**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**TRƯỜNG KỸ THUẬT & CÔNG NGHỆ**



**ISO 9001:2015**

**ĐINH TẤN MÃI**

**XÂY DỰNG HỆ THỐNG THEO DÕI DỰ ÁN**

**PHẦN MỀM LÀM VIỆC NHÓM CỦA SINH VIÊN QUA GITHUB ACTION**

**KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

**Vĩnh Long, tháng…..năm…….**

**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**TRƯỜNG KỸ THUẬT & CÔNG NGHỆ**



**ISO 9001:2015**

**XÂY DỰNG HỆ THỐNG THEO DÕI DỰ ÁN**

**PHẦN MỀM LÀM VIỆC NHÓM CỦA SINH VIÊN QUA GITHUB ACTION**

**KHÓA LUẬN TỐT NGHIỆP**

**NGÀNH CÔNG NGHỆ THÔNG TIN**

Giảng viên hướng dẫn: **TS NGUYỄN BẢO ÂN**

Sinh viên thực hiện: **ĐINH TẤN MÃI**

Mã số sinh viên: **110121063**

Lớp: **DA21TTB**

Khoá: **2021**

**Vĩnh Long, tháng…..năm…….**

LỜI MỞ ĐẦU

Trong bối cảnh học tập và phát triển phần mềm hiện đại, việc làm việc nhóm và sử dụng các công cụ phát triển như GitHub ngày càng trở nên phổ biến trong môi trường đào tạo ngành Công nghệ thông tin. Tuy nhiên, một thách thức đặt ra là làm thế nào để giảng viên có thể theo dõi hiệu quả quá trình làm việc nhóm, đánh giá mức độ đóng góp của từng sinh viên cũng như kiểm tra chất lượng mã nguồn một cách khách quan và tự động.

GitHub Actions, GitHub API và Webhook là những công cụ mạnh mẽ hỗ trợ quá trình kiểm tra và đánh giá mã nguồn, được GitHub cung cấp miễn phí với khả năng tích hợp linh hoạt. Khi kết hợp với quy trình CI/CD, chúng tạo thành một hệ thống tự động hoá toàn diện từ nộp bài đến phân tích chất lượng và thống kê đóng góp. Việc ứng dụng các công cụ này không chỉ giúp nâng cao chất lượng đào tạo mà còn giúp sinh viên làm quen với quy trình phát triển phần mềm chuyên nghiệp ngay từ khi còn trên ghế nhà trường.

Đề tài này hướng tới việc xây dựng một hệ thống hỗ trợ giảng viên và sinh viên theo dõi tiến độ dự án phần mềm làm việc nhóm thông qua GitHub. Hệ thống cho phép sinh viên đăng nhập bằng GitHub OAuth, nộp bài qua repository riêng, tự động đánh giá chất lượng mã thông qua GitHub Actions và SonarCloud, đồng thời thống kê mức độ đóng góp của từng thành viên nhóm. Đây là giải pháp thiết thực nhằm nâng cao hiệu quả học tập, giảng dạy và đánh giá trong các môn học liên quan đến phát triển phần mềm.

LỜI CẢM ƠN

Em xin bày tỏ lòng biết ơn chân thành và sâu sắc nhất đến Thầy – Tiến sĩ Nguyễn Bảo Ân, Giảng viên Khoa Kỹ thuật & Công nghệ, Trường Đại học Trà Vinh , người đã trực tiếp hướng dẫn em trong suốt quá trình thực hiện đồ án tốt nghiệp.

Trong suốt thời gian thực hiện đề tài, Thầy không chỉ là người định hướng về mặt chuyên môn mà còn là người luôn đồng hành, động viên và hỗ trợ em vượt qua những khó khăn, vướng mắc cả trong học thuật lẫn trong quá trình triển khai thực tế. Thầy đã tận tình góp ý, đưa ra những lời khuyên xác đáng, đồng thời luôn tạo điều kiện thuận lợi để em có thể hoàn thiện đề tài một cách tốt nhất.

