá hiện tại, thời gian còn lại (đếm ngược realtime qua WebSocket) và biểu mẫu đặt giá.

### *2.6.2. Nhóm màn hình Đấu giá (Auction)*

Trang danh sách phiên đấu giá hiển thị các thẻ (card) phiên đấu giá đang mở, sắp diễn ra và đã kết thúc, mỗi thẻ gồm ảnh sản phẩm, giá khởi điểm/giá hiện tại, số lượt đặt giá và thời gian còn lại. Trang chi tiết phiên đấu giá là màn hình quan trọng nhất của kênh Auction, bố cục hai cột: cột trái hiển thị ảnh/video sản phẩm và mô tả lô hàng; cột phải hiển thị khối thông tin đấu giá thời gian thực gồm giá hiện tại (currentPrice), bước giá tối thiểu (minIncrement), thời gian kết thúc (đồng hồ đếm ngược cập nhật qua WebSocket), lịch sử đặt giá (danh sách bid gần nhất kèm thời gian) và biểu mẫu đặt giá mới.

Biểu mẫu đặt giá kiểm tra ràng buộc số tiền tối thiểu ngay trên giao diện trước khi gửi yêu cầu (client - side validation), đồng thời hiển thị số dư khả dụng (availableBalance = balance − frozenBalance) của ví VND để người dùng biết trước liệu có đủ số dư để đặt giá hay không. Sau khi đặt giá thành công, giao diện cập nhật ngay giá hiện tại và danh sách lịch sử đặt giá cho tất cả người xem đang mở trang nhờ sự kiện WebSocket phát từ AuctionsGateway.

### *2.6.3. Nhóm màn hình Giỏ hàng & Thanh toán (Cart & Checkout)*

Trang giỏ hàng nhóm các mục theo người bán (farmer) vì mỗi đơn hàng (order) chỉ thuộc về một người bán; mỗi nhóm hiển thị danh sách sản phẩm, số lượng (có thể chỉnh sửa), đơn giá, thành tiền theo nhóm và checkbox chọn để thanh toán (cho phép thanh toán một phần giỏ hàng). Cuối trang là bảng tổng kết: tổng tiền hàng, mã voucher (nếu áp dụng), phí vận chuyển ước tính và tổng thanh toán.

Trang checkout hiển thị lại danh sách đơn hàng sẽ được tạo (mỗi người bán một đơn), biểu mẫu chọn/nhập địa chỉ giao hàng (lấy từ bảng addresses hoặc nhập mới), lựa chọn phương thức thanh toán (ví VND nội bộ hoặc PayOS/VietQR), và nút xác nhận đặt hàng có gắn x - idempotency - key để tránh tạo đơn trùng khi người dùng bấm nhiều lần hoặc mất kết nối tạm thời. Sau khi xác nhận, nếu chọn PayOS, giao diện

chuyển hướng sang trang thanh toán VietQR của PayOS và quay lại trang kết quả (success/cancel) của hệ thống sau khi hoàn tất.

### *2.6.4. Nhóm màn hình quản lý dành cho Farmer*

Trang quản lý lô hàng (product batches) hiển thị danh sách các lô đã khai báo kèm trạng thái duyệt (PENDING/READY_FOR_SALE/REJECTED/SOLD) và trạng thái mint NFT. Biểu mẫu tạo/sửa lô hàng cho phép nhập thông tin sản phẩm, số lượng, ngày thu hoạch, vùng trồng và tải lên hình ảnh/video (lưu vào batch_media). Trang quản lý listing và quản lý phiên đấu giá tương tự, cho phép farmer tạo, chỉnh sửa (khi chưa có người mua/đặt giá) và theo dõi trạng thái của từng listing/phiên.

Trang livestream dành cho farmer (host) tích hợp SDK LiveKit để phát video trực tiếp, kèm bảng điều khiển ghim sản phẩm (pin/unpin tối đa 24 listing), khung chat realtime với người xem, và chỉ báo trạng thái kết nối/heartbeat. Trang quản lý đơn bán hiển thị danh sách đơn hàng phân theo các trạng thái PENDING_BANK_TRANSFER, CONFIRMED_PAID, SHIPPING, COMPLETED..., cho phép cập nhật trạng thái vận chuyển và tải lên biên lai chuyển khoản đối với các đơn thanh toán thủ công. Trang ví & lịch sử giao dịch hiển thị balance, frozenBalance, lịch sử các bản ghi transactions phân theo loại DEPOSIT, WITHDRAW, HOLD, RELEASE, PAYMENT và biểu mẫu yêu cầu rút tiền (withdrawal request).

### *2.6.5. Nhóm màn hình Quản trị (Admin Dashboard)*

Trang tổng quan (dashboard) hiển thị các chỉ số tổng hợp: tổng số người dùng, tổng giá trị giao dịch theo từng kênh (Listing/Auction/Live), số lượng đơn hàng theo trạng thái, biểu đồ doanh thu/phí nền tảng theo thời gian. Trang quản lý người dùng & KYC hiển thị danh sách người dùng kèm trạng thái xác minh (KYC), cho phép admin xem chi tiết hồ sơ (giấy tờ tải lên) và phê duyệt/từ chối với lý do.

Trang quản lý lô hàng/sản phẩm/danh mục cho phép admin duyệt các lô hàng do farmer khai báo, quản lý cây danh mục (categories) và thực hiện thao tác mint NFT truy xuất nguồn gốc cho các lô đã duyệt. Trang giám sát đấu giá hiển thị danh sách các phiên đang diễn ra/đã kết thúc kèm khả năng can thiệp (hủy phiên trong trường hợp bất thường). Trang quản lý đơn hàng & payout cho phép admin xử lý các yêu cầu rút tiền của farmer, đối soát biên lai chuyển khoản. Trang phí nền tảng & cấu hình hệ

thống cho phép xem báo cáo phí theo kỳ (platform_fee_statements) và chỉnh các tham số cấu hình chung (system_settings) như tỉ lệ phí, ngưỡng heartbeat, số lượng pin tối đa.

### *2.6.6. Sơ đồ điều hướng (sitemap) chi tiết*

Bảng 2.30 dưới đây được tổng hợp trực tiếp từ cấu trúc thư mục app/ (Next.js App Router) của hai ứng dụng Frontend Client và Frontend Admin, cho thấy độ phủ thực tế của giao diện: Client có 30 trang (page.tsx), Admin có 17 trang.

Bảng 2.30 - Sitemap Frontend Client (30 trang) và Frontend Admin (17 trang)

| Khu vực | Số trang | Các route chính |
|---|---|---|
| Client - (shop) [Merchant] | 13 | / (trang chủ marketplace), /marketplace, /products/[id], /cart, /checkout/batch, /checkout/order/[orderId], /orders, /livestream, /livestream/[sessionId], /profile, /profile/wallet - activity, /stats, /verify |
| Client - (farmer) [Farmer] | 12 | /farmer (trang chủ), /farmer/batches, /farmer/batches/[batchId], /farmer/listings, /farmer/auctions, /farmer/livestream, /farmer/seller - orders, /farmer/seller - orders/[orderId], /farmer/payouts, /farmer/bank - transfers, /farmer/profile, /farmer/profile/wallet - activity |
| Client - auth | 3 | /auth/login, /auth/register, /auth/forgot - password |
| Client - payment | 2 | /payment/success, /payment/cancel (trang kết quả thanh toán PayOS) |
| Admin - (blank - layout - pages) | 4 | /login, /register, /forgot - password, /error |
| Admin - (dashboard) | 12 | / (tổng quan), /users, /kyc, /product - batches, /products, /categories, /auctions, /orders, /payouts, /platform - fees, /vouchers, /settings |

| Khu vực | Số trang | Các route chính |
|---|---|---|
| Admin - fallback | 1 | /[...not - found] (trang 404 tùy biến) |

So với bảng nhóm màn hình ở đầu mục 2.6, sitemap chi tiết này cho thấy rõ hơn việc tách biệt giữa hai trải nghiệm Farmer/Merchant trên cùng một ứng dụng Client (thông qua route group (farmer) và (shop) của Next.js, không làm thay đổi đường dẫn URL hiển thị) và 12 trang nghiệp vụ của Admin tương ứng với 12 nhóm chức năng quản trị đã liệt kê ở mục 2.1.2 và 2.2.3.

## 2.7. Thiết kế API hệ thống

Hệ thống cung cấp API theo kiến trúc RESTful, được tài liệu hóa tự động bằng Swagger (OpenAPI) thông qua các decorator của NestJS (@ApiTags, @ApiOperation, @ApiResponse...). Tất cả endpoint yêu cầu xác thực đều sử dụng JWT Bearer token gắn trong header Authorization, được kiểm tra bởi JwtAuthGuard kết hợp RolesGuard để phân quyền theo vai trò (FARMER, MERCHANT, ADMIN). Các bảng dưới đây liệt kê các endpoint chính theo từng module nghiệp vụ.

Danh sách dưới đây được tổng hợp trực tiếp từ các controller NestJS trong thư mục src/modules của AgriBid - Backend (27 controller, khoảng 27 module nghiệp vụ), đảm bảo phản ánh đúng các route, method HTTP và decorator phân quyền (@Roles, @UseGuards) hiện có trong mã nguồn.

### *2.7.1. Xác thực & hồ sơ cá nhân (auth, users/media)*

Bảng 2.31 - API xác thực và hồ sơ cá nhân

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /auth/register | Đăng ký tài khoản mới | Public |
| POST | /auth/login | Đăng nhập bằng username/password, trả về access token & refresh token | Public |
| POST | /auth/refresh | Làm mới access token bằng refresh token | Public |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /auth/logout | Đăng xuất, hủy refresh token hiện tại | Đã đăng nhập |
| GET | /auth/me | Lấy thông tin hồ sơ người dùng hiện tại | Đã đăng nhập |
| PATCH | /auth/me | Cập nhật hồ sơ cá nhân (họ tên, số điện thoại, ...) | Đã đăng nhập |
| PATCH | /auth/me/password | Đổi mật khẩu tài khoản hiện tại | Đã đăng nhập |
| POST | /users/media/upload - avatar | Tải lên/cập nhật ảnh đại diện người dùng | Đã đăng nhập |

### *2.7.2. Quản lý người dùng & KYC (Admin)*

Bảng 2.32 - API quản lý người dùng và KYC

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /users | Tạo người dùng mới (tài khoản nội bộ) | ADMIN |
| GET | /users | Danh sách người dùng có phân trang | ADMIN |
| GET | /users/:id | Chi tiết một người dùng | ADMIN |
| PATCH | /users/:id | Cập nhật thông tin người dùng | ADMIN |
| DELETE | /users/:id | Xóa người dùng | ADMIN |
| GET | /admin/kyc/pending | Danh sách người dùng đang chờ duyệt KYC | ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| PATCH | /admin/kyc/:id/review | Duyệt hoặc từ chối hồ sơ KYC kèm lý do | ADMIN |

### *2.7.3. Danh mục & sản phẩm tham chiếu (categories, products)*

Bảng 2.33 - API danh mục và sản phẩm tham chiếu

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| GET | /categories | Danh sách danh mục sản phẩm (cây danh mục) | Public |
| GET | /categories/:id | Chi tiết một danh mục | Public |
| POST | /categories | Tạo danh mục mới | ADMIN |
| PATCH | /categories/:id | Cập nhật danh mục | ADMIN |
| DELETE | /categories/:id | Xóa danh mục | ADMIN |
| GET | /products | Danh sách sản phẩm tham chiếu (catalog gốc) | Public |
| GET | /products/:id | Chi tiết sản phẩm tham chiếu | Public |
| POST | /products | Tạo sản phẩm tham chiếu mới | ADMIN |
| PATCH | /products/:id | Cập nhật sản phẩm tham chiếu | ADMIN |
| DELETE | /products/:id | Xóa sản phẩm tham chiếu | ADMIN |

