**BỘ GIÁO DỤC VÀ ĐÀO TẠO**

**TRƯỜNG ĐẠI HỌC PHENIKAA**

\*\*\*\*\*



**tool Auto challenge**

|  |  |
| --- | --- |
| **Course/Học phần** | : KỸ THUẬT PHẦN MỀM |
| **Code/Mã học phần** | : CSE702025 |
| **Lớp/Class**  **Instructor/Giảng viên**  **Group/Nhóm** | :COUR01.LT5  : Trịnh Thanh Bình  : 9  HÀ NỘI – 2025 |

MỞ ĐẦU

Trong bối cảnh công nghệ thông tin phát triển mạnh mẽ, việc ứng dụng tự động hóa trong các trò chơi trực tuyến ngày càng trở nên phổ biến, nhằm nâng cao trải nghiệm người chơi và tối ưu hóa thời gian. Roblox với nền tảng đa dạng và cộng đồng người chơi rộng lớn, đã trở thành môi trường lý tưởng để phát triển các ứng dụng hỗ trợ người chơi.

Game **Anime Vanguard** trên Roblox là một tựa game chiến thuật hấp dẫn, thu hút người chơi bởi hệ thống nhân vật phong phú và các thử thách đa dạng. Tuy nhiên, để đạt được tiến độ trong game và thu thập tài nguyên như Gems, Gold Coins hay hoàn thành các nhiệm vụ (Bounties), người chơi cần dành nhiều thời gian và công sức. Điều này gây khó khăn cho những người chơi không có nhiều thời gian rảnh hoặc muốn tối ưu hóa hiệu quả chơi mà không phải can thiệp liên tục hoặc đơn giản là do lười.

Để giải quyết vấn đề này, chúng tôi đã phát triển một **hệ thống tự động hóa** dành riêng cho người chơi Anime Vanguard giúp tự động thực hiện các thử thách trong game, tự động farm tài nguyên, nâng cấp và quản lý các đơn vị trong game một cách hiệu quả. Hệ thống không chỉ giúp tiết kiệm thời gian mà còn nâng cao hiệu quả chơi, phù hợp với những người chơi bận rộn hoặc đơn giản là muốn trải nghiệm game thoải mái hơn.

Báo cáo này trình bày chi tiết về quá trình phát triển hệ thống, bao gồm phân tích yêu cầu, thiết kế kiến trúc và giao diện, quản lý dự án theo phương pháp Scrum, triển khai và kiểm thử, cũng như đánh giá kết quả và đề xuất hướng phát triển trong tương lai.

# GIỚI THIỆU ĐỀ TÀI

## Bối cảnh và vấn đề đặt ra

Trong thế giới game trực tuyến hiện đại, việc tối ưu hóa thời gian chơi và tài nguyên là nhu cầu thiết yếu của nhiều người chơi, đặc biệt là đối với các trò chơi chiến thuật như **Anime Vanguard** trên nền tảng Roblox. Trò chơi này yêu cầu người chơi phải dành nhiều thời gian để tham gia các thử thách, thu thập tài nguyên và nâng cấp nhân vật. Tuy nhiên, không phải ai cũng có đủ thời gian hoặc kiên nhẫn để thực hiện các nhiệm vụ này một cách thủ công.

Đối tượng người chơi chủ yếu là những người bận rộn, không có nhiều thời gian để cày cuốc, hoặc đơn giản là muốn tối ưu hóa trải nghiệm chơi game mà không phải can thiệp quá nhiều. Họ cần một giải pháp tự động hóa, giúp tiết kiệm thời gian và công sức, đồng thời vẫn đạt được hiệu quả cao trong trò chơi.

## Mục tiêu và phạm vi dự án

### Mục tiêu

Dự án này hướng đến việc phát triển một hệ thống tự động hóa toàn diện cho game **Anime Vanguard**, nhằm:

* **Tiết kiệm thời gian**: Giúp người chơi hoàn thành các nhiệm vụ và thu thập tài nguyên mà không cần phải trực tiếp tham gia.
* **Tối ưu hóa tài nguyên**: Tự động hóa quá trình thu thập và sử dụng tài nguyên trong game, giúp người chơi đạt hiệu quả cao mà không tốn nhiều công sức.
* **Cải thiện trải nghiệm người chơi**: Mang đến một trải nghiệm chơi game mượt mà, liên tục và không gián đoạn, phù hợp với những người chơi bận rộn hoặc muốn thư giãn.

### Phạm vi

Phạm vi bao gồm: phần mềm chạy trên Windows với GUI trực quan, tích hợp với trò chơi Roblox để thực hiện các hành động trong game với các mục chính:

* + Phát triển hệ thống tự động tham gia các thử thách trong game.
  + Tự động hóa quá trình thu thập tài nguyên và nâng cấp nhân vật.
  + Cung cấp giao diện người dùng thân thiện và dễ sử dụng.
  + Đảm bảo tính ổn định và hiệu suất cao của hệ thống.

## Đối tượng người dùng

Đối tượng chính của hệ thống là:

* + **Người chơi game Anime Vanguard**: Những người yêu thích trò chơi nhưng không có nhiều thời gian để tham gia trực tiếp.
  + **Người chơi muốn tối ưu hóa trải nghiệm**: Những người muốn đạt được hiệu quả cao trong game mà không cần phải dành quá nhiều thời gian.
  + **Người chơi mới**: Những người mới bắt đầu chơi và cần một công cụ hỗ trợ để làm quen với trò chơi.

## Công nghệ sử dụng

Phần mềm được phát triển bằng **AutoHotkey v2** — ngôn ngữ lập trình mã nguồn mở chuyên dành cho tự động hóa trên AutoHotkey v2 cho phép thiết lập **phím tắt**, **macro**, tự động kích chuột và nhập phím, hoàn toàn phù hợp với mục tiêu tự động hóa nhiệm vụ trong game.

## Phương pháp phát triển và tổ chức nhóm

Dự án áp dụng phương pháp **Scrum**, tổ chức theo các Sprint 1–2 tuần để đảm bảo tính linh hoạt và tiến độ liên tục với các thành viên chính:

|  |  |  |  |
| --- | --- | --- | --- |
| Thành viên | MSSV | Vai trò | Mức độ hoàn thành công việc |
| Đào Mạnh Vương | 23010586 | Product Owner/Dev | 10 |
| Nguyễn Hữu Hiển | 23010521 | Scrum Master/Dev | 10 |
| Nguyễn Văn Chiến | 23010515 | Scrum Master/Dev | 10 |
| Trần Trọng Minh | 23010563 | Dev | 10 |

# Tổng quan về mô hình Scrum

## Giới thiệu Scrum

Scrum là một mô hình phát triển phần mềm thuộc nhóm phương pháp Agile, tập trung vào việc phát triển linh hoạt, thích ứng nhanh với thay đổi và liên tục cải tiến. Scrum thường được sử dụng trong các dự án có yêu cầu thay đổi thường xuyên hoặc không thể xác định đầy đủ ngay từ đầu, phù hợp với đặc điểm của các dự án game tool như tool tự động cho các dạng game tower defense như Anime Vanguard nơi tính năng, map, hay cơ chế game có thể thay đổi theo từng bản cập nhật. Scrum hướng đến việc chia nhỏ quá trình phát triển thành các giai đoạn ngắn gọi là Sprint trong đó đội ngũ sẽ tập trung hoàn thành một phần chức năng có thể sử dụng hoặc kiểm thử được. Qua từng Sprint sản phẩm sẽ dần được hoàn thiện với sự phản hồi và cải tiến liên tục.

## Lý do chọn Scrum

Scrum được lựa chọn vì phù hợp với đặc điểm của nhóm và tính chất của dự án:

* + **Thời gian phát triển ngắn**: Scrum cho phép chia nhỏ công việc và có sản phẩm khả dụng sau từng Sprint.
  + **Game có tính chất update liên tục:** Game được cập nhập mỗi 2 tuần thêm event mới, thêm map mới, các unit và meta khác nhau.
  + **Đội nhóm chưa có nhiều kinh nghiệm làm việc nhóm**: Scrum giúp xác định rõ vai trò, quy trình và cách phối hợp giữa các thành viên.
  + **Dự án gồm nhiều tính năng nhỏ, độc lập**: Việc chia backlog và chọn task phù hợp cho từng Sprint giúp dễ quản lý và phát triển.
  + **Cần kiểm thử từng chức năng riêng biệt**: Scrum cho phép kiểm tra từng phần nhỏ (auto farm, detect map, webhook,…) ngay sau khi hoàn thành.
  + **Có thể thay đổi và cải tiến liên tục**: Khi người dùng hoặc tester có phản hồi, nhóm có thể linh hoạt cập nhật backlog và thay đổi kế hoạch phát triển.
  + **Khuyến khích học tập và cải tiến sau mỗi Sprint**: Qua Sprint Retrospective, nhóm có thể nhìn lại và nâng cao kỹ năng làm việc nhóm theo thời gian.

## Các vai trò chính

### Product Owner (PO)

Là người định hướng và xác định yêu cầu, tính năng cần có của sản phẩm. Product Owner là người làm việc trực tiếp với người dùng cuối hoặc thị trường để xây dựng và quản lý danh sách Product Backlog – những tính năng cần phát triển.

Trong dự án này, Đào Mạnh Vương giữ vai trò Product Owner. Vương là người đề xuất ý tưởng ban đầu và xác định các tính năng quan trọng như: auto challenge, nhận diện map, GUI thống kê, gửi webhook thông báo…

### Scrum Master

Là người đảm bảo nhóm phát triển làm việc đúng với quy trình Scrum, hỗ trợ loại bỏ các rào cản, đảm bảo tính liên kết và tiến độ của cả nhóm.

