

**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

# **ĐỒ ÁN TỐT NGHIỆP**

**Hệ thống theo dõi, cảnh báo các giao dịch thoái vốn  
trên blockchain Cosmos Hub**

**Phạm Thành Phong**  
phong.pt173297@sis.hust.edu.vn

**Ngành Kỹ thuật máy tính**

**Giảng viên hướng dẫn:** PGS. TS. Trương Thị Diệu Linh

\_\_\_\_\_  
Chữ kí GVHD

**Khoa:** Kỹ thuật máy tính

**Trường:** Công nghệ thông tin và Truyền thông

**HÀ NỘI, 08/2022**

# LỜI CẢM ƠN

Hôm nay là một ngày rất đặc biệt, ngày mà em chính thức kết thúc khoảng thời gian tươi đẹp trên ghế nhà trường, những kỷ niệm đẹp, một thời thanh xuân đang nhớ cùng ngôi trường Đại học Bách khoa Hà Nội yêu dấu. Trước tiên, em muốn cảm ơn người đã sinh thành và nuôi em khôn lớn, để em có cơ hội được học tập tại trường Đại học Bách Khoa Hà Nội, được tiếp cận với những kiến thức mang lại lợi ích thiết thực cho xã hội.

Tiếp theo không thể không cảm ơn những thầy cô đã giảng dạy em, những người đã giúp em hoàn thiện hơn bản thân và tự tin bước vào đời. Đặc biệt, em xin chân thành cảm ơn cô Trương Thị Diệu Linh người đã giúp đỡ em vượt qua những khó khăn trong quá trình làm đồ án tốt nghiệp. Cuối cùng, em xin chân thành cảm ơn team Aura - một đơn vị thuộc công ty FPT Software đã hỗ trợ em rất nhiều để hoàn thành đồ án tốt nghiệp này. Aura là một start-up trong lĩnh vực công nghệ blockchain dành riêng cho NFT. Sản phẩm chính của team là Aura Network - một nền tảng Layer-1 blockchain tập trung phát triển và mở rộng mảng NFT, tối đa hóa việc sử dụng NFT trên các ngành công nghiệp khác nhau. Aura Network là giao thức cho phép người dùng mint, đánh giá, theo dõi dữ liệu hay tham gia giao dịch NFT, tất cả hướng đến mục tiêu giúp xây dựng cơ sở hạ tầng vững chắc để hỗ trợ các loại tài sản giao dịch trên thị trường tài chính phi tập trung một cách đơn giản và dễ dàng hơn.

Trong quá trình làm đồ án với sự giới hạn về thời gian và trình độ của bản thân còn hạn chế nên không thể tránh khỏi những thiếu sót. Vậy nên, em rất mong sẽ nhận được những góp ý của thầy cô và tất cả các bạn sinh viên khác để đồ án của em được hoàn thiện hơn. Em xin chân thành cảm ơn !

# TÓM TẮT NỘI DUNG ĐỒ ÁN

Việt Nam là một nước đang trong quá trình phát triển mạnh mẽ. Bên cạnh tạo được công ăn việc làm cho hàng triệu người dân, Việt Nam đã trở thành một môi trường lý tưởng cho các hoạt động đầu tư tài chính và thu hút lớn nguồn vốn từ nước ngoài. Nhờ đó, người dân đã tiếp cận được những cách thức tạo ra nguồn thu nhập mới không chỉ từ công việc thường ngày mà còn đến từ hoạt động đầu tư của chính họ. Bên cạnh các hình thức đầu tư truyền thống như gửi tiết kiệm, mua vàng, đầu tư bất động sản, thì sự phát triển mạnh mẽ của thị trường chứng khoán và các quỹ đầu tư cũng là một điểm nổi bật. Nhưng từ khi công nghệ 4.0 tiếp cận đến Việt Nam thì một hình thức đầu tư mới hình thành và rất có tiềm năng để phổ biến rộng rãi sau này, đó chính là “tiền ảo”. Là một ứng dụng của công nghệ chuỗi khối blockchain, “tiền ảo” đã chứng minh được về tính kỹ thuật của nó và thu hút một lượng lớn nhà đầu tư trên thế giới, trong đó Việt Nam cũng không phải là ngoại lệ khi các nhà đầu tư ở Việt Nam rất quan tâm đến công nghệ này. Cơ hội mà “tiền ảo” đem đến cho các nhà đầu tư rất lớn nhưng rủi ro mà nó mang lại thì cũng không hề nhỏ. Trong đồ án tốt nghiệp kỹ sư này, em nghiên cứu về công nghệ chuỗi khối blockchain và ứng dụng cụ thể của nó là “tiền ảo”, bên cạnh đó em xây dựng một hệ thống hỗ trợ cho hoạt động đầu tư trở nên thuận tiện và an toàn hơn.

Đồ án tốt nghiệp của em sẽ thực hiện xây dựng một hệ thống thời gian thực để lưu trữ các giao dịch thoái vốn (undelegate) của các nhà đầu tư và cảnh báo thông tin cho người sử dụng hệ thống về hoạt động thoái vốn với lượng “tiền ảo” có khả năng ảnh hưởng đến giá trị đồng “tiền ảo” mà họ quan tâm, cụ thể trong đây là đồng ATOM của mạng lưới Cosmos Hub, để người dùng biết và xử lý, hoặc cũng có thể lưu lại để làm phân tích sau này.

Đồ án đã đề xuất một hệ thống phù hợp, ổn định để tương tác với các mạng lưới blockchain và một công cụ hỗ trợ thực tế cho các nhà đầu tư. Và sau cùng, đồ án đã cài đặt thành công hệ thống và đang trong quá trình thử nghiệm thực tế.

## MỤC LỤC

<b>CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI.....</b>	<b>1</b>
1.1 Đặt vấn đề.....	1
1.2 Mục tiêu và phạm vi đề tài.....	1
1.3 Định hướng giải pháp.....	2
1.4 Bố cục đồ án .....	2
<b>CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU.....</b>	<b>3</b>
2.1 Khảo sát hiện trạng .....	3
2.2 Tổng quan chức năng .....	3
2.2.1 Biểu đồ use case tổng quát .....	3
2.2.2 Quy trình nghiệp vụ .....	4
2.3 Đặc tả chức năng .....	5
2.4 Yêu cầu phi chức năng .....	8
<b>CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG.....</b>	<b>9</b>
3.1 Mật mã học .....	9
3.1.1 Mật mã bất đối xứng .....	10
3.1.2 Hàm băm (Hash).....	10
3.1.3 Chữ ký số .....	11
3.2 Blockchain .....	11
3.3 Crypto .....	12
3.4 Thuật toán đồng thuận.....	12
3.4.1 Định nghĩa .....	12
3.4.2 Proof of Stake (PoS).....	12
3.4.3 Cosmos Hub.....	13

3.5 Công nghệ sử dụng .....	16
3.5.1 VueJS .....	16
3.5.2 Golang và Gin framework .....	17
3.5.3 MongoDB .....	17
3.5.4 Redis .....	18
<b>CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ .....</b>	<b>19</b>
4.1 Thiết kế kiến trúc.....	19
4.1.1 Lựa chọn kiến trúc phần mềm .....	19
4.1.2 Thiết kế API .....	19
4.1.3 Thiết kế giao diện .....	20
4.1.4 Thiết kế cơ sở dữ liệu .....	22
4.2 Xây dựng ứng dụng.....	23
4.2.1 Thư viện và công cụ sử dụng .....	23
4.2.2 Kết quả đạt được .....	23
4.2.3 Minh họa các chức năng chính .....	24
4.3 Kiểm thử.....	27
4.3.1 Kiểm thử chức năng hiển thị các sự kiện.....	27
4.3.2 Kiểm thử chức năng thông báo của discord.....	27
4.4 Triển khai .....	28
4.4.1 Chuẩn bị trước .....	28
4.4.2 Khởi chạy hệ thống .....	29
<b>CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT.....</b>	<b>33</b>
5.1 Về phần hệ thống .....	33
5.1.1 Cụm on-chain .....	33
5.1.2 Cụm tương tác .....	34
5.1.3 Cụm service .....	35

5.2 Về công nghệ .....	36
5.3 Về ứng dụng thực tiễn .....	36
<b>CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN .....</b>	<b>38</b>
6.1 Kết luận.....	38
6.1.1 So sánh với các hệ thống sẵn có.....	38
6.1.2 Đánh giá ưu nhược điểm và khả năng ứng dụng .....	38
6.2 Hướng phát triển.....	38



## DANH MỤC HÌNH VẼ

Hình 2.1	Biểu đồ usecase . . . . .	3
Hình 2.2	Biểu đồ nghiệp vụ . . . . .	5
Hình 4.1	Sơ đồ kiến trúc hệ thống . . . . .	19
Hình 4.2	Thiết kế mockup cho giao diện web . . . . .	20
Hình 4.3	Thiết kế mockup cho giao diện thông báo của discord . . . . .	21
Hình 4.4	Dữ liệu ngưỡng cảnh báo lưu trong cơ sở dữ liệu . . . . .	22
Hình 4.5	Dữ liệu sự kiện lưu trong cơ sở dữ liệu . . . . .	22
Hình 4.6	Giao diện trang "All validator" . . . . .	24
Hình 4.7	Kết quả khi mở rộng tìm kiếm . . . . .	24
Hình 4.8	Giao diện trang "By Delegator" . . . . .	24
Hình 4.9	Kết quả khi mở rộng tìm kiếm . . . . .	25
Hình 4.10	Kết quả với lệnh help . . . . .	25
Hình 4.11	Kết quả với lệnh config . . . . .	25
Hình 4.12	Kết quả với lệnh info . . . . .	26
Hình 4.13	Discord bot cảnh báo mức low . . . . .	26
Hình 4.14	Discord bot cảnh báo mức medium . . . . .	26
Hình 4.15	Discord bot cảnh báo mức high . . . . .	27
Hình 4.16	Discord bot cảnh báo mức warning . . . . .	27
Hình 4.17	Chuẩn bị ví . . . . .	28
Hình 4.18	Chuẩn bị token trong ví . . . . .	29
Hình 4.19	Chuẩn bị một lượng token để stake . . . . .	29
Hình 4.20	Kết quả khi live server khởi động . . . . .	29
Hình 4.21	Kết quả khi forwarder khởi động . . . . .	30
Hình 4.22	Forwarder cố gắng kết nối lại khi bị mất kết nối đến các live server . . . . .	30
Hình 4.23	Forwarder thông báo khi không có live server nào hoạt động . . . . .	30
Hình 4.24	Kết quả sau khi khởi động mongo . . . . .	31
Hình 4.25	Kết quả sau khi khởi động redis . . . . .	31
Hình 4.26	Kết quả sau khi khởi động data service . . . . .	32
Hình 4.27	Kết quả khi khởi động web client . . . . .	32
Hình 4.28	Kết quả khi khởi động discord bot . . . . .	32
Hình 5.1	Sơ đồ mạng lưới blockchain . . . . .	33
Hình 5.2	Sơ đồ thiết kế cụm tương tác . . . . .	34
Hình 5.3	Sơ đồ thiết kế cụm dịch vụ . . . . .	35



## DANH MỤC BẢNG BIỂU

Bảng 2.1	Đặc tả use case theo dõi sự kiện thông qua discord. . . . .	6
Bảng 2.2	Đặc tả use case hiển thị lịch sử các sự kiện . . . . .	7
Bảng 2.3	Đặc tả use case hiển thị các sự kiện theo delegator . . . . .	8
Bảng 4.1	Các cấu hình chung của giao diện trang web . . . . .	20
Bảng 4.2	Các cấu hình chung của giao diện thông báo discord . . . . .	21
Bảng 4.3	Danh sách công cụ và thư viện sử dụng. . . . .	23
Bảng 4.4	Kiểm thử chức năng hiển thị toàn bộ các sự kiện. . . . .	27
Bảng 4.5	Kiểm thử chức năng hiển thị toàn bộ các sự kiện. . . . .	28

## DANH MỤC THUẬT NGỮ VÀ TỪ VIẾT TẮT

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

# CHƯƠNG 1. GIỚI THIỆU ĐỀ TÀI

## 1.1 Đặt vấn đề

Hệ thống tài chính toàn cầu của chúng ta luân chuyển hàng nghìn tỷ đô la mỗi ngày và phục vụ hàng tỷ người. Nhưng hệ thống đầy rẫy các vấn đề, tăng thêm chi phí thông qua phí và sự chậm trễ, tạo ra xung đột thông qua thủ tục giấy tờ thừa thãi và rắc rối, đồng thời mở ra cơ hội cho gian lận và tội phạm. Nói một cách cụ thể, 45% trong số các trung gian tài chính, chẳng hạn như mạng thanh toán, sàn giao dịch chứng khoán và dịch vụ chuyển tiền, đối mặt với các tội phạm kinh tế hàng năm; con số là 37% cho toàn bộ nền kinh tế, và chỉ 20% và 27% đối với các lĩnh vực dịch vụ chuyên nghiệp và công nghệ, tương ứng. Không có gì ngạc nhiên khi chi phí quản lý tiếp tục tăng và vẫn là mối quan tâm hàng đầu của các chủ ngân hàng. Tất cả điều này làm tăng thêm chi phí, và người dùng cuối cùng phải chịu gánh nặng.

Rồi để khắc phục những vấn đề bất cập đó, blockchain đã ra đời và đem đến nhiều ứng dụng hơn thế nữa.