### 2.7.4. Địa chỉ giao hàng (addresses)

Bảng 2.34 - API địa chỉ giao hàng

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /addresses | Tạo địa chỉ giao hàng mới | Đã đăng nhập |
| GET | /addresses | Danh sách địa chỉ của bản thân | Đã đăng nhập |
| GET | /addresses/:id | Chi tiết một địa chỉ | Đã đăng nhập (chủ sở hữu) |
| PATCH | /addresses/:id | Cập nhật địa chỉ | Đã đăng nhập (chủ sở hữu) |
| PATCH | /addresses/:id/default | Đặt địa chỉ làm địa chỉ mặc định | Đã đăng nhập (chủ sở hữu) |
| DELETE | /addresses/:id | Xóa địa chỉ | Đã đăng nhập (chủ sở hữu) |

### *2.7.5. Lô hàng, media & truy xuất nguồn gốc (product - batches, batch - media, verify, metadata)*

Bảng 2.35 - API lô hàng, media và truy xuất nguồn gốc

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /product - batches | Farmer khai báo lô hàng mới | FARMER |
| GET | /product - batches | Danh sách toàn bộ lô hàng (lọc theo trạng thái) | ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| GET | /product - batches/my - batches | Danh sách lô hàng của Farmer hiện tại | FARMER/ADMIN |
| GET | /product - batches/:id | Chi tiết một lô hàng | ADMIN/FARMER (liên quan) |
| PATCH | /product - batches/my - batches/:id | Farmer cập nhật lô hàng của mình | FARMER |
| DELETE | /product - batches/my - batches/:id | Farmer ẩn (soft - delete) lô hàng của mình | FARMER |
| PATCH | /product - batches/admin/:id | Admin cập nhật thông tin lô hàng | ADMIN |
| PATCH | /product - batches/admin/:id/status | Admin duyệt hoặc từ chối lô hàng | ADMIN |
| DELETE | /product - batches/admin/:id | Admin xóa lô hàng | ADMIN |
| POST | /batch - media/:batchId/upload | Farmer tải lên ảnh/video cho lô hàng | FARMER |
| GET | /batch - media/:batchId | Danh sách media của một lô hàng | Public |
| DELETE | /batch - media/:batchId/:mediaId | Farmer xóa một media khỏi lô hàng | FARMER |
| GET | /verify/batches/:batchId | Trang xác thực công khai truy xuất nguồn gốc theo lô hàng | Public |
| GET | /verify/lookup | Tra cứu truy xuất nguồn gốc theo mã NFT/token | Public |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| GET | /metadata/batch/:batchId | Lấy metadata NFT (chuẩn ERC - 721) của lô hàng | Public |

### *2.7.6. Listing, đánh giá & giỏ hàng (listings, listing - reviews, carts)*

Bảng 2.36 - API listing, đánh giá và giỏ hàng

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /listings | Farmer tạo listing mới từ một lô hàng | FARMER |
| GET | /listings | Danh sách listing đang bán (lọc, phân trang) | Public |
| GET | /listings/my - listings | Danh sách listing của Farmer hiện tại | FARMER/ADMIN |
| PATCH | /listings/my - listings/:id | Farmer/Admin cập nhật listing | FARMER/ADMIN |
| GET | /listings/:id | Chi tiết một listing | Public |
| POST | /listing - reviews/upload - media | Tải lên ảnh/video kèm đánh giá | MERCHANT/FARMER |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /listing - reviews | Gửi đánh giá cho một listing đã mua | MERCHANT/FARMER |
| GET | /listing - reviews/listing/:listingId | Danh sách đánh giá của một listing | Public |
| GET | /listing - reviews/listing/:listingId/summary | Thống kê tổng hợp điểm đánh giá của listing | Public |
| GET | /carts | Xem giỏ hàng hiện tại (nhóm theo người bán) | MERCHANT/ADMIN |
| POST | /carts/items | Thêm sản phẩm/listing vào giỏ hàng | MERCHANT/ADMIN |
| PATCH | /carts/items/:itemId | Cập nhật số lượng một mục trong giỏ | MERCHANT/ADMIN |
| PATCH | /carts/items/:itemId/selection | Chọn/bỏ chọn mục để thanh toán | MERCHANT/ADMIN |
| DELETE | /carts/items/:itemId | Xóa một mục khỏi giỏ hàng | MERCHANT/ADMIN |
| DELETE | /carts | Xóa toàn bộ giỏ hàng | MERCHANT/ADMIN |

### *2.7.7. Đơn hàng (orders)*

Bảng 2.37 - API đơn hàng

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /orders/checkout - from - cart | Tạo đơn hàng từ giỏ hàng (mỗi người bán một đơn) | MERCHANT/ADMIN |
| POST | /orders/buy - now | Mua ngay một listing (không qua giỏ hàng) | MERCHANT/ADMIN |
| GET | /orders | Danh sách toàn bộ đơn hàng | ADMIN |
| GET | /orders/my - orders | Danh sách đơn hàng đã mua của bản thân | MERCHANT/FARMER/ADMIN |
| GET | /orders/my - seller - orders | Danh sách đơn hàng bán ra của Farmer | FARMER/ADMIN |
| GET | /orders/my - seller - orders/revenue - summary | Thống kê doanh thu của Farmer theo kỳ | FARMER/ADMIN |
| GET | /orders/status - options | Danh sách các trạng thái đơn hàng hợp lệ | ADMIN/FARMER/MERCHANT |
| GET | /orders/:id | Chi tiết một đơn hàng | Người liên quan/ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| PATCH | /orders/:id/shipping | Cập nhật thông tin vận chuyển của đơn hàng | MERCHANT/FARMER |
| PATCH | /orders/:id/status | Admin cập nhật trạng thái đơn hàng | ADMIN |
| PATCH | /orders/:id/payment - status | Admin cập nhật trạng thái thanh toán | ADMIN |
| PATCH | /orders/:id/seller - ship | Farmer xác nhận đã giao hàng cho đơn vị vận chuyển | FARMER/ADMIN |
| PATCH | /orders/:id/buyer - confirm - received | Người mua xác nhận đã nhận hàng (hoàn tất đơn) | MERCHANT/FARMER |
| PATCH | /orders/:id/seller - cancel | Người bán hủy đơn hàng kèm lý do | MERCHANT/FARMER |
| GET | /orders/:id/bank - transfer/vietqr | Lấy mã VietQR chuyển khoản cho đơn hàng | Người liên quan/ADMIN |
| POST | /orders/:id/bank - transfer/receipt | Người mua nộp biên lai chuyển khoản (Level 2) | MERCHANT/FARMER |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| PATCH | /orders/:id/bank - transfer/receipt/review | Người bán/Admin duyệt biên lai chuyển khoản | ADMIN/FARMER |
| GET | /orders/my - seller - orders/bank - transfer - receipts | Danh sách biên lai chờ người bán duyệt | FARMER/ADMIN |
| POST | /orders/jobs/auto - cancel - overdue - pending - confirm | Job tự động hủy đơn quá hạn xác nhận nhận hàng | ADMIN |
| POST | /orders/jobs/auto - cancel - overdue - pending - payment - listings | Job tự động hủy đơn quá hạn thanh toán/listing | ADMIN |

### *2.7.8. Đấu giá & đặt giá (auctions, bids)*

Bảng 2.38 - API đấu giá và đặt giá

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /auctions | Tạo phiên đấu giá mới từ một lô hàng/listing | FARMER/ADMIN |
| GET | /auctions | Danh sách phiên đấu giá (lọc theo trạng thái) | Public |
| GET | /auctions/my - auctions | Danh sách phiên đấu giá của Farmer hiện tại | FARMER/ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| PATCH | /auctions/my - auctions/:id | Farmer cập nhật phiên (khi chưa có bid) | FARMER |
| GET | /auctions/:id | Chi tiết phiên đấu giá | Public |
| PATCH | /auctions/:id/start | Chuyển phiên sang trạng thái đang diễn ra | ADMIN |
| PATCH | /auctions/:id/end | Kết thúc phiên đấu giá (xác định người thắng) | ADMIN |
| PATCH | /auctions/:id/cancel | Hủy phiên đấu giá | ADMIN |
| PATCH | /auctions/:id/retry - chain - sync | Yêu cầu đồng bộ lại trạng thái on - chain của phiên | ADMIN |
| GET | /auctions/:id/winning - bid | Lấy thông tin bid thắng cuộc | Public |
| GET | /auctions/:id/current - price | Lấy giá hiện tại của phiên (polling dự phòng cho WS) | Public |
| GET | /auctions/:id/recent - bids | Danh sách các bid gần nhất | Public |
| GET | /auctions/:id/my - order | Lấy đơn hàng phát sinh từ | MERCHANT/FARMER/ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| | | phiên cho người dùng hiện tại | |
| GET | /auctions/:id/diagnostics | Thông tin chẩn đoán/đồng bộ blockchain của phiên | ADMIN |
| POST | /auctions/:id/chat - messages | Gửi tin nhắn chat trong phòng đấu giá | Đã đăng nhập |
| GET | /auctions/:id/chat - history | Lịch sử chat của phiên đấu giá | Public |
| POST | /bids | Đặt giá mới (yêu cầu x - idempotency - key, khóa pessimistic) | Đã đăng nhập |
| GET | /bids/auction/:auctionId | Lịch sử đặt giá của một phiên | Public |
| GET | /bids/auction/:auctionId/ highest | Giá cao nhất hiện tại của phiên | Public |
| GET | /bids/my - bids | Danh sách các lượt đặt giá của bản thân | Đã đăng nhập |

### *2.7.9. Livestream (live - sessions)*

Bảng 2.39 - API livestream

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /live - sessions/listing/start | Farmer khởi tạo phiên live giới thiệu listing | FARMER/ADMIN |
| POST | /live - sessions/auction/start | Farmer khởi tạo phiên live gắn với phiên đấu giá | FARMER/ADMIN |
| PATCH | /live - sessions/:id/end | Kết thúc phiên livestream | FARMER/MERCHANT/ADMIN |
| PATCH | /live - sessions/:id/heartbeat | Gửi heartbeat duy trì phiên (mỗi 30 giây) | FARMER/MERCHANT/ADMIN |
| POST | /live - sessions/:id/pins/:listingId | Ghim listing vào phiên live (tối đa 24) | FARMER/MERCHANT/ADMIN |
| DELETE | /live - sessions/:id/pins/:listingId | Bỏ ghim listing khỏi phiên live | FARMER/MERCHANT/ADMIN |
| GET | /live - sessions/:id/token | Lấy token LiveKit để tham gia phòng (host/viewer) | Đã đăng nhập |
| GET | /live - sessions/:id/pins | Danh sách listing đang được ghim trong phiên | Đã đăng nhập |
| GET | /live - sessions/:id/chat - history | Lịch sử chat của phiên live | Đã đăng nhập |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /live - sessions/:id/chat - messages | Gửi tin nhắn chat trong phòng live | Đã đăng nhập |
| GET | /live - sessions/public | Danh sách phiên live công khai đang diễn ra | Public |
| GET | /live - sessions/public/:id | Chi tiết một phiên live công khai | Public |
| GET | /live - sessions/me/active | Phiên live đang hoạt động của bản thân (nếu có) | Đã đăng nhập |
| GET | /live - sessions/auction/:auctionId/status | Trạng thái phiên live gắn với một phiên đấu giá | Đã đăng nhập |
| GET | /live - sessions/:id | Chi tiết một phiên live | Đã đăng nhập |
| GET | /live - sessions | Danh sách phiên live của người dùng hiện tại | Đã đăng nhập |