Trong dự án này, vai trò Scrum Master được chia sẻ giữa các thành viên:Đào Mạnh Vương, Nguyễn Hữu Hiển, Nguyễn Văn Chiến.

Các bạn thay phiên hỗ trợ nhóm trong việc lập kế hoạch Sprint, tổ chức Daily Scrum và theo dõi tiến độ hoàn thành các task.

### Development Team

|  |  |
| --- | --- |
| Thành viên | Nhiệm vụ chính |
| Đào Mạnh Vương | Viết script auto challenge, auto bounty, phân tích màn chơi, thiết kế GUI, hệ thống tự Reconnect, Viết script các màn chơi |
| Nguyễn Hữu Hiển | Viết script auto đặt và nâng cấp units, target priority, xử lí cập nhập thông tin lên GUI, Viết script các màn chơi |
| Nguyễn Văn Chiến | Xây dựng hệ thống thống kê và điều khiển, cải tiến hệ thống bấm, chức năng tìm vị trí chính xác, Viết script các màn chơi |
| Trần Trọng Minh | Hệ thống phát hiện map, màn hình, chụp ảnh màn hình, gửi discord Webhook. |

# Phân tích nghiệp vụ & người dùng

## Đối tượng người dùng

|  |  |
| --- | --- |
| **Tiêu chí** | **Mô tả** |
| Người chơi game anime vanguard | Là học sinh, sinh viên, nhân viên văn phòng nhưng đa số sẽ là học sinh và sinh viên |
| Tuổi | 12-25 chiếm đa số người chơi |
| Thời gian online | 2-3 tiếng, cuối tuần sẽ tăng lên từ 5–6 tiếng |
| Mục tiêu chơi game | Thu thập Gems, Gold Coins, Trait Reroll hoàn thành Bounties, tham gia event boss để nhận phần thưởng giá trị từ đó nâng cấp đội hình, khỏe hơn để tham gia các sự kiện như Tournament, Pvp nơi mà người chơi sẽ thi đấu với nhau để giành phần thưởng |
| Thói quen | Chơi xen kẽ giữa công việc/học tập, thường bỏ vê việc học, công việc để cày các sự kiện nhằm tối ưu tài nguyên game cung cấp |
| Khó khăn | Không có thời gian theo dõi liên tục, dễ bỏ lỡ thời điểm tham gia challenge, khó cạnh tranh. Đối với học sinh thường bị cấm chơi game quá lâu hoặc quá mải chơi game dẫn đến không tập trung học tập, kết quả giảm sút… |

## Phân tích hoạt động của người dùng

### Theo dõi và tham gia challenge định kỳ

* + Mỗi 30 phút, game mở “Trait Reroll Challenge, Stats Chip Challenge…”.
  + Người chơi phải canh thời gian, nhanh chóng tham gia trong vài giây để không lãng phí thời gian vì challenge chỉ bắt đầu tính thời gian chờ sau khi qua màn.
  + Lặp lại liên tục trong nhiều giờ để thu thập Gems và Gold Coins đặc biệt là Trait Reroll.

### Bounty Hunts

* Mỗi ngày có giới hạn lượt Bounty; sau khi hết phải chờ reset, chỉ reset sau khi đã hoàn thành hết số lượt
* Người chơi thường phải tham gia chơi liên tục khoảng 10 lần để không bỏ phí mất tài nguyên.
* Việc ngồi “cày” liên tục khiến người chơi mỏi mệt, ảnh hưởng đến sức khỏe và hiệu suất công việc/học tập.

### Tham gia sự kiện boss (Weekly Boss Event)

* Boss event ra mắt mỗi tuần, yêu cầu đạt mức sức mạnh nhất định.
* Người chơi phải mất từ 4-6 tiếng để farm đủ tài nguyên, đáp ứng đủ nhu cầu tối thiểu cho các tài nguyên thiết yếu trong boss shop.

### Quản lý nâng cấp nhân vật/unit

* Mỗi ván game sẽ phải có chiến thuật khác nhau, phải tập trung farm tiền và đặt các tháp dẫn đến phải tập trung liên tục mà không thể treo máy

## Vấn đề và rủi ro khi không sử dụng công cụ

### Thiếu hụt tài nguyên → yếu thế so với người chơi khác

* Farm thủ công chậm, thu được Gems/Gold hạn chế.
* Không đủ tài nguyên để nâng cấp unit và mua vật phẩm, dẫn đến lực chiến kém, khó vượt qua map cao cấp.

### Không đủ sức mạnh để tham gia sự kiện mới → tụt hậu

* + Boss Event hàng tuần, các map mới đòi hỏi sức mạnh tối thiểu; người chơi không đạt yêu cầu sẽ mất cơ hội nhận thưởng độc quyền.
  + Khi sự kiện đặc biệt (event meta) xuất hiện, họ không thể tham gia, ảnh hưởng vị thế trong cộng đồng game.

### Tốn nhiều thời gian và công sức khi “cày” thủ công

* + Cần ngồi canh liên tục, bấm chuột và phím, theo dõi timer, rất dễ mỏi tay và thiếu tập trung.
  + Thời gian dành cho công việc, học tập và sinh hoạt cá nhân bị ảnh hưởng nặng nề.

### Dùng phần mềm hack → rủi ro bị khóa tài khoản

* + Phần lớn công cụ không được kiểm duyệt dễ chứa mã độc hoặc trigger anti-cheat.
  + Một khi bị phát hiện sử dụng hack trái phép thì tài khoản có nguy cơ bị ban vĩnh viễn, mất hết tiến độ.

## Nhu cầu nghiệp vụ (Business Needs)

### Tự động hoá hoàn toàn việc tham gia challenge & bounty

* + Thay vì phải kiểm tra mỗi 30 phút, hệ thống tự động nhận biết và join đúng lúc.
  + Tiết kiệm thời gian, không bỏ lỡ lượt chơi.

### Tự động farm map & nâng cấp units

* + Cho phép cấu hình sẵn chiến thuật: ưu tiên farm wave 1–3, sau đó nâng DPS.
  + Hệ thống tự động đặt và nâng cấp theo thứ tự ưu tiên đã thiết lập, đảm bảo tối ưu tài nguyên.

### Theo dõi và báo cáo thống kê tức thì

* + GUI hiển thị realtime Gems, Gold, Trait Rerolls, Số Bounties hoàn thành, thời gian chạy.
  + Nhật ký hoạt động ghi lại mọi hành động, giúp người dùng dễ dàng đối chiếu và điều chỉnh và cũng dễ dàng cho việc debug khi có lỗi xảy ra.

### Thông báo kết quả qua Discord webhook

* + Gửi report sau mỗi phiên chơi: map, gems/gold thu được, số lần retry, lỗi phát sinh.
  + Giúp người dùng nắm được kết quả ngay cả khi không ngồi trước máy.

## Kết luận

Phân tích nghiệp vụ và người dùng cho thấy nhu cầu rõ ràng về một công cụ tự động hoá **toàn diện**, đảm bảo:

* + **Tiết kiệm thời gian**: loại bỏ thao tác thủ công lặp đi lặp lại.
  + **Tối ưu tài nguyên**: hoạch định và thực thi chiến thuật nâng cấp hiệu quả.
  + **Đảm bảo an toàn**: hạn chế rủi ro bị phát hiện và bị khóa tài khoản.
  + **Giám sát & báo cáo**: cung cấp thông tin chi tiết, minh bạch, hỗ trợ người chơi đưa ra quyết định.

Tất cả những yếu tố này sẽ là cơ sở để xây dựng **Product Backlog**, phân chia các Sprint và thiết kế các module chức năng trong phần tiếp theo của báo cáo.

# Phân tích yêu cầu hệ thống

## TỔNG QUAN HỆ THỐNG

### Mục Đích Phát Triển

Auto Challenge Tool là một công cụ hỗ trợ chơi game Roblox được thiết kế để tự động hóa toàn bộ quá trình chơi, đặc biệt trong các game dạng "tower defense" hoặc có yếu tố chiến thuật lặp đi lặp lại. Công cụ này được lập trình bằng ngôn ngữ AutoHotkey phiên bản 2 (AHK v2), nhằm mô phỏng các thao tác của người dùng và tự động hóa các hành vi lặp lại trong game như farm tài nguyên, hoàn thành nhiệm vụ và tối ưu hóa hiệu suất người chơi.

Mục đích chính của hệ thống là:

* **Tăng cường trải nghiệm người chơi** bằng cách giảm thiểu thao tác thủ công, tránh lãng phí thời gian và gây nhàm chán vì các nhiệm vụ lặp đi lặp lại.
* **Tối ưu hóa tài nguyên trong game** có thể tự động đi các thử thách để thu thập tài nguyên giúp người chơi, không lãng phí tài nguyên khi có việc, bận không thể tham gia thu thập.
* **Tự động hóa quá trình chơi game** có thể tự động làm mọi thứ, không cần thao tác thủ công từ đó tiết kiệm thời gian.
* **Cung cấp giao diện thân thiện** giúp người dùng có thể dễ dàng tùy chỉnh và điều khiển theo nhu cầu cá nhân.
* **Cơ chế sử dụng dễ dàng:** có thể dễ dàng sử dụng qua các thao tác đơn giản như bấm f2 để bắt đầu

### Phạm Vi Áp Dụng

Auto Challenge Tool hoạt động như một hệ thống hỗ trợ tổng hợp, bao gồm các module chức năng như:

* **Tự động hóa các hành động trong game** như chọn map, nhận nhiệm vụ, đặt và nâng cấp unit.
* **Theo dõi và thống kê dữ liệu tài nguyên** mà người chơi thu thập được trong thời gian thực.
* **Gửi thông báo trạng thái, lỗi hoặc thành tích** thông qua Discord Webhook.
* **Cung cấp giao diện đồ họa người dùng (GUI)** để kiểm soát macro, cấu hình thông số và hiển thị log hoạt động.
* **Hỗ trợ nhiều map và chế độ chơi**, có khả năng thích nghi với các bản cập nhật game.

