 **TRƯỜNG ĐAI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**----🙣🕮🙡----**



**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ TRAO ĐỔI BẤT ĐỘNG SẢN TRÊN NỀN CHUỖI KHỐI BLOCKCHAIN**

**GVHD: Thầy Trần Thanh Tùng**

**SVTH MSSV**

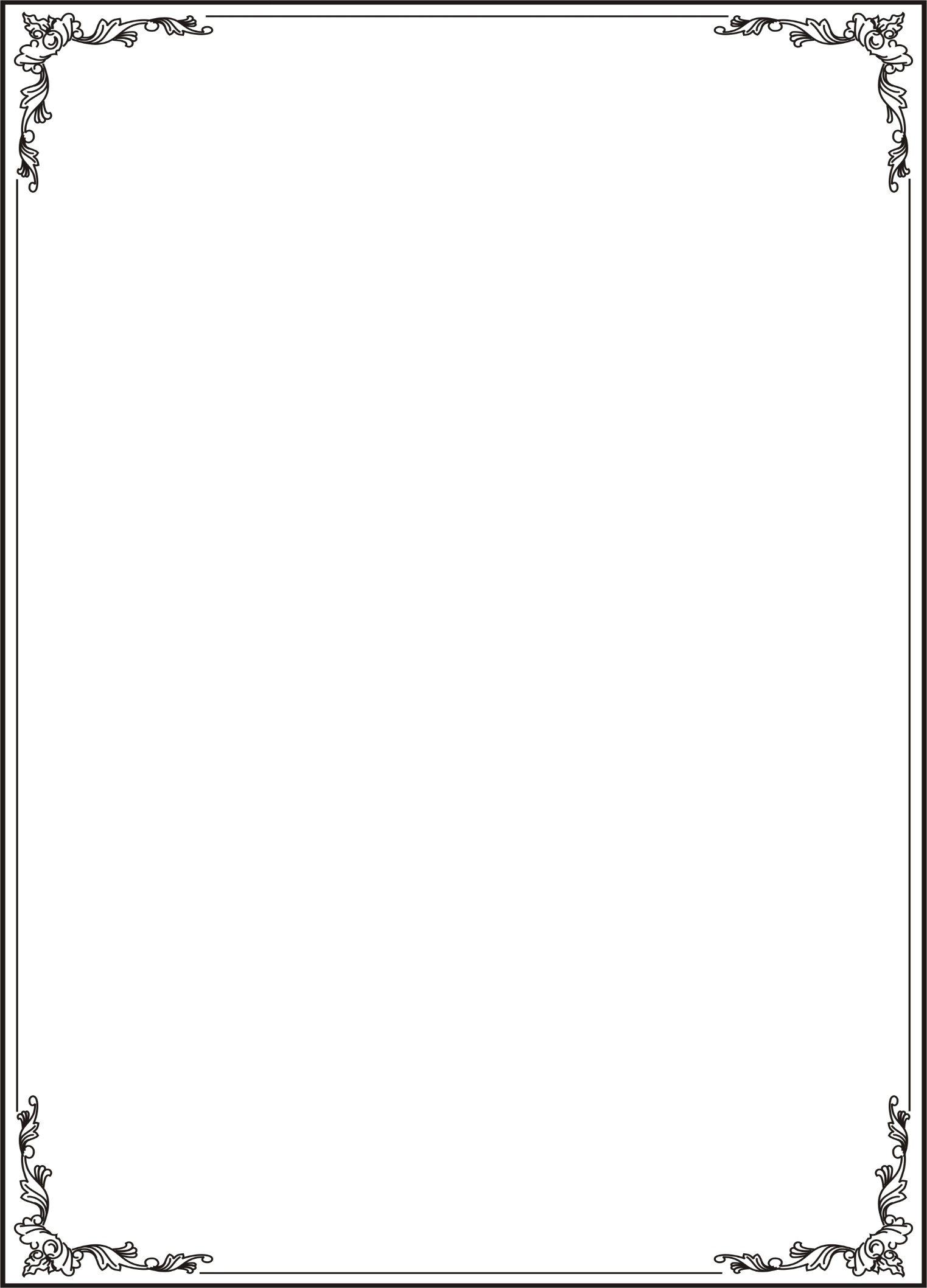
**Phan Bảo Khôi 15110065**

**Phan Dương Pha 15110094**

**Khóa: 2015 – 2019**

**Ngành: Công nghệ thông tin**

Tp. Hồ Chí Minh, tháng 7 năm 2019

**TRƯỜNG ĐAI HỌC SƯ PHẠM KỸ THUẬT TP.HCM**

**KHOA ĐÀO TẠO CHẤT LƯỢNG CAO**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**----🙣🕮🙡----**



**KHÓA LUẬN TỐT NGHIỆP**

**XÂY DỰNG HỆ THỐNG QUẢN LÝ VÀ TRAO ĐỔI BẤT ĐỘNG SẢN TRÊN NỀN CHUỖI KHỐI BLOCKCHAIN**

**GVHD: Thầy Trần Thanh Tùng**

**SVTH MSSV**

**Phan Bảo Khôi 15110065**

**Phan Dương Pha 15110094**

**Khóa: 2015 – 2019**

**Ngành: Công nghệ thông tin**

Tp. Hồ Chí Minh, tháng 7 năm 2019

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\***

Tp. Hồ Chí Minh, tháng 7 năm 2019

# **NHIỆM VỤ KHÓA LUẬN TỐT NGHIỆP**

Họ và tên sinh viên 1: Phan Bảo Khôi MSSV: 15110065

Họ và tên sinh viên 2: Phan Dương Pha MSSV:15110094

Ngành: Công nghệ thông tin Lớp: 15110CL2B

Giảng viên hướng dẫn: Thầy Trần Thanh Tùng ĐT: 0909090028

Ngày nhận đề tài: 12/2/2019 Ngày nộp đề tài: 1/7/2019

1. Tên đề tài: Xây dựng hệ thống trao đổi bất động sản trên nền chuỗi khối Blockchain
2. Các số liệu, tài liệu ban đầu: Khóa luận phát triển dựa trên Tiểu luận chuyên ngành
3. Nội dung thực hiện đề tài: Xây dựng hệ thống trao đổi bất động sản trên nền tảng web với các công nghệ ReactJS, Blockchain.
4. Sản phẩm: Trang web quản lý và trao đổi bất động sản chạy trên nền tảng Blockchain. Cho phép đăng bán các sản phẩm liên quan đến bất động sản.

TRƯỞNG NGÀNH GIẢNG VIÊN HƯỚNG DẪN

 **CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\***

# **PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN**

Họ và tên Sinh viên 1:………………………………………………….MSSV:…………………...

Họ và tên Sinh viên 2:………………………………………………….MSSV:…………………...

Tên đề tài:…………………………………………………………………………………………...

………………………………………………………………………………………………………

Họ và tên Giáo viên hướng dẫn:…………………………………………………………………….

**NHẬN XÉT**

1. Về nội dung đề tài và khối lượng thực hiện:

………………………………………………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

1. Ưu điểm:

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………………………………………………

1. Khuyết điểm:

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

1. Đề nghị cho bảo vệ hay không?

………………………………………………………………………………………………

1. Đánh giá loại:

………………………………………………………………………………………………

1. Điểm:………… (Bằng chữ:………………………………………………………..)

Tp. Hồ Chí Minh, tháng 7 năm 2019

Giáo viên hướng dẫn

(Ký và ghi rõ họ tên)

**CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**Độc lập – Tự do – Hạnh phúc**

**\*\*\*\*\*\*\***

# **PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN**

Họ và tên Sinh viên 1:………………………………………………….MSSV:…………………...

Họ và tên Sinh viên 2:………………………………………………….MSSV:…………………...

Tên đề tài:…………………………………………………………………………………………...

………………………………………………………………………………………………………

Họ và tên Giáo viên phản biện:…………………………………………………………………….

**NHẬN XÉT**

1. Về nội dung đề tài và khối lượng thực hiện:

………………………………………………………………………………………………………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………

1. Ưu điểm:

………………………………………………………………………………………………

………………………………………………………………………………………………

………………………………………………………………………………………………………………………………………………………………………………………………

1. Khuyết điểm:

………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………………

1. Đề nghị cho bảo vệ hay không?

………………………………………………………………………………………………

1. Đánh giá loại:

………………………………………………………………………………………………

1. Điểm:………… (Bằng chữ:………………………………………………………..)

Tp. Hồ Chí Minh, tháng 7 năm 2019

Giáo viên phản biện

(Ký và ghi rõ họ tên)

# **LỜI CẢM ƠN**

Chúng em muốn bày tỏ lòng biết ơn sâu sắc và chân thành đến thầy Trần Thanh Tùng, Thầy đã tận tình hướng dẫn và tạo mọi điều kiện cho chúng em trong quá trình tìm hiểu, nghiên cứu và thực hiện khóa luận tốt nghiệp. Dưới sự hướng dẫn của Thầy, chúng em đã học hỏi được rất nhiều kiến thức mới và hay. Sự tận tình và nhiệt huyết của Thầy là nguồn động lực dồi dào, thúc đẩy chúng em không ngừng nỗ lực để hoàn thành khóa luận một cách tốt nhất.

Chúng em xin chân thành cảm ơn các Thầy Cô ngành Công nghệ thông tin của khoa Đào tạo Chất Lượng Cao của Trường Đại học Sư Phạm Kỹ Thuật Thành phố Hồ Chí Minh đã truyền đạt những bài học, kiến thức quan trọng, tạo điều kiện cho chúng em học hỏi và có được nhiều kiến thức bổ ích, giúp chúng em có thể thực hiện khóa luận và đạt được kết quả như ngày hôm nay. Kiến thức hôm nay sẽ là hành trang giúp chúng em vững tin bước vào cuộc sống sau khi rời khỏi ghế nhà trường.

Chúng em cũng xin gửi lời tri ân đến gia đinh, bạn bè và người thân đã động viên và khích lệ chúng em trong thời gian học tập và thực hiện Khóa luận tốt nghiệp.

Mặc dù đã nỗ lực cố gắng song khóa luận này chắc chắn không thể tránh khỏi sai sót, khuyết điểm. Chúng em kính mong nhận được sự thông cảm và tận tình chỉ bảo, góp ý của quý Thầy Cô và các bạn.

Thành phố Hồ Chí Minh, tháng 6 năm 2019

Phan Bảo Khôi – Phan Dương Pha

# **TÓM TẮT**

Internet xuất hiện không chỉ phục vụ cho việc gửi email hay tải phần mềm mà nó còn là động lực để phát triển nền kinh tế toàn cầu. Trong thực tế, Internet đã trở thành trình điều khiển của nền kinh tế. Sự xuất hiện của Internet và các mạng cục bộ đã giúp cho việc trao đổi thông tin trở nên nhanh chóng, dễ dàng hơn. Email cho phép chúng ta nhận hay gửi thư ngay trên máy tính của mình, Ebusiness cho phép thực hiện giao dịch, buôn bán trên mạng… Cũng giống như Internet, blockchain xuất phát như một trào lưu với việc phát triển thêm nền tảng Ethereum để tạo ra các vụ thương thảo trên mạng.

Sự phát triển của Internet cũng đồng hành với những giao dịch trên mạng diễn ra thường xuyên hơn. Tuy nhiên, việc đảm bảo cho việc thành công của những giao dịch đó còn rất hạn chế và còn phụ thuộc nhiều vào bên trung gian. Đã có rất nhiều vụ lừa đảo khi thực hiện mua bán, trao đổi qua mạng.

Từ những rủi ro nêu trên nên các tổ chức tài chính cần những công nghệ mới, chính là Blockchain, được kì vọng không chỉ nhằm đảm bảo rằng các giao dịch trên mạng luôn an toàn mà còn đảm bảo tính an toàn và xa hơn nữa là cách mạng hóa các giải pháp bảo mật.

**MỤC LỤC**

[**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN HƯỚNG DẪN** iv](#_Toc12901457)

[**PHIẾU NHẬN XÉT CỦA GIÁO VIÊN PHẢN BIỆN** vi](#_Toc12901458)

[**LỜI CẢM ƠN** viii](#_Toc12901459)

[**TÓM TẮT** ix](#_Toc12901460)

[**DANH MỤC VIẾT TẮT** xii](#_Toc12901461)

[**DANH MỤC HÌNH ẢNH** xiii](#_Toc12901462)

[**DANH MỤC BẢNG BIỂU** xv](#_Toc12901463)

[**CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI** 16](#_Toc12901464)

[**1.** **Tên đề tài** 16](#_Toc12901465)

[**2.** **Mục tiêu của đề tài** 16](#_Toc12901466)

[**3.** **Đối tượng, phạm vi nghiên cứu của đề tài** 16](#_Toc12901467)

[**3.1.** **Đối tượng nghiên cứu** 16](#_Toc12901468)

[**3.2.** **Phạm vi nghiên cứu** 16](#_Toc12901469)

[**3.3.** **Ý nghĩa của đề tài** 16](#_Toc12901470)

[**3.4.** **Mô hình hệ thống** 17](#_Toc12901471)

[**Nghiệp vụ:** 17](#_Toc12901472)

[**CHƯƠNG 2. CƠ SỞ LÝ THUYẾT** 18](#_Toc12901473)

[**1.** **Sơ lược về Blockchain** 18](#_Toc12901474)

[**1.1.** **Các thành phần công nghệ của chuỗi khối** 18](#_Toc12901475)

[**1.2.** **Cách hoạt động của Blockchain.** 19](#_Toc12901476)

[**1.3.** **Các đặc điểm của Blockchain** 20](#_Toc12901477)

[**1.3.1.** **Mô hình mạng phân tán (Distributed Peer – 2 – Peer)** 20](#_Toc12901478)

[**1.3.2.** **Hàm Hash** 21](#_Toc12901479)

[**1.3.3.** **Chữ ký số (Digital Signature)** 21](#_Toc12901480)

[**1.3.4.** **Mã hóa đối xứng và bất đối xứng** 21](#_Toc12901481)

[**2.** **Sơ lược về Ethereum** 22](#_Toc12901482)

[**3.** **Giới thiệu về Smart Contract** 22](#_Toc12901483)

[**3.1.** **Khái niệm** 22](#_Toc12901484)

[**3.2.** **Cách vận hành của Smart Contract** 23](#_Toc12901485)

[**4.** **Solidity** 23](#_Toc12901486)

[**4.1.** **Solidity là gì?** 23](#_Toc12901487)

[**4.2.** **Các khái niệm cơ bản trong solidity** 23](#_Toc12901488)

[**5.** **Giới thiệu IPFS** 24](#_Toc12901489)

[**5.1.** **Khái niệm** 24](#_Toc12901490)

[**5.2.** **Những điểm mạnh của IPFS** 25](#_Toc12901491)

[**5.3.** **Cài đặt IPFS** 26](#_Toc12901492)

[**6.** **Giới thiệu ReactJS** 30](#_Toc12901493)

[**6.1.** **Giới thiệu** 30](#_Toc12901494)

[**6.2.** **React là gì** 30](#_Toc12901495)

[**6.3.** **Một số khái niệm cơ bản trong React** 31](#_Toc12901496)

[**6.3.1.** **Virtual DOM** 31](#_Toc12901497)

[**6.3.2.** **JSX là gì?** 31](#_Toc12901498)

[**6.3.3.** **Single Page Application (SPA)** 31](#_Toc12901499)

[**6.3.4.** **React - router là gì?** 33](#_Toc12901500)

[**6.3.5.** **Các thành phần trong React - router** 33](#_Toc12901501)

[**6.3.5.1.** **BrowserRouter và HashRouter** 33](#_Toc12901502)

[**6.3.5.2.** **Route** 34](#_Toc12901503)

[**7.** **Web3.js** 34](#_Toc12901504)

[**8.** **Metamask** 34](#_Toc12901505)

[**9.** **Truffle** 34](#_Toc12901506)

[**9.1.** **Khái niệm** 34](#_Toc12901507)

[**9.2.** **Cài đặt Truffle** 34](#_Toc12901508)

[**CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG** 38](#_Toc12901509)

[**1.** **Đăc tả yêu cầu** 38](#_Toc12901510)

[**1.1.** **Mục đích phạm vi của hệ thống** 38](#_Toc12901511)

[**1.2.** **Yêu cầu chức năng** 38](#_Toc12901512)

[**2.** **Thiết kế hệ thống** 39](#_Toc12901513)

[**2.1. Thiết kế kiến trúc hệ thống** 39](#_Toc12901514)

[**2.2. Thiết kế Use Case Diagram** 46](#_Toc12901515)

[**2.3.** **Sequence diagram** 68](#_Toc12901516)

[**2.4.** **Activity diagram** 72](#_Toc12901517)

[**CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI** 73](#_Toc12901518)

[**1.** **Kết quả đạt được** 73](#_Toc12901519)

[**1.** **Hướng phát triển** 73](#_Toc12901520)

[**TÀI LIỆU THAM KHẢO** 74](#_Toc12901521)

# **DANH MỤC VIẾT TẮT**

|  |  |
| --- | --- |
| IPFS | InterPlanetary File System |
| SPA | Single Page Applicaion |
| API | Application Programming Interface. |
| PoS | Proof-of-Stake. |
| PoW | Proof-of-Work |
| CA | Contrac account |
| DApp | Decentralized Application |
| ETH | Ethers |
| HTML | HyperText Markup Language |
| CSS | Cascading Style Sheets |

# **DANH MỤC HÌNH ẢNH**

[Hình 2. 1 Ví dụ về liên kết các khối trong Blockchain 18](#_Toc12863487)

[Hình 2. 2 Cách thức hoạt động của Blockchain 20](#_Toc12863488)

[Hình 2. 3 Sơ đồ tạo và kiểm tra chữ ký số 21](#_Toc12863489)

[Hình 2. 4 Mã hóa đối xứng 21](#_Toc12863490)

[Hình 2. 5 Mã hóa bất đối xứng với khóa công khai (trái) và khóa bí mật (phải) 22](#_Toc12863491)

[Hình 2. 6 Ví dụ tạo một contract trong solidity 23](#_Toc12863492)

[Hình 2. 7 Lỗi 404 trên HTTP 25](#_Toc12863493)

[Hình 2. 8 Màn hình kiểm tra version IPFS 27](#_Toc12863494)

[Hình 2. 9 Màn hình tạo 1 node mới trên IPFS 28](#_Toc12863495)

[Hình 2. 10 Remove toàn bộ boostrap có sẵn trên node 29](#_Toc12863496)

[Hình 2. 11 Hiển thị PeerID của node đó 29](#_Toc12863497)

[Hình 2. 12 Khởi động mạng Private IPFS 30](#_Toc12863498)

[Hình 2. 13 Điểm khác nhau giữa trang web truyền thống và trang web sử dụng Single Page Applicaion. 32](#_Toc12863499)

[Hình 2. 14 Khởi tạo truffle 35](#_Toc12863500)

[Hình 2. 15 Folder sau khi chạy câu lệnh khởi tạo 35](#_Toc12863501)

[Hình 2. 16 Contract Helloworld trong file Helloworld.sol 36](#_Toc12863502)

[Hình 2. 17 Cấu hình lại file 1\_initial\_migration.js 36](#_Toc12863503)

[Hình 2. 18 Cấu hình file truffle.js 37](#_Toc12863504)

[Hình 2. 19 Deploy thành công file Helloworld bằng truffle 37](#_Toc12863505)

[Hình 3. 1 Sơ đồ hệ thống Real Estate Dapp trên ethereum 39](#_Toc14327987)

[Hình 3. 2 Sơ đồ kiến trúc Real estate trên Dapp ethereum. 40](#_Toc14327988)

[Hình 3. 3 Quá trình tìm kiếm, phát hiện mạng trong Ethereum. 41](#_Toc14327989)

[Hình 3. 4 Quá trình tạo một giao dịch trong Real estate Dapp ethereum 42](#_Toc14327990)

[Hình 3. 5 Quá trình thay đổi trạng thái khi thực hiện một giao dịch trong Dapp ethereum 43](#_Toc14327991)

[Hình 3. 6 Quá trình mininng trong ethereum 44](#_Toc14327992)

[Hình 3. 7 Quá trình xác nhận một block mới 44](#_Toc14327993)

[Hình 3. 8 Mô hình dữ liệu 45](#_Toc14327994)

[Hình 3. 9 Danh sách tác nhân sử dụng hệ thống. 46](#_Toc14327995)

[Hình 3. 10 Usecase tổng quát 47](#_Toc14327996)

[Hình 3. 11 Usecase quản lý sản phẩm 48](#_Toc14327997)

[Hình 3. 12 Usecase quản lý đơn hàng mua 48](#_Toc14327998)

[Hình 3. 13 Usecase quản lý đơn bán. 48](#_Toc14327999)

[Hình 3. 14 Use case xem chi tiết sản phẩm 49](#_Toc14328000)

[Hình 4. 1 Màn hình Home page 75](#_Toc14328305)

[Hình 4. 2 màn hình hướng dẫn xác thực bằng metamask 75](#_Toc14328306)

[Hình 4. 3 Màn hình My Purchase 76](#_Toc14328307)

[Hình 4. 4 Màn hình Sell Listing 76](#_Toc14328308)

[Hình 4. 5 Thêm một sản phẩm mới 77](#_Toc14328309)

[Hình 4. 6 Xem chi tiết sản phẩm 78](#_Toc14328310)

[Hình 4. 7 Yêu cầu xác nhân giao dich bằng metamask 79](#_Toc14328311)

[Hình 4. 8 Xác nhận đơn hàng của người bán ( từ chối ) 80](#_Toc14328312)

[Hình 4. 9 Màn hình yêu cầu của người mua 80](#_Toc14328313)

[Hình 4. 10 Màn hình từ chối đơn hàng của người bán 81](#_Toc14328314)

[Hình 4. 11 Xác nhận hoàn thành đơn hàng của người mua và nhận xét về sản phẩm 82](#_Toc14328315)

# **DANH MỤC BẢNG BIỂU**

[Bảng 3. 1 Bảng đặc tả use case đăng nhập. 51](#_Toc12899126)

[Bảng 3. 2 Bảng đặc tả use case đăng xuất 52](#_Toc12899127)

[Bảng 3. 3 Bảng đặc tả use case xem danh sách bất động sản 53](#_Toc12899128)

[Bảng 3. 4 Bảng đặc tả use case xem chi tiết bất động sản 54](#_Toc12899129)

[Bảng 3. 5 Bảng đặc tả use case xem danh sách các đơn hàng đang mua 56](#_Toc12899130)

[Bảng 3. 6 Bảng đặc tả use case hủy đơn hàng 57](#_Toc12899131)

[Bảng 3. 7 Bảng đặc tả use case hoàn thành đơn hàng 59](#_Toc12899132)

# **CHƯƠNG 1. TỔNG QUAN ĐỀ TÀI**

1. **Tên đề tài**

Xây dựng hệ thống quản lý và trao đổi bất động sản trên nền chuỗi khối Blockchain.

1. **Mục tiêu của đề tài**

Các giao dịch bất động sản hiện nay thường phải phụ thuộc vào một số bên trung gian bao gồm các nhà môi giới, cơ sở dữ liệu tài sản của chính phủ, công ty ký quỹ, luật sư, thanh tra viên, thẩm định viên và công chứng viên. Công việc này tốn khá nhiều thời gian và chi phí thanh toán. Bên cạnh đó, cũng sẽ tốn một thời gian dài kể từ khi hợp đồng mua được ký cho đến khi hóa đơn bán hàng được ký kết, và việc chuyển nhượng tài sản thực tế.