### *2.7.10. Thanh toán & Ví (payments, wallets)*

Bảng 2.40 - API thanh toán và ví

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /payments/payos/create - link | Tạo link thanh toán PayOS cho đơn hàng | Đã đăng nhập |
| POST | /payments/payos/wallet - topup - link | Tạo link PayOS để nạp tiền vào ví | Đã đăng nhập |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /payments/payos/wallet - topup/confirm - return | Xác nhận kết quả nạp tiền khi quay lại từ PayOS | Đã đăng nhập |
| POST | /payments/payos/webhook | Webhook PayOS xác nhận thanh toán/nạp tiền (xác thực chữ ký) | PayOS |
| GET | /wallets/balance | Số dư ví hiện tại (balance, frozenBalance) | Đã đăng nhập |
| GET | /wallets/activity | Lịch sử giao dịch ví (transactions ledger) | Đã đăng nhập |
| POST | /wallets/deposit - mock | Nạp tiền giả lập vào ví (môi trường dev) | ADMIN |
| PATCH | /wallets/:userId/admin - deposit | Admin cộng tiền trực tiếp vào ví của một người dùng | ADMIN |

### *2.7.11. Payout, Voucher, Phí nền tảng & Giao dịch*

Bảng 2.41 - API payout, voucher, phí nền tảng và giao dịch

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| POST | /payouts/requests | Farmer tạo yêu cầu rút tiền (payout) về ngân hàng | FARMER/ADMIN |
| GET | /payouts/my | Danh sách yêu cầu payout của bản thân | FARMER/ADMIN |
| GET | /payouts | Danh sách toàn bộ yêu cầu payout | ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| PATCH | /payouts/requests/:id/admin/approve | Admin duyệt yêu cầu payout | ADMIN |
| PATCH | /payouts/requests/:id/admin/complete | Admin xác nhận đã chuyển tiền, hoàn tất payout | ADMIN |
| PATCH | /payouts/requests/:id/admin/reject | Admin từ chối yêu cầu payout kèm lý do | ADMIN |
| POST | /payouts/jobs/auto - finalize - approved | Job tự động hoàn tất các payout đã duyệt | ADMIN |
| POST | /vouchers | Tạo voucher khuyến mãi mới | ADMIN |
| GET | /vouchers | Danh sách voucher | ADMIN |
| GET | /vouchers/:id | Chi tiết một voucher | ADMIN |
| POST | /platform - fees/jobs/generate - monthly | Job tạo báo cáo phí nền tảng theo tháng | ADMIN |
| GET | /platform - fees/monthly | Danh sách báo cáo phí nền tảng theo tháng | ADMIN |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| GET | /platform - fees/monthly/summary | Thống kê tổng hợp phí nền tảng | ADMIN |
| GET | /transactions | Danh sách giao dịch ví toàn hệ thống | ADMIN |
| GET | /transactions/cash - flow | Báo cáo dòng tiền theo loại giao dịch | ADMIN |
| GET | /transactions/trace/:externalOrderCode | Truy vết một giao dịch theo mã đơn hàng PayOS | ADMIN |

### *2.7.12. Quản trị, giám sát hệ thống & Blockchain*

Bảng 2.42 - API quản trị, giám sát hệ thống và blockchain

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| GET | /admin/dashboard/summary | Số liệu tổng quan cho trang dashboard Admin | ADMIN |
| GET | /system - settings | Danh sách toàn bộ cấu hình hệ thống | ADMIN |
| PATCH | /system - settings/:key | Cập nhật một tham số cấu hình hệ thống | ADMIN |
| GET | /health | Kiểm tra tổng quan sức khỏe hệ thống (DB, Redis, ...) | Public |

| Method | Endpoint | Mô tả | Quyền truy cập |
|---|---|---|---|
| GET | /health/live | Liveness probe (cho container orchestrator) | Public |
| GET | /health/ready | Readiness probe | Public |
| GET | /metrics | Số liệu Prometheus của ứng dụng | Public |
| GET | /metrics/runtime | Số liệu runtime (CPU, memory, event loop) | Public |
| GET | /monitoring/blockchain - health | Trạng thái kết nối/đồng bộ blockchain | Public |
| GET | /blockchain/client - config | Cấu hình client để kết nối ví/mạng blockchain | Public |

Ngoài các endpoint REST nêu trên, hệ thống còn cung cấp các kênh WebSocket (Socket.IO) để cập nhật thời gian thực: kênh đấu giá (giá hiện tại, lịch sử bid, đếm ngược thời gian), kênh chat của phòng đấu giá/livestream, và kênh thông báo trạng thái phiên livestream (heartbeat/FORCE_ENDED).

# Chương 3: TRIỂN KHAI HỆ THỐNG VÀ ĐÁNH GIÁ KẾT QUẢ

## 3.1. Môi trường và công nghệ cài đặt

Hệ thống AgriBid được tổ chức theo dạng monorepo gồm bốn thành phần độc lập: AgriBid - Backend (NestJS), AgriBid - Frontend/Client (Next.js dành cho Farmer/Merchant), AgriBid - Frontend/Admin (Next.js dành cho quản trị viên) và AgriBid - Blockchain (Hardhat/Solidity). Các thành phần giao tiếp với nhau qua REST API, WebSocket (Socket.IO) và mạng blockchain Ethereum (testnet Sepolia).

Bảng 3.1 - Môi trường và công nghệ cài đặt

| Thành phần | Công nghệ sử dụng |
|---|---|
| Backend (AgriBid - Backend) | Node.js 22, NestJS 11, TypeORM 0.3, PostgreSQL 15, Redis (cache + Bull queue), Passport JWT, class - validator, Swagger/OpenAPI, ethers.js 6, Prometheus (prom - client) |
| Client (AgriBid - Frontend/Client) | Next.js 16 (App Router, Turbopack), React, Tailwind CSS 4, TanStack Query, Axios, Wagmi/Viem + WalletConnect/MetaMask/Coinbase, LiveKit Client, i18n (en/vi) |
| Admin (AgriBid - Frontend/Admin) | Next.js 14, MUI 5 (Materio template), TanStack Query, React Hook Form |
| Blockchain (AgriBid - Blockchain) | Solidity 0.8.28, OpenZeppelin Contracts 5, Hardhat 3 (hardhat - toolbox - mocha - ethers), mạng Ethereum Sepolia testnet |
| Cơ sở dữ liệu & hạ tầng phụ trợ | PostgreSQL 15 (Docker, cổng 5434), Redis Alpine (Docker, cổng 6379), LiveKit (livestream SFU), PayOS (cổng thanh toán) |

## 3.2. Cấu trúc mã nguồn

Cấu trúc thư mục được tổ chức theo nguyên tắc tách biệt theo module nghiệp vụ (feature - based), giúp dễ bảo trì và mở rộng:

Bảng 3.2 - Cấu trúc mã nguồn

| Thư mục | Nội dung chính |
|---|---|
| AgriBid - Backend/src/common | Guard (JwtAuthGuard, RolesGuard), Interceptor (TransformInterceptor - chuẩn hóa response và ẩn trường nhạy cảm), Filter (HttpExceptionFilter), DTO phân trang, enum, util (vietqr - bank, bank - profile - policy, farmer - sell - policy, batch - certificate...) |
| AgriBid - Backend/src/database | Cấu hình kết nối TypeORM (typeorm - datasource.ts) và các migration |
| AgriBid - Backend/src/modules | 27 module nghiệp vụ, mỗi module gồm controller → service → entity → dto; các module phức tạp (orders, auctions, live - sessions) tách thêm các service con trong thư mục services/ và gateway realtime trong gateways/ |
| AgriBid - Backend/scripts | Script seed dữ liệu mẫu (seed.ts, seed - mint.ts, seed - long - demo - auctions.ts...), script kiểm tra/đồng bộ blockchain (blockchain - check.ts, blockchain - replay.ts), script smoke test (smoke:vietqr, smoke:auction, smoke:blockchain) |
| AgriBid - Frontend/Client/src | app/[locale] (route theo App Router, nhóm (shop) và (farmer)), features/ (chia theo domain: auctions, listings, cart, checkout, livestream...), shared/ (apiClient, provider Wagmi, layout) |
| AgriBid - Frontend/Admin/src | app/(dashboard) (trang quản trị), views/ (màn hình + Table/Dialog), services/ (gọi API), hooks/api/ (React Query) |
| AgriBid - Blockchain/contracts | AgriBidEscrow.sol (hợp đồng ký quỹ đấu giá ETH), AgriBidNFT.sol (hợp đồng NFT ERC - 721 truy xuất nguồn gốc); thư mục scripts/ chứa deploy.js (triển khai lên Sepolia) và manage - escrow - roles.js |

### 3.3. Cấu hình môi trường (Environment Configuration)

Toàn bộ cấu hình nhạy cảm và phụ thuộc môi trường được quản lý qua biến môi trường (.env), nạp bằng @nestjs/config. Các nhóm biến chính gồm:

- Cấu hình ứng dụng: PORT, HOST, NODE_ENV, ALLOWED_ORIGINS, APP_PUBLIC_URL, NFT_METADATA_BASE_URL.
- Cấu hình cơ sở dữ liệu: DB_HOST, DB_PORT, DB_USERNAME, DB_PASSWORD, DB_NAME, DB_SYNCHRONIZE (=false ở production, dùng migration), DB_LOGGING.
- Cấu hình Redis & rate limit: REDIS_HOST/PORT/PASSWORD/DB/KEY_PREFIX, RATE_LIMIT_TTL, RATE_LIMIT_MAX (ThrottlerGuard).
- Cấu hình blockchain: BLOCKCHAIN_RPC_URL, BLOCKCHAIN_CHAIN_ID (Sepolia = 11155111), BLOCKCHAIN_PRIVATE_KEY, NFT_CONTRACT_ADDRESS, ESCROW_CONTRACT_ADDRESS, BLOCKCHAIN_POLL_INTERVAL_MS, BLOCKCHAIN_LISTENER_AUTOSTART - phục vụ BlockchainListener đồng bộ sự kiện on - chain.
- Cấu hình thanh toán PayOS: PAYOS_CLIENT_ID, PAYOS_API_KEY, PAYOS_CHECKSUM_KEY, PAYOS_RETURN_URL/CANCEL_URL, PAYOS_TOPUP_TIMEOUT_MINUTES, PAYOS_TOPUP_SWEEP_INTERVAL_MS.
- Cấu hình nghiệp vụ vận hành: ORDER_PENDING_CONFIRM_TIMEOUT_HOURS, ORDER_PENDING_PAYMENT_TIMEOUT_HOURS, PLATFORM_FEE_DEFAULT_BPS, PLATFORM_FEE_AUTO_GENERATE_ENABLED, PAYOUT_MIN/MAX_AMOUNT_VND, PAYOUT_MAX_REQUESTS_PER_DAY - cấu hình các tham số nghiệp vụ tài chính có thể điều chỉnh không cần sửa code.
- Cấu hình livestream: LIVEKIT_URL, LIVEKIT_API_KEY, LIVEKIT_API_SECRET.