## YÊU CẦU CHỨC NĂNG

### Giao Diện Thống Kê (GUI Stats Dashboard)

Auto Challenge Tool tích hợp một giao diện thống kê chuyên sâu, cung cấp dữ liệu toàn diện và trực quan về mọi hoạt động trong game, giúp người dùng dễ dàng giám sát tiến trình farm, tài nguyên thu được, và trạng thái hệ thống macro.

#### Hiển Thị Thời Gian Chạy Macro

* + Hệ thống sẽ tự động kích hoạt bộ đếm thời gian ngay khi người dùng khởi động macro.
  + Thời gian được hiển thị theo định dạng quốc tế chuẩn: Giờ:Phút:Giây (HH:MM:SS), cập nhật liên tục với độ trễ gần như bằng 0.
  + Đồng hồ thời gian thực giúp người dùng đo lường hiệu suất chạy macro, ước lượng thời gian hoàn thành một phiên chơi, hoặc lập kế hoạch thời gian farm.

#### Thống Kê Tài Nguyên

* + **Gems:** Thống kê tổng số đá quý người chơi đã thu thập được kể từ khi khởi chạy macro.
  + **Gold Coins:** Ghi nhận số lượng vàng tích lũy, bao gồm từ nhiệm vụ, chiến đấu hoặc phần thưởng map.
  + **Trait Rerolls:** Hiển thị số lượt người dùng đã thực hiện thay đổi trait hoặc kỹ năng của nhân vật/unit.
  + **Bounties Completed:** Tổng hợp số nhiệm vụ đặc biệt hoặc bounty đã hoàn thành để đánh giá hiệu quả và tiến độ.

#### Activity Log (Nhật ký hoạt động)

* + Hiển thị dạng dòng thời gian các sự kiện gần nhất: đặt unit, đổi map, hoàn thành nhiệm vụ, báo lỗi,...
  + Tự động cuộn xuống khi có bản ghi mới xuất hiện, giúp theo dõi dễ dàng mà không cần thao tác thủ công.
  + Lưu trữ toàn bộ nhật ký hoạt động vào file .log (kèm theo timestamp) để tiện kiểm tra, phân tích lại hoặc chia sẻ lỗi cho developer.

### Tự Động đi Challenge (Auto Challenge System)

Module này xử lý toàn bộ chuỗi hoạt động từ nhận, chọn lọc đến thực thi nhiệm vụ một cách hoàn toàn tự động, giúp người chơi tối đa hóa tài nguyên mà không cần can thiệp thủ công.

#### Quản Lý Nhiệm Vụ

* + Tự động kiểm tra danh sách nhiệm vụ có sẵn theo chu kỳ và nhận nhiệm vụ mới ngay khi có thể.
  + Hệ thống sử dụng thuật toán đánh giá độ khó (dựa trên mô tả nhiệm vụ) và gán nhãn (Challenge, Bounty, một số map đặc biệt).
  + Ưu tiên xử lý các nhiệm vụ có phần thưởng giá trị cao hoặc có tính giới hạn thời gian như challenge half hours.

#### Xử Lý Thử Thách

* + Phân tích logic map, đặc điểm kẻ địch boss, các map khó sẽ sử dụng kĩ năng đặc biệt…
  + Dựa trên từng map sẽ có chiến thuật khác nhau: chỉ đặt dps chính, đặt farm trước khi đặt dps, nâng cấp farm trước ở những map khó
  + Khi thất bại sẽ tự động từ bỏ map và đợi đến challenge tiếp theo thay vì mất thời gian vào một bản đồ quá khó.

### Tự Động Đặt Và Nâng Cấp Units

#### Chiến Thuật Đặt Unit

* Tự động xác định vị trí tối ưu cho trước để tiến hành phòng thủ
* Sử dụng tọa độ, công cụ di chuyển chuột chính xác để đặt tháp và phòng thủ
* Tùy từng map sẽ ưu tiên nâng dps hoặc nâng farm nhằm tối ưu tỉ lệ thắng và thời gian
* Nhận biết map khó để sử dụng các kĩ năng hoặc unit đặc biệt như: bắn toàn map, vô hiệu hóa kĩ năng…

#### Chiến Lược phân loại

* + Phân loại đơn vị thành nhóm: **DPS chính**, **Hỗ trợ**, **Farm**, **AoE**, **CC**.
  + Thực hiện nâng cấp theo cấu trúc ưu tiên đã cấu hình (ví dụ: nâng cấp DPS 3 lần trước khi nâng support).
  + Tự điều chỉnh thứ tự nâng cấp dựa trên map.

#### Auto Upgrade

* + Tự động nâng cấp farm units ở wave 1-3 để tối đa hóa lượng tài nguyên thu được về sau.
  + Khi tài nguyên đủ, ưu tiên nâng DPS hoặc những unit có khả năng gây sát thương diện rộng.
  + Hệ thống có logic kiểm tra dư tài nguyên để tránh upgrade sai mục tiêu hoặc nâng cấp vượt ngưỡng cần thiết.

### Gửi Thông Báo Qua Discord Webhook

#### Cấu Hình Webhook

* Cho phép người dùng nhập và lưu URL webhook Discord, đi kèm user ID nếu muốn gửi riêng cho một cá nhân.
* Thiết lập dạng thông báo cần nhận, ví dụ:
* Chỉ báo lỗi
* Chỉ khi hoàn thành challenge
* Thông báo theo chu kỳ (mỗi 30 phút)

#### Loại Thông Báo

* Gửi thông báo dạng embed đẹp mắt với thông tin:
  + Tên map, wave hoàn thành, thời gian hoàn thành
  + Tổng số gems/vàng kiếm được
  + Lỗi phát sinh và số lần retry
  + Thống kê cuối mỗi phiên chơi

#### Hệ Thống Macro Phím Tắt

|  |  |
| --- | --- |
| Phím Tắt | Chức Năng |
| F1 | Căn chỉnh lại vị trí cửa sổ Roblox để macro hoạt động chính xác theo tọa độ chuẩn |
| F2 | Kích hoạt toàn bộ hệ thống macro, bao gồm auto farm, auto đặt unit, nâng cấp |
| F3 | |  | | --- | |  |  |  | | --- | | Tắt hoàn toàn macro và dọn dẹp bộ nhớ tạm | |
| F4 | Tạm dừng hoặc tiếp tục macro, sử dụng trong trường hợp người dùng muốn can thiệp thủ công tạm thời |

## YÊU CẦU PHI CHỨC NĂNG

### Tính Ổn Định và Khả Năng Chịu Lỗi

* Sử dụng cơ chế bắt lỗi nâng cao: kiểm tra disconnect.
* Khi phát hiện lỗi, thực hiện reconnect và ghi vào logs, thông báo webhook đến người dùng.
* Log lỗi chi tiết với timestamp, loại lỗi, điều kiện phát sinh, để hỗ trợ debugging nhanh chóng.

### Giao Diện Người Dùng (UI/UX)

* Thiết kế theo mô hình Material UI hoặc Fluent Design.
* Sử dụng hiệu ứng visual nhẹ, chuyển cảnh mượt, màu sắc dịu mắt nhưng rõ ràng.
* Có xác nhận khi thao tác quan trọng (bắt đầu/dừng macro, xoá cấu hình).
* Tương thích màn hình độ phân giải cao (Full HD, 2K).

### Phản Hồi Nhanh và Hiệu Năng Cao

* Đảm bảo thời gian phản hồi nhanh cho các thao tác cơ bản.
* Áp dụng cơ chế ngắt, countinue khi đạt các điều kiện cụ thể.
* Giao diện luôn được cập nhật real-time mà không giật lag hoặc chậm trễ.

### Khả Năng Mở Rộng và Duy Trì

* Kiến trúc Module: từng thành phần (auto đặt unit, auto challenge, webhook, UI, pixel scan) hoạt động độc lập.
* Có khả năng hỗ trợ plugin mở rộng như thêm unit, tích hợp AI đặt chiến thuật,...
* Dễ dàng update và điều chỉnh khi có update, cập nhập map mới

## Kết luận

Phần phân tích yêu cầu hệ thống đã làm rõ toàn bộ nhu cầu chức năng và phi chức năng của công cụ tự động hóa Anime Vanguard trên nền tảng Roblox, được triển khai bằng AutoHotkey v2. Về chức năng, hệ thống cần đảm bảo tự động tham gia mọi loại thử thách (challenge, bounty), farm tài nguyên trên đa dạng bản đồ, tự động đặt và nâng cấp units theo chiến thuật tối ưu, đồng thời gửi báo cáo kết quả qua Discord webhook và hỗ trợ phím tắt điều khiển linh hoạt. Về phi chức năng, công cụ yêu cầu tính ổn định cao với khả năng tự phát hiện và xử lý lỗi (anti-disconnect, reconnect), giao diện GUI trực quan, phản hồi real-time, hiệu năng mượt mà, cùng kiến trúc module dễ mở rộng để ứng phó với các bản cập nhật game.