Việc sử dụng blockchain chủ yếu nhằm tận dụng các tiềm năng vốn có của nó để giúp chuyển giao tài sản một cách an toàn, nhanh chóng hơn, không còn tốn nhiều chi phí liên quan.

Thay vì tốn nhiều thời gian và chi phí cho phương pháp giao dịch tài sản truyền thông, người dùng có thể sử dụng Blockchain và Smart Contracts để quản lý các giao dịch, thanh toán. Bằng các giao diện dễ dùng, minh bạch trong việc chuyển tiền và nhận tiền.  Các mục tiêu này sẽ được tiến hành với những nội dung chính gồm:

* Tìm hiểu về  Blockchain trong việc quản lý và mua bán bất động sản trên nền tảng Blockchain (Phát triển lên từ Tiểu luận chuyên ngành).
* Cài đặt và tiến hành cấu hình mạng Private IPFS để lưu trữ dữ liệu.
* Áp dụng các công nghệ Blockchain, ReactJS, React - router, Single Page Application vào hệ thống.
* Xây dựng hệ thống trực quan, thân thiện với người dùng.

Và cuối cùng mục tiêu lớn nhất của chương trình chúng em là được người dùng

chấp nhận và sử dụng rộng rãi trong cuộc sống hàng ngày.

1. **Đối tượng, phạm vi nghiên cứu của đề tài**
   1. **Đối tượng nghiên cứu**

Tìm hiểu các bước cài đặt hệ thống mạng Private Blockchain.

Tìm hiểu các bước cài đặt hệ thống mạng Private IPFS.

Tìm hiểu quá trình xử lý các giao dịch ở trên hệ thống mạng Blockchain.

* 1. **Phạm vi nghiên cứu**

Tìm hiểu công nghệ blockchain xây dựng được private blockchain, sử dung ngôn ngữ solidity tạo smartcontract,

Tìm hiểu giao thức IPFS lữu trữ dữ liệu an toàn cũng như giảm dung lượng dữ liệu lưu trữ trực tiếp trên mạng blockchain.

Kết nối được fron-end với back-end. Xây dựng được ứng dụng kết hợp các công nghệ đã tìm hiểu

* 1. **Ý nghĩa của đề tài**

Với những mục tiêu và đối tượng cụ thể mà nhóm chúng em đã trình bày ở trên, nhóm chúng em hy vọng sản phẩm của mình sẽ được sử dụng rộng rãi trong cuộc sống hàng ngày giúp cho người dùng thuận tiện và an tâm hơn khi muốn mua bán bất động sản trực tuyến.

Tuy rằng đây vẫn còn là những công nghệ khá mới tại Việt Nam, nhưng nhóm chúng em tin rằng những thuận tiện mà sản phẩm này mang lại sẽ giúp mọi người sẽ dần sử dụng sản phẩm này hơn.

Với những mục tiêu và phạm vi nghiên cứu cụ thể đã được đề ra, chương tiếp theo chúng em sẽ trình bày những kiến thức và công việc liên quan để xây dựng đề tài.

* 1. **Mô hình hệ thống**

**Nghiệp vụ:**

* Người sử dụng ứng dụng giao diện web để thêm một bất đông sản, cập nhât thông tin, đóng bất động sản ( không hiển thị bán), thực hiện mua một bất động sản với tiền điện tử ETH, sử dụng ví metamask quản lý giao dịch cũng như yêu cầu người dùng ký xác nhận trên ví này khi giao dịch.
* Người sử dụng quản lý được danh sách các bất động sản, danh sách đơn hàng mua, danh dách yêu cầu mua từ những người khác trong hệ thống, người bán có thể chấp nhận hoăc từ chối yêu cầu từ người mua, người mua có thể rút lại yêu cầu mua trong thời gian cho phép cũng như cho nhận xét về sản phẩm sau khi hoàn thành quá trình mua hàng.
* Xây dựng api tương tác lấy thông tin, kiểm tra,… với mạng private blockchain, IPFS.

# **CHƯƠNG 2. CƠ SỞ LÝ THUYẾT**

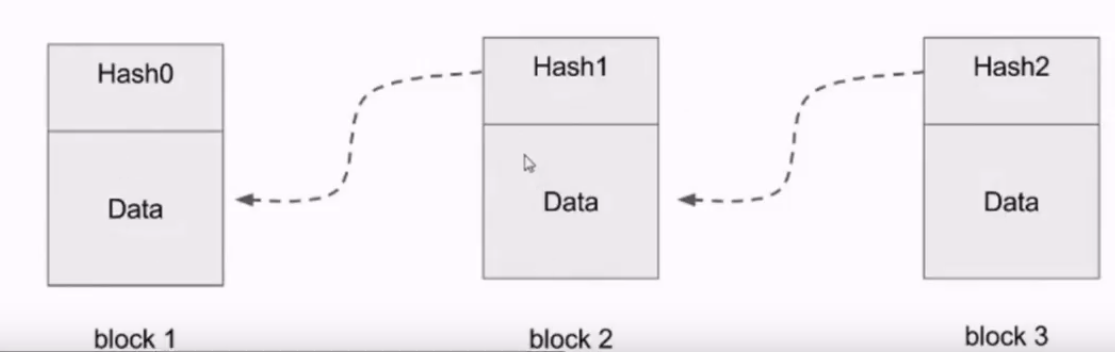
Với những mục tiêu cụ thể đã đề ra trong chương đầu tiên, chương này chúng tôi sẽ trình bày việc các kiến thức làm nền tảng cho việc xây dựng đề tài.

1. **Sơ lược về Blockchain**

Blockchain là một sổ cái kỹ thuật số được phân chia hay dễ hiểu hơn là cơ sở dữ liệu trong một mạng. Sổ cái được chia sẻ cho những người tham gia vào mạng lưới. Điều này cho thấy rằng trong toàn bộ hệ thống không phải chỉ có một vị trí duy nhất, một tài liệu có thể làm căn cứ đáng tin (authority) duy nhất, vì những lần sao chép cùng một phiên bản sổ cái được đặt ở nhiều nơi.

Tất cả các bản sao này được cập nhật khi dữ liệu hoặc giao dịch mới được ghi vào blockchain thông qua sự đồng thuận của tất cả mọi người tham gia. Người đào có trách nhiệm phê duyệt các giao dịch và giám sát mạng bằng cách giải quyết các công thức tinh vi với sự trợ giúp của máy tính. Nó là một hệ thống mạng ngang hàng P2P (peer – to – peer), loại bỏ tất cả mọi khâu trung gian, làm tăng cường an ninh, minh bạch và sự ổn định cũng như giảm thiểu chi phí và lỗi do con người gây ra.

Bằng cách cho phép phân phối các thông tin kỹ thuật số nhưng không được sao chép, công nghệ blockchain đã tạo ra xương sống cho một loại hình Internet mới.



Hình 2. 1 Ví dụ về liên kết các khối trong Blockchain

* 1. **Các thành phần công nghệ của chuỗi khối**

Ba thành phần công nghệ của Blockchain:

- Mạng ngang hàng ( Peer – to – Peer): Một nhóm các máy tính có khả năng giao tiếp với nhau mà không phải phụ thuộc vào một người cầm quyền ở trung tâm.

- Mật mã bất đối xứng: Cho phép những máy tính này gửi các tin nhắn được mã hóa đến những người nhận đã được xác định. Vì vậy bất kỳ ai cũng có thể biết định danh của người gửi, nhưng chỉ người nhận chỉ định mới có thể đọc được nội dung của tin nhắn. Ở Bitcoin và Ethereum, mật mã bất đối xứng được sử dụng để tạo ra một tập các giấy chứng nhận (credential) cho tài khoản của bạn, để chắc chắn rằng chỉ có duy nhất bạn mới có thể chuyển các token của bạn (tiền trong ví của bạn).

- Phép băm mật mã: Một cách để sinh một “dấu vân tay” (fingerprint) nhỏ, duy nhất cho bất kỳ dữ liệu nào, cho phép so sánh một cách nhanh chóng các tập dữ liệu lớn và một cách an toàn để xác nhận rằng dữ liệu đã được thay đổi hay chưa. Ở cả Bitcoin và Ethereum, cấu trúc dữ liệu cây được sử dụng để ghi lại thứ tự đúng tiêu chuẩn của các giao dịch, sau đó được băm vào một “dấu vân tay) làm cơ sở cho việc so sạn của các máy tính trong mạng.

* 1. **Cách hoạt động của Blockchain.**

Trong Blockchian có những tính năng rất đặc biệt, đó là có thể truyền tải dữ liệu mà không thông qua trung gian để xác nhận thông tin. Hệ thống Blockchain tồn tại nhiều nút độc lập với khả năng xác nhận thông tin. Mọi thông tin trong chuỗi khối có thể thay đổi hoặc bổ sung thêm khi có sự chấp thuận của tất cả các nút trên hệ thống. Hệ thống này có độ bảo mật cực cao, mọi hành động đánh cắp dữ liệu đều sẽ bị chặn đứng.

Hệ thống Blockchain vẫn có thể hoạt động bình thường ngay cả khi một phần của hệ thống bị sụp đổ, những máy tính và nút còn lại vẫn hoạt động để bảo vệ thông tin, giữ cho chuỗi khối không bị mất dữ liệu.



Hình 2. 2 Cách thức hoạt động của Blockchain

* 1. **Các đặc điểm của Blockchain**
     1. **Mô hình mạng phân tán (Distributed Peer – 2 – Peer)**

Một mạng phân bố ngang hàng peer-2-peer (p2p) bao gồm rất nhiều các nốt mạng liên kết trực tiếp với nhau, trong đó mỗi nốt mạng có vai trò và vị trí như nhau. Một ví dụ điển hình là Bittorrent, giao thức chia sẻ file qua Internet, hoặc Napster, ứng dụng chia sẻ nhạc trực tuyến những năm 1990s.

Mạng ngang hàng cho phép hàng triệu người sử dụng có thể kết nối trực tiếp với nhau, hình thành một khối khổng lồ với khả năng tính toán và băng thông của tất cả mạng tham gia gộp lại.

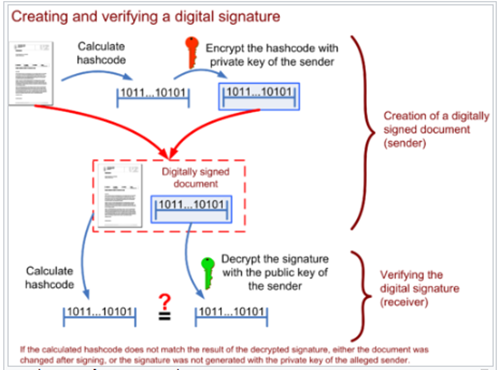
* + 1. **Hàm Hash**

Hàm Hash là một hàm mã hóa dữ liệu với dung lượng bất kỳ để thu được một kiểu dữ liệu có độ dài nhất định. Dữ liệu sau khi bị mã hóa đảm bảo không thể tạo ngược lại dữ liệu ban đầu. Mỗi dữ liệu đầu vào khác nhau sẽ tạo ra một đoạn mã Hash khác nhau mà không có một quy luật trước. Kết quả Hash của cùng một dữ liệu ban đầu sẽ luôn cho ra cùng một đoạn mã Hash duy nhất. Hash Sha-256 là hàm mã hóa dữ liệu trong đó kết quả hash là một chuỗi số gồm 256 bits.

* + 1. **Chữ ký số (Digital Signature)**

Chữ ký số là phương tiện được sử dụng để xác thực nhân dạng của người gửi tin nhắn hoặc của người ký tài liệu và để đảm bảo một điều chắc chắn rằng nội dung gốc của tin nhắn hoăc tài liệu đã gửi sẽ không bị thay đổi.

Chữ ký số dễ dàng chuyển giao , không thể bắt chước bởi bất kỳ người nào và có tự động dán nhãn thời gian. Chữ ký số có năng lực đảm bảo tài liệu gốc sẽ được gửi đến nơi và cả người gửi cũng không dễ để không công nhận nó sau này.



Hình 2. 3 Sơ đồ tạo và kiểm tra chữ ký số

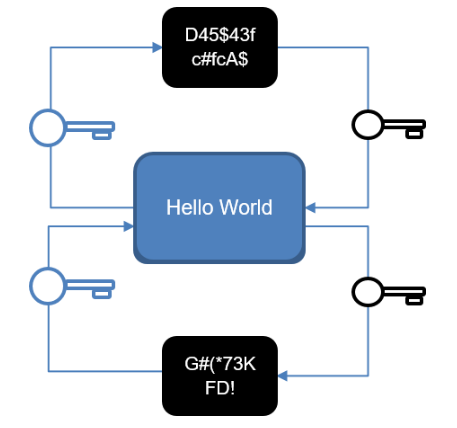
* + 1. **Mã hóa đối xứng và bất đối xứng**

Mã hóa đối xứng sử dụng một khóa duy nhất cho việc mã hóa và giải mã dữ liệu:



Hình 2. 4 Mã hóa đối xứng

Mã hóa bất đối xứng hay còn gọi là mã hóa công khai – bí mật, bao gồm một cặp khóa trong đó một khóa sẽ dùng để mã hóa và phải dùng khóa còn lại mới có thể giải mã; Cả hai khóa đều có thể dùng để mã hóa nhưng cần đến khóa còn lại để giải mã.



Hình 2. 5 Mã hóa bất đối xứng với khóa công khai (trái) và khóa bí mật (phải)

1. **Sơ lược về Ethereum**

Ethereum là một nền tảng phần mềm mở dựa trên công nghệ Blockchain cho phép các nhà phát triển xây dựng và triển khai các ứng dụng dựa trên hệ sinh thái phân tán (Decentralized applications).

Ethereum được xem là một loại tiền mã hóa (Cryptocurrency) tương tự như Bitcoin nhưng không chỉ giới hạn ở khái niệm tiền tệ, Ethereum có thể ứng dụng vào nhiều thứ khác thông qua Smart contract dựa trên công nghệ Blockchain.

1. **Giới thiệu về Smart Contract**
   1. **Khái niệm**

Smart Contract (Hợp Đồng Thông Minh) là một thuật ngữ mô tả khả năng tự đưa ra các điều khoản và thực thi thoả thuận của hệ thống máy tính bằng cách sử dụng công nghệ Blockchain. Toàn bộ quá trình của Smart Contract được thực hiện tự động và không có sự can thiệp từ bên ngoài. Các điều khoản của Smart Contract tương đương với một hợp đồng pháp lý và được ghi lại dưới ngôn ngữ của máy tính. Mục tiêu chính của Smart Contract là cho phép hai bên không xác định danh tính có thể giao dịch hay làm việc với nhau trên Internet mà không cần thông qua trung gian.

* 1. **Cách vận hành của Smart Contract**

Chúng chỉ tự động thực hiện những lệnh mà đã được lập trình sẵn từ trước. Đầu tiên, tài sản và điều khoản hợp đồng đều được mã hóa và chuyển vào một block thuộc Blockchain. Smart contract này tiếp đó sẽ được phân phối và sao chép lại bởi các node hoạt động trên nền tảng đó.

Sau khi có nhận lệnh triển khai thì hợp đồng sẽ được triển khai theo đúng như điều khoản định sẵn. Đồng thời, smart contract cũng sẽ tự động kiểm tra quá trình thực hiện những cam kết nêu trong hợp đồng.

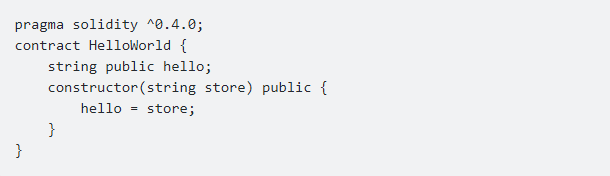
1. **Solidity**
   1. **Solidity là gì?**

Solidity là ngôn ngữ lập trình để cài đặt hợp đồng thông minh (smart contract). Solidity khá giống Javascript, hiện tại là ngôn ngữ phổ biến nhất cho lĩnh vực này.

* 1. **Các khái niệm cơ bản trong solidity**

**Contract:**

Mã của Solidity được đóng gói trong các hợp đồng. Một hợp đồng là khối xây dựng cơ bản của các ứng dụng Ethereum – tất cả các biến và chức năng thuộc về một hợp đồng, và đây là điểm khởi đầu của tất cả các dự án.



Hình 2. 6 Ví dụ tạo một contract trong solidity

**Version Pragma:**

Tất cả các mã nguồn đều bắt đầu với một phiên bản “version pragma“-một tuyên bố của phiên bản của trình biên dịch Solidity. Đây là để ngăn chặn các vấn đề với các phiên bản trình biên dịch tương lai có khả năng giới thiệu những thay đổi có thể phá vỡ mã của bạn. Ở ví dụ trên version pragma là: ^0.4.0

**Biến:**

Solidity yêu cầu khai báo biến và kiểu dữ liệu trước khi sử dụng.

VD: uint public numberOne = 20;

**Struct:**

Cho phép tạo các kiểu dữ liệu phức tạp hơn có nhiều thuộc tính. Kiểu cấu trúc này tương tự như trong ngôn ngữ lập trình C.

Ví dụ:

struct Person {

uint age;

string name;

}

**Array:**

Có hai loại mảng trong Solidity: mảng cố định và mảng động.

Bạn có thể khai báo một mảng như là public, và Solidity sẽ tự động tạo ra một phương thức getter cho nó. Cú pháp như sau: Person[] public people;

Các hợp đồng khác sau đó sẽ có thể đọc (nhưng không viết) cho mảng này. Vì vậy, đây là một mô hình hữu ích để lưu trữ dữ liệu công cộng trong hợp đồng.

**Keccak256 và Typecasting**

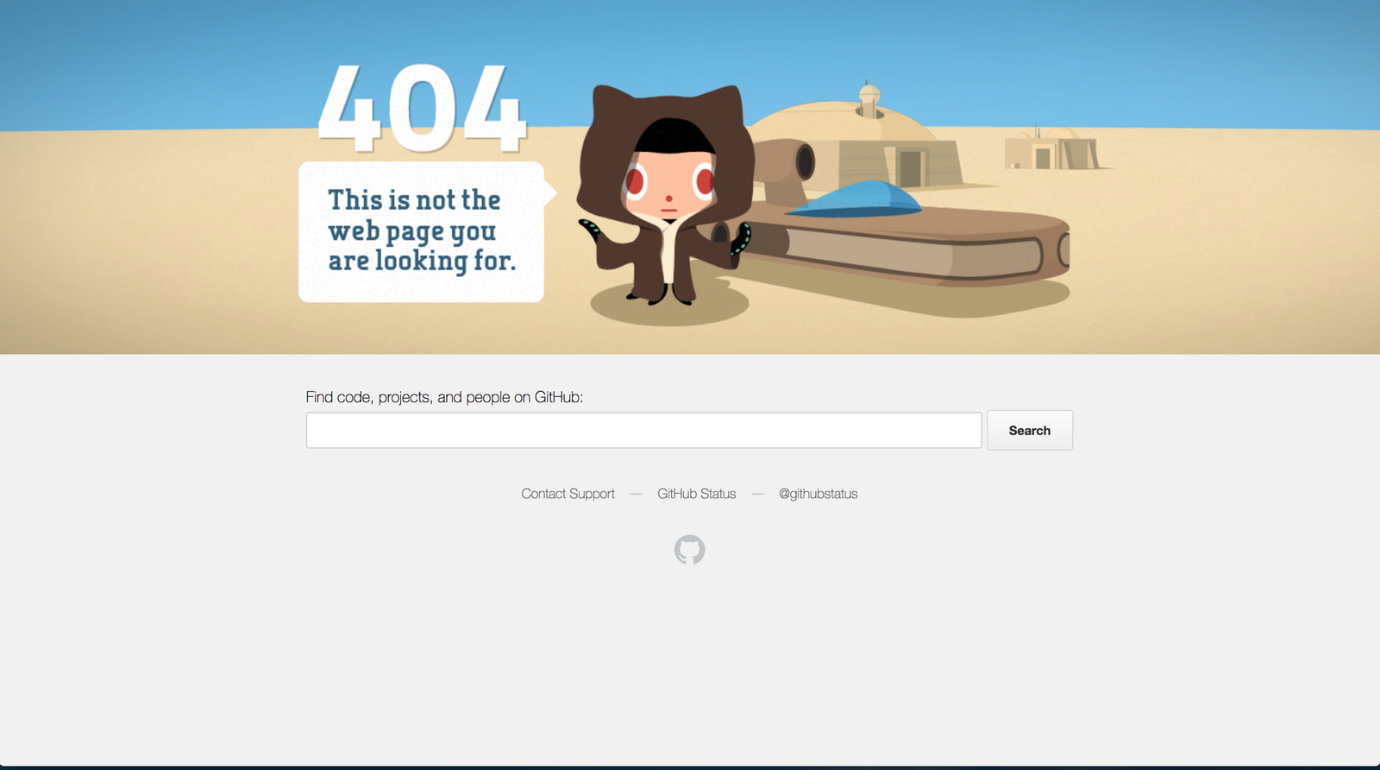
Ethereum có chức năng băm keccak256 được xây dựng trong, đó là phiên bản của SHA3. Một hàm băn cơ bản ánh xạ một chuỗi đầu vào thành 1 số thập lục phân 256-bit ngẫu nhiêu. Một thay đổi nhỏ trong chuỗi sẽ gây ra một thay đổi lớn trong băm. Nó hữu ích cho nhiều mục đích trong Ethereum. Typecasting: chuyển đổi giữa các kiểu dữ liệu

# **Giới thiệu IPFS**

* 1. **Khái niệm**

Internet hiện nay được xây đựng trên nền tảng giao thức HTTP (HyperText Transfer Protocol - giao thức truyền tải siêu văn bản). mặc dù được truyền tải cách rộng rãi nhưng HTTP cũng có một số hạn chế nổi bật như: mức độ bảo mật, khả năng truyền tải.

Giao thức HTTP làm cho các hoạt động trên internet phụ thuộc vào máy chủ. đặc biệt là lỗi “404 server not found” rất phổ biến hiện nay.



Hình 2. 7 Lỗi 404 trên HTTP

Lỗi trên thường gặp khi ta truy cập vào các nội dung đã cũ hoặc các máy chủ hiện thời không thể trả về kết quả mong muốn. đây là một hạn chế rất lớn của giao thức HTTP.

IPFS (InterPlanetary File System) là một giao thức phân phối mã nguồn mở của giao thức hypermedia hoạt động dựa trên nội dung và danh tính. Cụ thể hơn, nó sẽ phân phối dữ liệu theo hình thức P2P. Trong đó, các hoạt động của IPFS chủ yếu dựa vào khả năng tính toán băng thông của tất cả các máy tham gia chứ không tập trung vào một phần nhỏ các máy chủ trung tâm như các giao thức khác.

Cách thức hoạt động của IPFS sẽ tương tự như BitTorrent, đồng nghĩa với mỗi máy tính tham gia trong mạng lưới của nó sẽ đảm nhận cả việc download lẫn upload dữ liệu mà không cần có sự có mặt của một máy chủ trung tâm. So với HTTP, IPFS mang lại tiềm năng lớn nhờ cải thiện được tốc độ truyền tải, tránh sự phụ thuộc vào các máy chủ và có khả năng cải thiện cấu trúc của nền tảng Internet.