Trong những năm gần đây, cùng với sự phát triển của khoa học công nghệ, một ứng dụng nổi bật của blockchain- “tiền ảo”, “tài sản ảo” là những vấn đề “nóng” thu hút sự quan tâm của dư luận. Tại Việt Nam, tuy “tiền ảo” không được chấp nhận, không phải là phương tiện thanh toán hợp pháp nhưng hoạt động của nó vẫn có chiều hướng gia tăng bất chấp rủi ro. Được đánh giá là có nhiều ưu điểm trong giao dịch, các đồng tiền ảo thu hút sự quan tâm cao của nhà đầu tư (NĐT) và đã có những tác động nhất định đến thị trường tài chính, tiền tệ thế giới. Không phủ nhận những ưu điểm nhưng tiền ảo vẫn tiềm ẩn nhiều rủi ro cho NĐT, thậm chí cho cả thị trường tiền tệ. . . Làm thế nào để quản lý hiệu quả và hạn chế được những rủi ro từ đồng tiền ảo trong bối cảnh cuộc Cách mạng công nghệ 4.0 là vấn đề đặt ra cho các nhà quản lý và nhà đầu tư

## 1.2 Mục tiêu và phạm vi đề tài

Trước vấn đề bất cập đã nêu ở trên, em thấy việc cần thiết của một hệ thống hỗ trợ cho các nhà đầu tư theo dõi và phân tích để giảm thiểu rủi ro trong quá trình đầu tư tiền ảo của họ.

Hệ thống không chỉ phục vụ trên nền tảng web mà còn phục vụ cộng đồng trên Discord - nền tảng trao đổi phổ biến cho những người dùng tham gia nói riêng và toàn bộ hệ sinh thái blockchain nói chung bằng cách cảnh báo đến với người dùng Discord khi các giao dịch đó được họ quan tâm và đăng kí lại với hệ thống. Và đó

chính là điểm riêng biệt của hệ thống so với các hệ thống đang có sẵn chỉ phục vụ cho cộng đồng trên nền tảng web khiến cho việc cảnh báo thời gian thực đối với người dùng là không khả thi.

### 1.3 Định hướng giải pháp

Đề án có 2 đóng góp chính như sau:

- Đề án đề xuất một hệ thống ổn định tương cho tương tác với blockchain
- Đề án cung cấp một giải pháp đối với người dùng, những nhà đầu tư để nắm được những hoạt động tiềm năng ảnh hưởng đến lợi ích của họ

Trong quá trình xây dựng hệ thống, em đã giải quyết các vấn đề chính là:

- Để hệ thống hoạt động ổn định và bền bỉ, em đã thiết kế cụm các máy có cùng chức năng và sẵn sàng hoạt động khi máy chính bị trục trặc
- Để đảm bảo tính thời gian thực của hệ thống, em đã sử dụng message broker là Redis để điều phối dữ liệu đến Discord Bot cũng như Web client.
- Để phục vụ lưu trữ lâu dài và cung cấp công cụ cho quá trình phân tích sau này khi hệ thống được phát triển, em sử dụng MongoDB vì tính tiện lợi và phù hợp của nó.

### 1.4 Bố cục đề án

Các phần còn lại của báo cáo đề án tốt nghiệp sẽ có nội dung như sau.

Trong chương 2, em sẽ đưa ra các khảo sát và phân tích yêu cầu, nêu ra các chức năng tổng quan và phân rã nhỏ một số chức năng của ứng dụng.

Trong chương 3, em sẽ nêu ra các công nghệ chính để xây dựng hệ thống. Frontend sử dụng VueJS. Phía backend sử dụng Gin framework, thư viện Gorilla, ngôn ngữ lập trình Golang. Discord bot được viết bằng Python với thư viện discord.py. Cơ sở dữ liệu sử dụng MongoDB và message broker là Redis

Chương 4 sẽ trình bày về thiết kế kiến trúc của hệ thống, thiết kế cơ sở dữ liệu, các quá trình xây dựng, kiểm thử, triển khai hệ thống.

Chương 5 sẽ nêu lên những giải pháp đóng góp nổi bật mà hệ thống cung cấp.

Cuối cùng sẽ là chương 6, em sẽ đưa ra những ưu, nhược điểm và định hướng phát triển cho tương lai.

## CHƯƠNG 2. KHẢO SÁT VÀ PHÂN TÍCH YÊU CẦU

### 2.1 Khảo sát hiện trạng

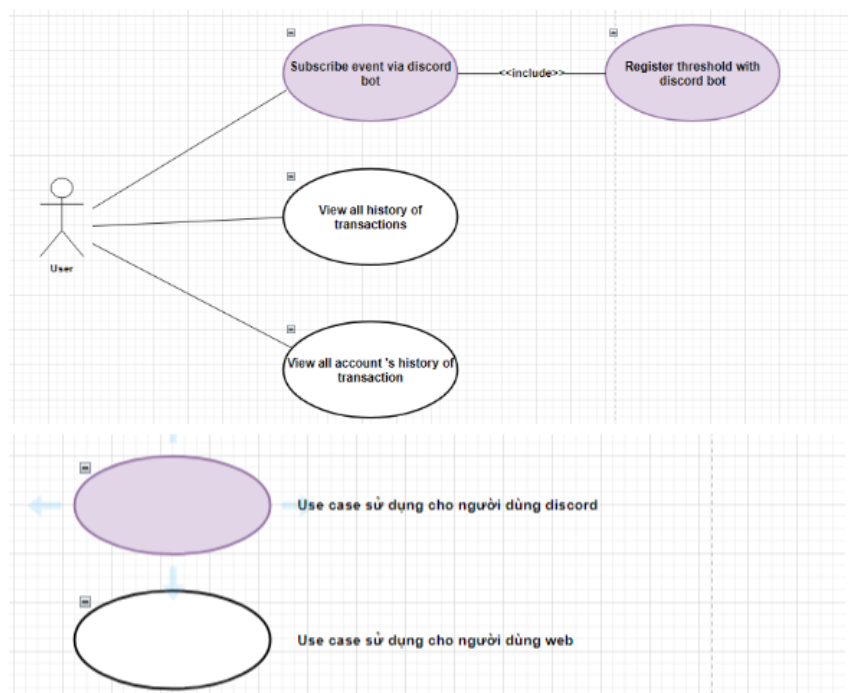
Hiện nay, trên thị trường, có rất nhiều doanh nghiệp và start-up làm về lĩnh vực blockchain. Đã có các web/app để người dùng có thể quan sát và truy vấn các giao dịch trên từng mạng lưới nhất định như:

- Etherscan: Công cụ truy vấn và phân tích dành riêng cho mạng lưới Ethereum
- MintScan: Công cụ truy vấn và phân tích dành riêng cho các mạng lưới xây dựng trên nền tảng Cosmos
- Flipsidecrypto: Trung tâm lưu trữ dữ liệu trên các mạng lưới như: Ethereum, Solana, Omosis,...

Tuy nhiên, theo như em tìm hiểu thì các giải pháp đó thường không chuyên biệt, không hướng đến một đối tượng người dùng cụ thể, một giải pháp để giải quyết một vấn đề cụ thể. Những công cụ đó thường lưu trữ tất cả các giao dịch và chỉ với mục đích duy nhất là cho phép người dùng truy vấn, tìm kiếm và đôi khi là chỉ biết được đầu ra, đầu vào của dòng tiền nhưng lại không biết được ý nghĩa của dòng tiền đó.

### 2.2 Tổng quan chức năng

#### 2.2.1 Biểu đồ use case tổng quát



Hình 2.1: Biểu đồ usecase

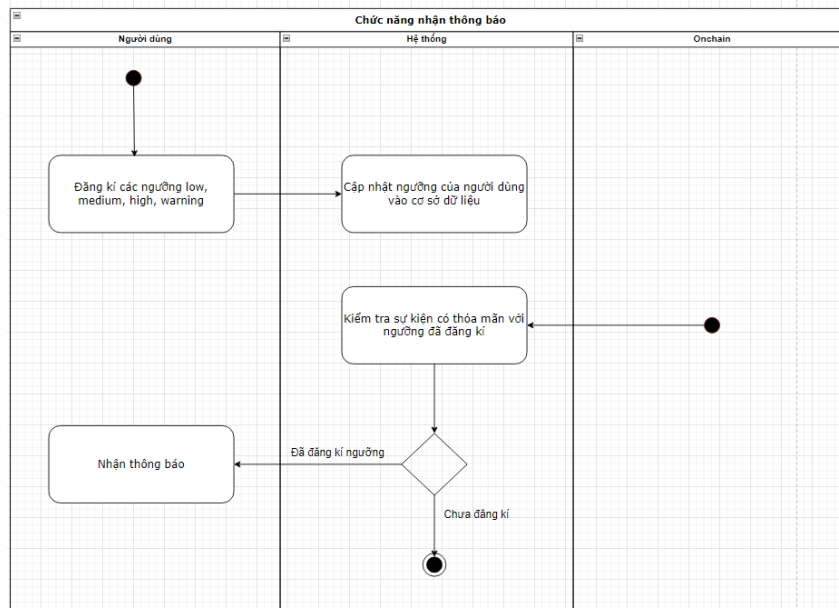
Trong hệ thống của em, User là người sử dụng hệ thống, bao gồm cả những người dùng của cộng đồng Discord và trên nền tảng web. Những người dùng ở nền tảng Discord được Discord Bot gửi thông báo trực tiếp ngay lập tức khi sự kiện đăng kí xảy ra. Trong khi đó, người dùng trên nền tảng web với mục đích chủ yếu để quan sát lịch sử các sự kiện và tìm kiếm.

Người dùng trên Discord ban đầu đăng kí ngưỡng thông báo với bot trong nhóm chung, khi bot bắt được sự kiện trên chain, bot sẽ dựa trên ngưỡng đó để gửi thông báo đến cho các user. Mỗi user có thể thiết lập ngưỡng với 4 mức độ khác nhau tăng dần: Low -> Medium -> High -> Warning.

Người dùng trên nền tảng web sau khi truy nhập vào địa chỉ chính thì sẽ thấy 2 tab chính. Tab thứ nhất là Home, tại đây, người dùng có thể thấy lịch sử toàn bộ các sự kiện theo thứ tự thời gian, có chức năng “View more” để mở rộng danh sách hiển thị, và các sự kiện mới đến sẽ ngay lập tức được hiển thị trực tiếp trên giao diện. Tab thứ 2 là Delegator, tab này có thanh Search để người dùng điền địa chỉ delegator, tab này cho phép người dùng tìm kiếm các giao dịch với delegator là địa chỉ đã được điền trong thanh Search.

### **2.2.2 Quy trình nghiệp vụ**

Đầu tiên, người dùng sẽ đăng kí các ngưỡng cảnh báo low, medium, high, warning với hệ thống. Khi hệ thống nhận được sự kiện, hệ thống truy xuất cơ sở dữ liệu để thông báo cho người dùng đã đăng kí ngưỡng cảnh báo với sự kiện đó. Nếu không, sự kiện sẽ bị bỏ qua.



**Hình 2.2:** Biểu đồ nghiệp vụ

## 2.3 Đặc tả chức năng

### a, Subscribe event via Discord Bot

Từ phía người dùng trên nền tảng Discord, một nền tảng phổ biến dành cho cộng đồng blockchain, có nhu cầu được nhận thông báo ngay lập tức khi sự kiện trên chain được bắt.

**Bảng 2.1:** Đặc tả use case theo dõi sự kiện thông qua discord.

<b>Mã use case</b>	UC1	<b>Tên use case</b>	Subscribe event via discord bot
<b>Người dùng</b>	Người dùng		
<b>Tiền điều kiện</b>	Người dùng đã đăng nhập vào discord, join group và đăng kí ngưỡng với bot		
<b>Luồng hoạt động chính</b>	<b>Thứ tự thao tác</b>	<b>Được thực hiện bởi</b>	<b>Hành động</b>
	1.	Người dùng	Yêu cầu bot hỗ trợ liệt kê các lệnh trong hệ thống bằng lệnh !help
	2.	Bot	Trả lời yêu cầu của người dùng bằng danh sách các lệnh có sẵn
	3.	Người dùng	Người dùng đăng kí ngưỡng bằng lệnh !config <low> <high> <medium> <warning>
	4.	Bot	Tiếp nhận yêu cầu đăng kí của người dùng để ghi nhận vào cơ sở dữ liệu của hệ thống
<b>Sự kiện thay thế</b>	<b>Thứ tự thao tác</b>	<b>Được thực hiện bởi</b>	<b>Hành động</b>
	4.a	Bot	Báo lỗi nếu câu lệnh của người dùng không thỏa mãn truyền các tham số hợp lý
<b>Hậu điều kiện</b>	Khi một sự kiện mà bot tiếp nhận được, nếu lượng tiền nằm trong đoạn low-medium thì bot sẽ gửi thông báo mức low đến người dùng, nếu trong khoảng medium-high thì bot sẽ gửi thông báo mức medium đến người dùng, nếu từ mức high đến warning thì sẽ gửi thông báo mức high và nếu trên mức warning thì sẽ thông báo mức warning		

**b, View all history of events**

Là một người dùng bình thường, em có nhu cầu muốn quan sát lịch sử các sự kiện một cách đơn giản hơn khi không cần tham gia vào Discord



**Bảng 2.2:** Đặc tả use case hiển thị lịch sử các sự kiện

<b>Mã use case</b>	UC2	<b>Tên use case</b>	View all history of events
<b>Người dùng</b>	Người dùng		
<b>Tiền điều kiện</b>	Không		
<b>Luồng hoạt động chính</b>	<b>Thứ tự thao tác</b>	<b>Được thực hiện bởi</b>	<b>Hành động</b>
	1.	Người dùng	Truy nhập vào đường dẫn của trang web
	2.	Người dùng	Lựa chọn “Home” trên thanh navbar
	3.	Hệ thống	Trả ra top 20 sự kiện undelegate gần nhất
	4.	Người dùng	Chọn nút “View more”
	5.	Hệ thống	Trả ra tiếp tối đa 20 bản ghi nữa
<b>Sự kiện thay thế</b>	6.	Hệ thống	Liên tục duy trì kết nối để tiếp nhận các sự kiện theo thời gian thức
	<b>Thứ tự thao tác</b>	<b>Được thực hiện bởi</b>	<b>Hành động</b>
	6.a	Bot	Reconnect khi hệ thống máy kết nối với Redis khi bị mất kết nối
<b>Hậu điều kiện</b>	Dữ liệu lịch sử các sự kiện sẽ được hiển thị trên giao diện web Người dùng có thể mở rộng danh sách hiển thị Dữ liệu sẽ liên tục được cập nhật		

**c, View all account’s history of events**

Là một người dùng bình thường, em có nhu cầu muốn quan sát lịch sử các sự kiện phát sinh từ một ví địa chỉ cố định một cách đơn giản hơn khi không cần tham gia vào Discord.