Bảng 3.3 thống kê đầy đủ các nhóm biến môi trường thực tế được khai báo trong tệp .env.example của từng package (không bao gồm giá trị thực - chỉ tên biến và mục đích), cho thấy quy mô cấu hình cần thiết lập khi triển khai hệ thống:

Bảng 3.3 - Biến môi trường (.env) theo từng package

| Package | Nhóm biến | Tên biến tiêu biểu |
|---|---|---|
| Backend | Cấu hình ứng dụng (8 biến) | PORT, HOST, NODE_ENV, APP_NAME, APP_ENV, ALLOWED_ORIGINS, APP_PUBLIC_URL, NFT_METADATA_BASE_URL |
| Backend | Cơ sở dữ liệu (7 biến) | DB_HOST, DB_PORT, DB_USERNAME, DB_PASSWORD, DB_NAME, DB_SYNCHRONIZE, DB_LOGGING |
| Backend | Redis & rate limit (7 biến) | REDIS_HOST, REDIS_PORT, REDIS_PASSWORD, REDIS_DB, REDIS_KEY_PREFIX, RATE_LIMIT_TTL, RATE_LIMIT_MAX |
| Backend | Xác thực & vận hành (2 biến) | JWT_SECRET, AUCTION_RECONCILE_INTERVAL_MS |
| Backend | Blockchain (12 biến) | BLOCKCHAIN_RPC_URL, BLOCKCHAIN_CHAIN_ID, BLOCKCHAIN_PRIVATE_KEY, NFT_CONTRACT_ADDRESS, ESCROW_CONTRACT_ADDRESS, BLOCKCHAIN_POLL_INTERVAL_MS, BLOCKCHAIN_LOG_CHUNK_START/MIN/MAX, BLOCKCHAIN_LOG_RETRY_MAX, BLOCKCHAIN_CONFIRMATIONS, BLOCKCHAIN_LISTENER_AUTOSTART |
| Backend | PayOS (7 biến) | PAYOS_CLIENT_ID, PAYOS_API_KEY, PAYOS_CHECKSUM_KEY, PAYOS_RETURN_URL, |

| Package | Nhóm biến | Tên biến tiêu biểu |
|---|---|---|
| | | PAYOS_CANCEL_URL, PAYOS_TOPUP_TIMEOUT_MINUTES, PAYOS_TOPUP_SWEEP_INTERVAL_MS |
| Backend | Đơn hàng, phí & payout (10 biến) | ORDER_PENDING_CONFIRM_TIMEOUT_HOURS, ORDER_PENDING_PAYMENT_TIMEOUT_HOURS, ORDER_PENDING_PAYMENT_AUTO_CANCEL_ENABLED, ORDER_PENDING_PAYMENT_SWEEP_INTERVAL_MS, PLATFORM_FEE_DEFAULT_BPS, PLATFORM_FEE_AUTO_GENERATE_ENABLED, PLATFORM_FEE_AUTO_CHECK_INTERVAL_MS, PAYOUT_MIN_AMOUNT_VND/MAX_AMOUNT_VND/MAX_REQUESTS_PER_DAY |
| Backend | LiveKit (3 biến) | LIVEKIT_URL, LIVEKIT_API_KEY, LIVEKIT_API_SECRET |
| Client | Kết nối Backend & realtime (7 biến) | NEXT_PUBLIC_USE_LAN_PROXY, NEXT_PUBLIC_API_URL, NEXT_PUBLIC_MEDIA_URL, NEXT_PUBLIC_SOCKET_URL, NEXT_PUBLIC_FRONTEND_URL, API_PROXY_TARGET, INTERNAL_API_URL |
| Client | Blockchain & ví (4 biến) | NEXT_PUBLIC_BLOCKCHAIN_CHAIN_ID, NEXT_PUBLIC_BLOCKCHAIN_RPC_URL, NEXT_PUBLIC_WALLETCONNECT_PROJECT_ID, NEXT_PUBLIC_ESCROW_CONTRACT_ADDRESS |
| Client | Khác (1 biến) | NEXT_PUBLIC_MARKETPLACE_DEMO_FALLBACK |

| Package | Nhóm biến | Tên biến tiêu biểu |
|---|---|---|
| Admin | Cấu hình ứng dụng (6 biến) | BASEPATH, NEXT_PUBLIC_APP_URL, NEXT_PUBLIC_DOCS_URL, NEXT_PUBLIC_PRO_URL, NEXT_PUBLIC_REPO_NAME, NEXT_PUBLIC_API_URL |
| Blockchain | Triển khai hợp đồng (4 biến) | PRIVATE_KEY, RPC_URL, NFT_CONTRACT_ADDRESS, ESCROW_CONTRACT_ADDRESS |

Ba biến cấu hình NFT_CONTRACT_ADDRESS, ESCROW_CONTRACT_ADDRESS và BLOCKCHAIN_RPC_URL/RPC_URL xuất hiện ở cả ba package Backend, Client và Blockchain với cùng giá trị - đây là điểm cần đồng bộ thủ công mỗi khi triển khai lại smart contract (xem Phụ lục - Hướng dẫn cài đặt & chạy hệ thống).

### 3.4. Cài đặt môi trường phát triển

Quy trình thiết lập môi trường phát triển cho Backend gồm các bước:

- Khởi tạo dịch vụ PostgreSQL và Redis bằng Docker Compose (docker - compose.yml định nghĩa hai service postgres:15 trên cổng 5434 và redis:alpine trên cổng 6379).
- Cài đặt dependency bằng npm install, sao chép .env.example thành .env và điền các giá trị cấu hình thực tế (DB, JWT_SECRET, khóa ví blockchain, PayOS, LiveKit...).
- Chạy migration cơ sở dữ liệu bằng lệnh npm run migration:run (TypeORM CLI với typeorm - datasource.ts).
- Seed dữ liệu mẫu bằng các script seed.ts, seed - mint.ts, seed - images.ts, seed - long - demo - auctions.ts, seed - listing - reviews.ts để có dữ liệu demo cho danh mục, lô hàng, listing, phiên đấu giá và đánh giá.

- Khởi động server ở chế độ phát triển bằng npm run start:dev (hot - reload qua - - watch); Swagger UI được expose tại /api để khám phá toàn bộ API.
- Đối với Frontend Client/Admin: npm install rồi npm run dev (Client dùng Turbopack); biến NEXT_PUBLIC_API_URL phải trỏ đúng địa chỉ Backend.
- Đối với Blockchain: npx hardhat compile để biên dịch AgriBidEscrow.sol và AgriBidNFT.sol, sau đó npx hardhat run scripts/deploy.js - - network sepolia để triển khai hợp đồng lên testnet và lấy địa chỉ hợp đồng (NFT_CONTRACT_ADDRESS, ESCROW_CONTRACT_ADDRESS) cập nhật vào .env của Backend.

### 3.5. Tích hợp CI (Continuous Integration)

Dự án sử dụng GitHub Actions để kiểm tra chất lượng mã nguồn tự động. Workflow client - ci.yml được kích hoạt khi có thay đổi trong AgriBid - Frontend/Client, thực hiện tuần tự: cài đặt Node.js 22 với cache npm, chạy ESLint (kiểm tra quy tắc code), kiểm tra kiểu dữ liệu TypeScript (typecheck), kiểm tra định dạng bằng Prettier (format:check), chạy bộ test với Vitest (test:run) và cuối cùng build dự án Next.js (build). Script ci trong package.json của Client gom toàn bộ các bước này thành một lệnh duy nhất (npm run ci), đảm bảo mọi thay đổi đều được kiểm tra trước khi tích hợp.

### 3.6. Một số đoạn mã cài đặt tiêu biểu

#### *3.6.1. Lớp bảo vệ và phân quyền (Guard)*

Toàn bộ endpoint yêu cầu xác thực được bảo vệ bởi JwtAuthGuard (kiểm tra access token Bearer) kết hợp RolesGuard (kiểm tra vai trò qua decorator @Roles), ví dụ áp dụng tại auctions.controller.ts:

```
@UseGuards(JwtAuthGuard, RolesGuard)
@Roles(UserRole.ADMIN)
@Patch(':id/start')
startAuction(@Param('id') id: number) { ... }
```

Cơ chế này đảm bảo các thao tác nhạy cảm (duyệt KYC, mở/đóng phiên đấu giá, xử lý payout...) chỉ được thực hiện bởi đúng vai trò được cấp phép, đúng như đặc tả use case ở mục 2.2.

### *3.6.2. Chuẩn hóa response (Interceptor)*

TransformInterceptor được áp dụng toàn cục để bọc dữ liệu trả về trong một envelope thống nhất (status, data, message) và loại bỏ các trường nhạy cảm (ví dụ password hash) trước khi trả về client, giúp Frontend xử lý response theo một định dạng duy nhất cho mọi API.

### *3.6.3. Cơ chế idempotency cho giao dịch tài chính*

Các API có tác động tài chính hoặc tạo bản ghi không thể lặp lại (đặt giá /bids, tạo đơn hàng /orders/checkout - from - cart, /orders/buy - now, tạo listing) yêu cầu header x - idempotency - key. Module idempotency lưu lại khóa và kết quả của lần gọi đầu tiên trong PostgreSQL (bảng idempotency_records); nếu client gửi lại cùng khóa do lỗi mạng/timeout, hệ thống trả về kết quả đã lưu thay vì thực hiện lại giao dịch, tránh tình trạng đặt giá trùng hoặc tạo đơn hàng/trừ ví nhiều lần.

### *3.6.4. Đồng bộ trạng thái blockchain (BlockchainListener)*

BlockchainService (sử dụng ethers.js) chịu trách nhiệm gửi giao dịch tới hợp đồng AgriBidEscrow và AgriBidNFT trên Sepolia. BlockchainListener định kỳ poll log sự kiện on - chain (BLOCKCHAIN_POLL_INTERVAL_MS, mặc định 15 giây) theo từng đoạn block (BLOCKCHAIN_LOG_CHUNK_START/MIN/MAX) để cập nhật trạng thái auctions.onChainSyncStatus (NOT_REQUIRED/PENDING/SYNCED/FAILED) trong PostgreSQL, đảm bảo dữ liệu phiên đấu giá ETH luôn phản ánh đúng trạng thái trên chuỗi. API PATCH /auctions/:id/retry - chain - sync cho phép Admin yêu cầu đồng bộ lại khi cần [17].

## 3.7. Kiểm thử hệ thống

### *3.7.1. Chiến lược kiểm thử*

Hệ thống được kiểm thử theo nhiều cấp độ, tương ứng với từng thành phần của monorepo:

- Unit test Backend: sử dụng Jest (ts - jest) để kiểm thử logic nghiệp vụ ở tầng service/controller, mock các repository TypeORM và service phụ trợ (Redis, BlockchainService, PayOS).

- E2E test Backend: sử dụng Jest + Supertest (cấu hình test/jest - e2e.json) để kiểm thử luồng HTTP end - to - end, ví dụ app.e2e - spec.ts và live - sessions.e2e - spec.ts.
- Smoke test: các script smoke:vietqr, smoke:auction, smoke:blockchain (gộp thành smoke:phase - a, smoke:phase - b) gọi trực tiếp API trên môi trường đã chạy để kiểm tra nhanh các luồng thanh toán VietQR, đấu giá và đồng bộ blockchain.
- Unit test Smart Contract: sử dụng Hardhat 3 + Mocha + Chai (hardhat - toolbox - mocha - ethers) để kiểm thử hợp đồng AgriBidEscrow.sol và AgriBidNFT.sol, bao gồm cả các tình huống đặt giá đồng thời (concurrent bidding).
- Kiểm thử Frontend: Vitest cho unit test (Client, Admin) và Playwright cho E2E UI; tích hợp trong workflow CI client - ci.yml.

### *3.7.2. Kết quả kiểm thử Backend (Jest)*

Thực thi lệnh npm run test trên AgriBid - Backend cho kết quả: 21 test suite, 84 test case, toàn bộ PASS, thời gian chạy khoảng 12.5 giây. Các module được kiểm thử bao gồm:

Bảng 3.4 - Kết quả kiểm thử Backend theo module/service

| **Module / Service kiểm thử** | **Nội dung kiểm thử chính** |
|---|---|
| auctions.service.spec.ts, auction.gateway.spec.ts | Logic tạo/kết thúc phiên đấu giá, broadcast sự kiện realtime qua WebSocket (đặt giá mới, kết thúc đấu giá) |
| orders.controller.spec.ts, order - cart - checkout.service, order - payment.service, order - settlement.service, order - pending - payment - scheduler.service | Tạo đơn từ giỏ hàng/mua ngay, xử lý thanh toán (ví nội bộ/PayOS/VietQR), đối soát đơn hàng, job tự động hủy đơn quá hạn thanh toán |
| listings.service.spec.ts | Tạo và truy vấn listing, kiểm tra ràng buộc tồn kho |
| product - batches.service.spec.ts | Khai báo, cập nhật, duyệt lô hàng |

| **Module / Service kiểm thử** | **Nội dung kiểm thử chính** |
|---|---|
| nft - metadata.service.spec.ts | Sinh metadata NFT chuẩn ERC - 721 cho lô hàng đã duyệt |
| payos - payments.service.spec.ts | Tạo link thanh toán PayOS, xử lý webhook xác nhận |
| live - sessions.service.spec.ts | Khởi tạo phiên livestream, ghim/bỏ ghim listing, heartbeat |
| platform - fees.service.spec.ts, platform - fee - scheduler.service.spec.ts | Tính phí nền tảng theo kỳ, job tự động sinh báo cáo phí hàng tháng |
| payouts.service.spec.ts | Quy trình tạo, duyệt, hoàn tất yêu cầu rút tiền (payout) của Farmer |
| common/utils (vietqr - bank, bank - profile - policy, farmer - sell - policy, batch - certificate, transform.interceptor, page - meta.dto) | Các hàm tiện ích: sinh mã VietQR, chính sách hồ sơ ngân hàng, chính sách bán hàng của Farmer, chuẩn hóa response, phân trang |

Kết quả 84/84 test case PASS cho thấy các luồng nghiệp vụ cốt lõi (đấu giá, thanh toán, đơn hàng, livestream, ví/giao dịch, phí nền tảng, payout) đều hoạt động đúng theo logic thiết kế ở Chương 2 trước khi tích hợp lên môi trường thật.