* 1. **Những điểm mạnh của IPFS**

Các nội dung khi người dùng upload lên sẽ được băm và sinh ra một đoạn mã. Mỗi nội dung giống nhau sẽ luôn cho các mã băm giống nhau, từ đó IPFS loại bỏ được sự trùng lặp. Nội dung khi một nút chia sẻ sẽ được chia thành các mảnh, sau đó chúng sẽ được đưa tới các nút gần nó trong mạng. Khi muốn tải về một tệp tin, một nút sẽ yêu cầu máy chủ tìm trong một bảng băm phân tán có thông tin của tất cả mọi nút trong mạng để tim ra được các mảnh của nội dung được lưu trữ ở những nút nào, công cuộc tìm kiếm được thực hiện một cách hiệu quả, IPFS mô tả rằng một nội dung có thể tìm ra sau tối đa 20 bước nhảy cho mạng 10 triệu nút. Việc nội dung được lưu ở các nút khác nhau gần đó sẽ giúp cho việc tải về chúng nhanh hơn, giúp tiết kiệm băng thông,

**Tránh sự phụ thuộc vào máy chủ:**

Bạn thường dễ dàng tìm thấy lỗi 404 khi tìm kiếm các nội dung đã cũ, khi chúng ta đang ngoại tuyến, hoặc máy chủ đang gặp phải sự cố và không thể hồi đáp thông tin cho người dùng. Đây cũng là vấn đề lớn nhất mà giao thức HTTP gặp phải, điều mà nó không thể cải thiện cũng như khắc phục.

Mặt khác, giao thức IPFS hoàn toàn bỏ qua khái niệm máy chủ, mà chỉ quan tâm tới nội dung tìm kiếm. Điều này không chỉ giúp chúng ta rút ngắn con đường tới thông tin, mà lại không lo gặp phải các máy chủ kém chất lượng, kém tin cậy.

**Không còn mô hình tập trung**

Với mô hình Internet ngày nay, chúng ta luôn đứng trước các vấn đề mang tính dây chuyền, mà khởi nguồn là cuộc chiến tranh giành người dùng đến các máy chủ trung tâm tốt hơn, được điều hành bởi một số tên tuổi lớn trong lĩnh vực công nghệ như Amazon, Google, Yahoo,…

Điều này dẫn đến các vấn đề như nhiều bí mật của chính phủ bị hé lộ, các công ty sử dụng phần mềm gián điệp, tin tặc thực hiện các vụ tấn công DDoS, ISPs công khai chặn các dịch vụ mà họ không muốn bạn truy cập, dữ liệu bị ngăn cản khỏi nội bộ một quốc gia, hay dữ liệu của bạn có thể chống lại chính chúng ta.

Với mô hình website phân tán (decentralized) của IPFS, các vấn đề này hoàn toàn được khắc phục và không còn chế độ quản lý phân cấp. Một trang web được tạo ra với mục đích chia sẻ thông tin sẽ luôn đến được với người dùng. Ở điều kiện lý tưởng nhất, bạn thậm chí có thể truy cập website khi không có kết nối mạng. Đây không chỉ là một điểm cộng lớn đối với các nhà phát triển, mà còn đánh dấu sự tiến bộ đối với với các quyền cá nhân, quyền riêng tư của người dùng.

**Giảm bớt chi phí**

Ưu điểm tiếp theo của mô hình IPFS đó là giảm bớt chi phí đối với cả người cung cấp nội dung và người dùng thông thường.

Có thể bạn chưa biết, nhưng dữ liệu được sử dụng trong giao thức HTTP trên thực tế rất tốn kém. Các nhà cung cấp dữ liệu được phép tính phí và đề ra các thỏa thuận đối với những nội dung ăn khách trên Internet.

* 1. **Cài đặt IPFS**

**Bước 1: Cài đặt IPFS**

Để sử dụng được IPFS, đầu tiên ta cần tải go-ipfs từ link https://ipfs.io/docs/install/. Sau đó di chuyển tới thư mục chứa file go-ipfs đã download về và giải nén file đó bằng câu lệnh:

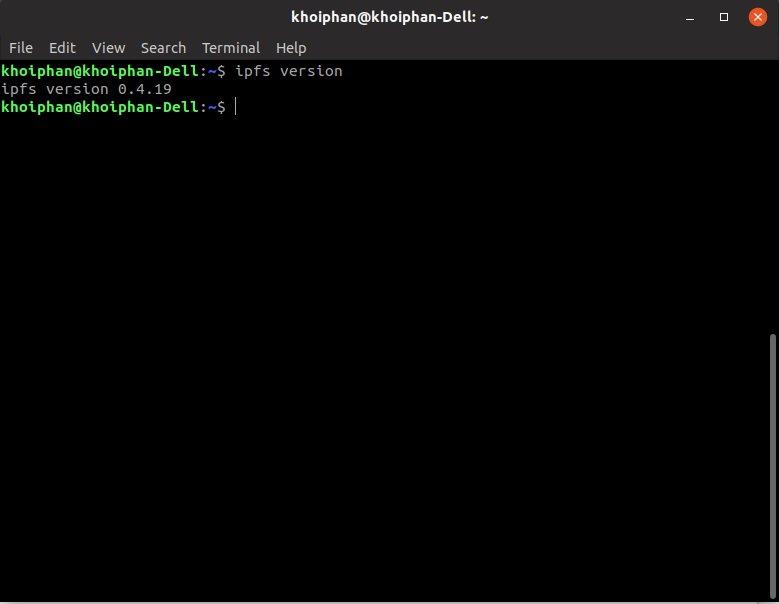
tar xvfz go-ipfs.tar.gz

Sau khi giải nén, ta được thư mục go-ipfs, ta thực hiện việc cài đặt bằng cách thực thi câu file script cài đặt install.sh:

./install.sh

Để kiểm tra xem thử ta đã cài đặt thành công hay chưa, ta thực hiện câu lệnh:

ipfs version



Hình 2. 8 Màn hình kiểm tra version IPFS

Như vậy là ta đã cài đặt thành công IPFS.

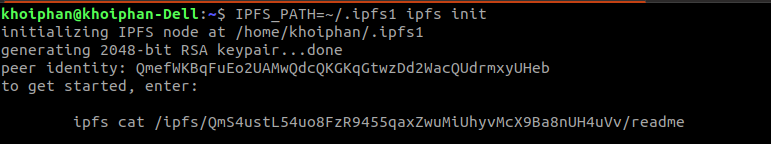
**Bước 2: Tạo node mới trong IPFS**

Sau khi cài đặt thành công, ta tiến hành khởi tạo 2 node để có thể tạo 1 mạng private IPFS demo. Có 2 loại node là Bootstrap node và Client node. Bootstrap node là node mà các node khác có thể kết nối tới được và có thể tìm được địa chỉ ID của các node khác. Để khởi tạo 1 node, ta dùng câu lệnh:

IPFS\_PATH=~/.ipfs1 ipfs init

Trong đó:

IPFS\_PATH: là câu lệnh để bắt buộc IPFS sử dụng đường dẫn tới file config .ipfs1: là tên thư mục của node



Hình 2. 9 Màn hình tạo 1 node mới trên IPFS

**Bước 3: Cài đặt Private IPFS network**

Ta phải cài đặt key generator. Trước tiên, ta cài đặt git trên Boostrap node (nếu đã cài đặt git từ trước thì không cần dùng câu lệnh này) bằng câu lệnh:

sudo apt-get install git

Tiếp theo, ta chạy câu lệnh để cài đặt swarm key generator. Lưu ý, để hai máy có thể kết nối đến mạng private này, thì ta cần phải dùng chung một fie swarm key generator bằng cách copy qua máy Client node. Nếu máy Client node cài đặt swarm key này thì hai máy sẽ không thể kết nối với nhau:

go get -u github.com/Kubuxu/go-ipfs-swarm-key-gen/ipfs-swarm-key-gen

Chạy file swarm key generator để tạo file swarm trong thư mục .ipfs trong máy:

./go/bin/ipfs-swarm-key-gen > ~/.ipfs/swarm.key

Sau đó copy file swarm.key cho các máy Client node.

Bước 4: Cài đăt Bootstrap node

Để kiểm tra file config của node, ta sử dụng câu lệnh:

IPFS\_PATH=~/.ipfs1 ipfs config show

Vì ta sẽ tạo private network nên ta phải remove hết tất cả các bootstrap có sẵn trong file config đó, để remove tất cả, ta sử dụng câu lệnh:

IPFS\_PATH=~/.ipfs1 ipfs bootstrap rm –all



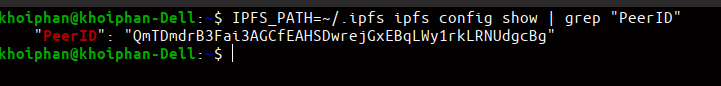
Hình 2. 10 Remove toàn bộ boostrap có sẵn trên node

Sau đó, ta tiến hành add các ip address và các Peer Identity của Bootstrap node vào các Client node và cả Bootstrap node.

Có thể lấy IP address bằng cách nhập câu lệnh : hostname -I

Peer Identity được tạo ra trong quá trình tạo node mới trên IPFS, có thể lấy PeerID bằng câu lệnh:

IPFS\_PATH=~/.ipfs ipfs config show | grep "PeerID"



Hình 2. 11 Hiển thị PeerID của node đó

Sau đó, tiến hành add bootstrap bằng câu lệnh:

IPFS\_PATH=~/.ipfs ipfs bootstrap add /ip4/<ip address of bootnode>/tcp/4001/ipfs/<peer identity hash of bootnode>

Ví dụ:

IPFS\_PATH=~/.ipfs ipfs bootstrap add /ip4/172.25.10.5/tcp/4001/ipfs/QmTDmdrB3Fai3AGCfEAHSDwrejGxEBqLWy1rkLRNUdgcBg

Chạy câu lệnh này trên Bootstrap node và Client node.

Bước 5: Khởi chạy private IPFS.

Để khởi chạy mạng private IPFS, ta dùng câu lệnh:

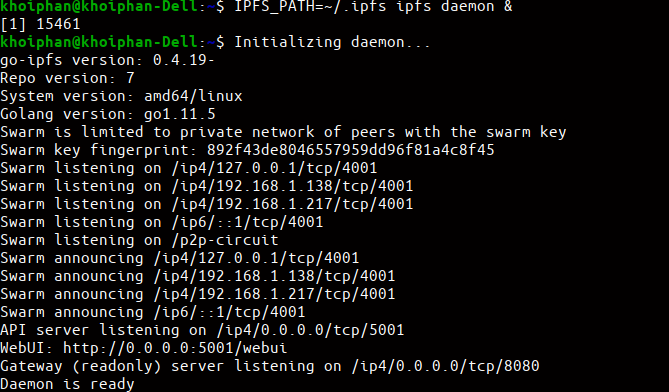
export LIBP2P\_FORCE\_PNET=1

IPFS\_PATH=~/.ipfs ipfs daemon &

Trong đó:

export LIBP2P\_FORCE\_PNET = 1 có nghĩa là ta sẽ buộc mạng này chạy private. Nếu không có mạng private nào được khởi tạo, khởi tạo daemon sẽ failed.

Sau khi chạy câu lệnh, màn hình sẽ xuất hiện như sau:

****

Hình 2. 12 Khởi động mạng Private IPFS

Như vậy là ta đã tạo thành công 1 mạng private IPFS.

# **Giới thiệu ReactJS**

* 1. **Giới thiệu**

React.js là một thư viện Javascript đang nổi lên trong những năm gần đây với xu hướng Single Page Application. Trong khi những framework khác cố gắng hướng đến một mô hình MVC hoàn thiện thì React nổi bật với sự đơn giản và dễ dàng phối hợp với những thư viện Javascript khác. Nếu như AngularJS là một Framework cho phép nhúng code Javascript trong code html thông qua các attribute như ng-model, ng-repeat...thì với ReactJS là một library cho phép nhúng code HTML trong code Javascript nhờ vào JSX, bạn có thể dễ dàng lồng các đoạn HTML vào trong Javascript. Tích hợp giữa Javascript và HTML vào trong JSX làm cho các component dễ hiểu hơn.

* 1. **React là gì**

React là một thư viện UI phát triển tại Facebook để hỗ trợ việc xây dựng những thành phần (components) UI có tính tương tác cao, có trạng thái và có thể sử dụng lại được. React được sử dụng tại Facebook trong production, và [www.instagram.com](http://www.instagram.com/) được viết hoàn toàn trên React.

Một trong những điểm hấp dẫn của React là thư viện này không chỉ hoạt động trên phía client, mà còn được render trên server và có thể kết nối với nhau. React so sánh sự thay đổi giữa các giá trị của lần render này với lần render trước và cập nhật ít thay đổi nhất trên DOM.

* 1. **Một số khái niệm cơ bản trong React**
     1. **Virtual DOM**

Công nghệ Virtual DOM giúp tăng hiệu năng cho ứng dụng. Việc chỉ node gốc mới có trạng thái và khi nó thay đổi sẽ tái cấu trúc lại toàn bộ, đồng nghĩa với việc DOM tree cũng sẽ phải thay đổi một phần, điều này sẽ ảnh hưởng đến tốc độ xử lý. React JS sử dụng Virtual DOM (DOM ảo) để cải thiện vấn đề này. Virtual DOM là một object Javascript, mỗi object chứa đầy đủ thông tin cần thiết để tạo ra một DOM, khi dữ liệu thay đổi nó sẽ tính toán sự thay đổi giữa object và tree thật, điều này sẽ giúp tối ưu hoá việc re-render DOM tree thật.

React sử dụng cơ chế one-way data binding – luồng dữ liệu 1 chiều. Dữ liệu được truyền từ parent đến child thông qua props. Luồng dữ liệu đơn giản giúp chúng ta dễ dàng kiểm soát cũng như sửa lỗi.

* + 1. **JSX là gì?**

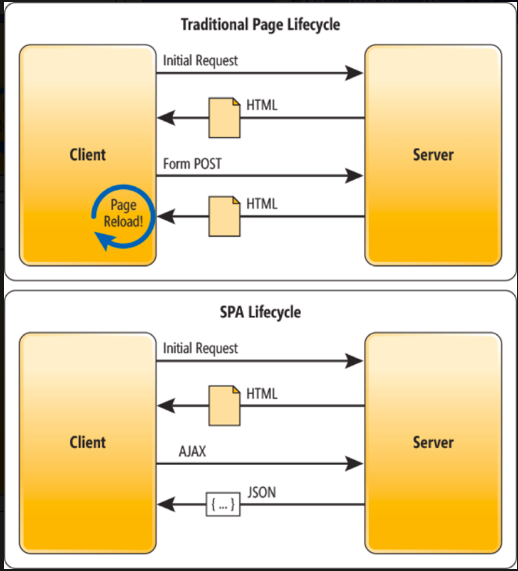
JSX là một dạng ngôn ngữ cho phép viết các mã HTML trong Javascript. Đặc điểm: Faster: Nhanh hơn. JSX thực hiện tối ưu hóa trong khi biên dịch sang mã Javacsript. Các mã này cho thời gian thực hiện nhanh hơn nhiều so với một mã tương đương viết trực tiếp bằng Javascript. Safer: an toàn hơn.

Ngược với Javascript, JSX là kiểu statically-typed, nghĩa là nó được biên dịch trước khi chạy, giống như Java, C++. Vì thế các lỗi sẽ được phát hiện ngay trong quá trình biên dịch. Ngoài ra, nó cũng cung cấp tính năng gỡ lỗi khi biên dịch rất tốt. Dễ dàng hơn, JSX kế thừa dựa trên Javascript, vì vậy rất dễ dàng để cho các lập trình viên Javascripts có thể sử dụng.

* + 1. **Single Page Application (SPA)**

Single page Application là một ứng dụng web giúp nâng cao trải nghiệm người dùng bằng cách sử dụng HTML5 và AJAX. Đầu tiên khi tải một trang web bất kỳ, SPA sẽ tải một trang HTML đơn, sau đó dựa trên request của người dùng, SPA sẽ tiếp tục tải các HTML khác trong cùng một trang đó.

Hiểu một cách đơn giản, thì toàn bộ resource của web bao gồm các file CSS, Javascript, master layout hay cấu trúc web page sẽ được load lần đầu tiên khi chúng ta bắt đầu duyệt môt website A nào đó. Ở những lần sau, khi chuyển trang khác, client sẽ gửi những ajax request để get dữ liệu cần thiết( thường là phần nội dung). Việc này mang đến trải nghiệm cho người dùng web tốt hơn, giảm thời gian phải load lại toàn bộ trang web cồng kềnh, tiết kiệm băng thông cũng như thời gian chờ đợi. Việc này là trái ngược hoàn toàn với trang web truyền thống khi toàn bộ trang web phải load lại mỗi khi chuyển trang.



Hình 2. 13 Điểm khác nhau giữa trang web truyền thống và trang web sử dụng Single Page Applicaion.

**Ưu điểm của SPA**

* Việc render html ở server là một điều đáng quan tâm nếu trang web của bạn có nhiều người dùng, cực kì tốn tài nguyên hệ thống. Với SPA, điều này chỉ xảy ra lần đầu tiên khi người dùng truy cập trang chủ, còn sau đó việc render sẽ do client đảm nhiệm.
* SPA tách biệt phần front-end và back-end ra, SPA giao tiếp với server chủ yếu qua JSON Rest API giúp cho dữ liệu gửi và trả giữa client và server được giảm đến mức tối thiểu. Việc phát triển, kiểm thử cũng có thể độc lập giữa front-end và back-end.
* SPA rất nhanh, vì các tài nguyên tĩnh (static resources) như HTML, CSS, Script chỉ được tải 1 lần duy nhất. Trong suốt quá trình sử dụng, chỉ có dữ liệu là được chuyển giao qua lại giữa client với server -> giảm thiểu băng thông cho server.
* Tăng trải nghiệm người dùng.

**Nhược điểm của SPA**

* Người dùng phải cho phép Javascript hoạt động trên trình duyệt. Nếu không SPA sẽ không hoạt động.
* Trình duyệt sẽ phải xử lý rất nhiều, nên vấn đề hiệu năng trên điện thoại tầm trung trở xuống là điều bạn cần quan tâm.
* Phức tạp: Việc phát triển SPA sẽ phức tạp hơn rất nhiều so với ứng dụng web truyền thống. Back-end developer cần phải biết Javascript cũng như biết cách sử dụng một trong những framework dùng để phát triển SPA (AngularJS, React, EmberJS…). Và việc viết unit test cho Javascript cũng có nhiều khó khăn hơn, tuy nhiên hiện nay cũng có rất nhiều framework support việc này.
  + 1. **React - router là gì?**

React-Router là một thư viện định tuyến (routing) tiêu chuẩn trong React. Nó giữ cho giao diện của ứng dụng đồng bộ với URL trên trình duyệt. React-Router cho phép bạn định tuyến "luồng dữ liệu" (data flow) trong ứng dụng của bạn một cách rõ ràng. Nó tương đương với sự khẳng định, nếu bạn có URL này, nó sẽ tương đương với Route này, và giao diện tương ứng.

* + 1. **Các thành phần trong React - router**
       1. **BrowserRouter và HashRouter**

React-Router cung cấp cho chúng 2 thành phần hay sử dụng đó là BrowserRouter & HashRouter. Hai thành phần này khác nhau ở kiểu URL mà chúng sẽ tạo ra và đồng bộ.

* BrowserRouter: Được sử dụng phổ biến hơn, nó sử dụng History API có trong HTML5 để theo dõi lịch sử bộ định tuyến của bạn.
* HashRouter: Sử dụng hash của URL (window.location.hash) để ghi nhớ mọi thứ.
  + - 1. **Route**

Route: Định nghĩa một ánh xạ (mapping) giữa một URL và một Component. Điều đó có nghĩa là khi người dùng truy cập theo một URL trên trình duyệt, một Component tương ứng sẽ được render trên giao diện.

Trong đó:

* **path**: Là đường dẫn trên URL.
* **exact**: Liúp cho route này này chỉ hoạt động nếu URL trên trình duyệt phù hợp tuyệt đối với giá trị của thuộc tính path của nó.
* **component**: Là component sẽ đươc load ra tương ứng với Route đó.

1. **Web3.js**

Web3.js là một thư viện của Ethereum giúp hỗ trợ xây dựng các DAPP trên nền tảng web.

1. **Metamask**

Metamask là một đường dẫn, nó cho phép ta truy cập vào trang web phân tán trong trình duyệt. Ngoài ra metamask còn cho phép ta chạy các ứng dụng khác như Ethereum DApp ngay trong trình duyệt mà không cần khởi động toàn bộ các node Ethereum dẫn đến trình duyệt trở nên nặng nề và chậm chạp.

Metamask bao gồm một mã nhận diện mã hóa vô cùng an toàn, cung cấp đến người dùng một giao diện để quản lý các thông tin cá nhân của bạn trên các trang web khác nhau và chấp nhận các giao dịch blockchain trên toàn hệ thống.

Cơ chế hoạt động của Metamask là thông qua các private key, ta có thể kết nối tới Ethereum blockchain và lưu trữ ethereum, token thuộc Ethereum một cách bảo mật nhất.

1. **Truffle**
   1. **Khái niệm**

Truffle là framework để phát triển ứng dụng phi tập trung trên Ethereum. Các chức năng bao gôm: Complie and build, testing, deployment, intergration.

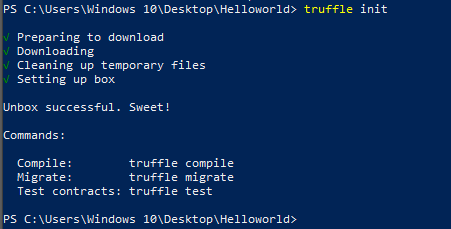
* 1. **Cài đặt Truffle**

Đầu tiên, để cài đặt truffle. Ta thực hiện câu lệnh:

npm install –g truffle

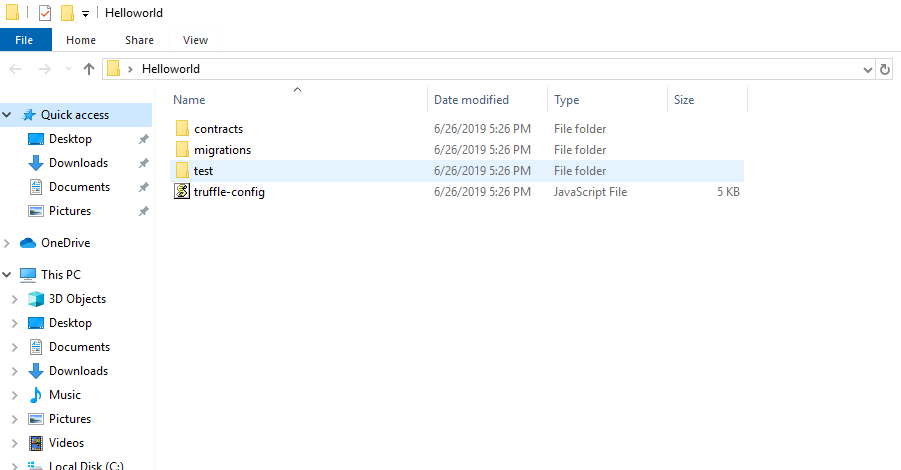
Tiếp theo tạo folder Helloworld. Tiếp theo, khởi tạo truffle bằng câu lệnh:

truffle init



Hình 2. 14 Khởi tạo truffle

Sau khi chạy câu lệnh truffle init, kết quả thu được sẽ như thế này:



Hình 2. 15 Folder sau khi chạy câu lệnh khởi tạo

Tiếp theo, tạo file helloworld.sol trong thư mục contract.



Hình 2. 16 Contract Helloworld trong file Helloworld.sol

Mở file 1\_initial\_migration.js trong thư mục migrations, import và thêm code như sau:



Hình 2. 17 Cấu hình lại file 1\_initial\_migration.js

Lưu ý đường dẫn import file, nó không phải là địa chỉ tương đối của cấu trúc project, mình phải import từ root "./helloworld.sol".