**Bảng 2.3:** Đặc tả use case hiển thị các sự kiện theo delegator

<b>Mã use case</b>	UC2	<b>Tên use case</b>	View all account's history of events
<b>Người dùng</b>	Người dùng		
<b>Tiền điều kiện</b>	Không		
<b>Luồng hoạt động chính</b>	<b>Thứ tự thao tác</b>	<b>Được thực hiện bởi</b>	<b>Hành động</b>
	1.	Người dùng	Truy nhập vào đường dẫn của trang web
	2.	Người dùng	Lựa chọn “Delegator” trên thanh navbar
	3.	Người dùng	Trên thanh search, người dùng điền địa chỉ của delegator muốn tìm kiếm
	4.	Hệ thống	Trả ra top 20 sự kiện mà địa chỉ trên đã undelegate
	5.	Người dùng	Chọn nút “View more”
	6.	Hệ thống	Trả ra tiếp tối đa 20 bản ghi nữa
	7.	Hệ thống	Liên tục duy trì kết nối để tiếp nhận các sự kiện theo thời gian thực
<b>Sự kiện thay thế</b>	<b>Thứ tự thao tác</b>	<b>Được thực hiện bởi</b>	<b>Hành động</b>
	7.a	Bot	Reconnect khi hệ thống mất kết nối với Redis
<b>Hậu điều kiện</b>	Dữ liệu lịch sử các sự kiện của một địa chỉ sẽ được hiển thị trên giao diện web Người dùng có thể mở rộng danh sách hiển thị Dữ liệu sẽ liên tục được cập nhật		

## 2.4 Yêu cầu phi chức năng

Các yêu cầu cần có của hệ thống:

- Hệ thống phải đảm bảo tính sẵn sàng, chống chịu lỗi
- Hệ thống phải đảm bảo tính chính xác, thời gian thực đến người dùng discord
- Hệ thống phải đảm bảo dữ liệu được lưu đầy đủ, hiển thị đến người dùng chính xác

## CHƯƠNG 3. CÔNG NGHỆ SỬ DỤNG

Trong chương này, em sẽ trình bày về các lý thuyết cần có cũng như các công nghệ sẽ được sử dụng trong đồ án:

### 3.1 Mật mã học

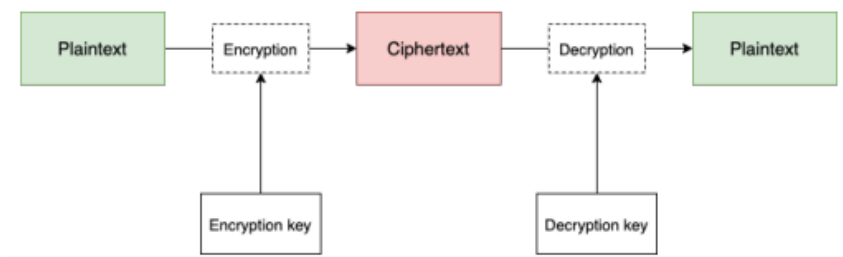
Theo định nghĩa, mật mã là ngành nghiên cứu các kỹ thuật toán học để cung cấp dịch vụ bảo mật thông tin. Trong thế giới ngày nay, mật mã thường đề cập đến quá trình gửi thông tin từ người này sang người khác một cách bảo mật. Nó có bốn tính năng quan trọng:

- Confidentiality: Chỉ những người được ủy quyền mới có thể truy cập được dữ liệu.
- Integrity: Thông tin trong quá trình chuyển từ người gửi đến người nhận không thể bị sửa đổi nếu không bị phát hiện
- Non-repudiation: Người gửi thông tin không thể phủ nhận hoạt động của mình.
- Authentication: Danh tính của người gửi và người nhận phải được xác nhận với nhau. Ngoài ra, điểm đến và nguồn gốc của thông tin phải rõ ràng.

Một số thuật ngữ quan trọng trong mật mã là:

- Bản rõ: Phần thông tin chưa được mã hóa.
- Bản mã: Đoạn thông tin được biến đổi hoặc mã hóa.
- Mã hóa: Quá trình mã hóa bản rõ thành bản mã tại phía người gửi.
- Giải mã: Quá trình chuyển đổi bản mã thành bản rõ ở phía người nhận.
- Khóa: Một tham số xác định kết quả đầu ra của thuật toán mật mã.

Các thuật ngữ và quy trình liên quan đến giao tiếp an toàn giữa hai bên có thể được tóm tắt trong hình dưới đây:



Có hai loại mật mã được phân loại theo cách sử dụng khóa:

- Mật mã đối xứng: Một khóa chung được sử dụng cho cả hai giai đoạn mã hóa và giải mã.

- Mật mã không đối xứng: Một cặp khóa (khóa công khai - khóa riêng) được sử dụng để mã hóa và giải mã thông tin.

### 3.1.1 Mật mã bất đối xứng

Trong mật mã không đối xứng, mỗi người dùng tạo một cặp khóa công khai và khóa riêng (khóa bí mật). Khóa công khai được sử dụng để mã hóa và khóa riêng là để giải mã.

$$D_{sk}(E_{pk}(x)) = x \quad (3.1)$$

Như cách đặt tên, khóa công khai phải được công khai cho mọi người trong hệ thống. Biết khóa công khai, nhưng không thể biết được khóa riêng. Giao tiếp giữa người gửi (được gọi là Alice) và người nhận (Bob) xảy ra trong các bước sau: Đầu tiên, Alice và Bob phải đồng ý về thuật toán để tạo cặp khóa của họ. Alice có khóa công khai  $pk_a$  và khóa bí mật  $sk_a$ . Tương tự, Bob cũng có cặp khóa công khai và bí mật dưới dạng  $(pk_b, sk_b)$ . Alice và Bob sau đó trao đổi khóa công khai với nhau

Thứ hai, Alice sử dụng khóa công khai của Bob  $pk_b$  để mã hóa tin nhắn  $m$ , thu được bản mã  $C$  và gửi bản mã  $C$  tới Bob

$$C = E_{pk_b}(m) \quad (3.2)$$

Từ phía người nhận, Bob giải mã bản mã bằng cách sử dụng khóa bí mật  $sk_b$  để thu lại bản rõ

$$D_{sk_b}(C) = m \quad (3.3)$$

### 3.1.2 Hàm băm (Hash)

Một hàm băm  $H$  ánh xạ một tin nhắn có độ dài tùy ý  $n$  bit tới một dấu vân tay của độ dài cố định  $m$  bit. Giá trị băm còn được gọi là bản tóm tắt của giá trị gốc thông điệp.

Các tính năng chính của hàm băm bao gồm:

- Thông báo có độ dài cố định: Hàm băm luôn tạo ra đầu ra có cùng kích thước bất kể kích thước của đầu vào
- Thuộc tính một chiều: Dễ dàng tính toán nội dung băm từ thông điệp gốc nhưng hầu như không thể tính toán thông điệp chỉ khi biết giá trị băm
- Khả năng chống va chạm: Khó có thể tìm được hai bản tin  $m_1, m_2$  sao cho:

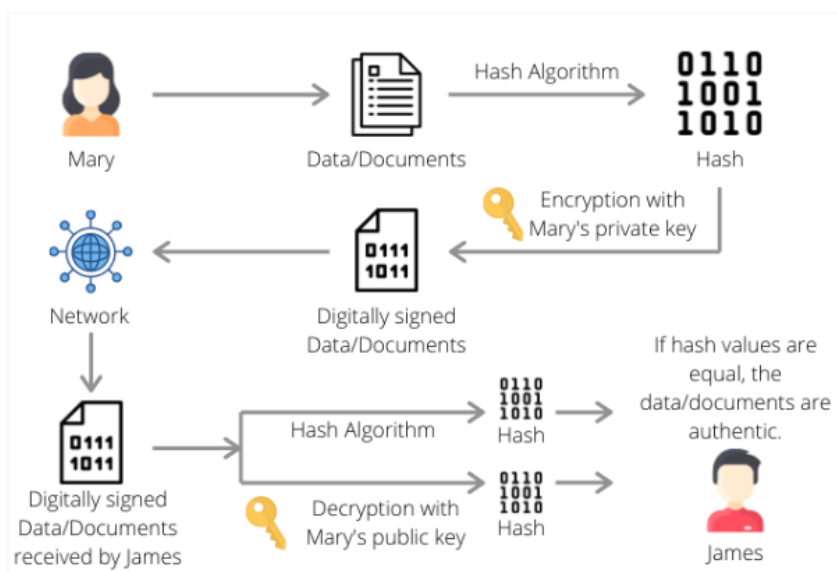
$$H(m_1) = H(m_2) \quad (3.4)$$

### 3.1.3 Chữ ký số

Lý do chữ ký điện tử ra đời là nhu cầu mang chữ ký tay của con người trong thế giới thực vào thế giới kỹ thuật số. Các hợp đồng giấy tờ cần phải được ký kết để được hợp lệ, và các phiên bản điện tử cũng vậy. Chữ ký điện tử phải chứng minh được:

- Tính toàn vẹn của dữ liệu: Thông tin là nguyên bản và không bị sửa đổi.
- Xác thực: Nguồn thông tin chính xác và chưa bị mạo danh bởi bất kỳ bên nào khác.
- Không thoái thác: Người ký không thể phủ nhận chữ ký của mình trên thông tin.

Sơ đồ chữ ký điện tử có hai bước cơ bản. Thuật toán ký có một tin nhắn và sử dụng khóa ký để xuất ra chữ ký trên tin nhắn. Các thuật toán xác minh sử dụng khóa xác minh, tin nhắn gốc và chữ ký để chứng minh tính hợp lệ của thông điệp:



## 3.2 Blockchain

Blockchain là một cơ sở dữ liệu phân cấp lưu trữ thông tin trong các khối (block) được liên kết với nhau bằng mã hóa và mở rộng theo thời gian để tạo thành một chuỗi (chain). Mỗi khối trong Blockchain sẽ được liên kết với khối trước đó, chứa thông tin về thời gian khởi tạo khối đó kèm một mã thời gian và dữ liệu giao dịch.

Hiểu đơn giản, Blockchain có thể được xem là một cuốn sổ cái điện tử được phân phối trên nhiều máy tính khác nhau, lưu trữ mọi thông tin giao dịch và đảm bảo các thông tin đó không thể bị thay đổi dưới bất kỳ hình thức nào. Mọi thông tin được lưu trên cuốn sổ cái đó sẽ được xác nhận bởi hàng loạt máy tính được kết nối trong một mạng lưới chung. Sẽ không một cỗ máy nào có khả năng thay đổi, viết đè lên hay xóa dữ liệu trong cuốn sổ cái đó.

### 3.3 Crypto

Crypto (hay Cryptocurrency) là dạng tiền điện tử do các dự án trên blockchain ban hành, được sử dụng như một phương tiện giao dịch trên các blockchain như thưởng cho miner, mua vốn đầu tư dự án,... Crypto sử dụng thuật toán mã hóa để đảm bảo mật thông tin giao dịch dưới dạng kỹ thuật số và kiểm soát việc tạo ra các đơn vị mới thông qua công nghệ blockchain.

### 3.4 Thuật toán đồng thuận

#### 3.4.1 Định nghĩa

Với tiền mã hóa, sổ dư của người dùng được ghi lại trong cơ sở dữ liệu – tức blockchain. Mọi người (hay chính xác hơn là mọi node) phải duy trì một bản sao cơ sở dữ liệu giống hệt nhau. Nếu không, thông tin sẽ bị xung đột. Từ đó, mạng lưới tiền mã hóa sẽ bị phá vỡ.

Mật mã khóa công khai đảm bảo rằng người dùng không thể tiêu tiền của nhau. Nhưng vẫn cần phải có một nguồn xác thực duy nhất để những người tham gia mạng lưới có thể xác định xem tiền đã được chi chưa.

Điểm chung tiên quyết là khi người dùng (trình xác thực) muốn thêm khối, họ phải stake một giá trị gì đó. Việc stake một giá trị khiến cho trình xác thực có xu hướng hành động trung thực. Nếu họ gian lận, họ sẽ mất những thứ họ đã stake. Những thứ có thể stake bao gồm sức mạnh tính toán, tiền mã hóa hoặc thậm chí danh tiếng.

Tại sao người dùng lại muốn mạo hiểm nguồn lực của chính họ? Bởi vì việc này mang đến cho họ cơ hội nhận phần thưởng. Phần thưởng thường là tiền mã hóa gốc của giao thức và được tạo thành từ các khoản phí do người dùng khác trả, các đơn vị tiền mã hóa mới được tạo hoặc cả hai.

Điều cuối cùng chúng ta cần là sự minh bạch. Chúng ta cần khả năng phát hiện khi ai đó đang gian lận. Lý tưởng nhất là họ phải tốn kém chi phí để sản xuất các khối, nhưng lại rất rẻ để bất kỳ ai muốn xác thực chúng. Điều này đảm bảo rằng các trình xác thực được người dùng thường xuyên kiểm tra.

#### 3.4.2 Proof of Stake (PoS)

Trong những ngày đầu của Bitcoin xuất hiện, Proof of Stake (PoS) đã được đề xuất như một giải pháp thay thế cho Proof of Work. Trong hệ thống PoS, không có khái niệm về công cụ đào, phần cứng chuyên dụng hoặc mức tiêu thụ năng lượng lớn. Tất cả những gì bạn cần là một chiếc PC thông thường.

Thực tế thì không phải tất cả. Bạn vẫn cần đầu tư nhiều thứ hơn. Trong PoS, bạn không chỉ trả quá nhiều tài nguyên bên ngoài (như điện hoặc phần cứng), mà

là tài nguyên từ bên trong – tiền mã hóa. Các quy tắc khác nhau với mọi giao thức, nhưng nhìn chung bạn cần có một số tiền tối thiểu để đủ điều kiện stake.