Bằng kiến thức sâu rộng, tinh thần trách nhiệm cao và sự tận tụy với sinh viên, Thầy đã giúp em không chỉ hiểu rõ hơn về chuyên môn mà còn học hỏi thêm nhiều kỹ năng thực tiễn và cách tư duy logic trong giải quyết vấn đề. Những buổi trao đổi, phản biện và định hướng từ Thầy là những trải nghiệm quý báu, giúp em hoàn thiện hơn từng bước trong quá trình làm đồ án.

Em cảm thấy vô cùng may mắn và trân trọng khi được Thầy hướng dẫn. Những kiến thức và kinh nghiệm học được từ Thầy chắc chắn sẽ là hành trang quý giá để em áp dụng và phát triển trong chặng đường sắp tới.

Một lần nữa, em xin gửi đến Thầy lời cảm ơn chân thành, kính chúc Thầy luôn mạnh khỏe, công tác tốt và tiếp tục truyền cảm hứng, kiến thức cho nhiều thế hệ sinh viên tiếp theo.

Sinh viên thực hiện

**Đinh Tấn Mãi**

NHẬN XÉT

*(Của giảng viên hướng dẫn trong đồ án, khóa luận của sinh viên)*

**Giảng viên hướng dẫn**

*(Ký và ghi rõ họ tên)*

UBND TỈNH VĨNH LONG **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TRÀ VINH Độc lập – Tự do – Hạnh Phúc**

**BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP**

*(Của giảng viên hướng dẫn)*

Họ và tên sinh viên:……………………………….. MSSV:……………………

Ngành:…………………………………………….. Khóa:……………………..

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

……………………………………………………………………………………….……………………………………………………………………………………….………………………………………………………………………………………

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

Chức danh: ……………………… …..Học vị: …………………………………….

**NHẬN XÉT**

1. Nội dung đề tài:

1. Ưu điểm:

1. Khuyết điểm:

1. Điểm mới đề tài:

1. Giá trị thực trên đề tài:

7. Đề nghị sửa chữa bổ sung:

8. Đánh giá:

Vĩnh Long*, ngày…… tháng……năm 20…*

Giảng viên hướng dẫn

*(Ký & ghi rõ họ tên)*

NHẬN XÉT

*(Của giảng viên chấm trong đồ án, khóa luận của sinh viên)*

**Giảng viên hướng dẫn**

*(Ký và ghi rõ họ tên)*

UBND TỈNH VĨNH LONG **CỘNG HOÀ XÃ HỘI CHỦ NGHĨA VIỆT NAM**

**TRƯỜNG ĐẠI HỌC TRÀ VINH Độc lập – Tự do – Hạnh Phúc**

**BẢN NHẬN XÉT ĐỒ ÁN, KHÓA LUẬN TỐT NGHIỆP**

*(Của cán bộ chấm đồ án, khóa luận)*

Họ và tên người nhận xét: …………………………………………………………..

Chức danh:………………………………………….. Học vị:………………………

Chuyên ngành:……………………………………………………………………….

Cơ quan công tác:……………………………………………………………………

Họ và tên sinh viên:………………………………………………………………….

Tên đề tài đồ án, khóa luận tốt nghiệp:………………………………………………

……………………………………………………………………………………….……………………………………………………………………………………….……………………………………………………………………………………….

**I. Ý KIẾN NHẬN XÉT**

1. Nội dung:

1. Điểm mới các kết quả của đồ án, khóa luận:

1. Ứng dụng thực tế:

**II. CÁC VẤN ĐỀ CẦN LÀM RÕ**

(Các câu hỏi của giáo viên phản biện)

**III. KẾT LUẬN**

(Ghi rõ đồng ý hay không đồng ý cho bảo vệ đồ án khóa luận tốt nghiệp)

Vĩnh Long*, ngày …..tháng….. năm 20…*

Người nhận xét

*(Ký & ghi rõ họ tên)*

MỤC LỤC

[CHƯƠNG 1. ĐẶT VẤN ĐỀ 1](#_Toc205594666)

[1.1. Lý do chọn đề tài 1](#_Toc205594667)