Mở Power Shell và sử dụng command sau để start server:

testrpc

Mặc định sẽ chạy ở localhost:8545

Mở file truffle.js để cấu hình network:



Hình 2. 18 Cấu hình file truffle.js

Mở thêm 1 Power Shell khác, dùng lệnh sau để compile:

truffle compile

Tiếp tục dùng lệnh sau dể migrate:

truffle migrate

Dùng truffle console bằng lệnh:

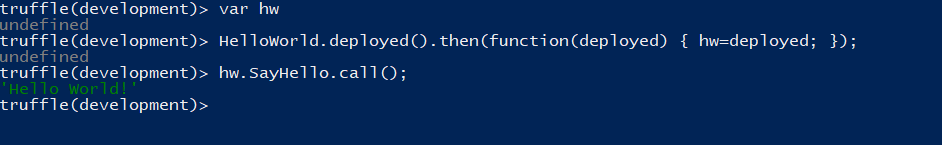
truffle console

Gõ từng lệnh sau:

var hw

HelloWorld.deployed().then(function(deployed) { hw=deployed; });

hw.SayHello.call();

[](https://s3-ap-southeast-1.amazonaws.com/kipalog.com/92m9mscsxp_image.png)

Hình 2. 19 Deploy thành công file Helloworld bằng truffle

# **CHƯƠNG 3: PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG**

Sau khi đã tìm hiểu rõ về nền tảng của các công nghệ khóa luận sẽ sử dụng, trong chương này, chúng ta sẽ đi vào việc sử dụng các công nghệ này, xây dựng chúng thành hệ thống trao đổi bất động sản trên nền tảng Blockchain.

## **Đăc tả yêu cầu**

### **Mục đích phạm vi của hệ thống**

Tìm hiểu công nghệ blockchain xây dựng được private blockchain, sử dung ngôn ngữ solidity tạo smartcontract,

Tìm hiểu giao thức IPFS lữu trữ dữ liệu an toàn cũng như giảm dung lượng dữ liệu lưu trữ trực tiếp trên mạng blockchain.

Kết nối được fron-end với back-end. Xây dựng được ứng dụng kết hợp các công nghệ đã tìm hiểu

### **Yêu cầu chức năng**

Để xây dựng thành công ứng dụng, nhóm phải giải quyết các vấn đề sau:

* + Xây dựng thành công hệ thống mạng Private Blockchain, từ đó đồng bộ được dữ liệu giữa các node trong hệ thống mạng.
  + Xây dựng smartcontract quản lý chức năng của hệ thống.
  + Sử dụng thư viên Web3 tương tác giữa front-end(React js) với mạng private blockchain cũng như các chức năng của smartcontract.
  + Người dùng tương tác với hệ thống có thể tạo được một bất động sản đăng bán trên hệ thống.

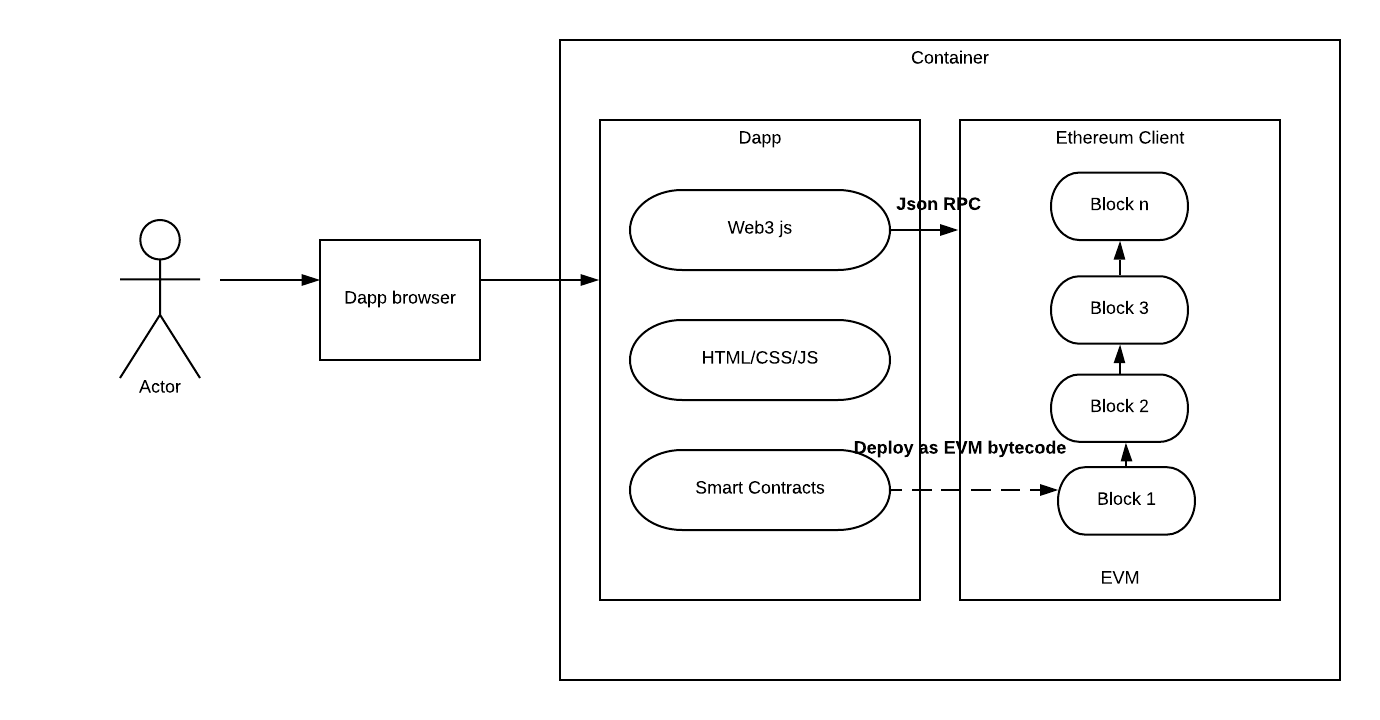
Các chức năng của hệ thống:

* Người sử dụng ứng dụng giao diện web để thêm một bất đông sản, cập nhât thông tin, đóng bất động sản ( không hiển thị bán), thực hiện mua một bất động sản với tiền điện tử ETH, sử dụng ví metamask quản lý giao dịch cũng như yêu cầu người dùng ký xác nhận trên ví này khi giao dịch.
* Người sử dụng quản lý được danh sách các bất động sản, danh sách đơn hàng mua, danh dách yêu cầu mua từ những người khác trong hệ thống, người bán có thể chấp nhận hoăc từ chối yêu cầu từ người mua, người mua có thể rút lại yêu cầu mua trong thời gian cho phép cũng như cho nhận xét về sản phẩm sau khi hoàn thành quá trình mua hàng.
* Xây dựng api tương tác lấy thông tin, kiểm tra,… với mạng private blockchain, IPFS.

## **Thiết kế hệ thống**

### **2.1. Thiết kế kiến trúc hệ thống**

#### **2.1.1. Mô hình hệ thống**



Hình 3. 1 Sơ đồ hệ thống Real Estate Dapp trên ethereum

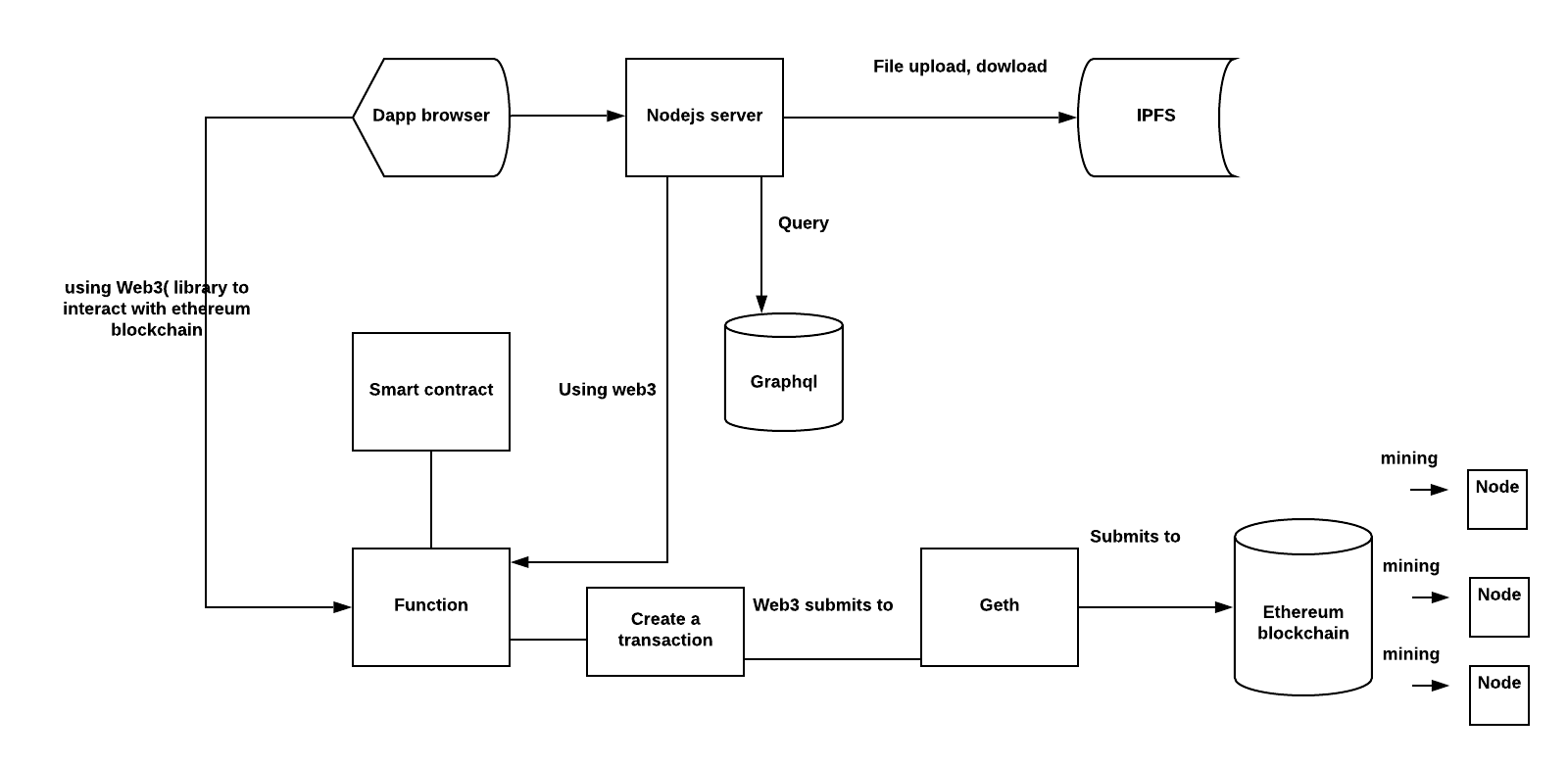
Trong sơ đồ hệ thống của ứng dụng gồm 2 phần chính ethereum client và front-end của Dapp:

Phần Ethereum client mô tả về quá trình phát triển smart contracts, smart contracts được viết bằng ngôn ngữ phổ biến như Solidity sau đó thông qua trình biên dịch truffle compiler được biên dịch thành operation code. Operation code được deploy vào blockchain của ethereum tại một logical address ( hay Block ).

Phần front-end của Dapp là nơi chứa private key & public key, điều này khác biệt hoàn với các ứng dụng truyền thống là thông tin dùng đăng nhập ứng dụng của người dùng sẽ không được lưu trữ ở cơ sở dữ liệu hay một bên thứ ba nào khác, đảm bảo tính an toàn cũng như bảo mật cho tài khoản đẵng nhập của người dùng. Người dùng sẽ tương tác với ứng dụng thông qua Web UI ( HTML/CSS/JS), trong ứng dụng nhóm chúng em có sử dụng React Js ( thư viện viết bằng javascript ) xây dựng giao diện. Đây cũng là nơi người dùng tương tác, hiển thị dữ liệu của smart contracs.

#### **Kiến trúc hệ thống**

Khi bắt đầu vào vào việc thiết kế nhóm chúng em cũng đã tìm hiểu những tài liệu liên quan và thách thức là phải xây dựng một kiến trúc hệ thống. Các ứng dụng truyền thống được xây dựng trên mô hình client-server, đối với ứng dụng hiện tại có thêm một thành phần Blockchain.

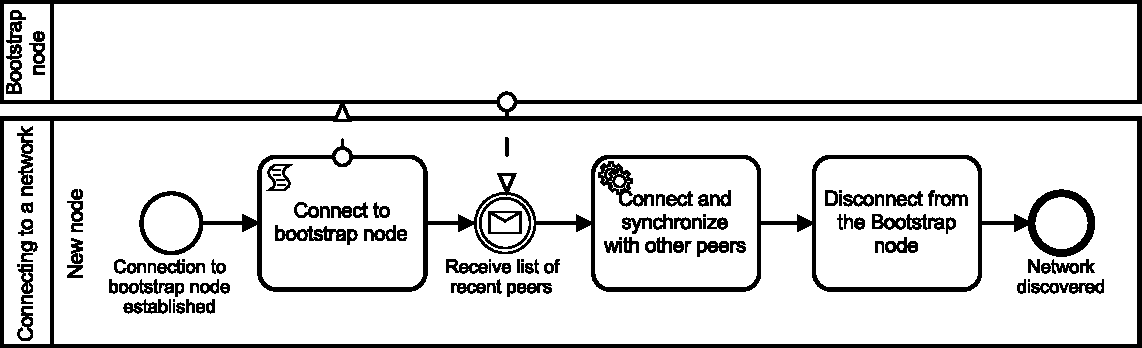


Hình 3. 2 Sơ đồ kiến trúc Real estate trên Dapp ethereum.

Kiến trúc đặc thù của ứng dụng trên Ethereum không có máy chủ, trong đó thì mọi tương tác diễn ra hoàn toàn giữa máy khách ( client ) và mạng ethereum blockchain. Vậy để truy vấn, đọc thông tin từ blockchain nhóm chúng em sử dụng Web3 đây là một thư viện javascript giúp tương tác với Ethereum blockchain. Khi ứng dụng đã kết nối được với ethereum blockchain thì việc quản lý, xác nhận, từ chối những giao dịch của smart contracts với ethereum là không tránh, metamask sẽ giúp người dùng thực hiện điều đó. Metamask quản lý tài khoản, thông tin, số dư, xác nhận giao dịch, mật khẩu của người dùng, ứng dụng giúp bảo mật an toàn cho tài khoản vì mọi thông tin đều được lưu dưới máy người dùng. ( còn tiếp )

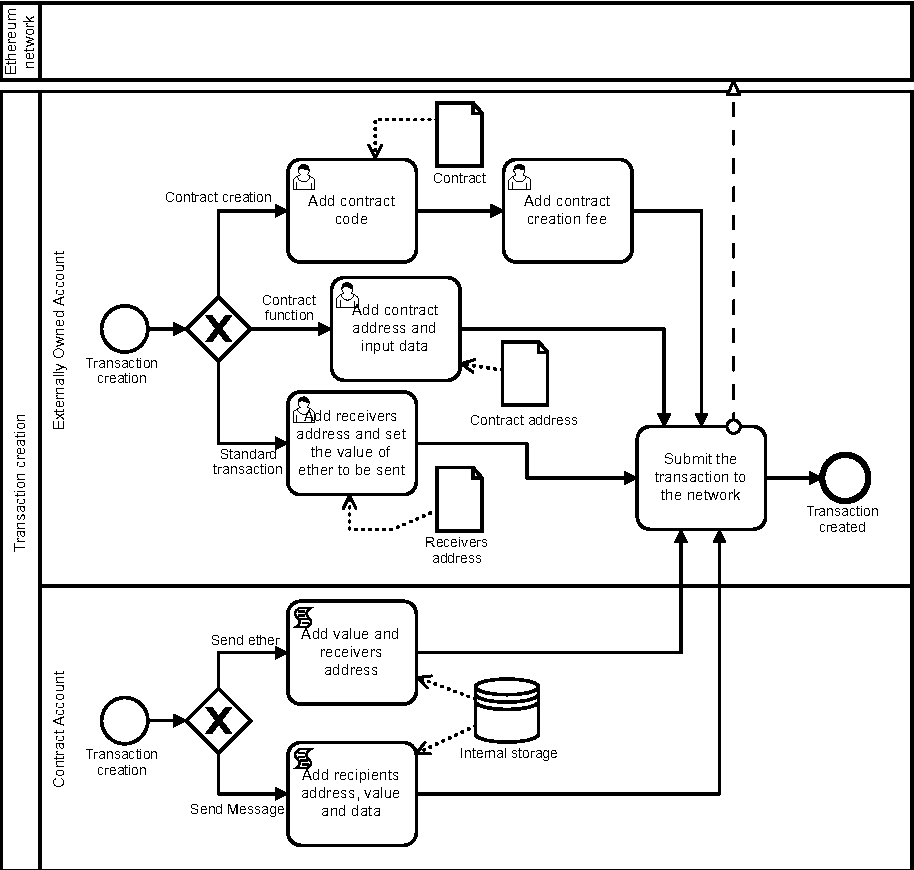
Để phát triển ứng dụng phi tập trung (DApp), cần thực hiện các bước sau:

* Thiết kế và thực hiện hợp đồng thông minh bằng ngôn ngữ cấp cao (Solidity).
* Biên dịch hợp đồng để tạo tệp nhị phân.
* Triển khai hợp đồng trên mạng Ethereum Blockchain bằng các ứng dụng khách Ethereum (truffle).
* Xây dựng ứng dụng Web (Front-end) tương tác với các hợp đồng thông minh. Đối với trường hợp sử dụng này, công nghệ blockchain yêu cầu hai phần:
* Hợp đồng thông minh Ethereum sử dụng Solidity làm ngôn ngữ lập trình nằm trong khối Máy ảo Ethereum (EVM).
* App Ứng dụng phân tán (DApp) bao gồm ứng dụng front-end và back-end dùng tương tương tác với hợp đồng thông minh và người dùng (chủ bất động sản và người mua bất động sản).



Hình 3. 3 Quá trình tìm kiếm, phát hiện mạng trong Ethereum.

Quá trình khám phá mạng, được thấy trong Hình 3.3, bắt đầu bằng cách thiết lập một kết nối đến một node bootstrap (một node mà IP đến từ mã nguồn,được giả định là luôn trực tuyến và được kết nối với các node thông thường khác). Một lần được kết nối, node bootstrap sẽ chia sẻ IP của các xung quanh được kết nối với nó và node mới sẽ đồng bộ hóa với các node khác.

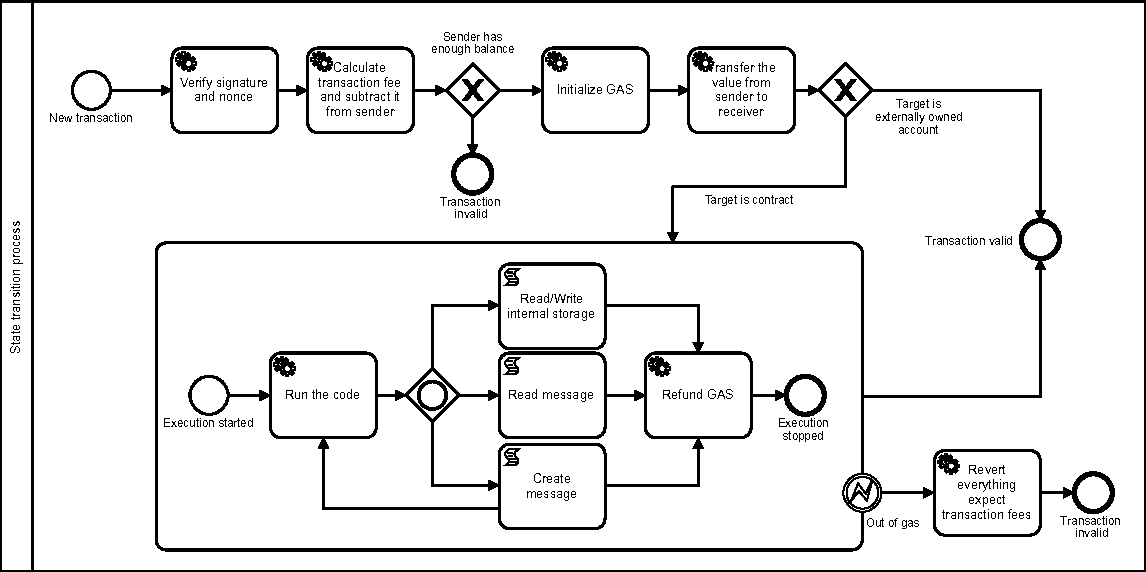


Hình 3. 4 Quá trình tạo một giao dịch trong Real estate Dapp ethereum

Quá trình tạo giao dịch nhìn rất phức tạp trong Ethereum, vì nó hỗ trợ các hợp đồng thông minh. Ban đầu, EOA có 2 lựa chọn (vì các chức năng hợp đồng không thể được gọi nếu không có hợp đồng), hoặc để gửi một giao dịch tiêu chuẩn đến một EOA khác (ví dụ chuyển ether) hoặc tạo hợp đồng mới. Khi hợp đồng đã được tạo, EOA có thể tạo giao dịch mới có thể gọi các chức năng của hợp đồng thông minh, được mô tả trong dữ liệu của giao dịch. Tài khoản hợp đồng có thể gửi tin nhắn. Thông báo trả về cơ bản giống như các giao dịch, khác biệt là nó được tạo bởi CA. CA-s có thể gửi ether đến một địa chỉ được lưu trữ trong bộ lưu trữ nội bộ của hợp đồng, hoặc tạo và gửi tin nhắn đến các hợp đồng khác.

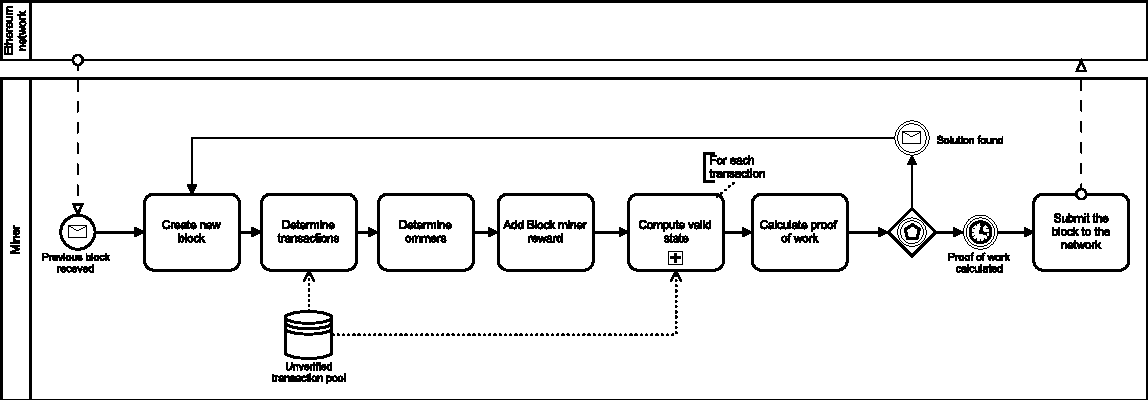
Trước khi xem xét quá trình xác nhận các giao dịch trong Ethereum, chúng ta cần hiểu khái niệm về chuyển đổi trạng thái của giao dịch. Ethereum giữ một trạng thái và giao dịch của người dùng sẽ làm thay đổi trạng thái đó. Nếu không có lỗi phát sinh trong việc thay đổi trạng thái, nó sẽ trở thành trạng thái mới trong khối tiếp theo. Quá trình chuyển đổi trạng thái bao gồm các bước sau:

* Xác minh cấu trúc của giao dịch (chữ ký và nonce).
* Xác minh rằng người gửi có đủ số dư để trả phí giao dịch.
* Khởi tạo GAS với số lượng đầu vào đã cho là START GAS.
* Chuyển giá trị từ người gửi sang người nhận. Nếu EOA không tồn tại, một EOA sẽ được tạo. Nếu mục tiêu là CA, mã được thực thi cho đến khi hoàn thành hoặc cho đến khi hết GAS. Hợp đồng có thể Đọc hoặc Ghi vào bộ nhớ trong của nó, đọc dữ liệu tin nhắn hoặc tạo một tin nhắn mới được gửi đến một bản sao khác hoặc tới EOA.
* Nếu thao tác chuyển không thành công vì người gửi EOA không có đủ số dư hoặc thực thi hết GAS thì tất cả các thay đổi từ khi thực hiện sẽ được chuyển về trạng thái ban đầu, ngoại trừ phí giao dịch sẽ không được hoàn trả lại.