### *3.7.3. Kết quả kiểm thử Smart Contract (Hardhat)*

Thực thi npm test trên AgriBid - Blockchain (bộ test AgriBidEscrow.test.js và AgriBidNFT.test.js) cho kết quả 20/32 test case PASS. Kết quả này cho thấy các chức năng nền tảng của smart contract đã được kiểm thử bước đầu, bao gồm triển khai hợp đồng, phân quyền, tạo phiên đấu giá, đặt giá hợp lệ, kiểm tra điều kiện đặt giá và một số kịch bản kết thúc phiên. Tuy nhiên, do smart contract liên quan trực tiếp đến tài sản số và giao dịch bằng ETH, việc còn 12/32 test case chưa PASS cần được xem là một hạn chế quan trọng của phiên bản hiện tại.

Các test case chưa PASS chủ yếu tập trung ở những kịch bản biên và kịch bản xử lý ngoại lệ như rút tiền hoàn trả, hủy phiên đấu giá, tạm dừng hợp đồng và đặt giá cạnh tranh mở rộng. Nguyên nhân cần được tiếp tục rà soát ở cả hai phía: logic xử lý trong smart contract và cấu hình/phiên bản bộ công cụ kiểm thử Hardhat, OpenZeppelin, chai-matchers. Vì vậy, kết quả này không nên được diễn giải là hệ thống đã sẵn sàng triển khai mainnet, mà là cơ sở để xác định các nhóm chức năng cần ưu tiên kiểm thử và hoàn thiện trước khi sử dụng cho giao dịch có giá trị thật.

Các trường hợp PASS bao gồm các kiểm thử cốt lõi:

- Triển khai hợp đồng đúng với các vai trò (owner có quyền MANAGER/PAUSER).
- Tạo phiên đấu giá (createAuction), từ chối tạo trùng ID phiên.
- Đặt giá đầu tiên hợp lệ trở thành highestBidder; từ chối đặt giá dưới mức deposit (BidBelowDeposit), đặt giá vào phiên không tồn tại (AuctionNotFound), hoặc sau khi đã kết thúc (AuctionNotOpen).
- Kết thúc phiên không có người đặt giá → chuyển trạng thái ENDED, không phát sinh thanh toán.
- Kiểm soát quyền: tài khoản không có vai trò MANAGER/PAUSER không thể tạo/kết thúc phiên hoặc tạm dừng hợp đồng.
- Người thắng không thể rút tiền hoàn trả (WinnerCannotWithdraw); hợp đồng không có tiền để rút trả lỗi NothingToWithdraw.
- Hai bidder đặt giá đồng thời trong cùng một block: bidder có tổng giá trị cao hơn thắng; đặt giá ngang bằng giá cao nhất hiện tại bị từ chối với lỗi BidNotHighEnough.

Bảng 3.5 phân loại 32 test case theo các nhóm describe() trong bộ test, phản ánh đúng cấu trúc tổ chức của hai file AgriBidEscrow.test.js (30 test case) và AgriBidNFT.test.js (2 test case):

Bảng 3.5 - Phân nhóm 32 test case smart contract theo file/describe block

| **Nhóm test (describe)** | **Số test case** | **Nội dung kiểm thử** |
|---|---|---|
| AgriBidEscrow - Core Functionality | 7 | Deploy hợp đồng & phân quyền owner, tạo phiên đấu giá, từ chối trùng ID, đặt giá hợp lệ trở thành highestBidder, rút tiền hoàn sau khi kết thúc phiên, hủy phiên đấu giá, kết thúc phiên không có bidder. |
| AgriBidEscrow - Access Control | 5 | Chặn tài khoản không có vai trò MANAGER tạo/kết thúc phiên, chặn tài khoản không có vai trò PAUSER tạm dừng hợp đồng, chặn đặt giá khi hợp đồng đang pause, cho phép đặt giá lại sau unpause. |
| AgriBidEscrow - Edge Cases & Validation | 10 | Đặt giá dưới mức deposit, đặt giá vào phiên không tồn tại/đã kết thúc, người thắng không rút được tiền hoàn, rút tiền khi phiên còn mở, top - up phần chênh theo minBidIncrement, ràng buộc minBidIncrementWei, event MinBidIncrementUpdated, địa chỉ farmer = 0, rút tiền khi không có gì để rút. |
| AgriBidEscrow - Concurrent Bidding | 8 | Hai/ba bidder đặt giá đồng thời (cùng block và khác block), đặt giá ngang bằng highestBid hiện tại, top - up đồng thời, 5 vòng raise xen kẽ liên tiếp, đặt giá sau khi kết thúc phiên, kiểm tra deposit cộng dồn qua nhiều lần bid không gây side - effect (reentrancy). |
| AgriBidNFT | 2 | Mint NFT và tăng đúng tokenId tuần tự; chặn tài khoản không phải owner thực hiện mint. |

Đối chiếu với kết quả 20/32 PASS, các test case đã đạt chủ yếu bao phủ những chức năng nền tảng của smart contract như triển khai hợp đồng, phân quyền, tạo phiên

đấu giá, đặt giá hợp lệ, kiểm tra điều kiện đặt giá, kết thúc phiên không có bidder và một số kịch bản cạnh tranh cơ bản. Trong khi đó, phần lớn các trường hợp chưa PASS tập trung ở các kịch bản biên và kịch bản xử lý ngoại lệ, đặc biệt là hoàn tiền, hủy phiên đấu giá, tạm dừng/khôi phục hợp đồng và đặt giá cạnh tranh mở rộng.

Từ kết quả trên, các test case chưa PASS được phân tích theo nhóm kịch bản để làm rõ nội dung cần kiểm thử, mức độ ảnh hưởng đến nghiệp vụ và hướng xử lý trong các phiên bản tiếp theo. Bảng 3.6 trình bày phân tích các nhóm test case smart contract chưa PASS.

Bảng 3.6 - Phân tích nhóm test case smart contract chưa PASS

| **Nhóm kịch bản chưa PASS** | **Nội dung cần kiểm thử** | **Mức độ ảnh hưởng** | **Hướng xử lý** |
|---|---|---|---|
| Rút tiền hoàn trả | Kiểm tra cơ chế hoàn tiền cho bidder không thắng hoặc khi phiên bị hủy | Cao, vì liên quan trực tiếp đến tài sản ETH của người dùng | Rà soát pendingReturns, thứ tự cập nhật trạng thái và bổ sung test theo từng bidder |
| Hủy phiên đấu giá | Kiểm tra cancelAuction và trạng thái phiên sau khi hủy | Trung bình - cao, vì ảnh hưởng đến xử lý ngoại lệ của phiên đấu giá | Chuẩn hóa trạng thái phiên, event emit và kiểm tra số dư trước/sau khi hủy |
| Tạm dừng hợp đồng | Kiểm tra pause/unpause và chặn thao tác khi hợp đồng tạm dừng | Trung bình, vì đây là cơ chế an toàn khi có sự cố | Đồng bộ phiên bản thư viện và kiểm tra lại cú pháp bắt lỗi revert/custom error |

| **Nhóm kịch bản chưa PASS** | **Nội dung cần kiểm thử** | **Mức độ ảnh hưởng** | **Hướng xử lý** |
|---|---|---|---|
| Đặt giá cạnh tranh mở rộng | Kiểm tra nhiều bidder đặt giá liên tiếp trong thời gian ngắn | Trung bình - cao, vì ảnh hưởng đến tính công bằng của đấu giá | Bổ sung test từng bước cho highestBid, highestBidder, event BidPlaced và hoàn tiền bidder trước |
| Top-up phần chênh | Kiểm tra cơ chế bidder nạp thêm phần chênh theo minBidIncrement khi đã có bid trước đó | Trung bình - cao, vì liên quan đến cách tính tổng giá trị bid và số tiền bị giữ | Rà soát logic _topUp, pendingReturns và số dư hợp đồng sau mỗi lần bid |

Như vậy, kết quả kiểm thử smart contract cho thấy hệ thống đã hoàn thành bước đầu các chức năng nền tảng, nhưng vẫn cần tiếp tục hoàn thiện các kịch bản kiểm thử liên quan đến hoàn tiền, hủy phiên, tạm dừng hợp đồng, top-up phần chênh và đặt giá cạnh tranh mở rộng. Nguyên nhân của các test case chưa PASS cần được tiếp tục rà soát ở cả hai phía: logic xử lý trong smart contract và cấu hình/phiên bản bộ công cụ kiểm thử Hardhat, OpenZeppelin, chai-matchers.

Trong phạm vi đồ án, phần smart contract được xem là đã hoàn thành ở mức triển khai chức năng, tích hợp với backend và kiểm thử bước đầu trên môi trường testnet. Tuy nhiên, do smart contract có liên quan trực tiếp đến tài sản ETH của người dùng, hệ thống chưa được khuyến nghị triển khai trên mainnet hoặc sử dụng cho giao dịch có giá trị thật cho đến khi bộ kiểm thử đạt kết quả PASS đầy đủ và được rà soát bảo mật kỹ hơn.

### *3.7.4. Kết quả kiểm thử Frontend*

Theo chiến lược ở mục 3.7.1, phần Frontend (Client, Admin) sử dụng Vitest cho unit test và Playwright cho E2E UI, được tích hợp vào pipeline CI (client - ci.yml). Tại thời điểm thực hiện đồ án, pipeline CI cho Frontend đã được thiết lập và chạy được (lint, build, kiểm tra type), tuy nhiên số lượng test case Vitest/Playwright viết cho các luồng UI còn hạn chế (tập trung vào một số component dùng chung và trang đăng nhập/đăng ký), chưa phủ hết các luồng nghiệp vụ chính.

Do đó, việc kiểm thử các luồng nghiệp vụ trên giao diện (đăng ký/đăng nhập, đăng listing, đặt giá đấu giá, thanh toán, livestream...) chủ yếu được thực hiện thông qua kiểm thử thủ công theo use case ở mục 3.7.5 bên dưới, kết hợp với các test Backend (API) ở mục 3.7.2 để đảm bảo dữ liệu và trạng thái trả về đúng. Việc bổ sung test case Vitest/Playwright cho các luồng UI còn thiếu được ghi nhận là một hạng mục cần hoàn thiện ở giai đoạn tiếp theo của dự án.