Việc xác định rõ ràng các đầu vào (cấu hình người dùng, tín hiệu màn hình), quy trình xử lý (lập lịch tự động, điều khiển chuột-phím, pixel scan, logging) và đầu ra (thống kê, nhật ký, webhook) tạo nền tảng vững chắc cho khâu thiết kế kiến trúc và triển khai. Những yêu cầu này không chỉ đáp ứng nhu cầu tiết kiệm thời gian và công sức cho người chơi, mà còn giảm thiểu rủi ro bị khóa tài khoản do sử dụng hack không kiểm soát, đồng thời nâng cao trải nghiệm người dùng thông qua báo cáo chi tiết và khả năng tùy chỉnh linh hoạt.

Kết quả của phần phân tích sẽ là cơ sở để xây dựng Product Backlog, phân chia Sprint hiệu quả theo mô hình Scrum, và thiết kế sơ đồ Use Case, Class, Sequence, Activity phù hợp. Trên tinh thần đó, các bước tiếp theo sẽ tập trung vào việc cụ thể hóa kiến trúc hệ thống, thiết kế GUI, và lập kế hoạch Sprint, để đảm bảo công cụ phát triển đúng tiến độ, chất lượng và đáp ứng đầy đủ yêu cầu đã đề ra.

# Scrum Artifact

## Product Backlog

### Phân Tích & Yêu Cầu Game

[Ưu tiên: Cao] Phân tích lối chơi và cơ chế game:

* Nghiên cứu hệ thống điều khiển nhân vật: Tìm hiểu cách game nhận lệnh từ người chơi, các thao tác di chuyển, tấn công, sử dụng kỹ năng.
* Phân tích cơ chế đặt units và nâng cấp: Hiểu rõ cách người chơi đặt các units (DPS, tank, support) trên bản đồ và quy trình nâng cấp để tối ưu sức mạnh.
* Xác định các loại nhiệm vụ và phần thưởng: Liệt kê các nhiệm vụ trong game như challenge, bounty, và phần thưởng tương ứng để tool có thể tự động xử lý.

[Ưu tiên: Cao] Nghiên cứu bản đồ và thử thách:

* Lập danh sách tất cả các map trong game: Thu thập thông tin về các bản đồ hiện có.
* Phân tích đặc điểm và yêu cầu của từng map: Mỗi map có đặc điểm riêng về địa hình, số lượng kẻ địch, yêu cầu chiến thuật.
* Xác định vị trí đặt units tối ưu: Tìm ra vị trí chiến lược để đặt units nhằm tối đa hóa hiệu quả phòng thủ.

[Ưu tiên: Cao] Phân tích hệ thống chống gian lận:

* Tìm hiểu cơ chế phát hiện auto: Nghiên cứu cách game phát hiện hành vi tự động hóa, tránh bị khóa tài khoản.
* Nghiên cứu giới hạn tốc độ thao tác: Xác định tốc độ thao tác tối đa để tool hoạt động an toàn.
* Xác định các hành vi có thể bị phát hiện: Liệt kê các hành vi như click quá nhanh, thao tác lặp lại không tự nhiên.

### Hệ Thống Cốt Lõi

[Ưu tiên: Quan trọng] Module điều khiển:

* Xây dựng hệ thống mô phỏng chuột: Phát triển chức năng tự động nhấp chuột chính xác.
* Phát triển mô phỏng bàn phím: Tự động gửi các phím tắt cần thiết trong game.
* Tối ưu tốc độ và độ chính xác: Đảm bảo thao tác diễn ra mượt mà, không gây lỗi.

[Ưu tiên: Quan trọng] Hệ thống nhận diện:

* Module quét màn hình: Thu thập hình ảnh màn hình để phân tích trạng thái game.
* Nhận diện các nút và menu: Xác định vị trí các nút bấm, menu để tương tác tự động.
* Phát hiện trạng thái game: Nhận biết khi nào game đang ở trạng thái chờ, chiến đấu, hoặc kết thúc.

### Tính Năng Tự Động

[Ưu tiên: Cao] Auto farm các bản đồ: Tự động chơi và hoàn thành các map để thu thập tài nguyên.

[Ưu tiên: Cao] Tự động đi challenge và bounty:

* Auto đi khi được reset: Tự động bắt đầu nhiệm vụ khi có thể.
* Tự động hoàn thành: Thực hiện đầy đủ các bước trong nhiệm vụ.
* Thu thập phần thưởng: Tự động nhận thưởng sau khi hoàn thành.

### Giao Diện Người Dùng

[Ưu tiên: Cao] Cửa sổ điều khiển chính:

* Nút bắt đầu/dừng: Cho phép người dùng khởi động hoặc dừng tool dễ dàng.
* Hiển thị trạng thái: Thông báo trạng thái hiện tại của tool (đang chạy, tạm dừng,...).
* Cài đặt cơ bản: Cho phép thiết lập các tham số đơn giản.

[Ưu tiên: Cao] Giao diện cài đặt:

* Cài đặt Discord webhook: Thiết lập kết nối gửi thông báo.
* Cài đặt private server: Hỗ trợ người dùng chọn server riêng.
* Tùy chọn nâng cao: Các thiết lập chuyên sâu cho người dùng có kinh nghiệm.

### Tính Năng Tích Hợp

[Ưu tiên: Cao] Tích hợp Discord webhook:

* Gửi thông báo kết quả: Báo cáo kết quả chơi tự động lên kênh Discord.
* Báo cáo lỗi: Thông báo lỗi phát sinh trong quá trình chạy tool.
* Theo dõi tài nguyên: Cập nhật tình trạng tài nguyên hiện có.

### Kiểm Thử

[Ưu tiên: Cao] Xây dựng bộ test

* Test các tính năng cơ bản
* Test các trường hợp đặc biệt
* Test hiệu năng
* Tiêu Chí Hoàn Thành
* Đạt yêu cầu về hiệu năng
* Không còn lỗi nghiêm trọng
* Discord webhook hoạt động tốt
* Đã xử lý các trường hợp lỗi

### Yếu Tố Rủi Ro

* Game cập nhật làm hỏng tính năng
* Bị phát hiện bởi anti-cheat
* Vấn đề về hiệu năng
* Thay đổi API Discord
* Bố cục map phức tạp
* Yêu cầu tài nguyên cao

## Các sprint

### Sprint 0 – Chuẩn bị & Khởi động dự án

Cài đặt công cụ phát triển (AutoHotkey v2, VSCode, Git, Discord webhook test server)

Tạo kho lưu trữ code (GitHub/Drive nội bộ)

Phân tích chi tiết yêu cầu từ trò chơi Anime Vanguard

Liệt kê và phân loại các thử thách/map/nhiệm vụ trong game

Xây dựng sơ đồ Use Case, xác định mô hình tương tác người dùng

### Sprint 1 – Xây dựng hệ thống điều khiển macro cơ bản

Viết script mô phỏng bàn phím & chuột bằng AutoHotkey

Căn chỉnh cửa sổ Roblox (F1)

Tạo GUI cơ bản: Bắt đầu (F2), Dừng (F3), Tạm dừng (F4)

Kiểm tra khả năng đọc tọa độ và pixel scan đơn giản

### Sprint 3 – Nhận diện nhiệm vụ và Auto Challenge

Quét màn hình, xác định nút nhận nhiệm vụ

Phân loại nhiệm vụ: bounty, challenge, stats reroll

Auto nhận nhiệm vụ mỗi 30 phút (đồng hồ đếm ngược)

Tự join map khi có challenge diễn ra

### Sprint 4 – Xây dựng GUI thống kê & logging nâng cao

Thiết kế GUI dạng dashboard: hiển thị Gems, Gold, Trait Reroll, số lần challenge

Tạo đồng hồ hiển thị thời gian macro

Tích hợp logging ra file .log (ghi thời gian, hành động, lỗi)

Tự động cuộn nhật ký trong GUI

### Sprint 5 – Tích hợp Discord Webhook thông báo kết quả

Cho phép nhập URL Webhook từ GUI

Tạo template báo cáo đẹp (embed): Gems, Gold, Map, Retry

Gửi thông báo khi hoàn thành map/challenge

Gửi cảnh báo khi phát sinh lỗi (disconnect, fail map)

### Sprint 6 – Tối ưu hiệu năng & xử lý lỗi (Reconnect, Anti-Disconnect)

Phát hiện disconnect, tự động reconnect

Tối ưu tốc độ thao tác chuột/phím

Tối ưu bộ nhớ và dọn log khi cần

Thêm kiểm tra timeout, fail-safe logic

### Sprint 7 – Kiểm thử toàn hệ thống & chuẩn bị release

Kiểm thử đơn vị các mô-đun: auto unit, challenge, GUI, webhook

Kiểm thử tích hợp toàn hệ thống trên nhiều độ phân giải

Tạo checklist QA: hoạt động đúng map, đúng logic nâng cấp

Chuẩn bị tài liệu hướng dẫn sử dụng

### Sprint 8 – Triển khai và thu thập phản hồi người dùng

Phát hành bản dùng thử (Beta)