Hình 3. 5 Quá trình thay đổi trạng thái khi thực hiện một giao dịch trong Dapp ethereum

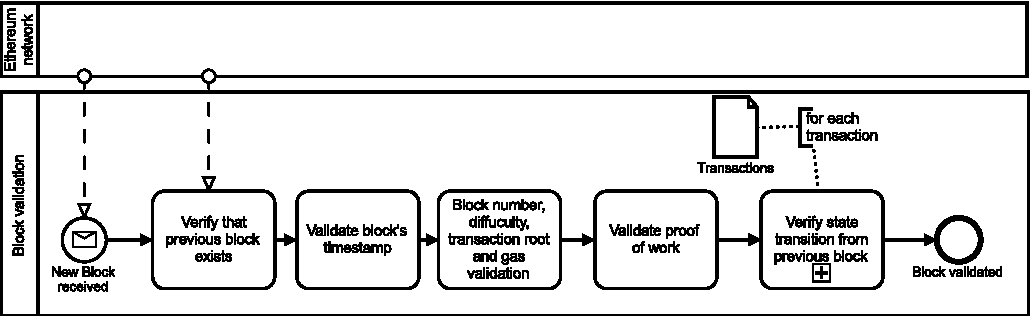
Khi một giao dịch được tạo trong ethereum thì cần phải được xác thực thông qua mining (quá trình khai thác hay còn gọi là “đào”, xác thực giao dịch) được thực hiện bởi miner ( người khai thác ).



Hình 3. 6 Quá trình mininng trong ethereum

Quá trình khai thác (mining process) trên hình bắt đầu khi một người khai thác (miner) phát hiện ra rằng các giao dịch mới được phát lên mạng (broadcasted to the network). Sau đó, người khai thác (miner) sẽ tìm thấy các ommers, là các khối hợp lệ, nhưng không phải là một phần của chuỗi chính và đưa chúng vào tiêu đề khối. Sau đó, người khai thác (miner) sẽ tính toán một trạng thái hợp lệ, có nghĩa là xác nhận các giao dịch và thông báo, thực thi mã và kiểm tra xem việc thực thi không hết Gas. Nếu một trạng thái hợp lệ được tính toán, người khai thác (miner) cũng phải cung cấp bằng chứng công việc(proof-of-work). Sau tất cả những điều này 26 người khai thác (miner) có thể phát khối (block) mới lên mạng. Nếu một khối (block) mới đã được phát sóng, người khai thác (miner) sẽ chuyển sang tạo khối tiếp theo.

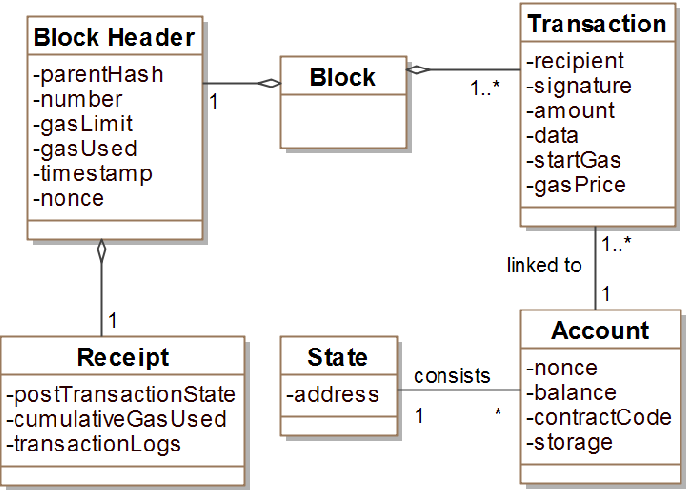
Quá trình xác nhận khối được mô tả trong Hình 3.7. Tất cả các thành phần khối khối được xác thực, bao gồm cả bằng chứng xác thực công việc (proof-of-work) và xác thực trạng thái (bằng cách thực hiện các giao dịch). Điều này được thực hiện bởi mọi nút đầy đủ trong mạng trước khi họ thêm khối mới vào blockchain.



Hình 3. 7 Quá trình xác nhận một block mới

Data models (mô hình dữ liệu): Mô hình dữ liệu được trình bày trong Hình 3.8. Tài khoản chứa một số nonce, bằng với số lượng giao dịch được gửi từ tài khoản đã cho và số dư của ether. Nếu tài khoản là một CA, đó cũng là mã hợp đồng và lưu trữ cho hợp đồng.

Giao dịch trong Ethereum là gói dữ liệu đã ký, lưu trữ một thông điệp sẽ được gửi từ EOA. Người nhận có thể là EOA hoặc CA. Đơn giản hóa, giao dịch chứa địa chỉ của người nhận, chữ ký của người gửi, từ đó có thể trích xuất địa chỉ, lượng ether, dữ liệu (trong trường hợp tạo hợp đồng hoặc gọi hàm) và start gas và gasPrice. Khối chứa danh sách các giao dịch. Ngoài ra, khối giữ một tiêu đề khối (block header), chứa nhiều thông tin nhất có lẽ là hàm băm(hash) của khối cha, block number, gas limit,thời gian tạo khối (timestamp) and nonce. Receipt là một thực thể lưu giữ thông tin về trạng thái giao dịch bài,tổng lượng Gas sử dụng tạo ra khối và nhật ký của các giao dịch. Hợp đồng có khả năng ghi vào nhật ký và chúng có thể được quan sát sau nếu cần.

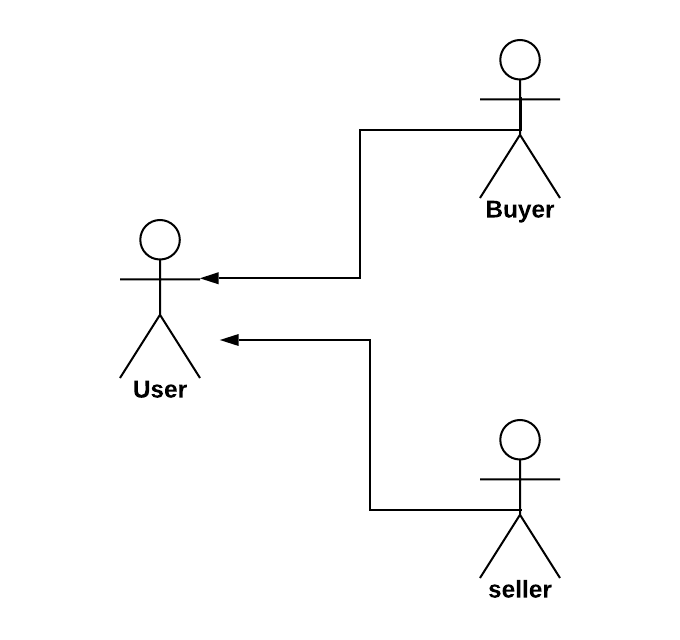


Hình 3. 8 Mô hình dữ liệu

### **2.2. Thiết kế Use Case Diagram**

#### **2.2.1. Danh sách tác nhân**

Các đối tượng giao tiếp với hệ thống bao gồm



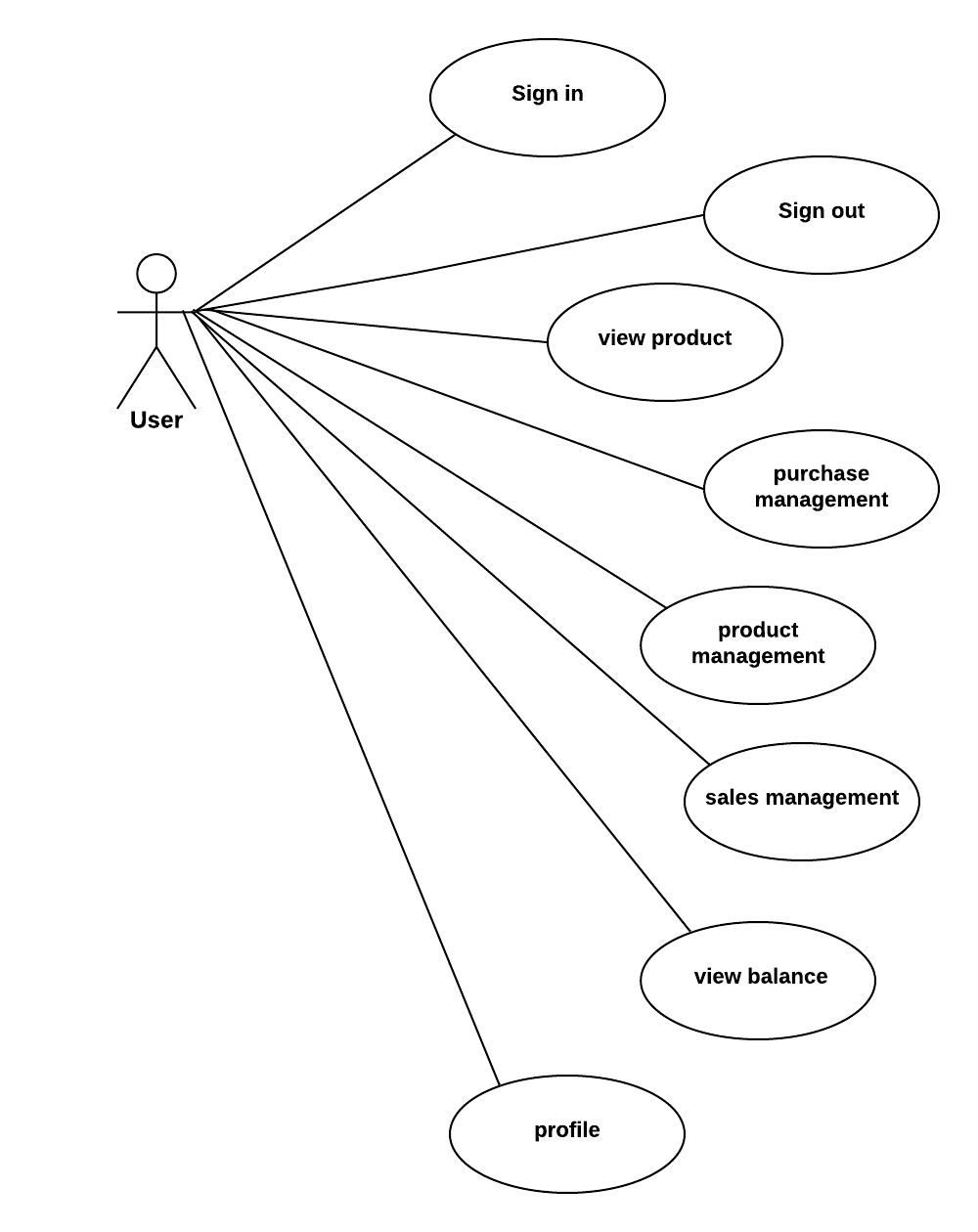
Hình 3. 9 Danh sách tác nhân sử dụng hệ thống.

Buyer: là những người sử dụng hệ thống với vai trò là người mua bất động sản.

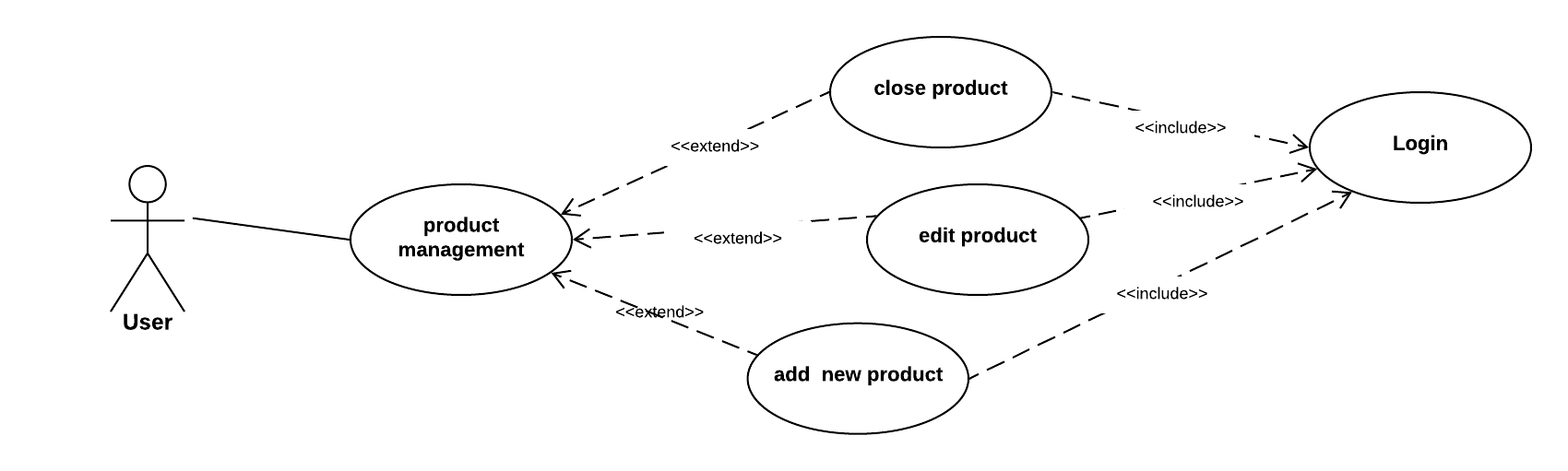
Seller: là những người sử dụng hệ thống với vai trò là người bán bất động sản.

#### **2.2.2. Use Case Diagram**

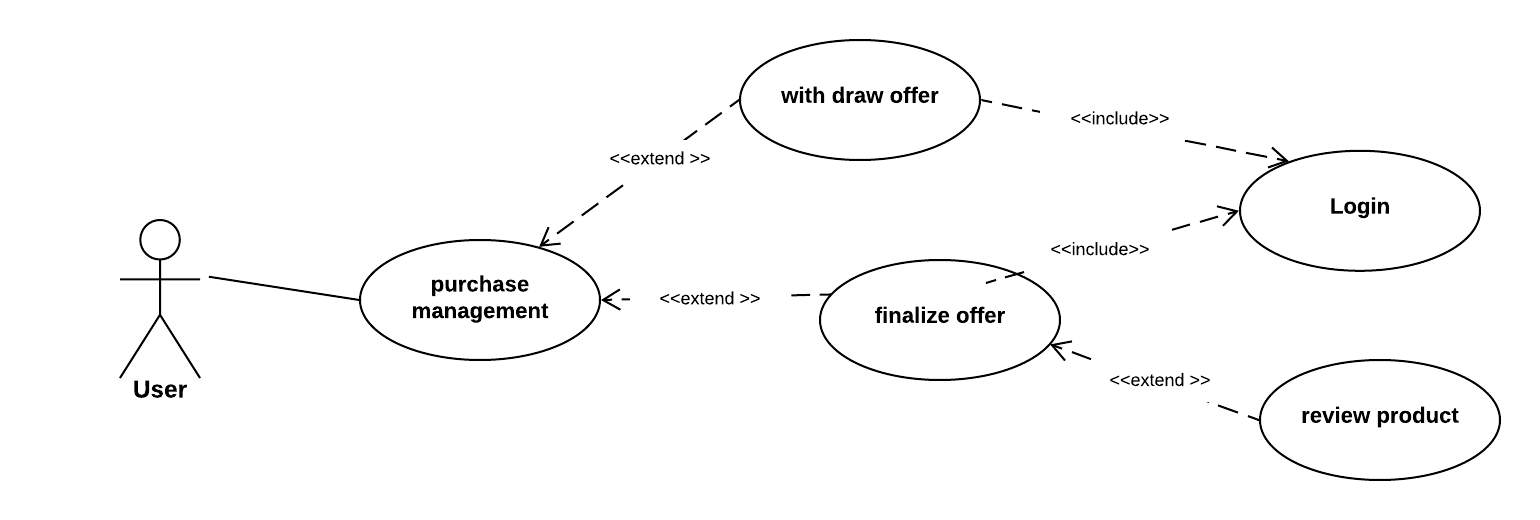
Use Case tổng quát



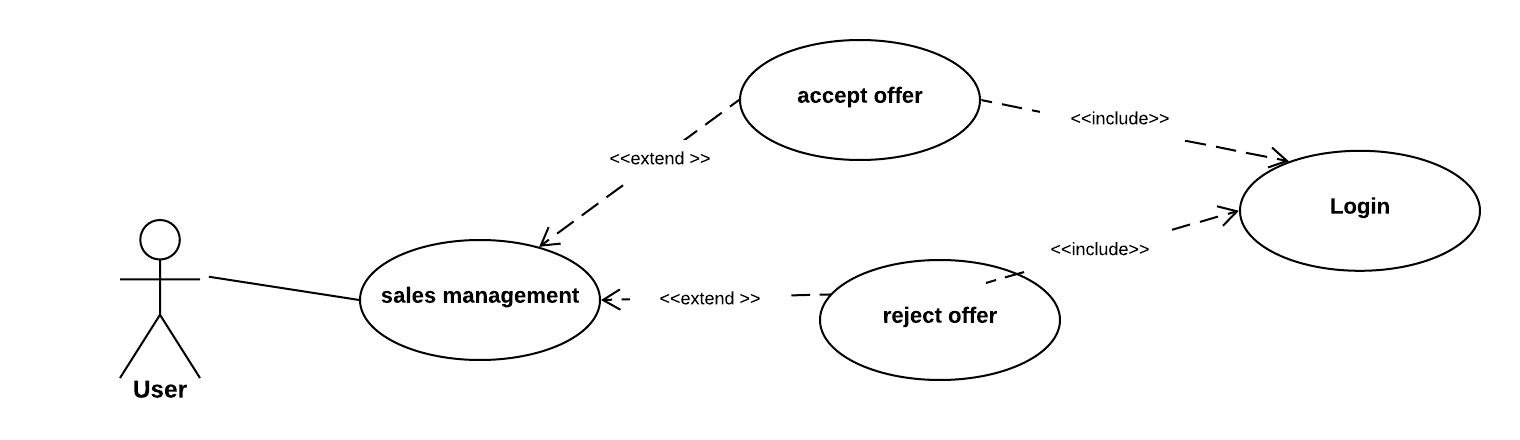
Hình 3. 10 Usecase tổng quát



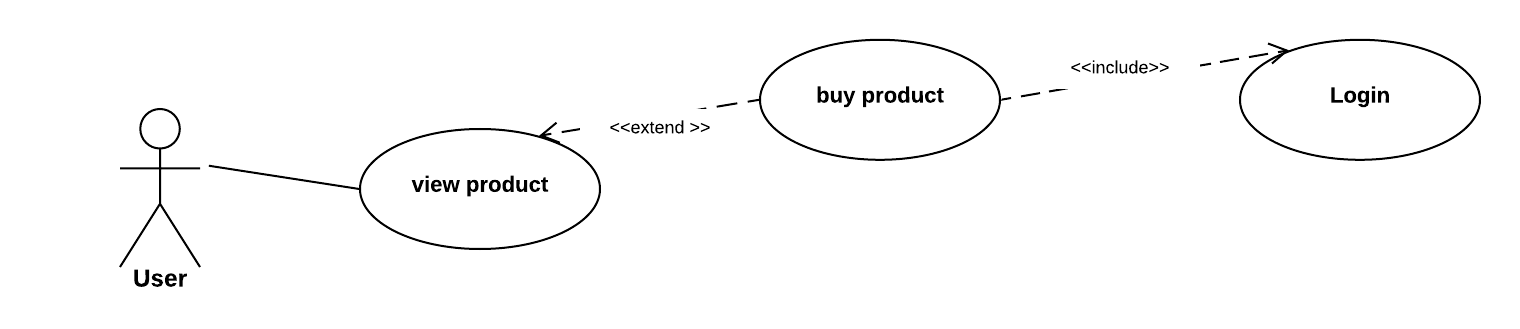
Hình 3. 11 Usecase quản lý sản phẩm



Hình 3. 12 Usecase quản lý đơn hàng mua



Hình 3. 13 Usecase quản lý đơn bán.



Hình 3. 14 Use case xem chi tiết sản phẩm

#### **2.2.3. Đặc tả Use Case**

Danh sách usecase

|  |  |  |  |
| --- | --- | --- | --- |
| Số Thứ Tự | Code | Tên Usecase | Mô Tả |
| 1 | UC01 | Đăng nhập | Người dùng đăng nhập vào hệ thống. |
| 2 | UC02 | Đăng xuất | Người dùng đăng nhập thoát khỏi hệ thống. |
| 3 | UC03 | Xem danh sách bất đông sản bán | Người dùng xem danh sách bất động sản đang bán ở trang chủ. |
| 4 | UC04 | Xem chi tiết bất động sản | Người dùng xem chi tiết thông tin bất động sản, thực hiện thao tác đặt mua. |
| 5 | UC05 | Xem danh sách các đơn hàng đang mua | Người dùng xem chi tiết trạng thái của các đơn hàng đặt mua( accept, reject, finalize, pending). |
| 6 | UC06 | Hủy đơn hàng | Người mua hủy yêu cầu mua và nhận lại tiền. |
| 7 | UC07 | Hoàn thành đơn hàng | Người mua kết thúc đơn hàng.  Bình luận về sản phẩm đã mua. |
| 8 | UC08 | Xem danh sách bất động sản đang được bán của người dùng | Người dùng xem danh sách các bất động sản đang bán của mình. |
| 9 | UC09 | Cập nhật thông tin | Người dùng cập nhật thông tin bất động sản đang bán. |
| 10 | UC10 | Đóng sản phẩm | Người dùng ngừng bán bất động sản hiện tại |
| 11 | UC11 | Thêm bất động sản | Người dùng thêm thông tin bất động sản mới. |
| 12 | UC12 | Xem danh sách các sản phẩm bất động sản bán. | Người bán xem danh sách các sản phẩm bất động sản đang được yêu cầu từ người mua. |
| 13 | UC13 | Chấp nhận yêu cầu | Người bán chấp nhận đơn hàng từ người mua. |
| 14 | UC14 | Từ chối yêu cầu | Người bán từ chối yêu cầu từ người mua. |
| 15 | UC15 | Xem thông tin số dư | Người dùng xem số dư tài khoản |
| 16 | UC16 | Xem thông tin | Người dùng xem thông tin cá nhân, cập nhật thông tin. |

Đặc tả

UC01: Đăng nhập

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Name** | | Đăng nhập | | **Code** | | UC01 |
| **Description** | | Người dùng đăng nhập vào hệ thống. | | | | |
| **Actor** | | User | | **Trigger** | |  |
| **Pre-condition** | | Người dùng đã có tài khoản trên metamask | | | | |
| **Post condition** | |  | | | | |
| **Actor** | | | **System** | | | |
| **Main Flow: Đăng nhập thành công** | | | | | | |
| 1 | Vào trang web | |  | | Hệ thống chuyển qua trang đăng nhập bằng metamask. | |
| 2 | Người dùng nhập thông tin: mật khẩu | |  | |  | |
| 3 | Người dùng nhấn “Login” | |  | | Đăng nhập thành công, chuyển vào trang web | |

**Activities**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Actor** | | | **System** | |
| **Alternative Flow: Đăng nhập thất bại** | | | | |
| 4 | Nhập sai mật khẩu hoặc chưa có tài khoản trên metamask |  | | Hiển thị thông báo lỗi. |

Bảng 3. 1 Bảng đặc tả use case đăng nhập.