Khi đó, bạn cần khóa tiền của mình trong một chiếc ví (bạn không thể di chuyển tiền của mình khi đang stake). Thông thường, bạn sẽ đồng ý với các trình xác thực khác về những giao dịch nào sẽ đi vào khối tiếp theo. Theo một nghĩa nào đó, bạn đang stake vào một khối có thể được chọn và giao thức sẽ chọn một khối.

Nếu khối của bạn được chọn, bạn sẽ nhận được một tỷ lệ phí giao dịch, tùy thuộc vào số tiền bạn stake. Bạn càng khóa nhiều tiền, bạn càng có thể kiếm được nhiều tiền hơn. Nhưng nếu bạn cố gắng gian lận bằng cách đề xuất các giao dịch không hợp lệ, bạn sẽ mất một phần (hoặc tất cả) phần tài sản mình đã stake. Do đó, chúng ta có một cơ chế tương tự như PoW - hành động trung thực sẽ có lợi hơn hành động không trung thực.

Nhìn chung, không có tiền mới được tạo ra để làm phần thưởng dành cho các trình xác thực. Do đó, đồng tiền gốc của blockchain phải được phát hành theo một số cách khác. Điều này có thể được thực hiện thông qua phân phối ban đầu (tức là ICO hoặc IEO) hoặc bằng cách khởi chạy giao thức với cơ chế PoW trước khi chuyển đổi sang PoS.

Cho đến nay, Proof of Stake thuần túy chỉ thực sự được triển khai với các loại tiền mã hóa nhỏ. Do đó, không rõ liệu nó có thể phục vụ như một giải pháp thay thế khả thi cho PoW hay không. Về mặt lý thuyết, nó được đánh giá là hiệu quả nhưng thực tế có thể sẽ rất khác.

Một khi PoS được triển khai trên một mạng lưới với một lượng lớn giá trị, hệ thống sẽ trở thành một sân chơi của lý thuyết trò chơi và các khuyến khích tài chính. Bất kỳ ai có bí quyết “hack” hệ thống PoS chỉ muốn làm như vậy nếu họ có thể thu được lợi ích từ nó – do đó, cách duy nhất để tìm hiểu xem nó có khả thi hay không là trực tiếp sử dụng mạng.

Chúng ta sẽ sớm thấy PoS được thử nghiệm trên quy mô lớn – Casper sẽ được triển khai như một phần của loạt các nâng cấp mạng Ethereum (được gọi chung là Ethereum 2.0).

### **3.4.3 Cosmos Hub**

#### **a, Tendermint**

TenderMint là phần mềm để sao lưu một cách an toàn và nhất quán một ứng dụng trên nhiều máy. Về tính an toàn có nghĩa là Tendermint hoạt động ngay cả khi có tới 1/3 máy bị lỗi theo những cách tùy ý. Về tính nhất quán có nghĩa là mọi máy không bị lỗi đều nhìn thấy cùng một nhật ký giao dịch và tính toán cùng một trạng

thái. Sao lưu an toàn và nhất quán là một vấn đề cơ bản trong các hệ thống phân tán; Nó đóng một vai trò quan trọng trong việc dung nạp lỗi của một loạt các ứng dụng, từ tiền tệ, đến bầu cử, đến dàn nhạc cơ sở hạ tầng, và hơn thế nữa. Khả năng chống chịu lỗi theo những cách tùy ý, bao gồm trở thành kẻ phá hoại, được gọi là chống chịu lỗi Byzantine (BFT).

TenderMint bao gồm hai thành phần kỹ thuật chính: blockchain engine và giao diện ứng dụng chung. Công cụ đồng thuận, được gọi là Core TenderMint, đảm bảo rằng các giao dịch tương tự được ghi lại trên mọi máy theo cùng một thứ tự. Giao diện ứng dụng, được gọi là giao diện blockchain ứng dụng (ABCI), cho phép các giao dịch được xử lý trong bất kỳ ngôn ngữ lập trình nào. Không giống như các giải pháp blockchain và đồng thuận khác, được đóng gói sẵn với các máy trạng thái được tích hợp Môi trường phát triển phù hợp với họ.

Tendermint được thiết kế để dễ sử dụng, đơn giản để hiểu, hiệu suất cao và hữu ích cho nhiều ứng dụng phân tán.

#### **b, Cosmos Hub**

Cosmos Hub là blockchain đầu tiên trong số hàng ngàn blockchain trong mạng lưới các blockchain có thể tương tác với nhau trên Tendermint. Token chính của Cosmos Hub là ATOM.

#### **c, ATOM**

Với ATOM, bạn có khả năng để đóng góp cho an ninh và quản trị của Cosmos Hub. Delegate ATOM của bạn cho một hoặc nhiều trong số 125 validator trên blockchain Cosmos Hub để kiểm thêm ATOM thông qua PoS. Bạn cũng có thể bỏ phiếu với ATOM của mình để ảnh hưởng đến tương lai của Cosmos Hub thông qua các đề xuất quản trị trên chuỗi.

#### **d, Validator**

Cosmos Hub dựa trên TenderMint, dựa trên một bộ validator chịu trách nhiệm thực hiện các khối mới trong blockchain. Các validator này tham gia vào giao thức đồng thuận bằng cách phát các phiếu bầu chứa chữ ký mật mã được ký bởi khóa riêng của mỗi validator.

Người giữ token có thể gửi ATOM của chính họ và có ATOM "được ủy quyền", hoặc đặt cọc cho validator, là các delegator. Hub Cosmos có 150 validator, nhưng theo thời gian, số lượng validator có thể tăng. Các validator được xác định bởi tổng số token ATOM được giao cho họ - 150 ứng cử viên xác nhận hàng đầu có quyền bầu cử nhiều nhất là các Cosmos validator hiện tại. Các validator và delegator của họ kiểm được ATOM như các quy định khối và token làm phí giao dịch thông qua



việc thực hiện giao thức đồng thuận Tendermint. Lưu ý rằng các validator có thể đặt tỷ lệ phần trăm hoa hồng cho các khoản phí mà các delegator của họ nhận được dưới dạng khuyến khích bổ sung. Bạn có thể tìm thấy một cái nhìn tổng quan về tất cả các validator hiện tại và khả năng bỏ phiếu của họ trên Mintscan.

Nếu các validator gian lận xác thực, hoặc offline trong một thời gian dài, ATOM được đặt cọc của họ (bao gồm cả ATOM của delegator cho họ) có thể bị cắt giảm. Hình phạt phụ thuộc vào mức độ nghiêm trọng của vi phạm.

### **e, Delegator**

Những người không thể hoặc không muốn vận hành các node xác thực vẫn có thể tham gia vào quá trình xác thực với tư cách là delegator. Thật vậy, các validator không được chọn dựa trên cổ phần tự gửi của họ mà dựa trên tổng cổ phần của họ, đó là tổng số cổ phần tự gửi của họ và cổ phần được giao cho họ từ các delegator. Đây là một tài sản quan trọng, vì nó làm cho các delegator trở thành một biện pháp bảo vệ chống lại các validator thể hiện hành vi xấu. Nếu một validator gian lận, các delegator của họ sẽ di chuyển các ATOM của họ ra khỏi họ, do đó giảm cổ phần của họ. Cuối cùng, nếu cổ phần của validator nằm ngoài top 125 validator có cổ phần cao nhất, họ sẽ bị đẩy khỏi nhóm validator.

Các delegator được chia sẻ lợi tức của các validator của họ, nhưng họ cũng chia sẻ rủi ro. Về mặt doanh thu, validator và delegator khác nhau ở chỗ các validator có thể áp dụng một khoản hoa hồng về doanh thu được chuyển đến delegator của họ trước khi được phân phối. Khoản cam kết này được biết đến với các delegator trước và chỉ có thể thay đổi theo các ràng buộc được xác định trước. Về rủi ro, các ATOM của các đại biểu có thể bị cắt giảm nếu validator của họ gian lận.

Để trở thành delegator, người nắm giữ ATOM cần tạo "delegate transaction", nơi họ chỉ định số lượng ATOM họ muốn ứng ra và validator nào. Một danh sách các ứng cử viên validator sẽ được hiển thị trong Cosmos Hub Explorers. Sau đó, nếu một delegator muốn rút lại một phần hoặc tất cả các cổ phần của họ, họ cần tạo một "Unbond transaction". Từ đó, các delegator sẽ phải đợi 3 tuần để lấy lại các ATOM của họ. Các delegator cũng có thể tạo một "rebond transaction" để chuyển ATOM từ validator này sang validator khác, mà không phải trải qua thời gian chờ 3 tuần.

### **f, Cosmos SDK và Gaia**

#### **(a) Cosmos SDK**

Cosmos SDK là một framework mã nguồn mở để xây dựng các blockchains Proof-Of Stake (POS), như Cosmos Hub, cũng như các blockchains Proof-

of-authority (POA). Các blockchain được xây dựng với Cosmos SDK thường được gọi là blockchain dành riêng cho ứng dụng.

Mục tiêu của Cosmos SDK là cho phép các nhà phát triển dễ dàng tạo ra các blockchain tùy chỉnh từ đầu có thể tương tác với các blockchain khác. Cosmos SDK Cosmos là khung giống như NPM để xây dựng các ứng dụng blockchain an toàn trên nền tảng Tendermint.

#### (b) Gaia

Gaia là tên của ứng dụng Cosmos SDK cho Cosmos Hub, cung cấp Gaia bao gồm Gaia Daemon và CLI để tương tác với blockchain Cosmos Hub

### 3.5 Công nghệ sử dụng

#### 3.5.1 VueJS

Vue.js là một framework rất linh động được dùng phổ biến để xây dựng nên các giao diện người dùng. Hoàn toàn khác với các framework nguyên khối thì Vue thường sở hữu thiết kế từ đầu theo những hướng cho phép cũng như khuyến khích làm việc để phát triển dễ dàng hơn các ứng dụng theo từng bước một.

Một khi đã phát triển lớp giao diện (view layer) thì người dùng chỉ cần sử dụng loại thư viện lõi của Vue. Ngoài ra, nếu như bạn kết hợp với các kỹ thuật tiên hướng hiện đại thì Vue cũng có thể đáp ứng được dễ dàng mọi nhu cầu xây dựng ứng dụng của một trang với độ phức tạp cao hơn.

Hiện nay, Vue đang giữ số stars cao nhất trên 160k; trong số đó thì framework frontend đang giữ vị trí lần lượt hiện nay là React(> 146k) và Angular(> 59.2k). Dưới đây là một số lý do nên sử dụng Vuejs.

- No build step required: Nếu sử dụng Vue thì bạn sẽ không cần phải trải qua quá nhiều bước để build mà có thể đi thẳng vào vấn đề một cách dễ dàng. Bởi vì, Vue không cần sử dụng build tool quá phức tạp mới có thể xây dựng ứng dụng, bạn chỉ cần khai báo một script là có thể phát triển một ứng dụng bằng Vue.
- Vue có thể tạo cấu trúc project nhanh chóng hơn nhờ vào command line interface.
- Hiện nay, tài liệu về Vue ngày càng đa dạng, rõ ràng về ngôn ngữ nên bạn có thể dễ dàng trở thành chuyên gia về nó.
- Vue sở hữu một hệ sinh thái vững chắc nên có thể cung cấp một số add-ons rất hữu ích cho việc xây dựng một ứng dụng frontend điển hình nhất. Nó có thể bao gồm: vue-router, vuex, vue-test-utils, vue-dev-tools, vue-cli,...

- Core Vue sở hữu tính năng tối thiểu bởi khả năng tập trung vào việc render giao diện cho người dùng và các tương tác của nó. Chính vì vậy, nó sẽ cung cấp tối thiểu những tính năng cần thiết cho việc thiết kế và xây dựng kiến trúc. Nó sẽ giúp bạn loại bỏ được các tính năng không cần thiết ra khỏi thư viện trong lõi Vue.js và đảm bảo cho framework nhỏ gọn và mềm dẻo hơn.

### 3.5.2 Golang và Gin framework

Go (hay Golang) là ngôn ngữ lập trình mã nguồn mở giúp xây dựng phần mềm dễ dàng, tin cậy và hiệu quả do các kỹ sư hàng đầu Google phát triển. Go (Golang) còn được biết đến là một ngôn ngữ static typed, mọi thứ trong Go đều phải có kiểu dữ liệu, trái với các ngôn ngữ dynamic typed như Javascript hoặc Python. Bản thân nó cũng là một Compile Programming Language, ngôn ngữ lập trình có trình biên dịch để ra được file thực thi.

Golang ra đời vì một sứ mệnh giúp tăng năng suất phần mềm, đặc biệt là ở lĩnh vực multicore processing (xử lý đa nhân), network (mạng) và những dự án có source code rất lớn. Và dưới đây là 1 một số đặc tính làm nên một ngôn ngữ rất mạnh và ưu chuộng hiện nay:

- Golang là static typed.
- Vue có thể tạo cấu trúc project nhanh chóng hơn nhờ vào command line interface.
- Golang build/compile rất nhanh
- Hỗ trợ lập trình concurrent (đồng thời) rất dễ dàng với Goroutine
- Cân bằng giữa hiệu năng và thời gian phát triển

Gin hay còn gọi là Gin-Gonic là một trong những dự án như vậy. Nó sử dụng một phiên bản tùy biến của gói httprouter vì tốc độ xử lý cực kỳ nhanh, điều này làm cho nó vô cùng hoàn hảo để phát triển API hiệu xuất cao. Song song đó, nó cung cấp các trình xử lý cho nhiều trường hợp sử dụng phổ biến: middleware, file uploading, logging, binding front-end HTML component với cấu trúc dữ liệu back-end, ...