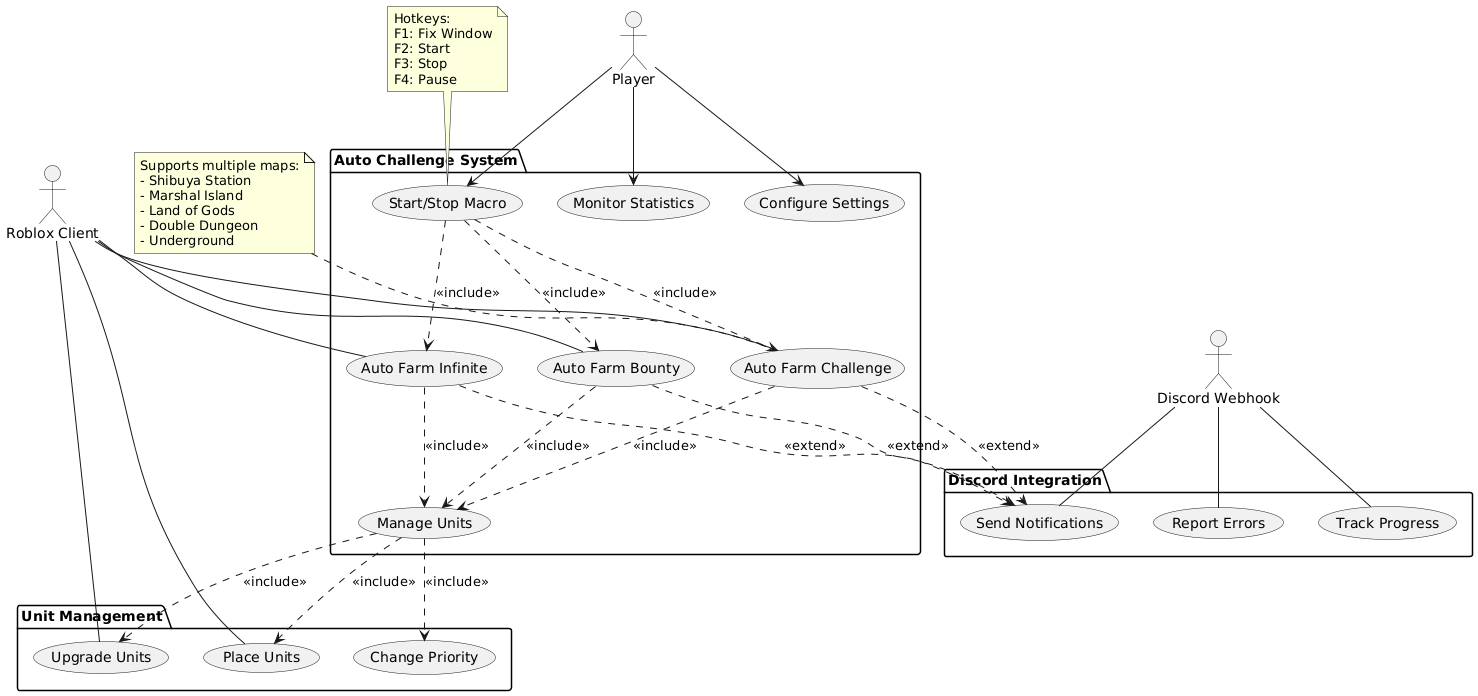
Thu thập ý kiến từ tester

Ghi nhận lỗi và đề xuất từ người dùng

Chuẩn bị danh sách backlog cho phiên bản nâng cấp

# Thiết kế hệ thống

## Sơ đồ use case



Hình . Sơ đồ use case của user

Use Case Diagram này thể hiện:

* Các Actor chính:
* Player (Người chơi)
* Discord Webhook (Hệ thống thông báo)
* Roblox Client (Game)

Các Use Case chính:

* Quản lý Macro (Start/Stop)
* Auto Farm các chế độ (Challenge/Bounty/Infinite)
* Quản lý Units
* Theo dõi thống kê
* Cấu hình hệ thống

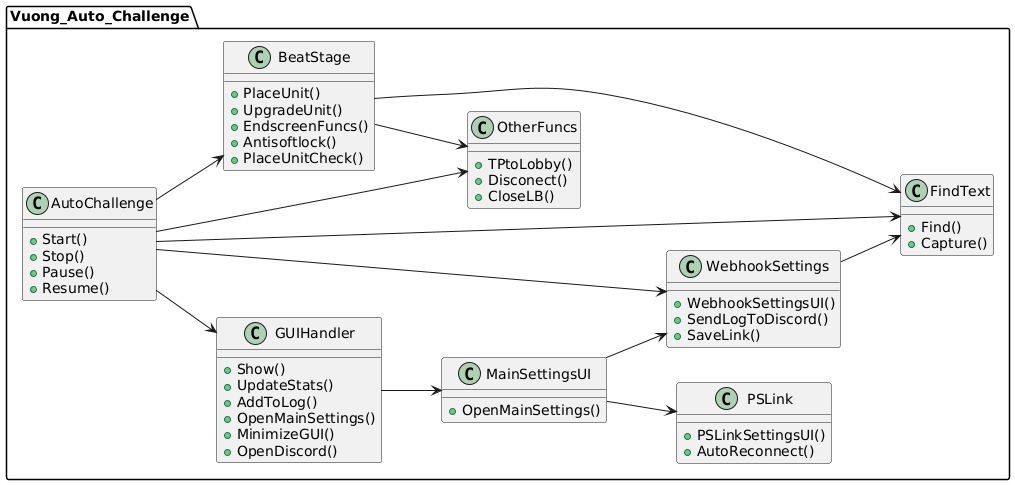
Các mối quan hệ:

* include: thể hiện các chức năng bắt buộc
* extend: thể hiện chức năng mở rộng
* association: thể hiện tương tác giữa actor và use case

Ghi chú:

* Phím tắt hệ thống
* Danh sách maps hỗ trợ

## Sơ đồ class



### Tổng quan kiến trúc

Sơ đồ lớp dưới đây mô tả các module chính, các chức năng tiêu biểu và mối quan hệ giữa các thành phần trong hệ thống tự động cày game Anime Vanguard.

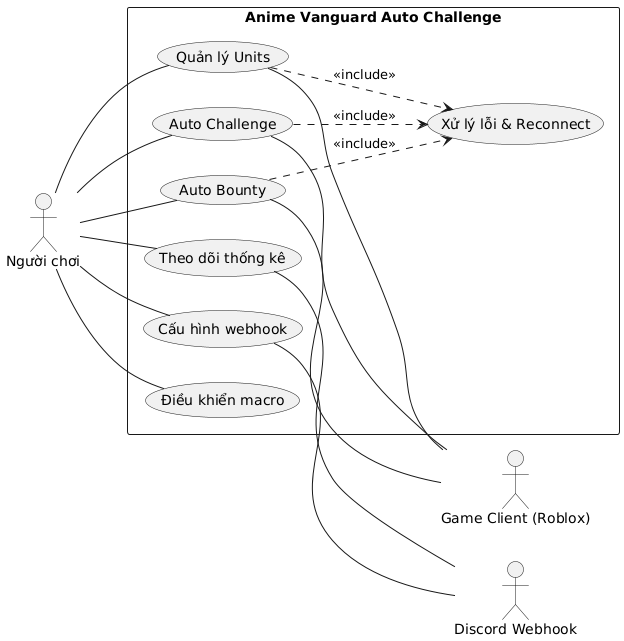
### Phân tích các thành phần chính

* **AutoChallenge**: Lớp trung tâm điều khiển toàn bộ quá trình tự động, bao gồm các hàm khởi động, dừng, tạm dừng và tiếp tục macro.
* **GUIHandler**: Quản lý giao diện người dùng, cập nhật thống kê, log, mở các cửa sổ cài đặt, thu nhỏ giao diện và liên kết Discord.
* **MainSettingsUI**: Giao diện cài đặt chính, cho phép truy cập các module cấu hình khác.
* **WebhookSettings**: Quản lý cấu hình webhook Discord, gửi log và lưu link webhook.
* **PSLink**: Quản lý chức năng tự động reconnect.
* **BeatStage**: Xử lý logic đặt và nâng cấp units, các hàm chống kẹt, kết thúc trận đấu.
* **FindText**: Nhận diện hình ảnh, text trên màn hình để xác định trạng thái game.
* **OtherFuncs**: Các hàm phụ trợ như dịch chuyển về lobby, disconnect, đóng bảng xếp hạng.

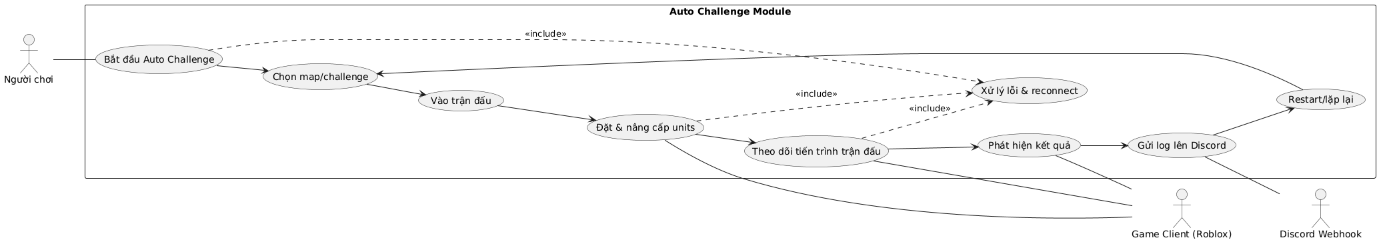
### Mối quan hệ giữa các lớp

* **AutoChallenge** sử dụng hầu hết các module còn lại để thực hiện các chức năng tự động.
* **GUIHandler** liên kết với **MainSettingsUI** để mở giao diện cài đặt.
* **MainSettingsUI** liên kết với **WebhookSettings** và **PSLink** để cấu hình webhook và reconnect.
* **BeatStage** sử dụng **FindText** để nhận diện trạng thái và **OtherFuncs** để xử lý các tình huống đặc biệt.
* **WebhookSettings** có thể sử dụng **FindText** để hỗ trợ gửi log theo trạng thái nhận diện.