UC02: Đăng xuất

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Đăng xuất | **Code** | UC02 |
| **Description** | Người dùng đăng xuất khỏi hệ thống. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập vào hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |
| --- | --- | --- | --- |
| **Actor** | | **System** | |
| **Main Flow: Đăng xuất thành công** | | | |
| 1 | Từ màn hi |  | Hệ thống tự đăng xuất. |

Bảng 3. 2 Bảng đặc tả use case đăng xuất

UC03: Xem danh sách bất đông sản bán

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem danh sách bất đông sản bán | **Code** | UC03 |
| **Description** | Người dùng xem danh sách bất động sản đang bán ở trang chủ. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: Hiển thị thành công** | | | | | | |
| 1 | Khi người dùng đăng nhập thành công vào hệ thống | | |  | | Hệ thống lấy thông tin bất động sản hiển cho người dùng |
| **Alternative Flow: Hiển thị thất bại** | | | | | | |
| 2 | |  |  | | Hiển thị thông báo lỗi cho người dùng | |

Bảng 3. 3 Bảng đặc tả use case xem danh sách bất động sản

UC04: Xem chi tiết bất động sản

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem chi tiết bất động sản | **Code** | UC04 |
| **Description** | Người dùng xem chi tiết thông tin bất động sản, thực hiện thao tác đặt mua | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: Xem chi tiết bất động sản thành công, đặt mua thành công** | | | | | | |
| 1 | Khi người dùng click vào một bất động sản | | |  | | Hệ thống đưa người dùng đến trang xem chi tiết với thông tin bất động sản đã chọn. |
| 2 | Khi người dùng thao tác trên slide bar | | |  | | Hệ thống tự thay tương ứng |
| 3 | Người dùng thao tác đặt mua | | |  | | Hệ thống kiểm tra thông tin tài khoản,yêu cầu người dùng ký giao dịch qua metamask.  Sản phẩm được thêm vào purchase ( trạng thái hiển thị pending).  Khi người bán chấp nhận đơn hàng |
| 4 | Người dùng xác nhận hoàn thành đơn hàng | | |  | | Hệ thống chuyển tiền cho người bán. Trạng thái sản phẩm trong purchase ( finalize) |
| **Alternative Flow: Xem chi tiết bất động sản thất bại, đặt mua thất bại** | | | | | | |
| 5 | |  |  | | Hiển thị thông báo lỗi cho người dùng  Đặt mua thất bại do số dư trong tài khoản không đủ. Hiển thị thông báo cho người dùng | |
| 6 | | Người bán từ chối đơn hàng |  | | Hiển thị trạng thái sản phẩm trong purchase là reject. | |

Bảng 3. 4 Bảng đặc tả use case xem chi tiết bất động sản

UC05: Xem danh sách các đơn hàng đang mua

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem danh sách các đơn hàng đang mua | **Code** | UC05 |
| **Description** | Người dùng xem chi tiết trạng thái của các đơn hàng đặt mua( accept, reject, finalize, pending). | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: Xem danh sách các đơn hàng đang mua thành công** | | | | | | |
| 1 | Khi người dùng click vào purchase | | |  | | Hệ thống đưa người dùng đến trang purchase. Hiển thị thành công các đơn hàng cũng như trạng thái các đơn hàng.  Hiển thị không có đơn hàng nếu như người dùng chưa thực hiện bất kỳ 1 giao dịch mua nào |
| **Alternative Flow: Xem danh sách các đơn hàng đang mua thất bại** | | | | | | |
| 2 | | Lỗi do không lấy được thông tin từ cơ sở dữ liệu |  | | Hiển thị thông báo lỗi cho người dùng | |

Bảng 3. 5 Bảng đặc tả use case xem danh sách các đơn hàng đang mua

UC06: Hủy đơn hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Hủy đơn hàng | **Code** | UC06 |
| **Description** | Người mua hủy yêu cầu mua và nhận lại tiền. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống.  Thực hiện yêu cầu mua 1 bất động sản tới người bán, trạng thái của bất động sản pending | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: Hủy đơn hàng thành công** | | | | | | |
| 1 | Khi người dùng click vào purchase | | |  | | Hệ thống đưa người dùng đến trang purchase. Hiển thị thành công các đơn hàng cũng như trạng thái các đơn hàng.  Hiển thị không có đơn hàng nếu như người dùng chưa thực hiện bất kỳ 1 giao dịch mua nào |
| 2 | Người dùng chọn withdraw offer | | |  | | Hệ thống yêu cầu người dùng xác thực qua metamask |
| 3 | Người dùng đồng ý | | |  | | Hệ thống thực hiện withdraw, hoàn trả tiền cho người dùng. Trạng thái đơn hàng chuyển thành withdraw offer |
| **Alternative Flow: Hủy đơn hàng thất bại** | | | | | | |
| 4 | |  |  | | Hiển thị thông báo lỗi cho người dùng | |

Bảng 3. 6 Bảng đặc tả use case hủy đơn hàng

UC07: Hoàn thành đơn hàng

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Hoàn thành đơn hàng | **Code** | UC07 |
| **Description** | Người mua kết thúc đơn hàng.Bình luận về sản phẩm đã mua. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống.  Thực hiện yêu cầu mua 1 bất động sản tới người bán, người mua chấp nhận đơn hàng, trạng thái của bất động sản accept | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow:** Hoàn thành đơn hàng **thành công** | | | | | | |
| 1 | Khi người dùng click vào purchase | | |  | | Hệ thống đưa người dùng đến trang purchase. Hiển thị thành công các đơn hàng cũng như trạng thái các đơn hàng.  Hiển thị không có đơn hàng nếu như người dùng chưa thực hiện bất kỳ 1 giao dịch mua nào |
| 2 | Người dùng chọn 1 đơn hàng trạng thái accept | | |  | | Hệ thống đưa người dùng đến trang chi tiết đơn hàng |
| 3 | Người dùng chọn finalize | | |  | | Hệ thống yêu cầu người dùng xác thực qua metamask. |
| 4 | Người dùng chọn đồng ý | | |  | | Hệ thống hiển thực hiện hoàn thành đơn hàng. Số tiền của đơn hàng được chuyển cho người bán. |
| **Alternative Flow:** Hoàn thành đơn hàng **thất bại** | | | | | | |
| 5 | |  |  | | Hiển thị thông báo lỗi cho người dùng | |

Bảng 3. 7 Bảng đặc tả use case hoàn thành đơn hàng

UC08: Xem danh sách bất động sản đang được bán của người dùng

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem danh sách bất động sản đang được bán của người dùng | **Code** | UC08 |
| **Description** | Người dùng xem danh sách các bất động sản đang bán của mình trên hệ thống. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống. | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: hiển thị thành công danh sách bất động** | | | | | | |
| 1 | Từ màn hình người dùng chọn mylisting | | |  | | Hệ thống chuyển qua trang danh sách bất động sản bán trên hệ thống của người dùng.  ( hệ thống cho phép người dùng thao tác cập nhật thông tin, cũng như ngừng bán bất động sản) |
| **Alternative Flow: hiện thị thất bại** | | | | | | |
| 5 | | Hệ thống không lấy được thông tin |  | | Hiển thị thông báo lỗi của người dùng | |

Bảng 3.8: Xem danh sách bất động sản đang được bán của người dùng

UC09: Cập nhật thông tin

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Cập nhật thông tin | **Code** | UC10 |
| **Description** | Người dùng cập nhật thông tin bất động sản đang bán | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đăng nhập hệ thống, có bất động sản đang được bán trên hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: cập nhật thông tin thành công** | | | | | | |
| 1 | Người dùng click editlisting của bất động sản trong mylisting | | |  | | Hệ thống chuyển qua trang chỉnh sửa thông tin sản phẩm |
| 2 | Người dùng thay đổi thông tin | | |  | | Hệ thống kiểm tra dữ liệu vào |
| 3 |  | | |  | | Hệ thống yêu cầu người dùng xác nhận qua metamask |
| 4 |  | | |  | | Hệ thống hiện thị thành công. |
| **Alternative Flow: cập nhật thông tin thất bại** | | | | | | |
| 5 | | Thông tin người dùng nhập vào thiếu hoặc sai |  | | Hiển thị thông báo lỗi để người dùng chỉnh sửa lại thông tin | |
| 6 | | Số dư trong tài khoản người dùng không đủ để thực hiện thay đổi thông tin |  | | Metamask thông báo tài khoản người dùng không đủ eth thực hiện giao dich. Người dùng mua thêm eth để thực hiện giao dich. | |

Bảng 3.9: Cập nhật thông tin.

UC10: Đóng sản phẩm

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Đóng sản phẩm | **Code** | UC10 |
| **Description** | Người dùng ngừng bán một bất động sản trong danh sách bán hiện tại. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** |  | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: ngừng bán bất động sản thành công** | | | | | | |
| 1 | Người dùng click close listing của bất động sản trong mylisting | | |  | | Hệ thống hiển thị thông báo yêu cầu người dùng xác nhận |
| 2 | Người dùng xác nhận | | |  | | Metamask yêu cầu người dùng xác nhận giao dich |
| 3 | Người dùng xác nhận | | |  | | Hiển thị thông báo thành công |
| **Alternative Flow: ngừng bán bất động sản thất bại** | | | | | | |
| 4 | | Người dùng hủy yêu cầu |  | | Hệ thống xác nhận người dùng hủy yêu cầu | |
| 5 | | Số dư trong tài khoản không đủ thực hiện yêu câu |  | | Metamask thông báo tìa khoản người dùng không đủ thực hiện giao dich. | |

Bảng 3.10: Ngưng bán sản phẩm

UC11: Thêm bất động sản

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Thêm bất động sản | **Code** | UC11 |
| **Description** | Người dùng thêm thông tin bất động sản mới. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đăng nhập thành công | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: thêm thành công** | | | | | | |
| 1 | Người click add listing từ màn hình chính | | |  | | Hệ thống chuyển qua trang thêm bất đông sản mới |
| 2 | Người dùng nhập thông tin | | |  | | Hệ thống hiển thi trang review your listing |
| 3 | Người dùng chọn Done | | |  | | Metamask yêu cầu người dùng xác nhận |
| 4 | Người dùng xác nhận | | |  | | Hệ thống hiển thị đơn hàng tạo thành công |
| **Alternative Flow: thêm thất bại** | | | | | | |
| 5 | | Người dùng nhập thông tin sai hoặc thiểu |  | | Hệ thống hiển thị thông báo lỗi để người dùng sửa lại | |
| 6 | | Sô dư tài khoản người dùng không đủ để thực hiện giao dịch |  | | Metamask thông báo số dư tài khoản không đủ thực hiện giao dịch. | |

Hình 3.11: Thêm thông tin bất đông sản mới.

UC12: Xem danh sách các bất động sản đang bán

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Xem danh sách các sản phẩm bất động sản đang bán | **Code** | UC12 |
| **Description** | Người bán xem danh sách các sản phẩm bất động sản đang được yêu cầu từ người mua. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đăng nhập hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: hiển thị danh sách bất động sản đang chờ xác nhận từ người bán.** | | | | | | |
| 1 | Người dùng chọn tab pending trong trang mysale | | |  | | Hệ thống hiện thị danh sách đang chờ người bán chấp nhận |
| **Alternative Flow: hệ thống hiện thị danh sách bất động sản đang chờ xác nhận từ người bán thất bại** | | | | | | |
| 5 | |  |  | | Hiên thị thông báo lỗi | |

Bảng 3.12: Xem danh sách bất động sản chờ xác nhận từ người bán

UC13: Chấp nhận yêu cầu

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Chấp nhận yêu cầu | **Code** | UC13 |
| **Description** | Người bán chấp nhận đơn hàng từ người mua. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống | | |
| **Post condition** |  | | |

**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: chấp nhận yêu cầu thành công** | | | | | | |
| 1 | Người dùng chọn một bất động sản trong mysales ở tab pending | | |  | | Hệ thống hiển thị thông tin bất động sản |
| 2 | Người bán chấp nhận yêu cầu từ người mua | | |  | | Hệ thống xác nhận, metamask yêu cầu người dùng xác nhân |
| 3 | Người dùng xác nhận trên metamask | | |  | | Metamask xác nhận thành công  Hệ thống trả về xác nhận đơn hàng thành công (trạng thái đơn hàng chuyển sang accept tư f người bán) |
| **Alternative Flow: chấp nhận yêu cầu thất bại** | | | | | | |
| 5 | |  |  | | Metamask thông báo số dư tài khoàn không đủ để thực hiện xác nhận giao dich. | |

Bảng 3.13: người bán chập nhận yêu cầu từ người mua

UC14: Từ chối yêu cầu

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | Từ chối yêu cầu | **Code** | UC14 |
| **Description** | Người bán từ chối yêu cầu từ người mua. | | |
| **Actor** | User | **Trigger** |  |
| **Pre-condition** | Người dùng đã đăng nhập hệ thống | | |
| **Post condition** |  | | |

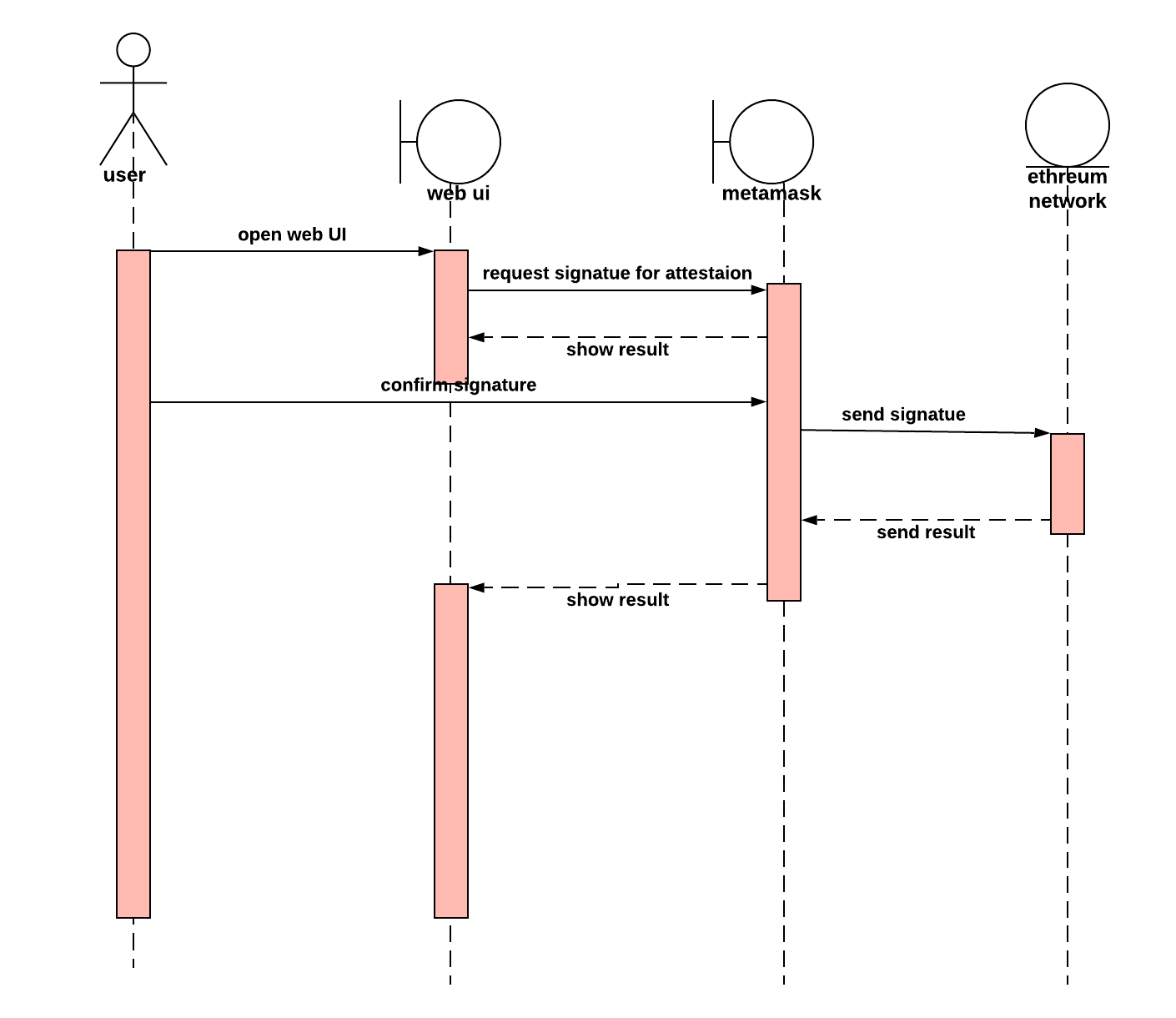
**Activities**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Actor** | | | | **System** | | |
| **Main Flow: từ chối yêu cầu thành công** | | | | | | |
| 1 | Người dùng chọn một bất động sản trong mysales ở tab pending | | |  | | Hệ thống hiển thị thông tin bất động sản |
| 2 | Người dùng từ chối yêu cầu từ người mua | | |  | | Hệ thống thống báo xác nhận qua metamask |
| 3 | Người dùng xác nhận | | |  | | Thông báo từ chối đơn hàng thành công |
| **Alternative Flow: từ chối yêu cầu thất bại** | | | | | | |
| 5 | |  |  | | Metamask thông báo số dư tài khoản người dùng không đủ để thực hiện giao dịch | |

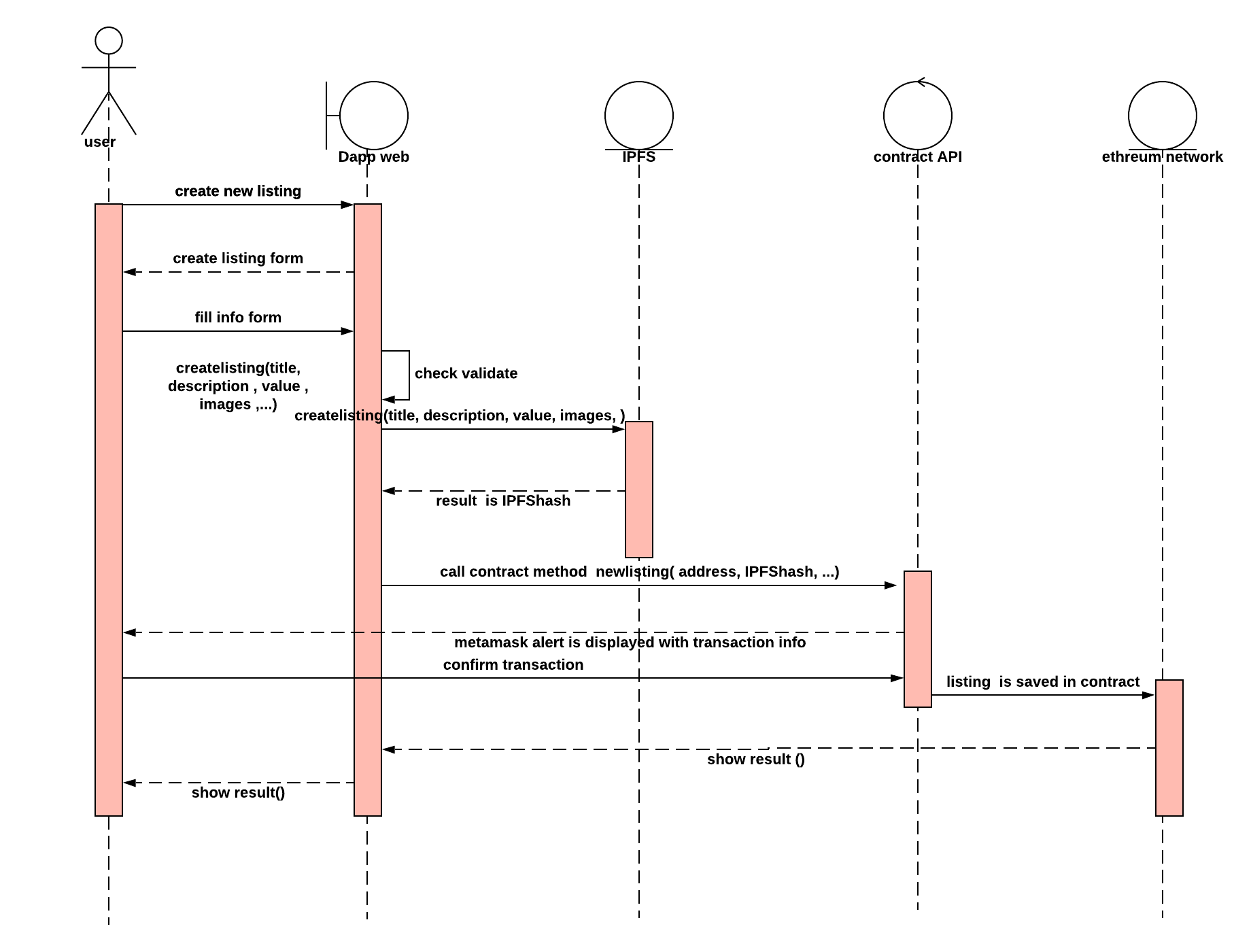
Bảng 3.14: Từ chối yêu cầu.