Gin-Gonic là web framework của Golang được dùng nhiều nhất

### 3.5.3 MongoDB

MongoDB là một database hướng tài liệu (document), một dạng NoSQL database. Vì thế, MongoDB sẽ tránh cấu trúc table-based của relational database để thích ứng với các tài liệu như JSON có một schema rất linh hoạt gọi là BSON. MongoDB sử dụng lưu trữ dữ liệu dưới dạng Document JSON nên mỗi một collection sẽ các các kích cỡ và các document khác nhau. Các dữ liệu được lưu trữ trong document kiểu

JSON nên truy vấn sẽ rất nhanh.

Các đặc trưng của MongoDB gồm có:

- Các ad hoc query: hỗ trợ search bằng field, các phép search thông thường, regular expression searches, và range queries
- Indexing: bất kì field nào trong BSON document cũng có thể được index.
- Replication: có ý nghĩa là “nhân bản”, là có một phiên bản giống hệt phiên bản đang tồn tại, đang sử dụng. Với cơ sở dữ liệu, nhu cầu lưu trữ lớn, đòi hỏi cơ sở dữ liệu toàn vẹn, không bị mất mát trước những sự cố ngoài dự đoán là rất cao. Vì vậy, người ta nghĩ ra khái niệm “nhân bản”, tạo một phiên bản cơ sở dữ liệu giống hệt cơ sở dữ liệu đang tồn tại, và lưu trữ ở một nơi khác, để phòng có sự cố.
- Aggregation: Các Aggregation operation xử lý các bản ghi dữ liệu và trả về kết quả đã được tính toán. Các phép toán tập hợp nhóm các giá trị từ nhiều Document lại với nhau, và có thể thực hiện nhiều phép toán đa dạng trên dữ liệu đã được nhóm đó để trả về một kết quả duy nhất
- Lưu trữ file: MongoDB được dùng như một hệ thống file tận dụng những function trên và hoạt động như một cách phân phối qua sharding.

#### **3.5.4 Redis**

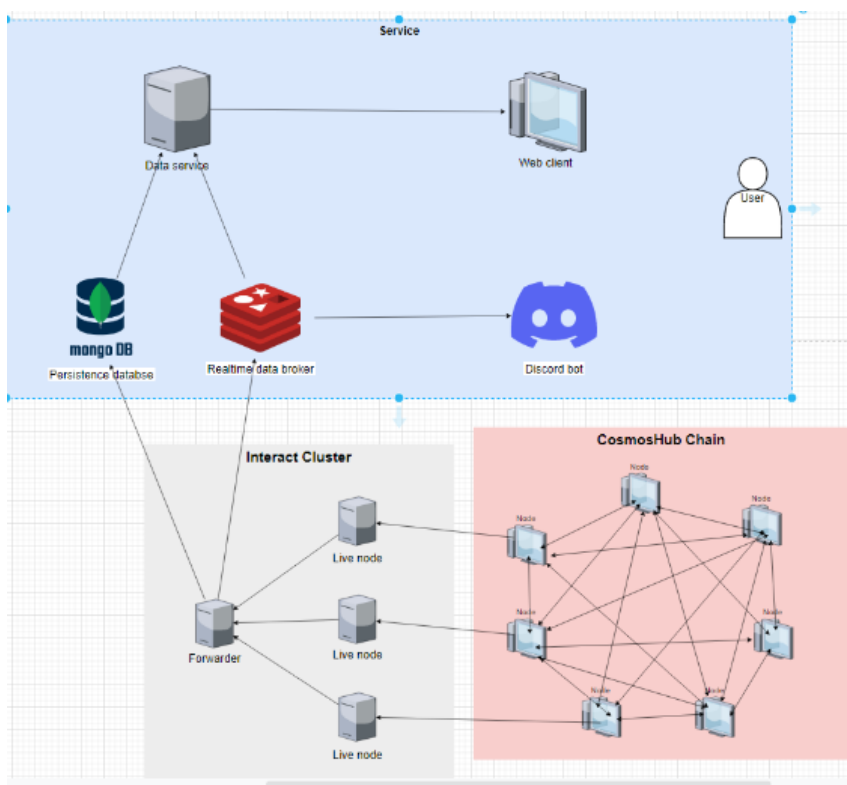
Redis (Remote Dictionary Server) là một kho lưu trữ dữ liệu key-value mã nguồn mở với tốc độ cao. Redis được sử dụng để lưu trữ dữ liệu có cấu trúc, có thể được sử dụng như một database, bộ nhớ cache hay một message broker.

## CHƯƠNG 4. THỰC NGHIỆM VÀ ĐÁNG GIÁ

### 4.1 Thiết kế kiến trúc

#### 4.1.1 Lựa chọn kiến trúc phần mềm

Đây là sơ đồ toàn bộ hệ thống mà em đã xây dựng, hệ thống được cấu tạo từ 3 thành phần chính là Service, Interact và Onchain



**Hình 4.1:** Sơ đồ kiến trúc hệ thống

Phần sau sẽ giải thích chi tiết các thành phần trong hệ thống

#### 4.1.2 Thiết kế API

Để phục vụ cho người dùng trên nền tảng web cũng như các developer có thể sử dụng các dịch vụ của hệ thống, em đã cung cấp các API:

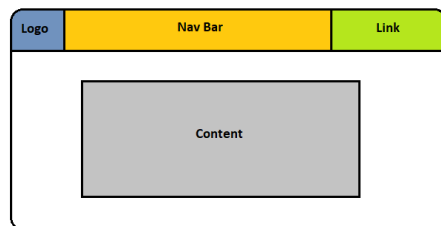
- API `Get /unbond/:delegator` dùng để lấy ra lịch sử tất cả các giao dịch undelegate của delegator được chỉ định, delegator là địa chỉ ví của một delegator trên blockchain, nếu delegator có giá trị là “all”, lấy tất cả các giao dịch undelegate trên blockchain của tất cả các delegator
- API `/websocket` và `/websocket/delegator/:delegator` cung cấp kết nối websocket đến web client để cập nhật dữ liệu liên tục

Để phục vụ cho người dùng trên nền tảng discord, bot của em sẽ cung cấp các

câu lệnh tương tác với người dùng bao gồm:

- ‘!help’: lệnh sẽ giúp người dùng liệt kê danh sách và chức năng của các câu lệnh khác trong hệ thống.
- ‘!info’: lệnh sẽ giúp người dùng hiển thị các ngưỡng cảnh báo đã đăng kí với hệ thống
- ‘!config <low> <medium> <high> <warning>’: đăng kí các ngưỡng cảnh báo với từng mức low, medium, high, warning

#### 4.1.3 Thiết kế giao diện

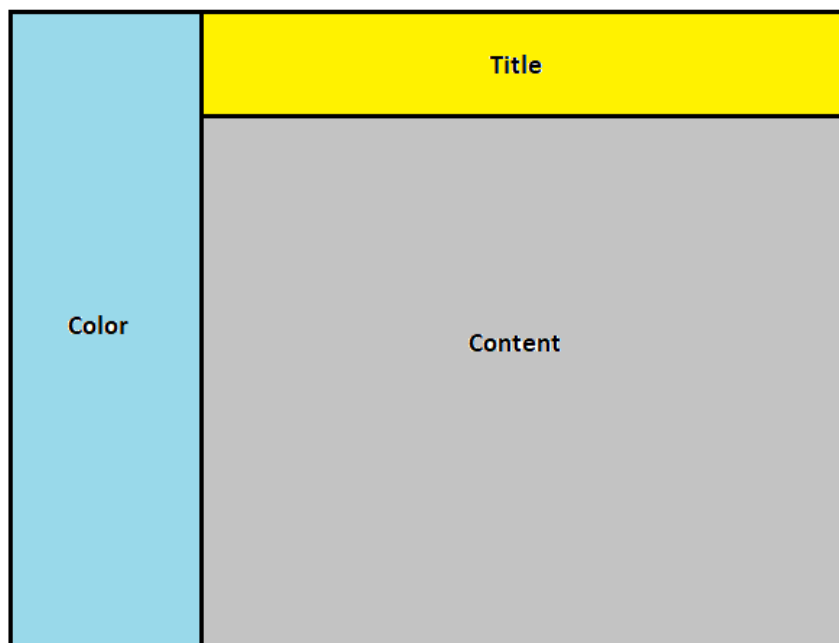


**Hình 4.2:** Thiết kế mockup cho giao diện web

Hình trên là thiết kế mockup cho giao diện web, giao diện web của em được xây dựng đơn giản gồm 4 thành phần chính là Logo, Nav Bar, Link, Content. Trong đó chỉ có phần Content thay đổi theo chức năng của hệ thống.

**Bảng 4.1:** Các cấu hình chung của giao diện trang web

Thuộc tính	Cấu hình
Độ phân giải màn hình	HD (1280x720), Full HD (1920x1080)
Màu chữ	Trắng (#ffffff)
Màu nền	Đen (#000000)
Font chữ	Sans-serif
Font chữ đậm	800
Màu viền	Vàng (#eaed4c)



**Hình 4.3:** Thiết kế mockup cho giao diện thông báo của discord

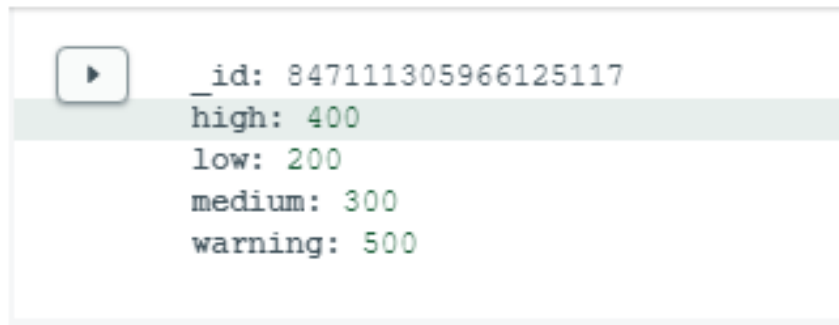
Hình trên là thiết kế mockup cho giao diện thông báo của discord, giao diện web của em được xây dựng đơn giản gồm 4 thành phần chính là Color, Title, Content. Trong đó chỉ có phần Color là màu ứng với mức cảnh báo, Title là tiêu đề mức cảnh báo và Content là nội dung cảnh báo.

**Bảng 4.2:** Các cấu hình chung của giao diện thông báo discord

Thuộc tính	Cấu hình
Độ phân giải màn hình	HD (1280x720), Full HD (1920x1080)
Màu chữ	Trắng (#ffffff)
Màu nền	Đen (#000000)
Font chữ	Sans-serif
Font chữ đậm	600
Màu cảnh báo	Đỏ (#f22707), Cam (#ff860d), Vàng (#eef51b), Xanh lá (#56f725)

#### 4.1.4 Thiết kế cơ sở dữ liệu

##### a, Dữ liệu cho cấu hình người dùng Discord



**Hình 4.4:** Dữ liệu ngưỡng cảnh báo lưu trong cơ sở dữ liệu

Bao gồm các trường:

- `_id`: là id của người dùng Discord.
- `high`: ngưỡng high mà người có `_id` đã đăng kí
- `low`: ngưỡng low mà người có `_id` đã đăng kí
- `medium`: ngưỡng medium mà người có `_id` đã đăng kí
- `warning`: ngưỡng warning mà người có `_id` đã đăng kí

##### b, Dữ liệu của sự kiện undelegate



**Hình 4.5:** Dữ liệu sự kiện lưu trong cơ sở dữ liệu

- `_id`: định danh cho document
- `validator`: địa chỉ ví validator trong giao dịch
- `amount`: lượng token rút ra khỏi địa chỉ ví validator vào ví delegator
- `time`: thời gian diễn ra sự kiện
- `completion_time`: thời gian mở khóa token
- `delegator`: địa chỉ ví delegator trong giao dịch
- `tx_hash`: mã hash của transaction, dùng để tìm kiếm thông tin của giao dịch



trên các explorer

- tx\_fee: phí thực hiện của giao dịch

## 4.2 Xây dựng ứng dụng

### 4.2.1 Thư viện và công cụ sử dụng

Đây là danh sách các công cụ, thư viện, framework sử dụng trong hệ thống của em

**Bảng 4.3:** Danh sách công cụ và thư viện sử dụng.

Mục đích	Công cụ	Địa chỉ URL
IDE lập trình frontend	Visual Studio Code	<a href="https://code.visualstudio.com/">https://code.visualstudio.com/</a>
Ngôn ngữ lập trình backend	Golang	<a href="https://go.dev/">https://go.dev/</a>
Ngôn ngữ lập trình backend	Python	<a href="https://www.python.org/">https://www.python.org/</a>
Framework lập trình backend	Gin framework	<a href="https://github.com/gin-gonic/gin">https://github.com/gin-gonic/gin</a>
Các thư viện lập trình backend	Gorilla websocket, discord.py	<a href="https://discordpy.readthedocs.io/en/stable/">https://discordpy.readthedocs.io/en/stable/</a> <a href="https://github.com/gorilla/websocket">https://github.com/gorilla/websocket</a>
Công nghệ frontend	VueJS	<a href="https://vuejs.org/">https://vuejs.org/</a>
Cơ sở dữ liệu	MongoDB	<a href="https://www.mongodb.com/">https://www.mongodb.com/</a>
Thư viện lập trình cho frontend	Vuex, Vue router	<a href="https://vuex.vuejs.org/">https://vuex.vuejs.org/</a> <a href="https://router.vuejs.org/">https://router.vuejs.org/</a>
Message broker	Redis	<a href="https://redis.io/">https://redis.io/</a>

### 4.2.2 Kết quả đạt được

Sau khi hoàn thiện, hệ thống của em đã hoạt động đúng với kỳ vọng, đáp ứng được các mục đích ban đầu của đề tài


### 4.2.3 Minh họa các chức năng chính

**a, Quan sát danh sách toàn bộ sự kiện**

[illegible]

**Hình 4.6:** Giao diện trang "All validator"

### Sau khi “View more”



All events By date

🔍 📄 🔄 ⌂

## All events

Integrator address	Validator address	Amount	Time
0x00	0x00	5000000000	2020-01-01T00:00:00Z
0x00	0x00	2000000000	2020-01-01T00:00:00Z
0x00	0x00	8000000000	2020-01-01T00:00:00Z
0x00	0x00	3000000000	2020-01-01T00:00:00Z
0x00	0x00	3000000000	2020-01-01T00:00:00Z
0x00	0x00	5000000000	2020-01-01T00:00:00Z
0x00	0x00	7000000000	2020-01-01T00:00:00Z
0x00	0x00	8000000000	2020-01-01T00:00:00Z
0x00	0x00	10000000000	2020-01-01T00:00:00Z
0x00	0x00	4000000000	2020-01-01T00:00:00Z

View more

#### Hình 4.7: Kết quả khi mở rộng tìm kiếm

### **b, Quan sát danh sách các sự kiện theo delegator**

The screenshot displays the AURA Network Delegator Dashboard. At the top, there is a navigation bar with the AURA Network logo and the text 'All events By Delegator'. Below the navigation bar is a search bar with the placeholder text 'Search by delegator address...'. Underneath the search bar is a table with four columns: 'Validator address', 'Amount', 'Time', and 'Transaction hash'. Below the table is a button labeled 'View score'.