## Sơ đồ hoạt động

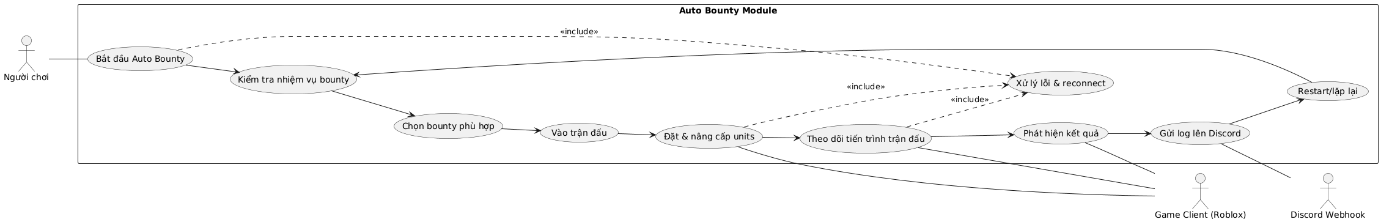


### Auto Challenge



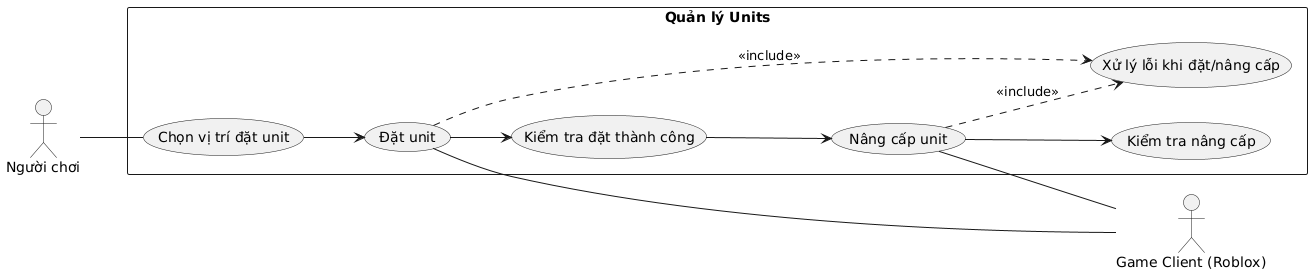
**Mô tả**: Tự động vào các thử thách, đặt units, điều khiển trận đấu và lặp lại nhiều trận.

### Auto Bounty



**Mô tả**: Tự động farm các nhiệm vụ bounty theo thời gian thực, hoàn thành nhanh, nhận phần thưởng.

### Quản lý Units



**Mô tả**: Chọn đơn vị, nâng cấp, định nghĩa đội hình cho farm.

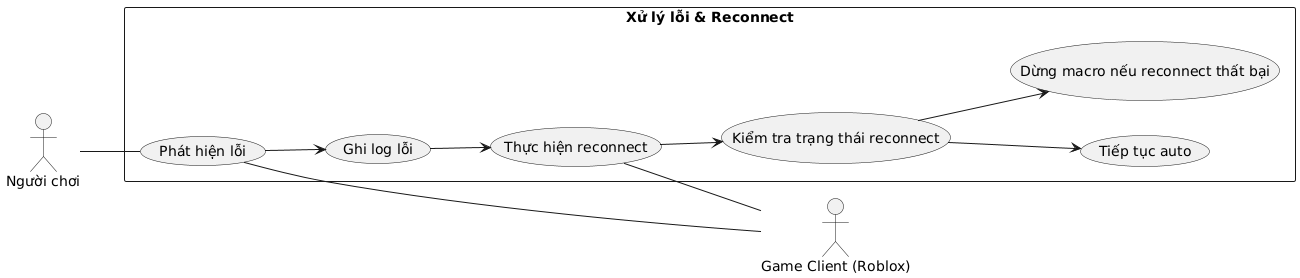
### Theo dõi thống kê

**Mô tả**: Ghi nhận hiệu suất farm (tổng số trận, thời gian, tỉ lệ thắng, tài nguyên thu được...) và gửi lên Discord.

### Cấu hình webhook

**Mô tả**: Cho phép người chơi nhập URL webhook Discord, cấu hình định dạng và nội dung gửi đi.

### Tự động kết nối lại

**Xử lý lỗi & Reconnect *(Chức năng phụ dùng lại bởi nhiều chức năng chính)***

**Mô tả**: Phát hiện các lỗi như disconnect, lag, fail load map, xử lý lại hành động hoặc tự động reconnect để tiếp tục quy trình farm.

## Sơ đồ Package

AutoChallenge/ => Thư mục gốc của ứng dụng AutoHotkey

|

|--------Lib => Chứa các thư viện, hàm dùng chung

| |

| |---------BeatStage.ahk => Xử lý các màn chơi tự động

| |---------FindSpot.ahk => Tìm vị trí trên màn hình

| |---------FindText.ahk => Nhận diện văn bản

| |---------Function.ahk => Các hàm tiện ích

| |---------GUI.ahk => Xử lý giao diện người dùng

| |---------MainSetting.ahk => Cài đặt chính của ứng dụng

| |---------PrivateServer.ahk => Kết nối server riêng

| |---------Var.ahk => Biến toàn cục

| |---------WebHookSetting.ahk=> Cài đặt webhook Discord

| |---------AHKv2-Gdip-master => Thư viện xử lý ảnh

| |---------Discord-Webhook-master => Thư viện gửi webhook Discord

|

|-------Images => Chứa các hình ảnh sử dụng trong app

|

|-------Settings => Chứa các file cấu hình, dữ liệu người dùng

| |

| |---------DiscordUSERID.txt

| |---------DoDailyChallenge.txt

| |---------PSLink.txt

|

|-------AutoChallenge.ahk => File chính chạy ứng dụng

## Sơ đồ tuần tự

### Chức năng auto challenge

Ảnh có chứa văn bản, biểu đồ, Song song, hàng

Nội dung do AI tạo ra có thể không chính xác.

#### Mô Tả Chức Năng

Chức năng Auto Challenge được thiết kế để tự động tham gia các màn Thử Thách (Challenge Mode) trong game theo chu kỳ định kỳ (mỗi 30 phút). Mục tiêu là tối ưu hóa thời gian chơi bằng cách tự động hóa toàn bộ quá trình kiểm tra, tham gia và hoàn thành thử thách, kể cả trong điều kiện người chơi vắng mặt.

#### Quy Trình Bắt Đầu

Khi người dùng nhấn phím F2, macro được kích hoạt thông qua hàm InitializeMacro(). Trong quá trình khởi tạo, hệ thống thực hiện các thao tác sau:

* **Kiểm tra cửa sổ Roblox**: Sử dụng hàm WinExist(RobloxWindow) để xác định xem cửa sổ Roblox có đang hoạt động hay không.
* **Cấu hình lại cửa sổ**: Nếu cửa sổ tồn tại, hệ thống sẽ tự động di chuyển và thay đổi kích thước cửa sổ Roblox về vị trí (0,0) với kích thước tiêu chuẩn 800x600 để đảm bảo khả năng nhận diện hình ảnh và tương tác chính xác.
* **Khởi tạo biến hệ thống**: Ghi nhận thời điểm bắt đầu macro. Khởi tạo các biến đếm, bộ đếm thời gian, và các biến đánh dấu để phục vụ việc theo dõi quá trình auto challenge.

#### Quy Trình Auto Challenge

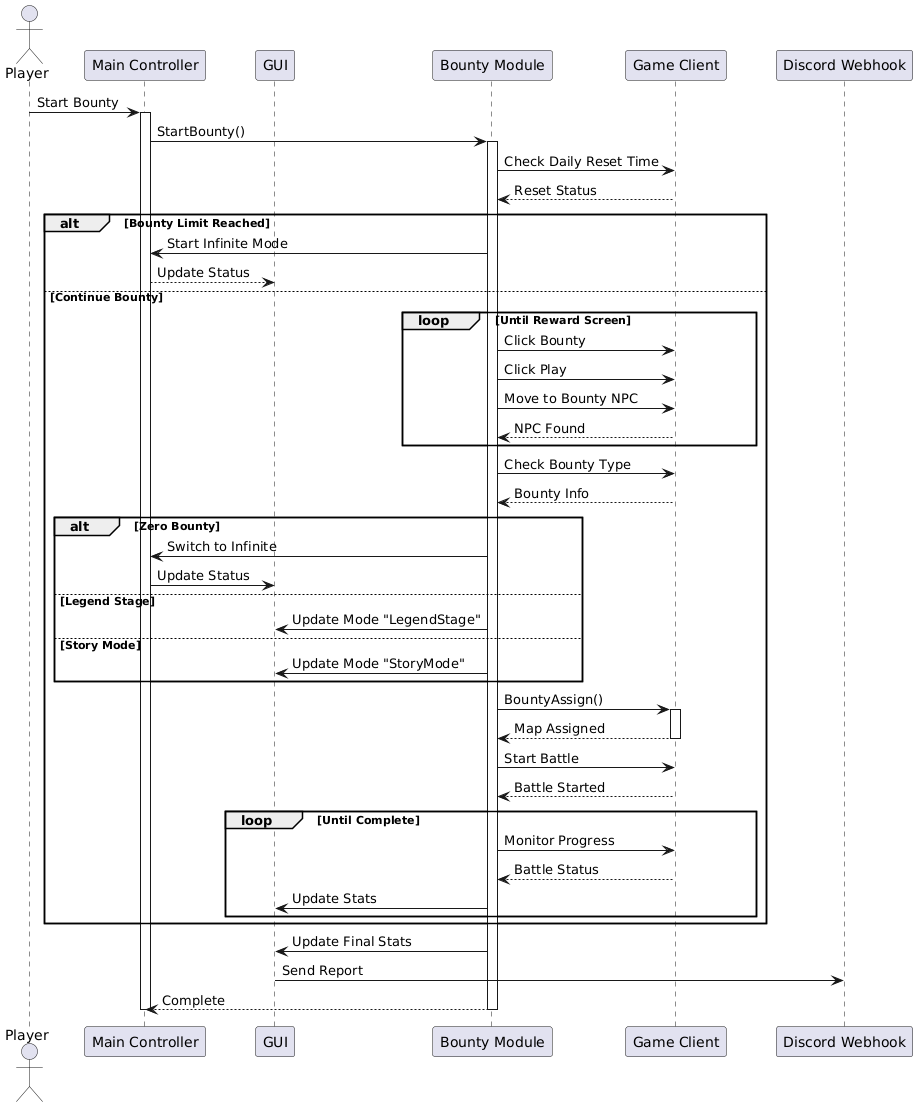
Quy trình chính của hệ thống là tự động tham gia các thử thách (challenge) trong game thông qua các bước sau:

* **Vào Menu Challenge:** Giao diện trong game được điều hướng tới phần "Challenge Menu" bằng cách mô phỏng click chuột vào vị trí tương ứng.
* **Kiểm Tra Loại Challenge:** Tùy thuộc vào cấu hình, hệ thống sẽ kiểm tra và lựa chọn một trong ba loại thử thách:
* **Trait Reroll Challenge**: Lựa chọn thử thách thay đổi trait (tính năng phụ), nếu được cấu hình bật.
* **Regular Challenge**: Sử dụng để kiểm tra xem đã hoàn thành challenge hay chưa nếu rồi thì bắt đầu bounty chưa thì sẽ bắt đầu chinh phục challenge.

#### Xác Định Map Challenge

Sau khi xác định loại challenge, hệ thống gọi hàm ChallengeAssign() để xác định chính xác bản đồ thử thách. Các map có thể bao gồm: Shibuya Station, Marshal Island, Land of Gods, Double Dungeon, Under Ground, Shibuya Aftermath, Golden Castle, Kuinshi Palace, Sand Village, Spirit Society, Edge of Heaven. Mỗi map có thể yêu cầu chiến thuật khác nhau (sẽ được xử lý trong phần tiếp theo).

### Chức năng auto bounty



#### Mô Tả Chức Năng

Chức năng **Auto Bounty** được thiết kế để tự động thực hiện các nhiệm vụ săn tiền thưởng (bounty) trong game, giúp người chơi tiết kiệm thời gian và tối ưu hóa tiến độ farm. Hệ thống sẽ tự động nhận nhiệm vụ, dịch chuyển đến khu vực mục tiêu, đánh bại kẻ thù cần thiết, và lặp lại quy trình khi hoàn thành hoặc thất bại.

#### Kiểm Tra Điều Kiện

* Khi người dùng kích hoạt Auto Bounty, hàm StartBounty() được gọi.
* Hệ thống kiểm tra giới hạn daily bounty bằng biến BountyCheck.
* Nếu đã đạt giới hạn, tự động chuyển sang chế độ Infinite Mode.

#### Quy Trình Vào Menu Bounty

* **Các bước thao tác trong game**: 1.Click vào Areas 2.Click vào nút Play. 3.Di chuyển đến vị trí NPC Bounty 4.Tương tác với NPC bằng phím E.
* **Xử Lý Loại Bounty:** Phân loại dựa trên FindText: gồm LegendStage và StoryModevà phân loại bản đồ được xử lý cũng dựa trên FindText
* **Xử Lý Trong Game:** Điều kiện bắt đầu trận đấu: FindText(..., BossStart), cập nhật tiến trình và trạng thái qua Monitor Progress, kết quả được gửi đến GUI và Discord Webhook.
* **Giám Sát Và Báo Cáo:** Tích hợp báo cáo: GUI cập nhật loại map và trạng thái trận đấu và Discord Webhook gửi logs.

### Chức năng findspot

Ảnh có chứa văn bản, ảnh chụp màn hình, biểu đồ, Song song

Nội dung do AI tạo ra có thể không chính xác.

#### Mô Tả Chức Năng

Chức năng **FindSpot** được thiết kế để tự động hóa quá trình đặt unit vào vị trí thích hợp và khởi động trận chiến trong game. Chức năng này bao gồm các bước xử lý camera, kiểm tra vị trí có hợp lệ hay không, điều hướng đơn vị và bắt đầu trận đánh nếu điều kiện được thỏa mãn.

Quy trình đảm bảo vị trí unit hợp lệ trước khi bước vào combat, đồng thời có khả năng phục hồi trong trường hợp xảy ra lỗi như đặt sai vị trí hoặc góc nhìn không phù hợp.

#### Quy Trình Xử Lý

Khởi Tạo Quá Trình

* Người chơi kích hoạt macro bằng phím F2
* Hệ thống bắt đầu thực hiện module FindSpot()
* Chọn unit Speedwagon từ slot 4

Vòng Lặp Tìm Vị Trí

* Điều chỉnh góc camera liên tục
* Xoay camera để có góc nhìn tối ưu
* Kiểm tra vị trí hiện tại xem có hợp lệ để đặt unit không
* Nếu không hợp lệ, hệ thống teleport về spawn, reset camera, và lặp lại kiểm tra

Xử Lý Kết Quả

* Khi phát hiện vị trí hợp lệ, tiến hành:
* Gọi BeatStage() để bắt đầu combat
* Cập nhật registry với tọa độ unit
* Tinh chỉnh lại góc nhìn để tối ưu quan sát trong trận

### Chức năng BeatStage

Ảnh có chứa văn bản, biểu đồ, ảnh chụp màn hình, Song song

Nội dung do AI tạo ra có thể không chính xác.

#### Mô Tả Chức Năng

Chức năng BeatStage chịu trách nhiệm tự động triển khai quá trình chiến đấu trong game. Quá trình này bao gồm việc kiểm tra bản đồ hiện tại, triển khai các đơn vị farm và combat, theo dõi kết quả trận đấu và gửi thông báo sau khi hoàn thành.

#### Quy Trình Xử Lý

**Khởi Tạo:**

* Player kích hoạt quá trình thông qua StartChallenge().
* Main Controller gọi module BeatStage.

**Lấy Thông Tin Map:**

* Module BeatStage truy vấn thông tin bản đồ hiện tại từ Game Client.

**Triển Khai Unit:**

* Đặt các đơn vị farm ở vị trí phù hợp.
* Thực hiện kiểm tra việc đặt unit (PlaceUnitCheck()).
* Đặt các đơn vị gây sát thương (damage units) để chuẩn bị combat.
* Kiểm tra xác nhận các đơn vị được đặt đúng vị trí.

**Xử Lý Nâng Cấp (nếu cần):**

* Nâng cấp farm: Được thực hiện nếu lượng tiền yêu cầu farm cao hơn mức mặc định.
* Thay đổi ưu tiên tấn công: Áp dụng nếu vị trí unit cần thay đổi chiến thuật.
* Nâng cấp đơn vị gây sát thương: Khi cần tăng sức mạnh combat.

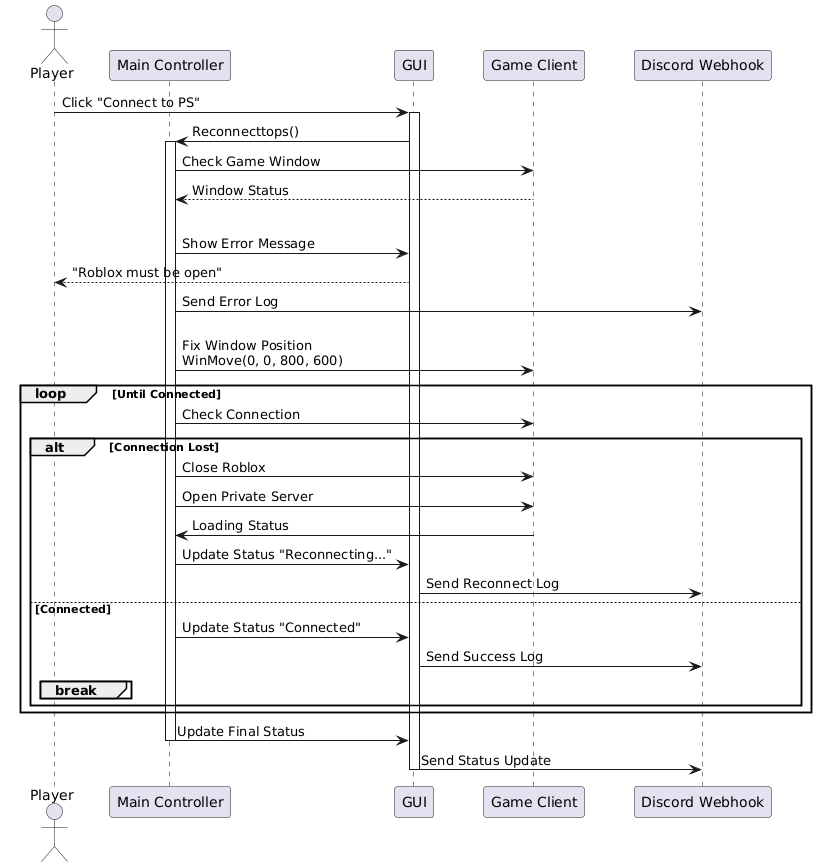
**Theo Dõi Trận Đấu:**

* Module liên tục kiểm tra trạng thái trận đấu (Monitor Battle Result) từ Game Client.
* Nếu trận đấu hoàn tất:
* Gọi các hàm kết thúc trận (Endscreen Tasks).
* Thu thập phần thưởng.
* Gửi kết quả trận đấu lên Discord qua webhook.