### **Sequence diagram**

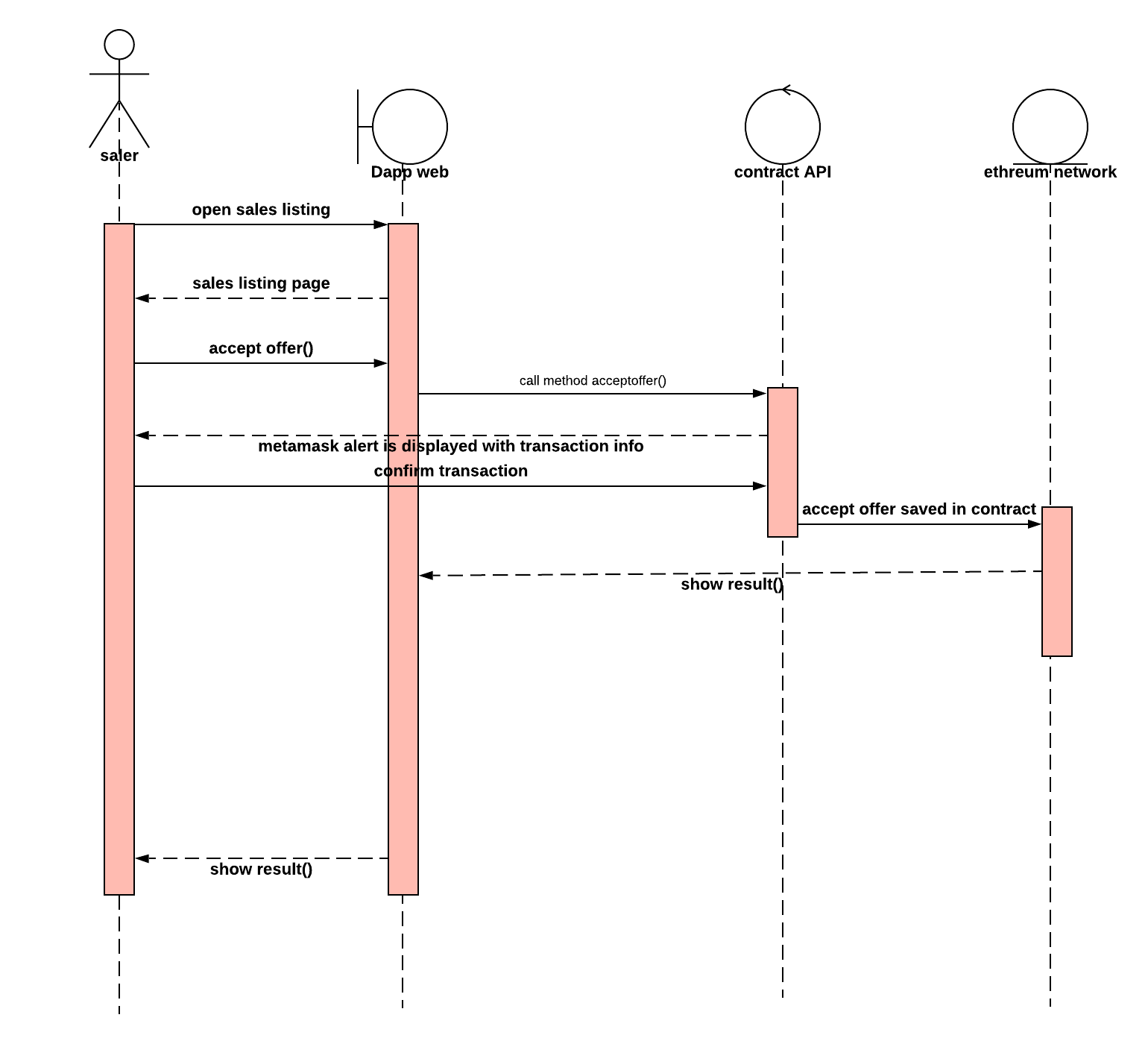
Lược đồ tuần tự chức năng đăng nhập dùng metamask



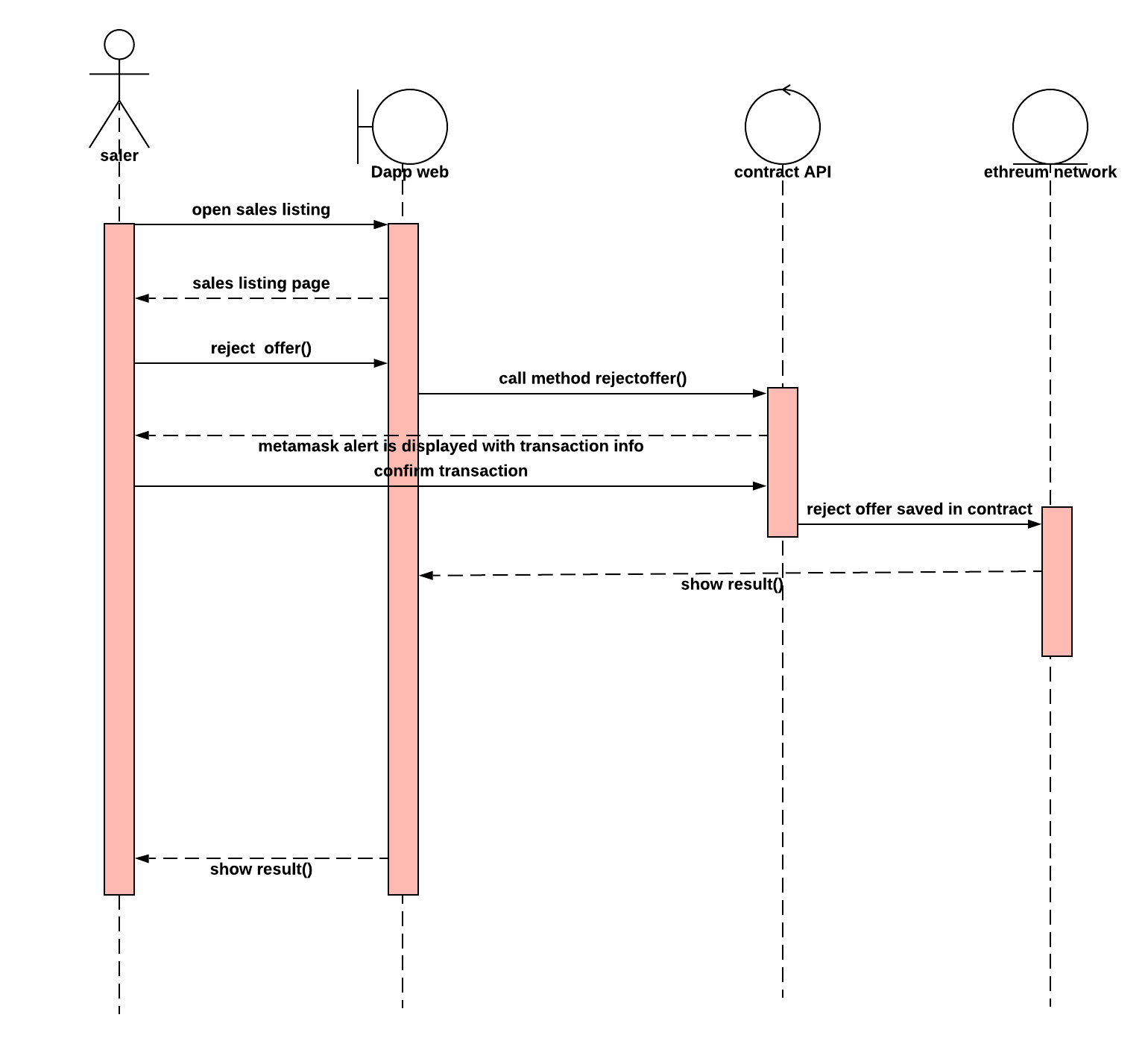
Tạo một bất động sản mới



Người bán chấp nhận đơn hàng từ người mua

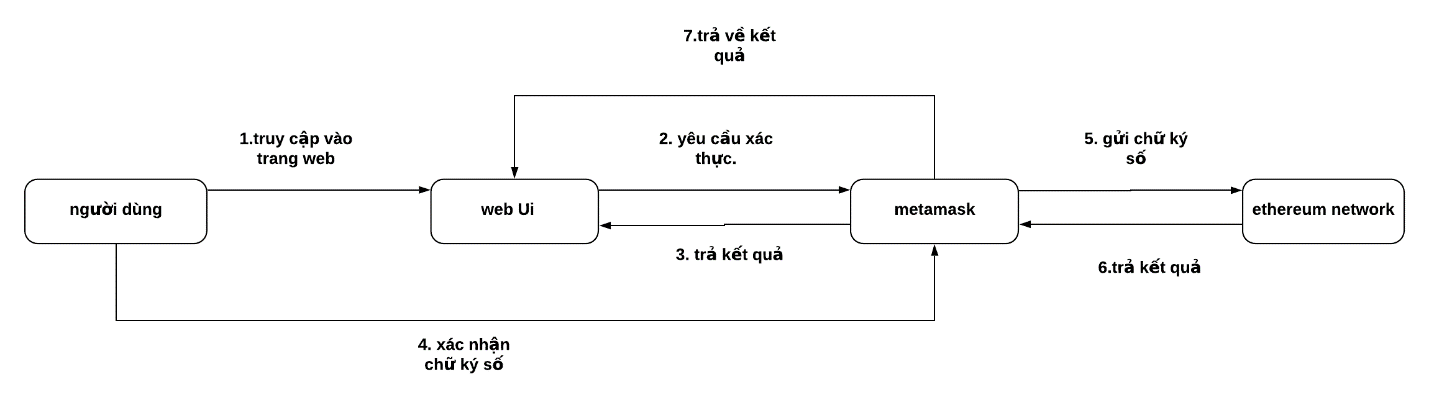


Người bán từ chối đơn hàng từ người mua

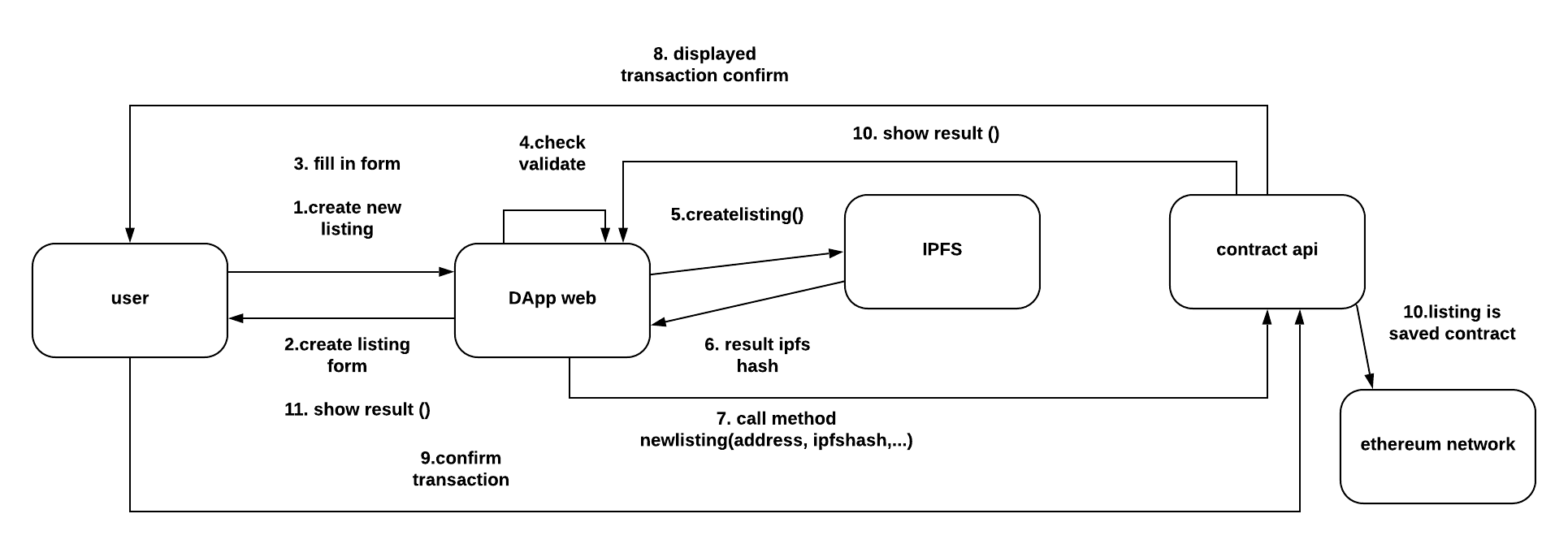


### **Activity diagram**

Chức năng nhập



Tạo một bất động sản mới



# **CHƯƠNG 4. KẾT QUẢ ĐẠT ĐƯỢC VÀ HƯỚNG PHÁT TRIỂN ĐỀ TÀI**

1. **Kết quả đạt được**

**Ưu điểm:**

* Trang web có giao diện trực quan, dễ sử dụng, đơn giản, được tổ chức một cách chặt chẽ và khoa học.
* Giúp người bán dễ quản lý các sản phẩm của mình.
* Áp dụng công nghệ Blockchain nên đảm bảo an toàn cho người bán và người mua.
* Có thông báo hiển thị tình trạng của sản phẩm trong quá trình mua/bán.

**Nhược điểm:**

Nhóm đã hoàn thành được phần cơ bản các chức năng cần thiết của một trang web quản lý và trao đổi bất động sản. Tuy nhiên, do năng lực của bản thân và thời gian có hạn, đề tài vẫn không tránh khỏi những sai sót, và trang web vẫn còn hạn chế nhiều về mặt giao diện, chức năng.

Vẫn còn nhiều chức năng nhóm chưa thực hiện được. Vẫn còn nhiều thiết sót về mặt logic và nghiệp vụ.

Chưa thể xử lý hết các trường hợp có thể xảy ra lỗi.

* 1. **Hướng phát triển**

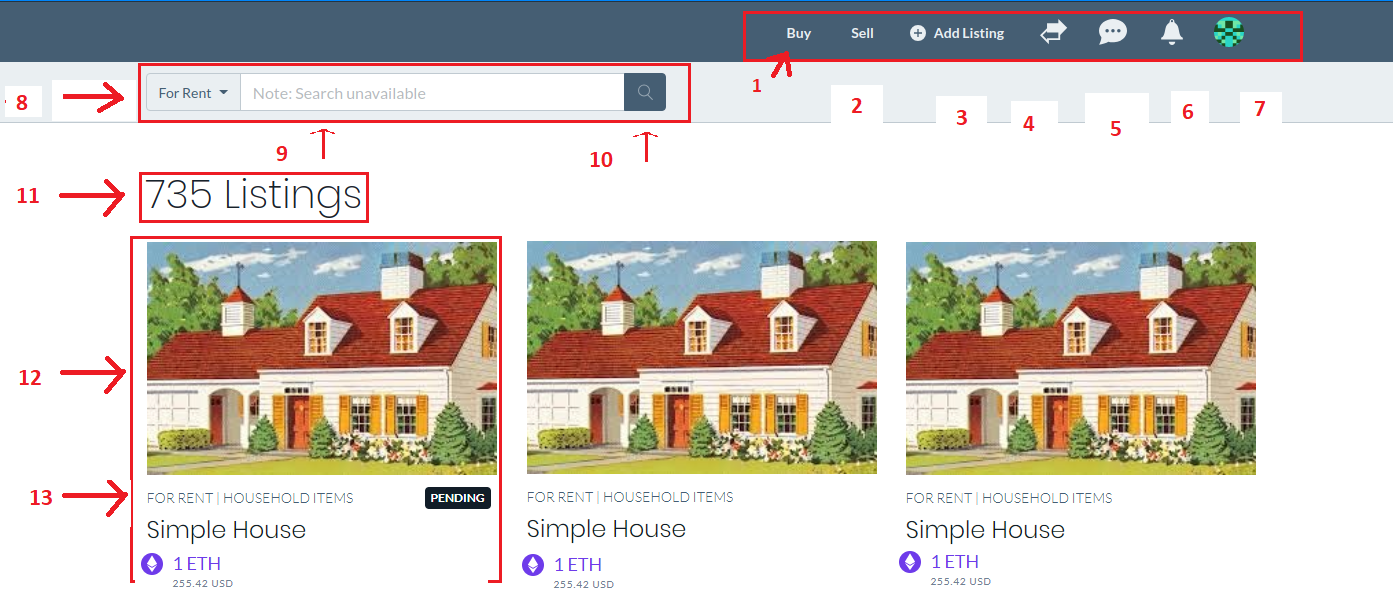
Nhóm sẽ tiếp tục phát triển và thực hiện nhiều ý tưởng hơn nữa trong thời gian tới. Sau đây là một số ý tưởng mà nhóm đã có được nhưng vẫn chưa thể thực hiện ngay lúc này:

* Cho phép người bán có thể áp dụng các khuyến mãi lên sản phẩm của mình. Ví dụ như nếu mua và trả tiền 1 lần sẽ nhận được các ưu đãi như giảm giá hoặc các ưu đãi khác. Hoặc có thể mua sản phẩm theo hình thức trả góp.
* Có thể phát triển để đăng bán các căn hộ chung cư của các dự án lớn.

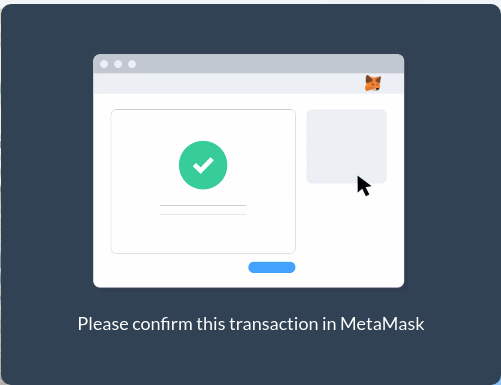
# **TÀI LIỆU THAM KHẢO**

1. <https://medium.com/@s_van_laar/deploy-a-private-ipfs-network-on-ubuntu-in-5-steps-5aad95f7261b>
2. <https://medium.com/birds-view/mapping-the-decentralized-world-of-tomorrow-5bf36b973203>
3. <https://www.freecodecamp.org/news/how-to-design-a-secure-backend-for-your-decentralized-application-9541b5d8bddb/>
4. <https://kipalog.com/posts/Cong-cu-cho-phat-trien-DApps-tren-Ethereum---HelloWorld-demo>
5. <https://github.com/ethereum/web3.py>
6. <https://web3js.readthedocs.io/en/1.0/getting-started.html>
7. <https://medium.com/@s_van_laar/deploy-a-private-ipfs-network-on-ubuntu-in-5-steps-5aad95f7261b>
8. <https://blog.zeppelin.solutions/designing-the-architecture-for-your-ethereum-application-9cec086f8317>