[1.2. Mục tiêu 1](#_Toc205594668)

[1.3. Nội dung nghiên cứu 2](#_Toc205594669)

[1.4. Đối tượng và phạm vi nghiên 3](#_Toc205594670)

[1.5. Phương pháp nghiên cứu 4](#_Toc205594671)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 6](#_Toc205594672)

[2.1. Tổng quan về GitHub và làm việc nhóm qua GitHub 6](#_Toc205594673)

[2.1.1 Git và GitHub là gì? 6](#_Toc205594674)

[2.1.2 Các tính năng hỗ trợ làm việc nhóm 7](#_Toc205594675)

[2.1.3 Lợi ích khi làm việc nhóm qua GitHub 8](#_Toc205594676)

[2.1.4 Mô hình làm việc nhóm qua GitHub 9](#_Toc205594677)

[2.2. Tổng quan CI/CD 10](#_Toc205594678)

[2.2.1 Continuous Integration (CI) - Tích hợp Liên tục 10](#_Toc205594679)

[2.2.2 Continuous Delivery (CD) - Phân phối Liên tục 12](#_Toc205594680)

[2.2.3 Quy trình CI/CD 13](#_Toc205594681)

[2.3. GitHub Actions 18](#_Toc205594682)

[2.4. Tổng quan về GitHub API và Webhook 21](#_Toc205594683)

[2.4.1 GitHub API 21](#_Toc205594684)

[2.4.2 Github Webhook 23](#_Toc205594685)

[2.5. Công cụ đánh giá mã nguồn 25](#_Toc205594686)

[2.5.1 Mục tiêu đánh giá mã nguồn 25](#_Toc205594687)

[2.5.2 Phân loại công cụ đánh giá mã nguồn 26](#_Toc205594688)

[2.5.3 Quy trình đánh giá mã nguồn 26](#_Toc205594689)

[2.5.4 Kết hợp công cụ đánh giá mã nguồn với GitHub Actions 28](#_Toc205594690)

[2.6. Tổng quan về hệ thống đánh giá lập trình tự động 30](#_Toc205594691)

[2.7. Các công nghệ sử dụng trong hệ thống 30](#_Toc205594692)

[2.7.1 Quy trình tạo môn học, đề tài, nhóm 30](#_Toc205594693)

[2.7.2 Quy trình kết nối GitHub, nhận webhook 30](#_Toc205594694)

[2.7.3 Quy trình nộp bài và đánh giá mã nguồn tự động 31](#_Toc205594695)

[2.7.4 Quy trình hiển thị thống kê đóng góp 31](#_Toc205594696)

[2.8. Tích hợp AI vào quy trình đánh giá lập trình 32](#_Toc205594697)

[2.9. Các công nghệ xây dựng trang web 32](#_Toc205594698)

[2.9.1 Ngôn ngữ HTML 32](#_Toc205594699)

[2.9.2 Bảng định kiểu CSS 32](#_Toc205594700)

[2.9.3 Ngôn ngữ Javascript 32](#_Toc205594701)

[2.9.4 NextJs 32](#_Toc205594702)

[2.9.5 TailwindCSS 32](#_Toc205594703)

[2.9.6 Shadcn/ui 32](#_Toc205594704)

[2.9.7 TypeScript 32](#_Toc205594705)

[2.9.8 Node.js 32](#_Toc205594706)

[2.9.9 Express 32](#_Toc205594707)

[2.9.10 Socket.IO 32](#_Toc205594708)

[2.9.11 MySQL 32](#_Toc205594709)

[CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU 33](#_Toc205594710)

[3.1. Mô tả bài toán thực tế 33](#_Toc205594711)

[3.2. Phân tích yêu cầu người dùng 33](#_Toc205594712)

[3.3. Lựa chọn công nghệ 33](#_Toc205594713)

[3.4. Thiết kế kiến trúc hệ thống tổng thể 33](#_Toc205594714)

[3.5. Thiết kế cơ sở dữ liệu 34](#_Toc205594715)