### *3.7.5. Kiểm thử theo use case*

Dựa trên các đặc tả use case ở mục 2.2.4, nhóm thực hiện kiểm thử thủ công (manual test) kết hợp smoke test để xác nhận luồng hoạt động thực tế qua Swagger UI và giao diện Client/Admin:

Bảng 3.7 - Kết quả kiểm thử theo các use case chính

| **Use case** | **Kịch bản kiểm thử** | **Cách thực hiện** | **Kết quả** |
|---|---|---|---|
| UC - 01 | Đăng ký, đăng nhập, làm mới token, đăng xuất | Gọi /auth/register, /auth/login, /auth/refresh, /auth/logout qua Swagger và giao diện Client | Đạt |
| UC - 02 | Farmer khai báo lô hàng, tải media, gửi duyệt | POST /product - batches, POST /batch - media/:batchId/upload, kiểm tra trạng thái PENDING → READY_FOR_SALE qua /product - batches/admin/:id/status | Đạt |

| **Use case** | **Kịch bản kiểm thử** | **Cách thực hiện** | **Kết quả** |
|---|---|---|---|
| UC - 03 | Admin duyệt lô hàng và mint NFT truy xuất nguồn gốc | PATCH /product - batches/admin/:id/status (action: APPROVE → READY_FOR_SALE), kiểm tra GET /metadata/batch/:batchId và GET /verify/lookup trả về đúng tokenId trên Sepolia | Đạt |
| UC - 04 | Tạo listing, thêm vào giỏ hàng, thanh toán ví nội bộ | POST /listings, POST /carts/items, POST /orders/checkout - from - cart với INTERNAL_WALLET, kiểm tra số dư ví thay đổi đúng (script smoke:vietqr/smoke:auction) | Đạt |
| UC - 05 | Tạo phiên đấu giá, đặt giá VND và ETH, kết thúc phiên | POST /auctions, POST /bids, PATCH /auctions/:id/end; với ETH theo dõi qua AgriBidEscrow trên Sepolia và đối chiếu auctions.onChainSyncStatus | Đạt với luồng cốt lõi gồm tạo phiên, đặt giá và kết thúc phiên. Các kịch bản mở rộng liên quan đến refund, cancel, pause/unpause và đặt giá cạnh tranh trên smart contract cần tiếp tục hoàn thiện như đã phân tích ở mục 4.3. |

| **Use case** | **Kịch bản kiểm thử** | **Cách thực hiện** | **Kết quả** |
|---|---|---|---|
| UC - 06 | Livestream gắn listing/đấu giá, ghim sản phẩm, chat | POST /live - sessions/listing/start, POST /live - sessions/:id/pins/:listingId, kiểm tra token LiveKit và kênh chat realtime | Đạt |
| UC - 07 | Nạp/rút tiền qua PayOS và VietQR, payout cho Farmer | POST /payments/payos/wallet - topup - link, /payments/payos/webhook (giả lập), POST /payouts/requests, duyệt qua PATCH /payouts/requests/:id/admin/approve | Đạt |
| UC - 08 | Áp dụng voucher cho đơn hàng | POST /vouchers (Admin tạo), kiểm tra áp dụng giảm giá khi checkout | Đạt |

### *3.7.6. Nhận xét, đánh giá*

Kết quả kiểm thử cho thấy các luồng nghiệp vụ chính của hệ thống như xác thực, quản lý lô hàng, truy xuất nguồn gốc, mua bán, đấu giá, livestream, ví, thanh toán và payout đã hoạt động đúng theo thiết kế ở mức unit test Backend và kiểm thử thủ công theo use case. Cụ thể, bộ kiểm thử Backend đạt 84/84 test case PASS, cho thấy các service nghiệp vụ cốt lõi đã được kiểm tra tương đối đầy đủ trước khi tích hợp lên môi trường thực tế.

Đối với smart contract, hệ thống đã triển khai và kiểm thử bước đầu các chức năng nền tảng như phân quyền, tạo phiên đấu giá, đặt giá hợp lệ, kiểm tra điều kiện đặt giá và một số kịch bản kết thúc phiên. Tuy nhiên, do bộ kiểm thử smart contract hiện mới đạt 20/32 test case PASS, các kịch bản liên quan đến hoàn tiền, hủy phiên, tạm dừng hợp đồng, top-up phần chênh và đặt giá cạnh tranh mở rộng cần được tiếp tục rà soát. Đây là hạn chế quan trọng vì smart contract có liên quan trực tiếp đến tài sản ETH của người dùng.

Nhìn chung, hệ thống đã đáp ứng được các yêu cầu kiểm thử chính trong phạm vi đồ án, đặc biệt ở các luồng Backend, giao diện nghiệp vụ và kiểm thử use case. Tuy nhiên, trước khi triển khai trong môi trường giao dịch thực tế hoặc mainnet, cần hoàn thiện bộ kiểm thử smart contract để đạt kết quả PASS đầy đủ, đồng thời bổ sung kiểm thử bảo mật cho các hàm liên quan đến nhận, giữ và hoàn trả ETH.

## 3.8. Đánh giá kết quả và so sánh hệ thống

### *3.8.1. Mục đích và phương pháp đánh giá*

Mục 3.7 đã trình bày kết quả kiểm thử đúng/sai của từng module theo unit test, smoke test và use case. Chương này tập trung trả lời câu hỏi ở mức cao hơn: hệ thống AgriBid hoạt động tốt đến đâu về hiệu năng, có đáp ứng được đặc thù realtime của đấu giá/livestream không, và có những ưu - nhược điểm gì khi so sánh với các nền tảng tham chiếu đã nêu ở mục 1.2 và Bảng 1.1 của Chương 1. Phương pháp đánh giá gồm ba phần: (1) đo hiệu năng API và độ trễ realtime trên môi trường phát triển (development) bằng cách gọi trực tiếp các script smoke test và theo dõi log thời gian xử lý; (2) đánh giá chi phí giao dịch (gas) của hai smart contract trên Sepolia testnet thông qua các giao dịch thực thi trong bộ test Hardhat; (3) so sánh định tính các tiêu chí chức năng và mô hình giao dịch giữa AgriBid và các nền tảng tham chiếu.

### *3.8.2. Kết quả đánh giá hiệu năng*

Bảng 3.8 tổng hợp thời gian phản hồi trung bình của một số API tiêu biểu, đo trên môi trường development (NestJS chạy local, PostgreSQL/Redis container, 10 lần gọi liên tiếp cho mỗi API thông qua script smoke test):

Bảng 3.8 - Thời gian phản hồi trung bình của một số API tiêu biểu (môi trường development)

| **API** | **Thời gian phản hồi trung bình** | **Ghi chú** |
|---|---|---|
| GET /listings (phân trang, lọc danh mục) | ~80 ms | Truy vấn có join categories/product_batches, sử dụng index theo status và categoryId. |

| **API** | **Thời gian phản hồi trung bình** | **Ghi chú** |
|---|---|---|
| POST /bids (đặt giá đấu giá VND) | ~120 ms | Bao gồm kiểm tra hợp lệ giá đặt, cập nhật highestBid trong transaction, phát sự kiện WebSocket tới các client đang xem phiên. |
| POST /orders/checkout - from - cart (INTERNAL_WALLET) | ~150 ms | Bao gồm tạo đơn hàng, trừ ví, ghi nhận transaction, áp voucher (nếu có) trong một database transaction. |
| Độ trễ broadcast sự kiện đấu giá qua Socket.IO (bid mới → client khác nhận) | < 100 ms | Đo trong mạng nội bộ (localhost), chưa tính độ trễ mạng thực tế khi triển khai production. |
| Độ trễ tham gia phòng livestream LiveKit (room join → nhận luồng) | ~1 - 2 s | Phụ thuộc thời gian khởi tạo kết nối WebRTC, đo trên môi trường test với một publisher và một subscriber. |

Kết quả cho thấy các API chính đều có thời gian phản hồi dưới 200 ms trong môi trường development, đáp ứng yêu cầu trải nghiệm thời gian thực cho chức năng đấu giá (đặt giá và nhận cập nhật trong vòng dưới 100 ms). Độ trễ tham gia livestream (1–2 giây) là mức chấp nhận được cho ứng dụng bán hàng qua video, tương đương các nền tảng livestream commerce phổ biến.

### *3.8.3. Chi phí giao dịch trên smart contract*

Bảng 3.9 ước lượng chi phí gas của một số giao dịch chính trên hai smart contract AgriBidEscrow.sol và AgriBidNFT.sol, dựa trên báo cáo gas (gas report) sinh ra khi chạy bộ test Hardhat trên Sepolia testnet:

Bảng 3.9 - Ước lượng chi phí gas của các giao dịch chính trên smart contract (Sepolia testnet)

| **Giao dịch** | **Gas ước tính** | **Ghi chú** |
|---|---|---|
| createAuction (tạo phiên đấu giá) | ~120.000 - 150.000 gas | Ghi thông tin phiên đấu giá mới (farmer, mức giá khởi điểm, deposit tối thiểu, thời gian kết thúc) vào storage. |
| placeBid (đặt giá ETH) | ~60.000 - 90.000 gas | Cập nhật highestBidder/highestBid, cộng dồn deposit của bidder; chi phí tăng nhẹ khi cần hoàn trả cho bidder trước đó. |
| withdrawRefund (rút tiền hoàn) | ~35.000 - 45.000 gas | Chuyển ETH theo mô hình pull - payment, hạn chế rủi ro reentrancy so với việc hoàn trả tự động (push). |
| mint (AgriBidNFT, mint NFT truy xuất nguồn gốc) | ~90.000 - 110.000 gas | Mint token ERC - 721 mới và gán tokenURI trỏ tới metadata JSON của lô hàng. |

Chi phí gas của các giao dịch trên đều ở mức trung bình - thấp so với các hợp đồng phức tạp khác trên Ethereum, do thiết kế ưu tiên lưu trữ tối thiểu trên on - chain (chỉ lưu các trường cốt lõi phục vụ ký quỹ và xác thực, phần dữ liệu mô tả chi tiết được lưu off - chain trong PostgreSQL và đồng bộ qua BlockchainListener). Việc áp dụng mô hình pull - payment cho withdrawRefund giúp giảm rủi ro tấn công reentrancy nhưng yêu cầu người dùng tự thực hiện thêm một giao dịch để nhận lại tiền hoàn, đây là điểm đánh đổi giữa an toàn và trải nghiệm người dùng.

### *3.8.4. So sánh AgriBid với các nền tảng tham chiếu*

Dựa trên phần khảo sát hiện trạng và các hệ thống liên quan ở mục 1.2, đồng thời kế thừa các tiêu chí so sánh đã trình bày trong Bảng 1.1 (Chương 1), Bảng 3.10 tiếp tục so sánh AgriBid với hai nhóm nền tảng tham chiếu: (1) các nền tảng thương mại điện

tử nông sản/marketplace nói chung, và (2) các nền tảng đấu giá trực tuyến hoặc livestream commerce, theo các tiêu chí chức năng chính.

Bảng 3.10 - So sánh AgriBid với các nền tảng tham chiếu theo tiêu chí chức năng

| **Tiêu chí** | **Marketplace nông sản thông thường** | **Nền tảng đấu giá/livestream commerce** | **AgriBid** |
|---|---|---|---|
| Mua bán cố định (Listing) | Có | Một số có | Có |
| Đấu giá kiểu English Auction realtime | Không | Có (đa dạng ngành hàng) | Có, chuyên biệt cho nông sản theo lô hàng |
| Bán hàng qua livestream | Một số có | Có | Có, tích hợp ghim listing/đấu giá trong phòng live |
| Truy xuất nguồn gốc bằng NFT/blockchain | Không | Không | Có (AgriBidNFT, ERC - 721 trên Ethereum) |
| Thanh toán đa kênh (ví nội bộ, PayOS, VietQR, ETH) | Thường chỉ 1–2 kênh nội địa | Tùy nền tảng, ít hỗ trợ tiền điện tử | Có cả kênh VND và ETH qua smart contract |
| Ký quỹ/escrow minh bạch cho đấu giá | Không áp dụng | Tùy nền tảng, thường xử lý off - chain | Có (AgriBidEscrow, on - chain, pull - payment) |

Điểm khác biệt nổi bật của AgriBid là sự kết hợp đồng thời ba hình thức giao dịch (Listing, Auction, Live Session) trong cùng một nền tảng chuyên biệt cho nông sản theo lô hàng, cùng với việc ứng dụng NFT để truy xuất nguồn gốc và smart contract escrow để minh bạch hóa quá trình đấu giá bằng ETH - những yếu tố mà các nền tảng

marketplace nông sản thông thường hiện chưa có. Hạn chế hiện tại là hệ thống mới triển khai trên testnet (Sepolia) và chưa có dữ liệu thực tế từ người dùng thật, nên các số liệu hiệu năng và chi phí gas ở mục 3.8.2–3.8.3 mang tính ước lượng trên môi trường thử nghiệm, cần được đo lại khi triển khai production với lưu lượng thực tế.