#### Hình 4.8: Giao diện trang "By Delegator"

Sau khi nhập địa chỉ delegator và “Search”:

cosmos12hqn3a9a2d6aeef9c64zuezdcmrkaak

Q

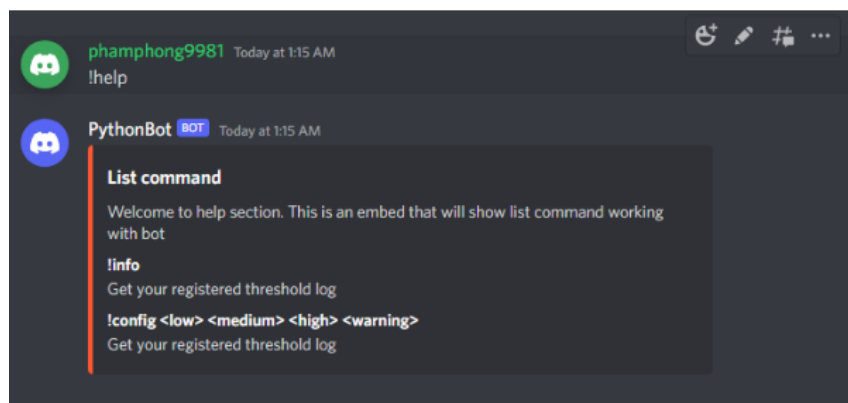
Validator address	Amount	Time	Transaction hash
cosmosvaloper17kshy3tqgshu59a0crrvppg7y0pnpqg	5000 uat	2025-07-31T13:59:56+07:00	85665C3C03E2D54D52F8794368A33F6FAA0A987C6474703C4D669E
cosmosvaloper76t6dt5v6dflnsh7ew3g3y9k6lcnw3gpgs	300 uat	2025-07-31T16:30:00+07:00	85670CCAC248AC7B674D274D932FDC49AA092B330667D4D349E85944F
cosmosvaloper76t6dt5v6dflnsh7ew3g3y9k6lcnw3gpgs	300 uat	2025-07-31T16:30:00+07:00	85670CCAC248AC7B674D274D932FDC49AA092B330667D4D349E85944F
cosmosvaloper76t6dt5v6dflnsh7ew3g3y9k6lcnw3gpgs	300 uat	2025-07-31T16:46:32+07:00	469fE43D0983C34607A831367A3A3AE44F0CD8C93935556A39FACA00A4
cosmosvaloper76t6dt5v6dflnsh7ew3g3y9k6lcnw3gpgs	300 uat	2025-07-31T16:49:29+07:00	469fE43D0983C34607A831367A3A3AE44F0CD8C93935556A39FACA00A4

View more

### Hình 4.9: Kết quả khi mở rộng tìm kiếm

### c, Các lệnh trên discord

Với câu lệnh “!help”:



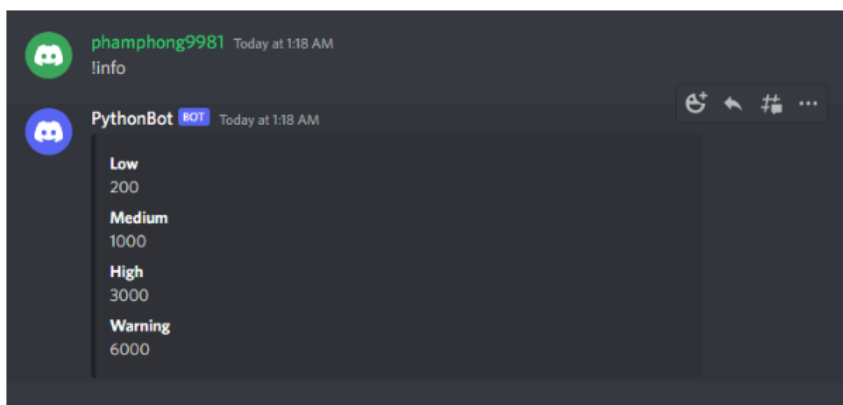
#### Hình 4.10: Kết quả với lệnh help

Với câu lệnh “!config <low> <medium> <high> <warning>”



### Hình 4.11: Kết quả với lệnh config

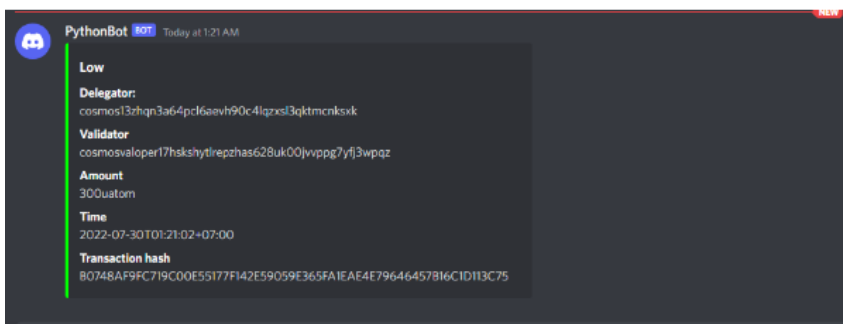
Với câu lệnh “!info”:



**Hình 4.12:** Kết quả với lệnh info

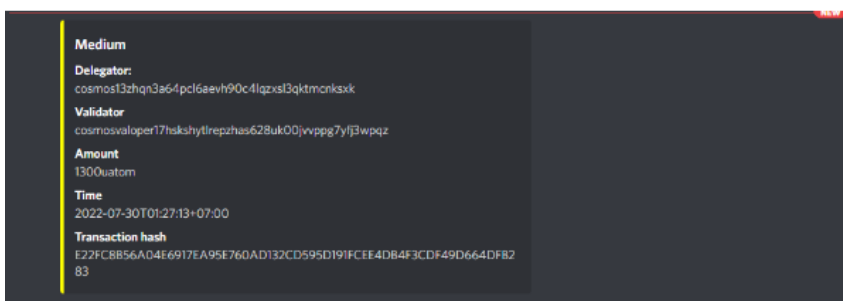
#### d, Các cảnh báo trên discord

Khi người dùng nhận được cảnh báo mức Low:



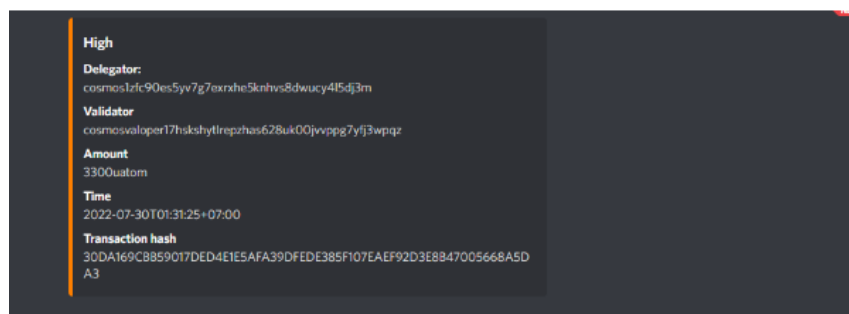
**Hình 4.13:** Discord bot cảnh báo mức low

Khi người dùng nhận được cảnh báo mức Medium:



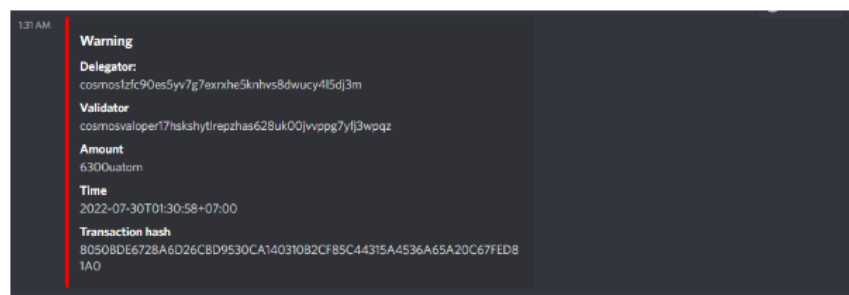
**Hình 4.14:** Discord bot cảnh báo mức medium

Khi người dùng nhận được cảnh báo mức High:



Hình 4.15: Discord bot cảnh báo mức high

Khi người dùng nhận được cảnh báo mức Warning:



Hình 4.16: Discord bot cảnh báo mức warning

### 4.3 Kiểm thử

#### 4.3.1 Kiểm thử chức năng hiển thị các sự kiện

Bảng 4.4: Kiểm thử chức năng hiển thị toàn bộ các sự kiện.

Test case	Đầu vào	Đầu ra mong muốn	Kết quả
Hiển thị top 20 sự kiện gần nhất	Không	Danh sách 20 sự kiện gần nhất	Đạt
Mở rộng danh sách hiển thị	Không	20 sự kiện gần nhất tiếp theo được hiển thị	Đạt
Hiển thị top 20 sự kiện gần nhất theo delegator	Địa chỉ ví của delegator	20 giao dịch gần nhất của delegator	Đạt

#### 4.3.2 Kiểm thử chức năng thông báo của discord

**Bảng 4.5:** Kiểm thử chức năng hiển thị toàn bộ các sự kiện.

Test case	Đầu vào	Đầu ra mong muốn	Kết quả
Cảnh báo các khi có sự kiện xảy ra ở mức low	Ngưỡng được thiết lập	Thông báo	Đạt
Cảnh báo các khi có sự kiện xảy ra ở mức medium	Ngưỡng được thiết lập	Thông báo	Đạt
Cảnh báo các khi có sự kiện xảy ra ở mức high	Ngưỡng được thiết lập	Thông báo	Đạt
Cảnh báo các khi có sự kiện xảy ra ở mức warning	Ngưỡng được thiết lập	Thông báo	Đạt

## 4.4 Triển khai

### 4.4.1 Chuẩn bị trước

Để tương tác và tạo các giao dịch trên blockchain, chúng ta cần cài đặt Gaiad, mọi hướng dẫn cài đặt sẽ có trong đường dẫn <https://hub.cosmos.network/main/getting-started/installation.html>. Do tính chất của đồ án tốt nghiệp, em sẽ không đi sâu giải thích về gaiad, mọi thông tin và hướng dẫn đều có trong document của Cosmos Hub ở đường dẫn <https://hub.cosmos.network/main/hub-overview/overview.html>.

Bên cạnh đó, ta cần ví Cosmos Hub có một lượng token đủ thanh toán phí giao dịch và đã stake một lượng tiền vào 1 validator bất kì Ví này chúng ta sẽ dung với mục đích tạo testcase, thay vì phải chờ sự kiện xuất hiện trên hệ thống.

Đây là ví em đã tạo thông qua gaiad:

```
name: phamphong
type: local
address: cosmos13zhqn3a64pcl0aevh98c4lqzxs13qktmcnksxk
pubkey: '{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"Anty0Qr6AgZfdly71/cKPdG3iADC0iz79Zm7aZfjSp8i"}'
mnemonic: ""
```

**Hình 4.17:** Chuẩn bị ví

Lượng token em đang có trong ví:

```
C:\Users\Admin>gaiad-v7.0.2-windows-amd64 query bank balances cosmos13zhqn3a64pcl6aevh90c4lqzxs13qktmcnksxk
balances:
- amount: "4900710"
  denom: uatom
pagination:
  next_key: null
  total: "0"
```

**Hình 4.18:** Chuẩn bị token trong ví

Em đã stake một lượng token ( 400000uatom ) vào validator 01Node có địa chỉ cosmosvaloper178h4s6at5v9cd8m9n7ew3hg7k9eh0s6wptxpcn, phí giao dịch là 200uatom:

```
C:\Users\Admin>gaiad-v7.0.2-windows-amd64 tx staking delegate cosmosvaloper178h4s6at5v9cd8m9n7ew3hg7k9eh0s6wptxpcn 400000uatom --from cosmos13zhqn3a64pcl6aevh90c4lqzxs13qktmcnksxk --fees 200uatom
{"body":{"messages":[{"@type":"/cosmos.staking.v1beta1.MsgDelegate","delegator_address":"cosmos13zhqn3a64pcl6aevh90c4lqzxs13qktmcnksxk","validator_address":"cosmosvaloper178h4s6at5v9cd8m9n7ew3hg7k9eh0s6wptxpcn","amount":{"denom":"uatom","amount":"400000"}}], "memo":"","timeout_height":"0","extension_options":[],"non_critical_extension_options":[],"auth_info":{"signer_infos":[{"fee":{"amount":[{"denom":"uatom","amount":"200"}],"gas_limit":"200000","payer":"","granter":""},"signatures":[]]}]}]}
confirm transaction before signing and broadcasting [y/N]: y
code: 0
codespace: ""
data: ""
events: []
gas_used: "0"
gas_wanted: "0"
height: "0"
info: ""
logs: []
raw_log: "[{}]"
timestamp: ""
tx: null
txhash: 07C2182214DE014241768C39E474076CFD00E298C7B72FC46F8043968311615F
```

**Hình 4.19:** Chuẩn bị một lượng token để stake

#### 4.4.2 Khởi chạy hệ thống

##### a, Live server

Đây là kết quả sau khi em bật từng live server:

```
2022/07/27 09:35:04 Stopped connection to Node
PS C:\Users\Admin\Documents\GitHub\CosmosHubSubscribeEvent> go run live.go
2022/07/27 09:35:04 Trying to connect to node ...
2022/07/27 09:35:04 Connected to Node
2022/07/27 09:35:04 Trying to connect to forwarder ...
2022/07/27 09:35:05 Connected to Forwarder
2022/07/27 09:35:05 {
  "jsonrpc": "2.0",
  "id": 0,
  "result": {}
}
```

**Hình 4.20:** Kết quả khi live server khởi động

Ban đầu live server sẽ cố gắng kết nối với node, chỉ khi kết nối với node thành công, live server mới thực hiện kết nối với Forwarder.