**Kết Thúc:**

* Trả lại quyền điều khiển cho Main Controller.
* Quay lại sảnh (lobby) để chuẩn bị cho lần chạy tiếp theo.

### Chức năng Reconnect



#### Mô Tả Chi Tiết

Chức năng Reconnect được thiết kế để đảm bảo việc kết nối lại với PrivateServerRoblox một cách tự động, khi phát hiện tình trạng mất kết nối hoặc cửa sổ game gặp lỗi. Hệ thống tự kiểm tra tình trạng cửa sổ game, xử lý lỗi kết nối, cập nhật trạng thái cho người dùng thông qua GUI và gửi thông báo qua Discord webhook.

**Khởi Tạo Reconnect**

* Người chơi thực hiện lệnh kết nối lại qua GUI (nút **"Connect to PS"**)
* Hệ thống bắt đầu quy trình Reconnecttops() để xử lý tình huống mất kết nối

**Xử Lý Kết Nối**

* Kiểm tra cửa sổ **Roblox** có đang hoạt động không
* Nếu không phát hiện, hiển thị lỗi và gửi log Discord
* Nếu có, điều chỉnh lại kích thước và vị trí cửa sổ để đảm bảo hiển thị chuẩn
* Bắt đầu vòng lặp kiểm tra kết nối

**Cập Nhật GUI**

* Hiển thị trạng thái hiện tại: "Reconnecting...", "Connected", hoặc "Error"
* Tự động cập nhật khi có thay đổi trạng thái kết nối

**Thông Báo Discord**

* Gửi log ngay khi quá trình reconnect bắt đầu
* Gửi thông báo kết quả cuối cùng: thành công hoặc thất bại
* Ghi nhận thông tin phục vụ theo dõi và debug

# Triển khai và Kiểm thử

## Cài đặt môi trường

### Cài đặt AutoHotKey v2

* Truy cập trang chính thức: https://www.autohotkey.com/v2/
* Nhấn Download
* Trong mục tải xuống, chọn AutoHotkey v2.0, sau đó chọn Installer (.exe) để cài đặt.

### Cài đặt Roblox

* Shift Lock switch: Tắt
* Camera Mode và Movement Mode: Mặc định
* FPS tối đa: 60
* Ngôn ngữ trải nghiệm: Tiếng Anh
* Độ nhạy camera: 0.15 (tuỳ chọn)
* Chế độ đồ họa: Thủ công, mức thấp nhất (1)
* Thống kê hiệu suất / Voice Chat / Tai nghe: Tắt (khuyến nghị)

### Cài đặt trong game Anime Vanguard

* Tự động bỏ qua đợt sóng: Bật
* Chi tiết thấp, tắt hiệu ứng, tắt chỉ số sát thương, tắt tên kẻ địch: Bật
* Tỷ lệ UI (UI Scale): 1
* Chọn đơn vị khi đặt: Bật
* Hoạt ảnh trang trí: Mặc định

### Cài đặt Window

* HDR: Tắt
* Độ phân giải: 1920×1080 ở 100% scale
* Ngôn ngữ hệ thống: Tiếng Anh (Hoa Kỳ)
* Bàn phím: QWERTY

## Cách chạy Tool

### Cài đặt tool

* Truy cập repo: [vuong200514/KTPM](https://github.com/vuong200514/KTPM) tải xuống dưới dạng tệp zip
* Sử dụng winrar hoặc 7zip giải nén tệp
* Mở file autochallenge.ahk để bắt đầu sử dụng

### Cấu hình Webhook gửi thông báo qua Discord

**Bước 1: Mở giao diện cấu hình Webhook**

* Chạy script VuongMacro.ahk
* Nhấn nút Settings → Webhook Settings

**Bước 2: Tạo Webhook trên Discord**

* Mở Discord → Chọn server → Chuột phải vào kênh → Edit Channel
* Vào tab Integrations → Webhooks
* Nhấn New Webhook, đặt tên → Copy Webhook URL

**Bước 3: Dán vào GUI**

* Dán Webhook URL vào ô tương ứng trong GUI

**Bước 4: Lưu cấu hình**

* Nhấn Save Settings, script sẽ lưu URL

### Tổ hợp phím chức năng

|  |  |
| --- | --- |
| **Phím tắt** | Tác dụng |
| F1 | Thay đổi kích thước Roblox |
| F2 | |  | | --- | |  |  |  | | --- | | Khởi động Tool | |
| F3 | Dừng Tool |
| F4 | Tạm dừng tool |

## Một số hình ảnh

### Hình UI của tool



### Hình Discord Webhook thông báo kết quả

Ảnh có chứa văn bản, ảnh chụp màn hình, Phần mềm đa phương tiện, phần mềm

Nội dung do AI tạo ra có thể không chính xác.

# Đánh giá dự án

## Ưu điểm

* Giảm công sức thao tác: Công cụ đã hỗ trợ người chơi loại bỏ hầu hết thao tác thủ công, đặc biệt là các nhiệm vụ lặp đi lặp lại như challenge và bounty, giúp tiết kiệm đáng kể thời gian và công sức.
* Tự động hóa toàn diện: Hệ thống đảm nhận đầy đủ các khâu từ tham gia nhiệm vụ, đặt và nâng cấp unit, thống kê tài nguyên cho đến gửi thông báo kết quả qua Discord webhook.
* Dễ sử dụng, cấu hình đơn giản: Giao diện người dùng trực quan, tích hợp phím tắt điều khiển, phù hợp cho cả người mới tiếp cận công cụ hỗ trợ.
* Kiến trúc module rõ ràng: Các chức năng được tách riêng theo từng module (GUI, AutoChallenge, AutoFarm, Webhook…), giúp dễ dàng bảo trì và mở rộng.
* Khả năng tương thích và ổn định: Công cụ hoạt động tốt trên nhiều độ phân giải màn hình và được thử nghiệm trên nhiều loại bản đồ, chế độ chơi khác nhau.

## Nhược điểm

* Phụ thuộc vào bố cục giao diện game: Vì sử dụng pixel detection và FindText nên nếu giao diện game thay đổi (màu sắc, vị trí nút…) thì công cụ có thể gặp lỗi hoặc cần chỉnh sửa lại.
* Chưa ứng dụng AI: Việc nhận diện các nút, map và đối tượng trong game vẫn còn đơn giản, dễ bị sai sót trong tình huống phức tạp.
* Chưa có khả năng tự học chiến thuật: Các quy tắc đặt và nâng cấp unit đều được cố định sẵn, chưa có khả năng thích nghi với chiến thuật mới hoặc tự tối ưu.

## Hướng phát triển

* Nâng cấp nhận diện hình ảnh bằng OpenCV hoặc tích hợp AI để tăng độ chính xác, giảm phụ thuộc vào pixel cố định.
* Cải tiến giao diện người dùng theo hướng chuyên nghiệp hơn, hỗ trợ tùy biến cao và hiển thị biểu đồ thống kê trực quan.
* Cho phép lưu nhiều cấu hình sử dụng cho từng bản đồ hoặc chế độ chơi khác nhau.

Kết luận

Dự án phát triển công cụ tự động hóa cho game **Anime Vanguard** trên nền tảng **Roblox** đã gặt hái nhiều kết quả tích cực, hoàn thành mục tiêu đề ra là tiết kiệm thời gian, tối ưu tài nguyên và nâng cao trải nghiệm người chơi. Bằng cách áp dụng phương pháp **Scrum**, nhóm đã xây dựng thành công các tính năng quan trọng như tự động tham gia challenge, farm tài nguyên, đặt và nâng cấp unit, cùng với giao diện người dùng thân thiện và hệ thống thông báo qua **Discord webhook**. Công cụ này giúp người chơi tiết kiệm công sức nhờ khả năng tự động hóa hiệu quả, tối ưu quá trình thu thập và sử dụng tài nguyên, đồng thời cung cấp thông tin thời gian thực thông qua GUI dễ sử dụng. Ngoài ra, tính năng gửi báo cáo qua Discord hỗ trợ người dùng theo dõi tiến trình từ xa, còn cơ chế tự động khắc phục lỗi và reconnect giúp nâng cao tính ổn định. Tuy nhiên, công cụ vẫn đối mặt với thách thức do game liên tục cập nhật và nguy cơ bị hệ thống chống gian lận phát hiện, đòi hỏi phải kiểm tra và nâng cấp thường xuyên. Nhìn chung, công cụ tự động hóa này không chỉ đáp ứng tốt nhu cầu của người chơi mà còn mở ra hướng phát triển mới cho các ứng dụng hỗ trợ game, tạo tiền đề cho các cải tiến trong tương lai và khẳng định vai trò là trợ thủ đắc lực cho cộng đồng game thủ Anime Vanguard.

Tài liệu tham khảo

**Coursera** - Cung cấp các khóa học Laravel: [Tìm khóa học trên Coursera](https://www.coursera.org)

AHK LIB – Cung cấp các function: [AHK Wiki](https://autohotkey.wiki/libraries)

AutoHotkey Foundation. (2023):  [*AutoHotkey v2 Documentation*.](https://www.autohotkey.com/docs/v2/)

AutoHotkey Community. (2022): [*FindText Library for AHK v2*.](https://github.com/AutoHotkey/AutoHotkeyv2)

Discord Developer Portal. (2023): [*Webhooks Documentation*.](https://discord.com/developers/docs/resources/webhook)