### *3.8.5. Nhận xét chung*

Tổng hợp từ kết quả kiểm thử (Mục 3.7) và đánh giá ở Mục 3.8, hệ thống AgriBid đáp ứng được các yêu cầu phi chức năng cơ bản (thời gian phản hồi API dưới 200 ms, độ trễ realtime dưới 100 ms cho đấu giá, chi phí gas ở mức hợp lý cho các giao dịch cốt lõi) và có sự khác biệt rõ ràng so với các nền tảng tham chiếu nhờ kết hợp đấu giá, livestream và truy xuất nguồn gốc bằng blockchain trong cùng một hệ thống chuyên biệt cho nông sản. Các hạn chế về môi trường kiểm thử (đặc biệt là phần smart contract, đã nêu ở mục 3.7.3) và việc chưa có số liệu từ môi trường production là cơ sở cho các đề xuất ở phần hướng phát triển trong Kết luận.

# KẾT LUẬN

## 1. Kết quả đạt được

Sau quá trình phân tích, thiết kế và cài đặt, đồ án đã xây dựng được nền tảng AgriBid - hệ thống mua bán và đấu giá nông sản trực tuyến với các kết quả cụ thể:

- Xây dựng hoàn chỉnh Backend NestJS với 27 module nghiệp vụ, hơn 155 endpoint REST API được tài liệu hóa bằng Swagger, áp dụng phân quyền theo vai trò (FARMER, MERCHANT, ADMIN) cho từng endpoint.
- Thiết kế và triển khai cơ sở dữ liệu PostgreSQL với các bảng phản ánh đầy đủ nghiệp vụ: người dùng/KYC, lô hàng & media, danh mục/sản phẩm, listing, giỏ hàng, đơn hàng, đấu giá/đặt giá, livestream, ví/giao dịch, voucher, payout, phí nền tảng.
- Tích hợp blockchain Ethereum (Sepolia testnet): hợp đồng AgriBidEscrow.sol quản lý ký quỹ và thanh toán cho phiên đấu giá bằng ETH, hợp đồng AgriBidNFT.sol (ERC - 721) mint NFT truy xuất nguồn gốc cho từng lô hàng đã được duyệt, cùng BlockchainListener đồng bộ trạng thái on - chain về cơ sở dữ liệu.
- Tích hợp thanh toán đa kênh: ví nội bộ (INTERNAL_WALLET), cổng PayOS (tạo link thanh toán, webhook xác nhận, nạp tiền vào ví) và chuyển khoản VietQR (biên lai Level 2, quy trình duyệt thủ công).
- Xây dựng tính năng livestream bán hàng/đấu giá tích hợp LiveKit, cho phép Farmer mở phiên live, ghim sản phẩm/listing, và người xem tương tác qua chat realtime.
- Xây dựng giao diện Client (Next.js) cho Farmer/Merchant với đầy đủ luồng marketplace, đấu giá, livestream, ví MetaMask, và giao diện Admin (Next.js/MUI) phục vụ quản trị KYC, danh mục, lô hàng, đơn hàng, đấu giá, payout, phí nền tảng.
- Thực hiện kiểm thử tự động (84/84 unit test Backend PASS, 20/32 test hợp đồng thông minh PASS) và kiểm thử thủ công theo 8 use case chính, xác nhận các luồng nghiệp vụ trọng yếu hoạt động đúng thiết kế.

Để đánh giá mức độ hoàn thành đề tài một cách có căn cứ, Bảng K.1 đối chiếu 5 mục tiêu đã đề ra ở phần Mở đầu (mục 2) với các kết quả đạt được tương ứng đã trình bày ở mục 1 trên:

Bảng K.1 - Đối chiếu mục tiêu đề tài và kết quả đạt được

| **Mục tiêu đề ra (Mở đầu, mục 2)** | **Kết quả đạt được (Kết luận, mục 1)** |
|---|---|
| Xây dựng backend NestJS/TypeORM/PostgreSQL/Redis/Bull/Socket.IO cung cấp API cho Listing, Auction, Live Session, ví điện tử VND, PayOS và VietQR. | Hoàn thành: 27 module nghiệp vụ, hơn 155 endpoint REST tài liệu hóa Swagger, phân quyền theo vai trò; tích hợp ví INTERNAL_WALLET, PayOS và VietQR. |
| Xây dựng giao diện Client (Next.js 16/React 19) cho Farmer/Merchant: marketplace, đấu giá realtime, livestream, giỏ hàng, thanh toán, portal Farmer. | Hoàn thành: giao diện Client đầy đủ luồng marketplace, đấu giá, livestream tích hợp LiveKit, ví MetaMask. |
| Xây dựng giao diện Admin (Next.js 14 + MUI): duyệt KYC, lô hàng, quản lý danh mục, đơn hàng/đấu giá, payout, phí nền tảng. | Hoàn thành: giao diện Admin phục vụ quản trị KYC, danh mục, lô hàng, đơn hàng, đấu giá, payout, phí nền tảng. |
| Triển khai hai smart contract Ethereum (Hardhat, Solidity 0.8.28, OpenZeppelin): AgriBidEscrow.sol (đấu giá ETH, pull - payment) và AgriBidNFT.sol (NFT ERC - 721 truy xuất nguồn gốc). | Hoàn thành: hai hợp đồng đã triển khai trên Sepolia testnet, có BlockchainListener đồng bộ on - chain/off - chain; 20/32 test case smart contract PASS (còn 12 test case FAIL, xem mục 2 - Hạn chế). |
| Đảm bảo an toàn giao dịch tài chính: database transaction + pessimistic locking, idempotency - key cho API quan trọng, đồng bộ hai chiều backend - blockchain. | Hoàn thành: cơ chế idempotency - key (x - idempotency - key + băm SHA - 256) và frozenBalance đã được áp dụng; 84/84 unit test Backend PASS xác nhận các luồng tài chính trọng yếu hoạt động đúng thiết kế. |

Đối chiếu với các mục tiêu đã đề ra, hệ thống AgriBid đã cơ bản hoàn thành các nhóm chức năng chính gồm backend, frontend client, frontend admin, ví nội bộ,

thanh toán, đấu giá, livestream, quản trị và đồng bộ blockchain. Đối với mục tiêu smart contract, hệ thống đã triển khai được hai hợp đồng AgriBidEscrow và AgriBidNFT trên Sepolia testnet, đồng thời tích hợp cơ chế đồng bộ on-chain/off-chain thông qua BlockchainListener. Tuy nhiên, bộ kiểm thử smart contract hiện mới đạt 20/32 test case PASS, do đó phần hợp đồng thông minh cần tiếp tục được rà soát và hoàn thiện trước khi có thể xem là đủ điều kiện triển khai trong môi trường giao dịch thực tế.

**2. Hạn chế**

- Một số kịch bản kiểm thử hợp đồng thông minh (rút tiền hoàn trả, hủy phiên, tạm dừng hợp đồng, đặt giá đồng thời mở rộng) chưa cho kết quả PASS đầy đủ do chênh lệch phiên bản bộ công cụ kiểm thử Hardhat/chai - matchers, cần được rà soát và nâng cấp.
- Hệ thống mới được kiểm thử trên mạng Sepolia testnet, chưa triển khai và đánh giá hiệu năng trên mạng chính thức (mainnet) hoặc môi trường production với lưu lượng lớn.
- Quy trình duyệt biên lai chuyển khoản VietQR và duyệt payout hiện còn phụ thuộc nhiều vào xử lý thủ công của Admin, có thể gây chậm trễ khi số lượng giao dịch tăng cao.
- Chưa có cơ chế kiểm thử tải (load test/stress test) cho các API có lưu lượng cao như đặt giá đấu giá realtime và kênh WebSocket.

**3. Hướng phát triển**

- Đồng bộ phiên bản bộ công cụ kiểm thử Hardhat/chai - matchers và bổ sung các kịch bản kiểm thử còn thiếu cho hợp đồng AgriBidEscrow, hướng tới kiểm thử bảo mật (audit) trước khi triển khai mainnet.
- Tự động hóa quy trình đối soát thanh toán VietQR (tích hợp API ngân hàng/OCR biên lai) để giảm phụ thuộc vào xử lý thủ công.
- Bổ sung kiểm thử tải cho các API realtime (đấu giá, livestream) và tối ưu kiến trúc WebSocket khi mở rộng quy mô người dùng.

- Mở rộng hệ thống truy xuất nguồn gốc: tích hợp thêm dữ liệu IoT (cảm biến nhiệt độ, độ ẩm trong quá trình vận chuyển) gắn vào metadata NFT.
- Phát triển ứng dụng di động (mobile app) cho Farmer/Merchant để thuận tiện trong việc cập nhật lô hàng, theo dõi phiên đấu giá và livestream.
- Tích hợp thêm các nguồn dữ liệu xác thực độc lập cho truy xuất nguồn gốc, chẳng hạn như thiết bị IoT, cảm biến môi trường, dữ liệu kiểm định chất lượng từ bên thứ ba hoặc chứng nhận từ cơ quan chuyên môn. Việc này giúp nâng cao độ tin cậy của dữ liệu đầu vào trước khi thông tin lô hàng được ghi nhận và phát hành NFT trên blockchain.

# TÀI LIỆU THAM KHẢO

[1] World Bank, Digital Technology for Traceability in Vietnam's Fruit and Vegetable Value Chains, World Bank Group, 2022. [Online]. Available: https://documents1.worldbank.org/curated/en/099724310182330688/pdf/IDU03bd6b7680aa0804d260a4120c0c008620286.pdf. [Accessed: Jun. 15, 2026].

[2] A. Wongkitrungrueng and N. Assarut, "The role of live streaming in building consumer trust and engagement with social commerce sellers," Journal of Business Research, vol. 117, pp. 543–556, 2020, doi: 10.1016/j.jbusres.2018.08.032.

[3] P. Bajari and A. Hortaçsu, "Economic insights from Internet auctions," Journal of Economic Literature, vol. 42, no. 2, pp. 457–486, 2004, doi: 10.1257/0022051041409075.

[4] A. Kamilaris, A. Fonts, and F. X. Prenafeta-Boldú, "The rise of blockchain technology in agriculture and food supply chains," Trends in Food Science & Technology, vol. 91, pp. 640–652, 2019, doi: 10.1016/j.tifs.2019.07.034.

[5] K. Demestichas, N. Peppes, T. Alexakis, and E. Adamopoulou, "Blockchain in agriculture traceability systems: A review," Applied Sciences, vol. 10, no. 12, Art. no. 4113, 2020, doi: 10.3390/app10124113.

[6] J. F. Galvez, J. C. Mejuto, and J. Simal-Gandara, "Future challenges on the use of blockchain for food traceability analysis," TrAC Trends in Analytical Chemistry, vol. 107, pp. 222–232, 2018, doi: 10.1016/j.trac.2018.08.011.

[7] N. Kshetri, "Blockchain's roles in meeting key supply chain management objectives," International Journal of Information Management, vol. 39, pp. 80–89, 2018, doi: 10.1016/j.ijinfomgt.2017.12.005.

[8] F. Tian, "An agri-food supply chain traceability system for China based on RFID & blockchain technology," in Proc. 13th International Conference on Service Systems and Service Management (ICSSSM), Kunming, China, 2016, pp. 1–6, doi: 10.1109/ICSSSM.2016.7538424.

[9] W. Entriken, D. Shirley, J. Evans, and N. Sachs, "ERC-721: Non-Fungible Token Standard," Ethereum Improvement Proposals, EIP-721, 2018. [Online]. Available: https://eips.ethereum.org/EIPS/eip-721. [Accessed: Jun. 15, 2026].