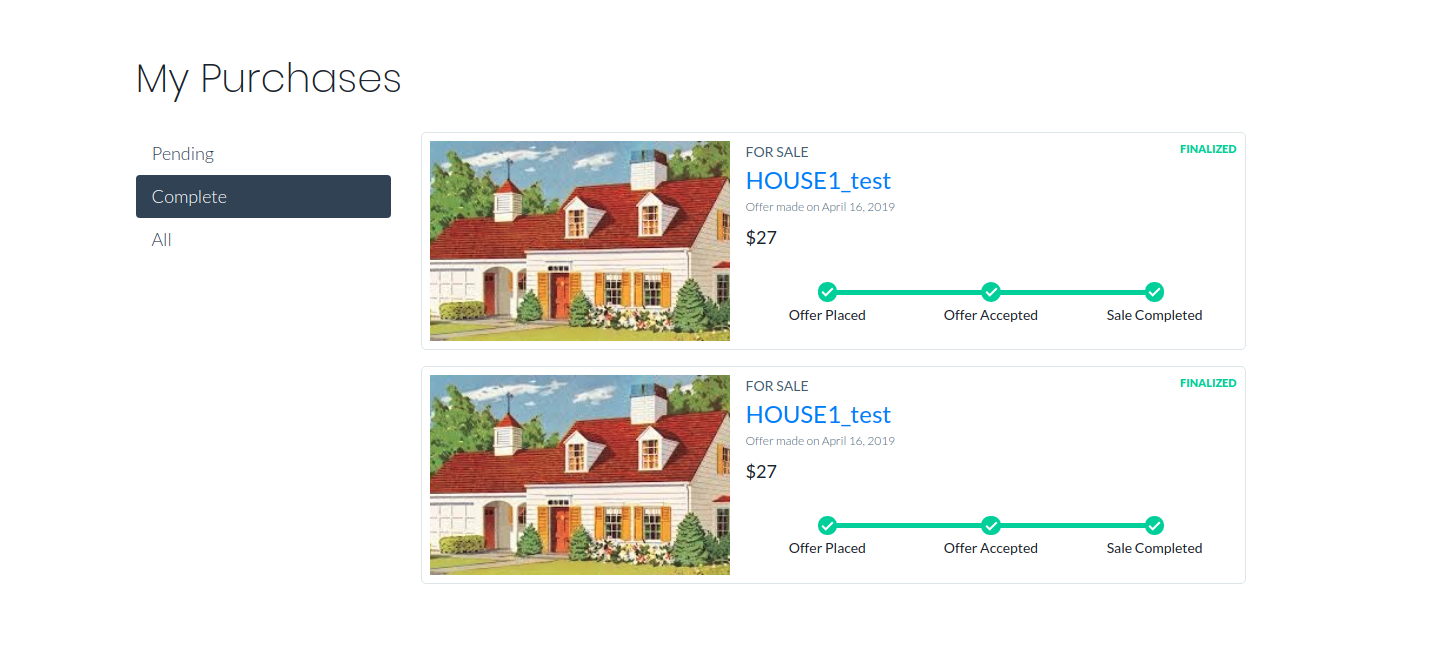
# **PHỤ LỤC 1: GIAO DIỆN SẢN PHẨM**



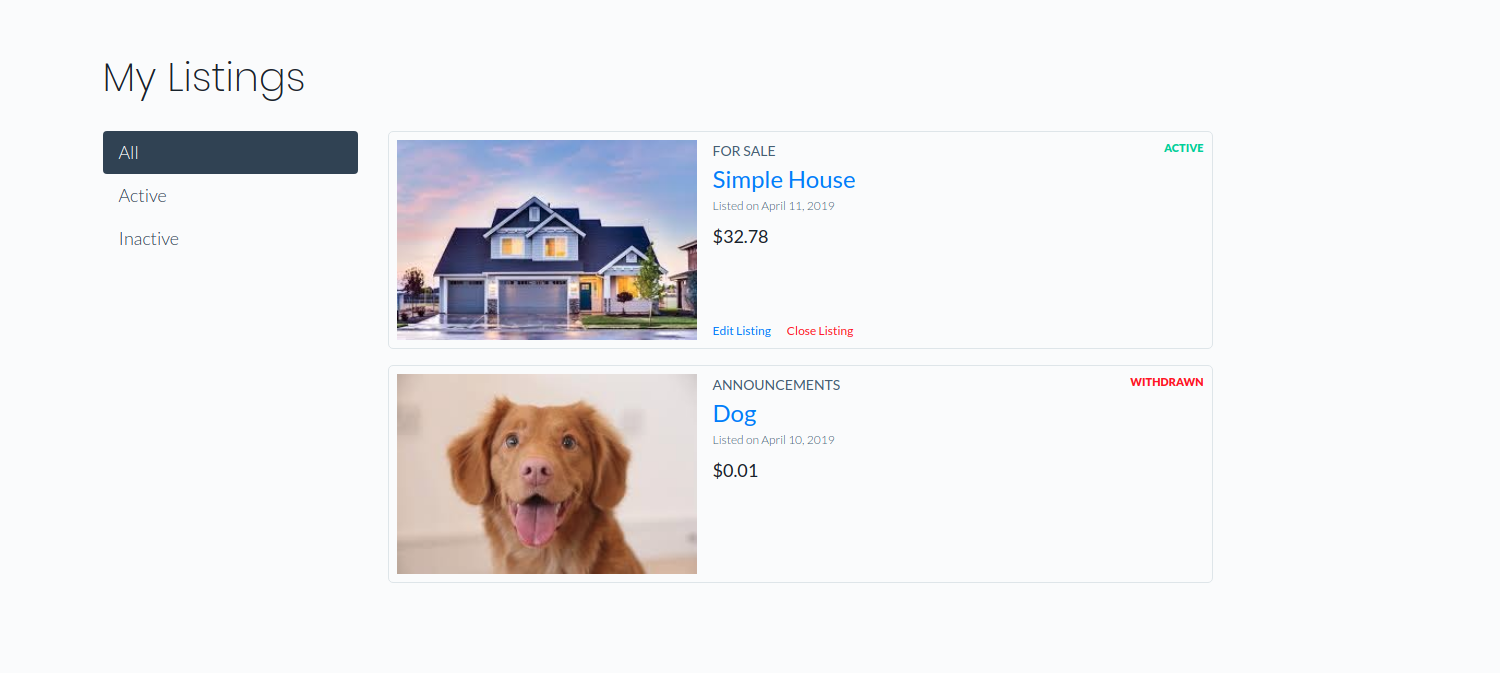
Hình 4. 1 Màn hình Home page



Hình 4. 2 màn hình hướng dẫn xác thực bằng metamask



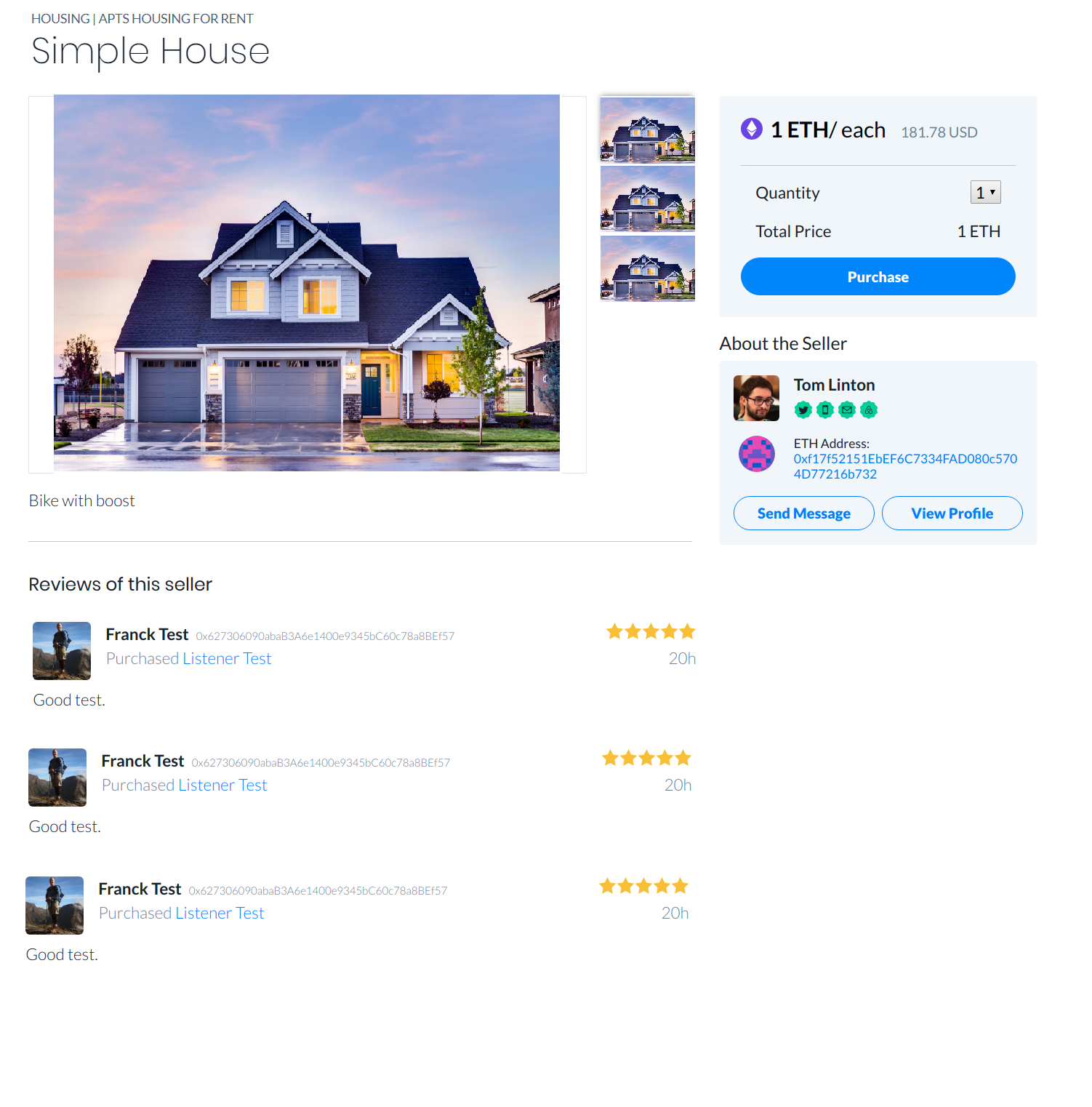
Hình 4. 3 Màn hình My Purchase



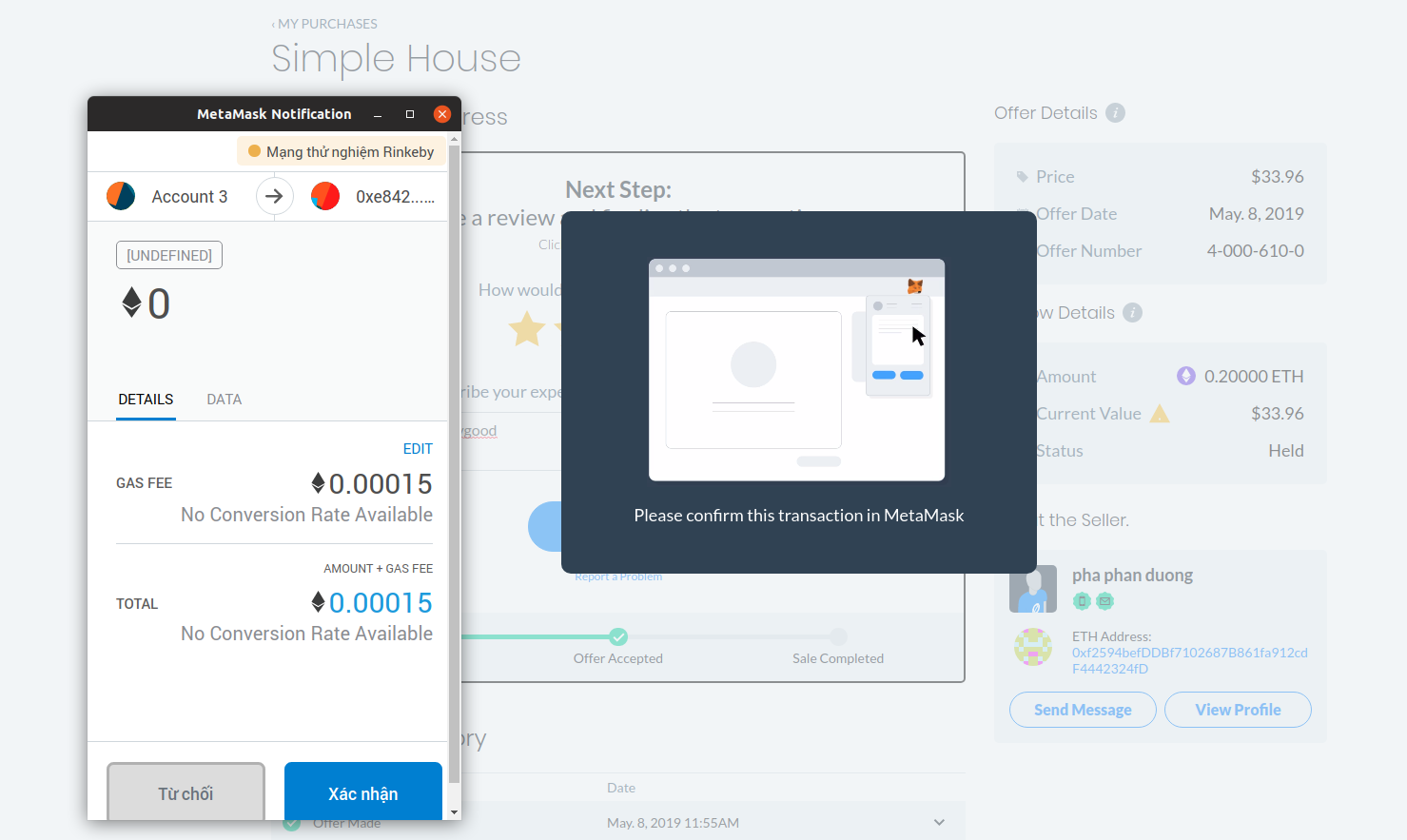
Hình 4. 4 Màn hình Sell Listing



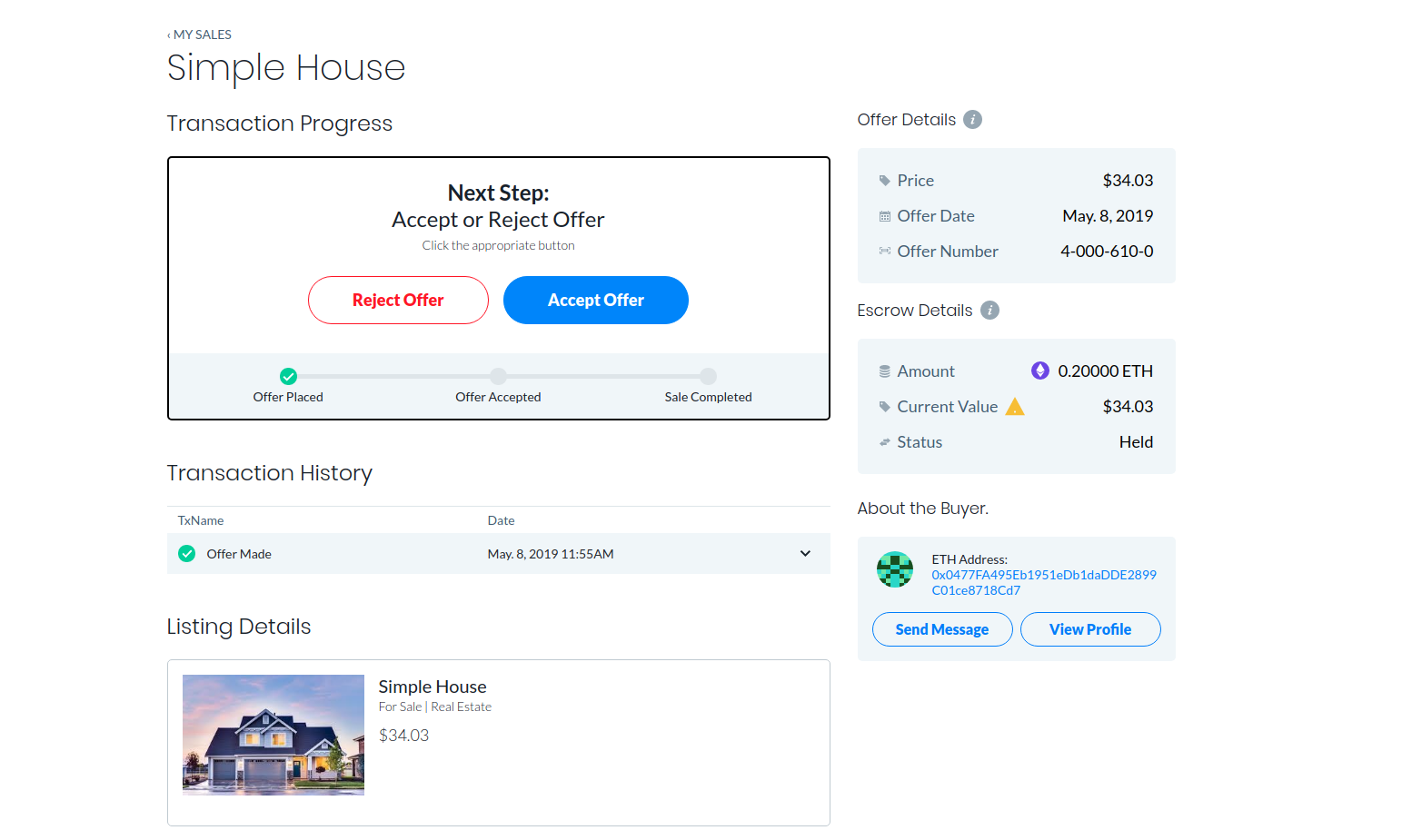
Hình 4. 5 Thêm một sản phẩm mới



Hình 4. 6 Xem chi tiết sản phẩm



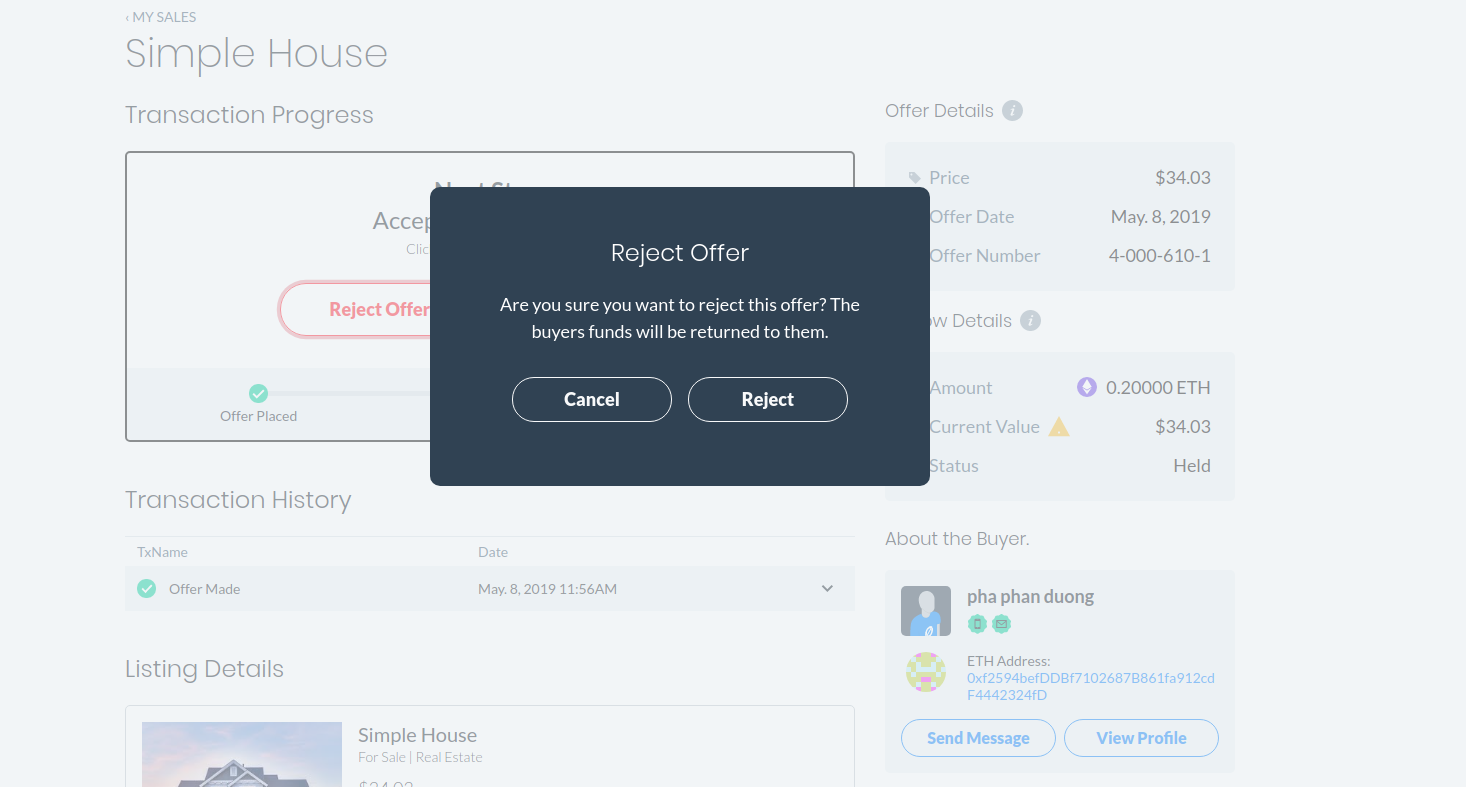
Hình 4. 7 Yêu cầu xác nhân giao dich bằng metamask



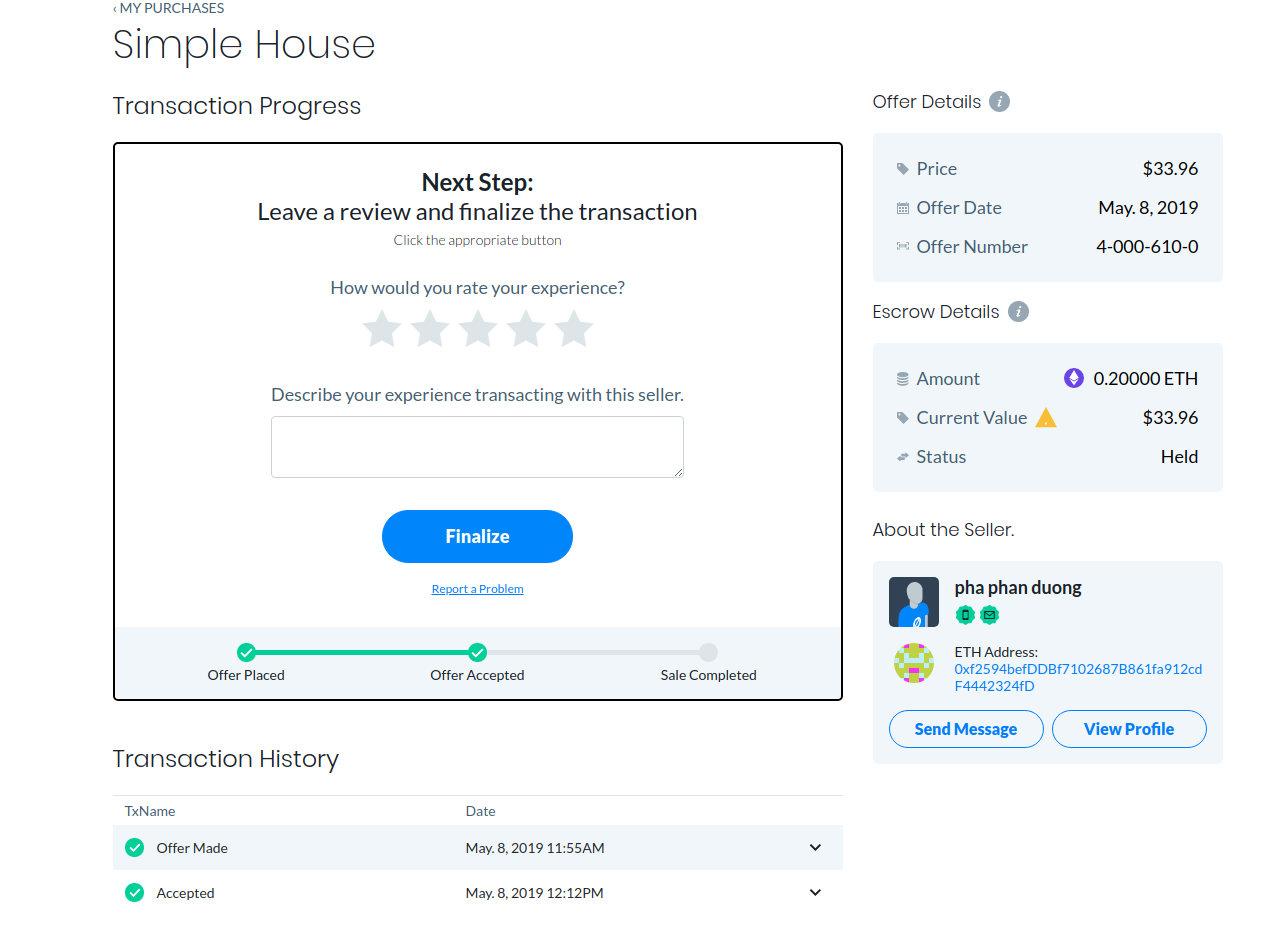
Hình 4. 8 Xác nhận đơn hàng của người bán ( từ chối )



Hình 4. 9 Màn hình yêu cầu của người mua



Hình 4. 10 Màn hình từ chối đơn hàng của người bán



Hình 4. 11 Xác nhận hoàn thành đơn hàng của người mua và nhận xét về sản phẩm

# **PHỤ LỤC 2: HIỆN THỰC CÁC CHỨC NĂNG CỦA ỨNG DỤNG TRONG SMARTCONTRACT.**

Dưới đây là một số chức năng cơ bản được hiện thực qua smartcontract.

Cấu trúc của listing trong

function \_createListing(

address \_seller,

bytes32 \_ipfsHash, // IPFS JSON with details, pricing, availability

uint \_refund

)

Cấu trúc của offer tạo bởi Buyer

function makeOffer(

uint listingID,

bytes32 \_ipfsHash, // IPFS hash containing offer data

uint \_finalizes, // Timestamp an accepted offer will finalize

address \_affiliate, // Address to send any required commission to

uint256 \_commission, // Amount of commission to send in Origin Token if offer finalizes

uint \_value, // Offer amount in ERC20 or Eth

ERC20 \_currency, // ERC20 token address or 0x0 for Eth

address \_arbitrator // Escrow arbitrator

)

Chức năng accept offer của Saler

// @dev Seller accepts offer

function acceptOffer(uint listingID, uint offerID, bytes32 \_ipfsHash) public {

Listing storage listing = listings[listingID];

Offer storage offer = offers[listingID][offerID];

require(msg.sender == listing.seller, "Seller must accept");

require(offer.status == 1, "status != created");

require(

listing.deposit >= offer.commission,

"deposit must cover commission"

);

if (offer.finalizes < 1000000000) { // Relative finalization window

offer.finalizes = now + offer.finalizes;

}

listing.deposit -= offer.commission; // Accepting an offer puts Origin Token into escrow

offer.status = 2; // Set offer to 'Accepted'

emit OfferAccepted(msg.sender, listingID, offerID, \_ipfsHash);

}

Chức nẵng cập nhật thông tin

function \_updateListing(

address \_seller,

uint listingID,

bytes32 \_ipfsHash, // Updated IPFS hash

uint \_additionalDeposit // Additional deposit to add

) private {

Listing storage listing = listings[listingID];

require(listing.seller == \_seller, "Seller must call");

if (\_additionalDeposit > 0) {

tokenAddr.transferFrom(\_seller, this, \_additionalDeposit);

listing.deposit += \_additionalDeposit;

}

emit ListingUpdated(listing.seller, listingID, \_ipfsHash);

}

Chức năng rút lại offer của Buyer

function withdrawOffer(uint listingID, uint offerID, bytes32 \_ipfsHash) public {

Listing storage listing = listings[listingID];

Offer storage offer = offers[listingID][offerID];

require(

msg.sender == offer.buyer || msg.sender == listing.seller,

"Restricted to buyer or seller"

);

require(offer.status == 1, "status != created");

refundBuyer(listingID, offerID);

emit OfferWithdrawn(msg.sender, listingID, offerID, \_ipfsHash);

delete offers[listingID][offerID];

}

Chức năng xác thực hoàn thành đơn hàng của Buyer

function finalize(uint listingID, uint offerID, bytes32 \_ipfsHash) public {

Listing storage listing = listings[listingID];

Offer storage offer = offers[listingID][offerID];

if (now <= offer.finalizes) { // Only buyer can finalize before finalization window

require(

msg.sender == offer.buyer,

"Only buyer can finalize"

);

} else { // Allow both seller and buyer to finalize if finalization window has passed

require(

msg.sender == offer.buyer || msg.sender == listing.seller,

"Seller or buyer must finalize"

);

}

require(offer.status == 2, "status != accepted");

paySeller(listingID, offerID); // Pay seller

if (msg.sender == offer.buyer) { // Only pay commission if buyer is finalizing

payCommission(listingID, offerID);

}

emit OfferFinalized(msg.sender, listingID, offerID, \_ipfsHash);

delete offers[listingID][offerID];

//test update listing.seller = listing.buyer

listings.push(Listing({

seller: offer.buyer,

deposit: listing.deposit,

depositManager: listing.depositManager,

refund: listing.refund

}));

}