##### b, Forwarder

Sau khi khởi chạy 3 live server gắn với 3 node khác nhau, nếu việc kết nối với các node thành công, các live server sẽ thực hiện kết nối với Forwarder. Đây là kết quả khi em chạy Forwarder:

```

PS C:\Users\Admin\Documents\GitHub\CosmosHubSubscribeEvent> go run forwarder.go
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
Node127.0.0.1:57545 up
map[0xc0002d4160:127.0.0.1:57545]
Node127.0.0.1:57546 up
map[0xc0002d4160:127.0.0.1:57545 0xc0002d4420:127.0.0.1:57546]
Node127.0.0.1:57547 up
map[0xc0002d4160:127.0.0.1:57547 0xc0002d4160:127.0.0.1:57545 0xc0002d4420:127.0.0.1:57546]
Connected to live server 127.0.0.1:57545

```

**Hình 4.21:** Kết quả khi forwarder khởi động

Forwarder sẽ mở websocket lắng nghe các kết nối từ các live server, mỗi kết nối đến sẽ được lưu vào mảng danh sách, như ta thấy trong kết quả trên, có 3 live server đang kết nối đến Forwarder là 127.0.0.1:57547, 127.0.0.1:57545 và 127.0.0.1:57546. Tuy nhiên tại một thời điểm, Forwarder sẽ chỉ nhận dữ liệu đến từ 1 live server và trên kết quả hiển thị trên thì đó là 127.0.0.1:57545

Khi một node trong mạng gặp vấn đề, live server gắn với node đó sẽ tự động ngắt kết nối websocket với Forwarder, Forwarder sẽ tìm các live server còn lại trong danh sách để chuyển hướng nhận dữ liệu, và đây là kết quả khi node gắn với live server 127.0.0.1:57545 gặp vấn đề:

```

Connected to live server 127.0.0.1:57545
Node 127.0.0.1:57545 down: websocket: close 1000 (normal)
Connected to live server 127.0.0.1:57546

```

**Hình 4.22:** Forwarder cố gắng kết nối lại khi bị mất kết nối đến các live server

Như kết quả trên, Forwarder đã chuyển qua tiếp nhận dữ liệu từ live server 127.0.0.1:57546 khi node gắn với live server 127.0.0.1:57545 gặp vấn đề

Khi tất cả các node đều bị gặp vấn đề thì Forwarder liên tục phải tìm kiếm và đương nhiên dữ liệu trong trường hợp này sẽ không được tiếp nhận bởi hệ thống, trường hợp này chúng ta cần phải có tính toán cụ thể ban đầu về số lượng node và live server cần dựng lên để hạn chế rủi ro đó:

```

Node 127.0.0.1:63186 down: websocket: close 1000 (normal)
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up
There arent any available live sever which could connect. Wait 1 second for live server up

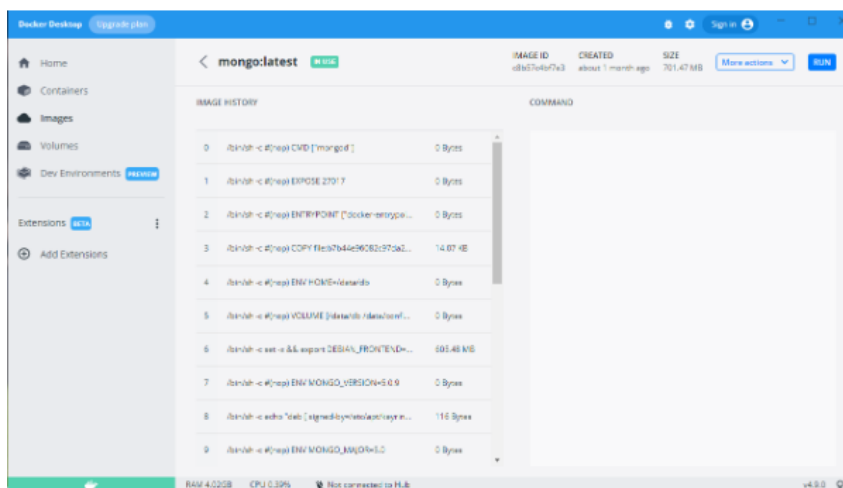
```

**Hình 4.23:** Forwarder thông báo khi không có live server nào hoạt động



### c, Persistence database (MongoDB)

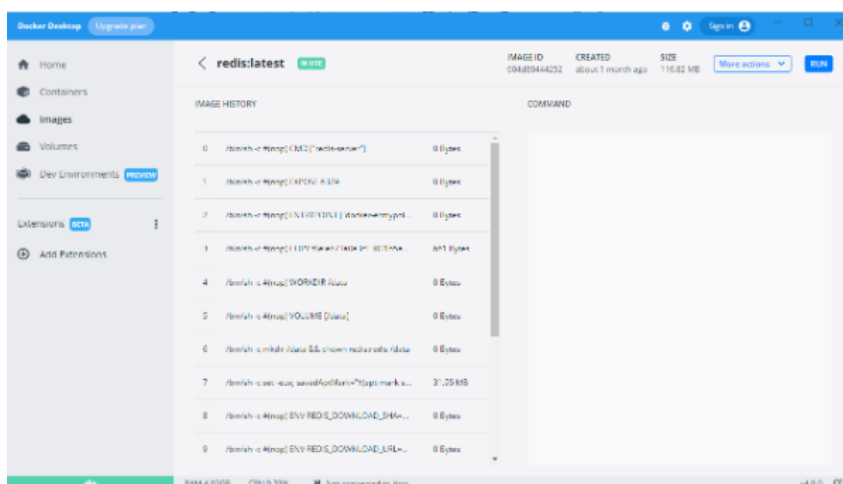
Trong đồ án tốt nghiệp, em sử dụng Docker là nền tảng để chạy mongoDB, đây là kết quả khi khởi chạy MongoDB tại <http://localhost:27017>



Hình 4.24: Kết quả sau khi khởi động mongo

### d, Realtime data broker (Redis)

Tương tự MongoDB, em cũng sử dụng Docker là nền tảng để chạy Redis, đây là kết quả chạy Redis tại <http://localhost:6379>



Hình 4.25: Kết quả sau khi khởi động redis

### e, Data service

Kết quả sau khi khởi chạy Data service lắng nghe tại cổng 8088:

```

PS C:\Users\Admin\Documents\GitHub\CosmosHubSubscribeEvent> go run backend.go
GIN-debug [WARNING] Creating an Engine instance with the Logger and Recovery middleware already attached.

GIN-debug [WARNING] Running in "debug" mode. Switch to "release" mode in production.
- using env:   export GIN_MODE=release
- using code:  gin.SetMode(gin.ReleaseMode)

GIN-debug GET /unbond/:delegator --> datn/backend.Run.func1 (4 handlers)
GIN-debug GET /websocket --> datn/backend.serveClient (4 handlers)
GIN-debug GET /websocket/delegator/:delegator --> datn/backend.serveClientByDelegator (4 handlers)
GIN-debug [WARNING] You trusted all proxies, this is NOT safe. We recommend you to set a value.
Please check https://pkg.go.dev/github.com/gin-gonic/gin#readme-don-t-trust-all-proxies for details.
GIN-debug Listening and serving HTTP on :8088

```

**Hình 4.26:** Kết quả sau khi khởi động data service

Data service cung cấp 2 chức năng chính:

- API Get /unbond/:delegator dùng để lấy ra lịch sử tất cả các giao dịch undelegate của delegator được chỉ định, delegator là địa chỉ ví của một delegator trên blockchain, nếu delegator có giá trị là “all”, lấy tất cả các giao dịch undelegate trên blockchain của tất cả các delegator
- API /websocket và /websocket/delegator/:delegator cung cấp kết nối websocket đến web client để cập nhật dữ liệu liên tục

#### f, Web client

Kết quả khi em khởi chạy thành công giao diện web:

```

PROBLEMS 22 OUTPUT DEBUG CONSOLE TERMINAL JUPYTER
DONE Compiled successfully in 9817ms

App running at:
- Local: http://localhost:8081/
- Network: http://192.168.1.203:8081/

Note that the development build is not optimized.
To create a production build, run npm run build.

```

**Hình 4.27:** Kết quả khi khởi động web client

Giao diện web hiện đang chạy trên cổng 8081 của localhost

#### g, Discord bot

Discord bot được lập trình bằng ngôn ngữ python, dựa trên thư viện discord.py, documentation của thư viện có ở đường dẫn <https://discordpy.readthedocs.io/en/stable/>

Kết quả khi em khởi động discord bot:

```

PS C:\Users\Admin\Documents\GitHub\CosmosHubSubscribeEvent> python .\discord_python\discord_bot.py

```

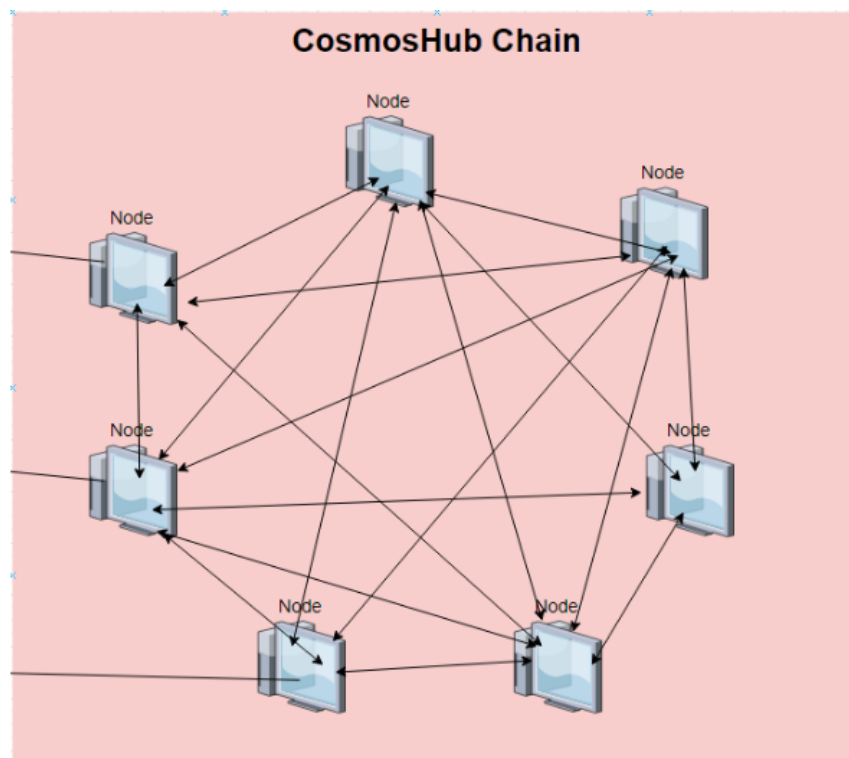
**Hình 4.28:** Kết quả khi khởi động discord bot

## CHƯƠNG 5. CÁC GIẢI PHÁP VÀ ĐÓNG GÓP NỔI BẬT

### 5.1 Về phần hệ thống

Ở phần 4.1.1, em đã trình bày tổng quan về hệ thiết kế hệ thống mà em đã xây dựng. Trong phần này, em sẽ trình bày chi tiết các thành phần trong hệ thống 4.1 mà em đã xây dựng để chứng minh hệ thống là giải pháp rất phù hợp ứng với yêu cầu đề tài