[10] NestJS, "Documentation," [Online]. Available: https://docs.nestjs.com. [Accessed: Jun. 15, 2026].

[11] TypeORM, "TypeORM Documentation," [Online]. Available: https://typeorm.io. [Accessed: Jun. 15, 2026].

[12] PostgreSQL Global Development Group, "PostgreSQL 15 Documentation," [Online]. Available: https://www.postgresql.org/docs/15/. [Accessed: Jun. 15, 2026].

[13] Redis, "Redis Documentation," [Online]. Available: https://redis.io/docs/latest/. [Accessed: Jun. 15, 2026].

[14] LiveKit, "LiveKit Documentation," [Online]. Available: https://docs.livekit.io/intro/overview/. [Accessed: Jun. 15, 2026].

[15] Hardhat, "Hardhat Documentation," [Online]. Available: https://hardhat.org/docs/getting-started. [Accessed: Jun. 15, 2026].

[16] OpenZeppelin, "OpenZeppelin Contracts Documentation," [Online]. Available: https://docs.openzeppelin.com/contracts/5.x. [Accessed: Jun. 15, 2026].

[17] Ethers, "Ethers.js Documentation," [Online]. Available: https://docs.ethers.org/v6/. [Accessed: Jun. 15, 2026].

[18] payOS, "payOS API Documentation," [Online]. Available: https://payos.vn/docs/api/. [Accessed: Jun. 15, 2026].

[19] Next.js, "Next.js Documentation," [Online]. Available: https://nextjs.org/docs. [Accessed: Jun. 15, 2026].

# PHỤ LỤC

## 1. Bảng từ viết tắt và thuật ngữ

Bảng PL.1 giải thích các từ viết tắt và thuật ngữ chuyên ngành được sử dụng xuyên suốt báo cáo, theo thứ tự xuất hiện từ Mở đầu đến Chương 3:

Bảng PL.1 - Bảng từ viết tắt và thuật ngữ

| **Từ viết tắt / thuật ngữ** | **Giải thích** |
|---|---|
| KYC | Know Your Customer - quy trình xác minh danh tính người dùng (Farmer) trước khi cho phép bán hàng trên hệ thống. |
| NFT | Non - Fungible Token - token không thể thay thế, dùng làm chứng chỉ truy xuất nguồn gốc duy nhất cho từng lô hàng nông sản. |
| ERC - 721 | Chuẩn token không thể thay thế (NFT) trên Ethereum, được AgriBidNFT.sol triển khai để mint chứng chỉ lô hàng. |
| ERC | Ethereum Request for Comments - các chuẩn kỹ thuật cho hợp đồng thông minh trên Ethereum (ví dụ ERC - 721). |
| Idempotency / Idempotency - key | Cơ chế đảm bảo một yêu cầu API được gửi nhiều lần (do lỗi mạng, double click...) chỉ tạo ra một kết quả duy nhất, thực hiện qua header x - idempotency - key và băm SHA - 256. |
| EDR | Ethereum Development Runtime - môi trường chạy mạng Ethereum cục bộ tích hợp trong Hardhat 3, dùng để biên dịch và kiểm thử smart contract. |
| PayOS | Cổng thanh toán trực tuyến của Việt Nam, được tích hợp để tạo link thanh toán, nhận webhook xác nhận và nạp tiền vào ví nội bộ. |
| VietQR | Chuẩn mã QR thanh toán liên ngân hàng tại Việt Nam, dùng cho luồng chuyển khoản trực tiếp kèm xác nhận biên lai thủ công. |
| JWT | JSON Web Token - cơ chế xác thực dựa trên token, dùng cho access token/refresh token trong NestJS (Passport JWT strategy). |
| RBAC | Role - Based Access Control - kiểm soát truy cập theo vai trò (FARMER, MERCHANT, ADMIN), cài đặt qua RolesGuard. |

| **Từ viết tắt / thuật ngữ** | **Giải thích** |
|---|---|
| ACID | Atomicity, Consistency, Isolation, Durability - bốn tính chất của transaction cơ sở dữ liệu, đảm bảo tính toàn vẹn cho các nghiệp vụ tài chính (giữ/giải phóng tiền, đặt giá). |
| Pessimistic locking | Cơ chế khóa bi quan ở mức bản ghi cơ sở dữ liệu (SELECT ... FOR UPDATE), ngăn race condition khi nhiều giao dịch cùng cập nhật số dư ví hoặc giá đấu cao nhất. |
| REST / RESTful API | Representational State Transfer - kiến trúc giao tiếp client - server qua HTTP theo tài nguyên (resource), được Backend triển khai cho hơn 100 endpoint. |
| ABI | Application Binary Interface - định nghĩa giao diện gọi hàm của smart contract, dùng bởi ethers.js để Backend/Frontend tương tác với AgriBidEscrow và AgriBidNFT. |
| RPC | Remote Procedure Call - giao thức Backend/Frontend dùng để gửi yêu cầu tới node Ethereum (qua BLOCKCHAIN_RPC_URL / NEXT_PUBLIC_BLOCKCHAIN_RPC_URL). |
| Pull - payment | Mô hình thanh toán trong AgriBidEscrow.sol: hợp đồng giữ tiền ký quỹ và người thắng/người hoàn tiền phải tự gọi hàm rút (withdraw) thay vì hợp đồng tự đẩy tiền, giúp hạn chế rủi ro reentrancy. |
| Webhook | Cơ chế PayOS gọi ngược về Backend để thông báo kết quả thanh toán (thành công/thất bại) theo thời gian thực. |
| WebRTC | Web Real - Time Communication - công nghệ truyền video/âm thanh thời gian thực, nền tảng cho tính năng livestream qua LiveKit. |
| Sepolia | Mạng thử nghiệm (testnet) của Ethereum, dùng để triển khai và kiểm thử AgriBidEscrow.sol/AgriBidNFT.sol trước khi (nếu) lên mainnet. |
| Frozen balance | Số dư bị tạm giữ trong ví nội bộ của người dùng khi tham gia đấu giá VND, được giải phóng hoặc trừ tùy kết quả phiên đấu giá. |

## 2. Hướng dẫn cài đặt và chạy hệ thống

Phần này tổng hợp quy trình cài đặt, triển khai smart contract và kiểm tra hệ thống, dựa trên các tài liệu vận hành thực tế của dự án

(BLOCKCHAIN_DEPLOY_CHECKLIST.md, DEMO_PHASE_A_CHECKLIST.md, DEMO_PHASE_B_CHECKLIST.md) [20].

### *2.1. Cài đặt và chạy Backend, Frontend*

- Khởi động PostgreSQL và Redis bằng Docker Compose, sao chép .env.example → .env và điền giá trị thực tế cho từng package (xem Bảng 3.1 - Chương 3).
- Backend: npm install, npm run migration:run để khởi tạo schema, npm run seed (và các script seed - mint, seed - images, seed - long - demo - auctions, seed - listing - reviews) để có dữ liệu demo, npm run start:dev để chạy ở chế độ phát triển (Swagger tại /api).
- Frontend Client/Admin: npm install, npm run dev; đảm bảo NEXT_PUBLIC_API_URL trỏ đúng địa chỉ Backend (Admin cần cùng host với Backend để cookie refresh hoạt động).

### *2.2. Triển khai smart contract (Sepolia testnet)*

Quy trình triển khai và đồng bộ địa chỉ hợp đồng giữa 3 package (Blockchain → Backend → Client), theo BLOCKCHAIN_DEPLOY_CHECKLIST.md:

- Bước 1 - Biên dịch và kiểm thử: cd AgriBid - Blockchain && npm install && npm run compile && npm test (chạy bộ test Hardhat/Mocha/Chai cho AgriBidEscrow và AgriBidNFT).
- Bước 2 - Triển khai: cp .env.example .env, điền PRIVATE_KEY (ví triển khai) và RPC_URL (Sepolia, chain ID 11155111), sau đó npm run deploy:sepolia; ghi lại NFT_CONTRACT_ADDRESS và ESCROW_CONTRACT_ADDRESS được in ra.
- Bước 3 - Cấp quyền cho ví Backend trên hợp đồng escrow: npm run roles:escrow - - - - action=grant - - role=manager - - account=0x... và tương tự với - - role=pauser, dùng địa chỉ ví hệ thống (BLOCKCHAIN_PRIVATE_KEY của Backend).
- Bước 4 - Cập nhật Backend .env: điền BLOCKCHAIN_RPC_URL, BLOCKCHAIN_CHAIN_ID=11155111, BLOCKCHAIN_PRIVATE_KEY, NFT_CONTRACT_ADDRESS, ESCROW_CONTRACT_ADDRESS, BLOCKCHAIN_LISTENER_AUTOSTART,

BLOCKCHAIN_POLL_INTERVAL_MS, sau đó khởi động lại Backend; kiểm tra ABI khớp trong blockchain.constant.ts.

- Bước 5 - Kiểm tra nhanh (smoke test): curl http://localhost:8000/blockchain/client - config phải trả về đúng địa chỉ hợp đồng và chain ID vừa cấu hình.
- Bước 6 - Cập nhật Frontend Client .env: NEXT_PUBLIC_BLOCKCHAIN_CHAIN_ID, NEXT_PUBLIC_BLOCKCHAIN_RPC_URL và NEXT_PUBLIC_ESCROW_CONTRACT_ADDRESS phải khớp với giá trị đã cấu hình ở Backend, để ví MetaMask của người dùng kết nối đúng mạng và đúng hợp đồng.
- Lưu ý: với môi trường phát triển cục bộ có thể dùng Hardhat node (chain ID 31337) thay cho Sepolia, áp dụng quy trình tương tự.

### *2.3. Kiểm tra hệ thống bằng smoke test (demo checklist)*

Hai bộ checklist demo (Phase A - nghiệp vụ VND/PayOS/VietQR, Phase B - đấu giá ETH on - chain/NFT) cung cấp script smoke test tự động để xác minh hệ thống hoạt động đúng trước khi demo:

- Phase A (VietQR, đấu giá VND, chat, NFT/verify): chạy node scripts/vietqr - controls - smoke.mjs (pass khi in ra "All VietQR control smoke checks passed.") và node scripts/auction - demo - smoke.mjs (pass khi gọi POST chat - messages và chat - history thành công với một phiên đấu giá đang HAPPENING). Tài khoản demo: farmer_vip_1, merchant_vip_1, admin_vip_1 (mật khẩu password123).
- Phase B (đấu giá ETH, NFT on - chain): trước tiên kiểm tra B0 - các biến ESCROW_CONTRACT_ADDRESS, NFT_CONTRACT_ADDRESS, BLOCKCHAIN_RPC_URL, BLOCKCHAIN_CHAIN_ID đã khớp giữa Backend và Client, mạng MetaMask đúng (Hardhat 31337 hoặc Sepolia). Nếu phiên đấu giá BLOCKCHAIN thiếu onChainId, chạy npm run blockchain:sync - auctions, sau đó npm run smoke:blockchain (pass khi client - config trả về

đúng và thấy phiên đấu giá BLOCKCHAIN có onChainId), hoặc npm run smoke:phase - b để chạy toàn bộ.

- Quy trình demo B1 minh họa luồng đặt giá ETH thực tế: đăng nhập merchant_vip_1, lưu địa chỉ ví ETH khớp MetaMask, mở phiên đấu giá BLOCKCHAIN, gửi giao dịch placeBid - tx hash hiển thị và BlockchainListener đồng bộ bid vào cơ sở dữ liệu.

## 3. Link Github

### *3.1. Backend*

- https://github.com/HieuChiChau/AgriBid-Backend

### *3.2. Frontend Client*

- https://github.com/HieuChiChau/AgriBid-Frontend-Client

### *3.3. Frontend Admin*

- https://github.com/HieuChiChau/AgriBid-Frontend-Admin

### *3.4. Blockchain*

- https://github.com/HieuChiChau/AgriBid-Blockchain