#### 5.1.1 Cụm on-chain

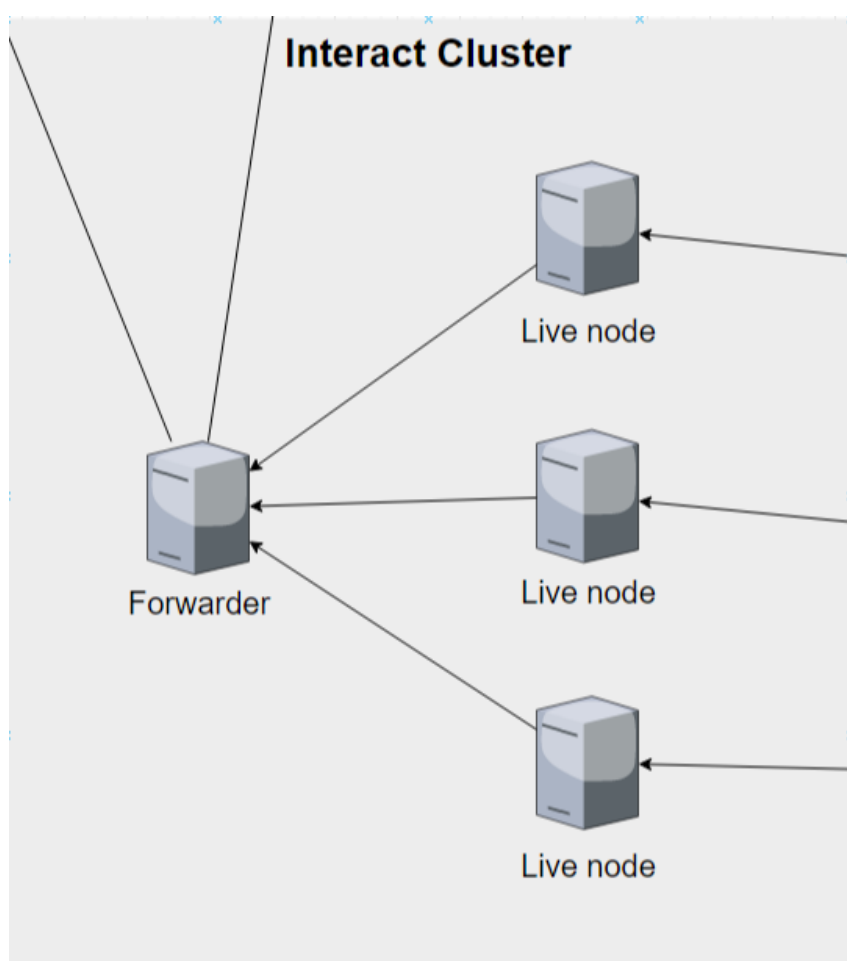


**Hình 5.1:** Sơ đồ mạng lưới blockchain

Trước tiên khi muốn tương tác với hệ thống blockchain nào đó, ở đây là Cosmos Hub, chúng ta phải là một hoặc cần một node trong mạng lưới. Node ở đây nghĩa là một máy ở trong mạng lưới, tham gia vào quá trình lưu trữ sổ cái. Khi xây dựng thành công node đó lên, việc ta cần làm chỉ cần là tương tác từ bên ngoài đến các node đó một cách dễ dàng. Ở trong hệ thống của em, em đang dùng 3 node của Aura, WhisperNode, chainapsis đã dựng lên có địa chỉ là: <https://rpc.cosmos.network>, <https://rpc-cosmoshub.whispernode.com>, <https://rpc-cosmoshub.blockapsis.com>. Để đảm bảo tính ổn định của hệ thống, chống việc khi một node gặp vấn đề thì sẽ luôn có 2 node khác thay thế nhiệm vụ của nó. Ngoài ra, chúng ta có thể tham khảo thêm địa chỉ các node được công khai của các tổ chức lớn khác ở <https://github.com/cosmos/chain-registry/blob/master/cosmoshub/chain.json>

Để giải thích rõ hơn tại sao em đã xây dựng 3 node, thì bởi vì trong một mạng lưới blockchain như Cosmos Hub, nhờ được xây dựng trên nền tảng là Tendermint, các node đã được đồng bộ với nhau về mặt dữ liệu. Nghĩa là các node đều có chung "sổ cái" để lưu trữ các giao dịch. Nhưng nếu một node có vấn đề thì sẽ thế nào? Khi đó, nếu chỉ tương tác với 1 node thì trong thời gian đó, cả hệ thống dường như sẽ ngừng hoạt động. Vì vậy khi ta tăng số lượng node khác nhau mà hệ thống tác thì khả năng chịu lỗi trong hệ thống sẽ càng tốt hơn. Tùy theo tính toán và khả năng mà chúng ta có thể tăng hoặc giảm số lượng đó ở một mức thích hợp.

### 5.1.2 Cụm tương tác



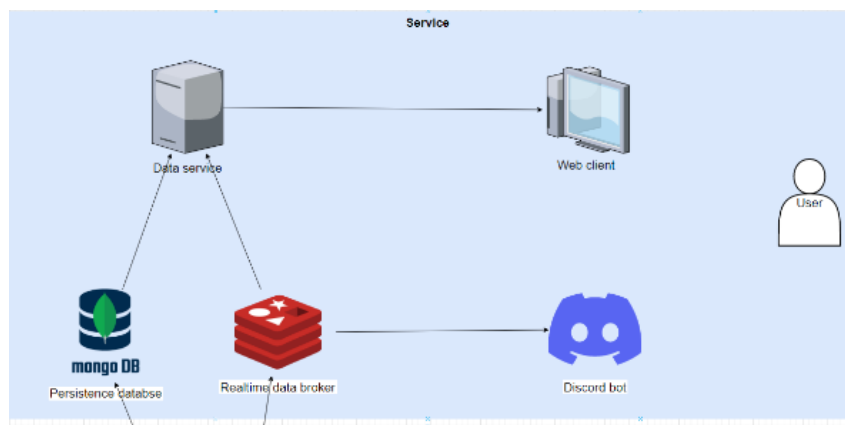
**Hình 5.2:** Sơ đồ thiết kế cụm tương tác

Với mỗi node trong blockchain, ta xây dựng 1 live server để tương tác với node đó, cụ thể ở đây em dùng websocket để duy trì kết nối đó. Mỗi khi node trong blockchain ghi nhận được sự kiện, ngay lập tức thông tin qua websocket đã được truyền tải đến live server. Các live server sẽ lại duy trì với chung 1 forwarder nhưng forwarder đó chỉ nhận dữ liệu từ 1 live server, 2 live server còn lại sẽ trong trạng

thái chờ. Khi node đang hoạt động gặp sự cố, live server sẽ biết và báo cho Forwarder để chuyển node nhận data. Forwarder có trách nhiệm tổng hợp thông tin, quản lý lỗi, điều phối thông tin đến database và message broker.

Như đã nói ở trên, em đã đề xuất để các live server kết nối chung cùng đến một forwarder. Forwarder ở đây giống như master trong mô hình master-slave, khi đó master sẽ nhận kết quả từ các slave và điều phối cũng như tổng hợp dữ liệu để truyền tải đồng bộ và hiệu quả nhất. Bất kể khi live server hay node ứng với nó gặp phải vấn đề, luôn có 2 cặp node-live server khác sẵn sàng thay thế làm tiếp nhiệm vụ gần như ngay lập tức. Và ngay cả sau khi gặp lỗi, nếu các node trở lại hoạt động bình thường, các live server sẽ chủ động tái kết nối với forwarder để tham gia vào hàng chờ phục vụ. Qua đó chúng ta có thể thấy khả năng mà hệ thống ngừng hoạt động do ảnh hưởng từ mạng lưới gần như khó có thể xảy ra, chỉ khi 3 node cùng lúc ngừng hoạt động. Vì vậy, chúng ta cần phải tính toán, đưa ra các giá trị phù hợp số lượng để đảm bảo tính ổn định và cũng như chi phí hoạt động của hệ thống

### 5.1.3 Cụm service



**Hình 5.3:** Sơ đồ thiết kế cụm dịch vụ

Trong cụm service, em sẽ đi qua chức năng của từng thành phần:

- **Persistence database (MongoDB)** là nơi lưu trữ các sự kiện, do việc truy vấn trên blockchain gặp rất nhiều khó khăn do blockchain nếu muốn query thì sẽ phải theo page tuần tự. Với MongoDB là một NoSQL database rất phù hợp với bản chất dữ liệu, tính chất bài toán và khả năng truy vấn.
- **Realtime data broker (Redis)** là broker được dùng cho việc chuyển tiếp dữ liệu thực đến Discord Bot và cả giao diện web. Cũng có nhiều loại message broker nhưng em chọn Redis vì nó rất thông dụng trong các dự án thực tế của các doanh nghiệp cũng như đối với sinh viên.

- **Data Service:** là nơi cung cấp API chính cho việc truy vấn từ Persistence database và subscribe Realtime data broker để cung cấp websocket cho frontend.
- **Web Client:** là giao diện web phục vụ cho người dùng web, gọi api từ Data service và đồng thời subscribe realtime data thông qua websocket
- **Discord Bot:** bot được viết bằng Python, subscribe Realtime data từ Realtime data broker, khi có sự kiện sẽ thông báo cho người dùng.

Ta thấy rằng việc discord bot nhận dữ liệu trực tiếp từ Redis đã cho thấy được tính thời gian thực của hệ thống. Các bot sẽ nhận được trực tiếp dữ liệu thông qua cơ chế Pub/Sub của Redis. Việc Forwarder cùng lúc truyền tải trực tiếp đến MongoDB và redis sẽ giúp tạo ra tính nhất quán trong việc lưu trữ dữ liệu. Hơn thế nữa, ta có thể thấy sau khi dữ liệu được truyền tải đến Redis, data service cũng theo dõi để nhận được dữ liệu đó. Em đồng thời tạo web socket giữa Data service và Web client để giao diện web cũng sẽ được cập nhật liên tục với các sự kiện mới

## 5.2 Về công nghệ

Trong đồ án tốt nghiệp này, em lựa chọn ngôn ngữ lập trình Golang là ngôn ngữ chủ yếu để lập trình phía backend. Lý do chính là Golang là một ngôn ngữ mới, phổ biến và nổi bật nhất ở khả năng xử lý đa luồng. Xử lý đa luồng trong hệ thống của em được áp dụng trong từng server, vì để đảm bảo tính thời gian thực thì tính đa luồng cũng cần thiết phải kèm theo.

Cùng với đó là việc sử dụng một front-end framework hiện đại chính là Vuejs đã giúp em dễ dàng xây dựng cấu trúc và lập trình phía front-end nhanh hơn.

Việc sử dụng MongoDB cũng có lý do vì về thực tế, hệ sinh thái Cosmos vẫn đang trong quá trình phát triển. Việc cập nhật phiên bản thường xuyên diễn ra. Vì vậy, cấu trúc dữ liệu mà em nhận được từ mạng lưới Cosmos Hub luôn thay đổi qua từng phiên bản. So với việc lưu trữ trên các hệ cơ sở dữ liệu SQL thì gần như là không khả thi khi truy vấn. Còn so với các hệ quản trị cơ sở dữ liệu NoSQL cùng loại thì Mongo lại là công cụ phổ biến, thông dụng nhất, nó cũng cung cấp các lựa chọn chi phí thấp hoặc miễn phí nên em cũng có thể dễ dàng triển khai và phát triển

## 5.3 Về ứng dụng thực tiễn

Em thấy hệ thống của em có tính ứng dụng cao trong thực tế. Những ý tưởng ban đầu cho việc xây dựng và phát triển hệ thống cũng từ tìm hiểu nhu cầu thực tế của người dùng, những tính chất, mối quan hệ giữa các sự kiện với tác động đối với người dùng. Vì vậy, em nghĩ với những người có nhu cầu về đầu tư blockchain

thì giải pháp của em có thể giúp đỡ họ phần nào, để nhận thức sớm nhất để đưa ra các quyết định hợp lý cũng như cho họ khả năng tìm kiếm, phân tích các sự kiện đó sau này.

Hệ thống của em có khả năng mở rộng về sau, vì trong đồ án này, em chỉ lấy điển hình là sự kiện undelegate, nhưng sau này, khi có những mô hình kinh tế khác, những hình thức đầu tư mới thì với việc là tương tác trực tiếp với Tendermint thì các sự kiện trên mạng lưới đều có thể được hệ thống của em lấy về tương tự như sự kiện undelegate.

## CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

### 6.1 Kết luận

#### 6.1.1 So sánh với các hệ thống sẵn có

So với các hệ thống sẵn có trên thị trường, hệ thống của em đã có những tính năng đặc biệt như hướng đến phục vụ cộng đồng người dùng Discord, có khả năng chống chịu lỗi khá tốt nhờ có các nút phòng bị cho các rủi ro có thể xảy ra

#### 6.1.2 Đánh giá ưu nhược điểm và khả năng ứng dụng

Trong quá trình xây dựng hệ thống, cùng với kết quả thử nghiệm và chạy thử, em rút ra được những nhận xét sau đây:

##### a, Về ưu điểm

- Hệ thống chạy ổn định, những kết quả đều đúng như mong muốn ban đầu theo thiết kế
- Hệ thống có những thiết kế chống chịu lỗi như tự động thực hiện kết nối lại khi gặp vấn đề truyền tải, chuyển hướng nhận dữ liệu khi một node trong mạng gặp vấn đề,...
- Hệ thống đa dạng giao diện, phục vụ thuận tiện cho người dùng
- Hệ thống đã xây dựng nền tảng để có thể dễ dàng mở rộng và cải tiến ví dụ như ta có thể thêm các sự kiện khác như staking, redelegate, các giao dịch với số tiền lớn,...

##### b, Về nhược điểm

- Hệ thống hiện đang xây dựng trong ngữ cảnh là đồ án tốt nghiệp nên có thể có khả năng về lỗi và chưa đảm bảo an toàn trước nguy cơ bị tấn công.
- Hệ thống của em đang hoàn toàn chạy trên mạng cục bộ, nên hiện chưa gặp các vấn đề về đường truyền

### 6.2 Hướng phát triển

Có thể thấy, hiện tại blockchain vẫn đang là một công nghệ ở mức tiềm năng, đang được mọi người để ý sau những thành tựu nổi bật của Bitcoin và Ethereum. Vì chưa đạt được đến sự ổn định và công nhận mạnh mẽ từ nhiều nước trên thế giới, nên vẫn có những hoài nghi về công nghệ này cũng như ứng dụng thực tế của nó là tiền điện tử. Nhưng nếu sau này công nghệ này đạt đến đỉnh cao, tiền điện tử được công nhận, trở thành một hình thức đầu tư chính thức được cả thế giới công



nhận, khi đó hệ thống của em sẽ có chỗ đứng và tiềm năng phát triển mạnh mẽ vì tính ứng dụng thực tiễn mà nó mang lại

## TÀI LIỆU THAM KHẢO

- [1] Gersbach, Hans and Mamageishvili, Akaki and Schneider, Manvir *Staking Pools on Blockchains*. [Online]. Available: <https://arxiv.org/pdf/2203.05838.pdf> (visited on 2022/05)
- [2] Kwon, Jae and Buchman, Ethan *Cosmos Whitepaper*. [Online]. Available: [https://wikibitimg.fx994.com/attach/2020/12/16623142020/WBE16623142020\\_55300.pdf](https://wikibitimg.fx994.com/attach/2020/12/16623142020/WBE16623142020_55300.pdf) (visited on 2020/12/29)
- [3] Cosmos, IBC, *What is Cosmos (ATOM)?*. [Online]. Available: <https://pintu.co.id/en/academy/post/what-is-cosmos-atom>
- [4] Ducr  e, Jens, *Research–A blockchain of knowledge?*. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2096720920300051>

# PHỤ LỤC