[3.6. Thiết kế các luồng hoạt động chính của hệ thống 34](#_Toc205594716)

[3.6.1 Quy trình tạo môn học, đề tài, nhóm 34](#_Toc205594717)

[3.6.2 Quy trình kết nối GitHub, nhận webhook 35](#_Toc205594718)

[3.6.3 Quy trình nộp bài và đánh giá mã nguồn tự động 35](#_Toc205594719)

[3.6.4 Quy trình hiển thị thống kê đóng góp 35](#_Toc205594720)

[CHƯƠNG 4. Triển khai hệ thống 36](#_Toc205594721)

[4.1. Thiết lập môi trường phát triển 36](#_Toc205594722)

[4.2. Xây dựng backend với Express và Sequelize 36](#_Toc205594723)

[4.3. Xây dựng các API chức năng chính 36](#_Toc205594724)

[4.3.1 API người dùng và xác thực OAuth 36](#_Toc205594725)

[4.3.2 API môn học, đề tài, nhóm 37](#_Toc205594726)

[4.3.3 API webhook và xử lý commit từ GitHub 37](#_Toc205594727)

[4.3.4 API thống kê đóng góp 37](#_Toc205594728)

[4.4. Thiết kế và xây dựng giao diện bằng Next.js 38](#_Toc205594729)

[4.4.1 Giao diện giảng viên 38](#_Toc205594730)

[4.4.2 Giao diện sinh viên 38](#_Toc205594731)

[4.5. Thiết lập GitHub Actions và tích hợp SonarCloud 38](#_Toc205594732)

[4.6. Tích hợp toàn bộ hệ thống và xử lý dữ liệu thời gian thực 39](#_Toc205594733)

[CHƯƠNG 5. KẾT QUẢ VÀ ĐÁNH GIÁ 40](#_Toc205594734)

[5.1. Kết quả thử nghiệm với tài khoản sinh viên 40](#_Toc205594735)

[5.2. Kết quả thử nghiệm với tài khoản giảng viên 40](#_Toc205594736)

[5.3. Kiểm tra kết quả đánh giá mã nguồn qua GitHub Actions 40](#_Toc205594737)

[5.4. Đánh giá tính chính xác và hiệu quả thống kê đóng góp 40](#_Toc205594738)

[CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 42](#_Toc205594739)

[6.1. Kết luận 42](#_Toc205594740)

[6.2. Hạn chế 42](#_Toc205594741)

[6.3. Hướng phát triển trong tương lai 42](#_Toc205594742)

[PHỤ LỤC 43](#_Toc205594743)

[DANH MỤC TÀI LIỆU THAM KHẢO 44](#_Toc205594744)

DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH

[Ảnh 1.1 Placeholder.svg 2](#_Toc203916953)

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Ý nghĩa** |
| CI |  |
| CD |  |
| LLM |  |
| AI |  |
| API |  |

# 

# ĐẶT VẤN ĐỀ

## Lý do chọn đề tài

Trong bối cảnh chuyển đổi số và cuộc cách mạng công nghiệp 4.0 đang diễn ra mạnh mẽ, việc ứng dụng tự động hóa vào quy trình phát triển phần mềm ngày càng đóng vai trò quan trọng nhằm nâng cao hiệu suất, tính chính xác và chất lượng sản phẩm. Trong môi trường giáo dục đại học, đặc biệt là ngành Công nghệ thông tin, nhu cầu đánh giá bài nộp của sinh viên một cách khách quan, nhanh chóng và chính xác đang trở thành một yêu cầu thiết thực. Tuy nhiên, nhiều hoạt động đánh giá hiện nay vẫn còn thực hiện thủ công, dẫn đến tốn kém thời gian và công sức cho giảng viên, đồng thời thiếu công cụ phản hồi hiệu quả cho người học.

Từ thực tiễn đó, việc xây dựng một hệ thống hỗ trợ tự động theo dõi và đánh giá mã nguồn của sinh viên thông qua nền tảng GitHub và các công cụ kiểm tra chất lượng mã như GitHub Actions, SonarCloud là hướng tiếp cận phù hợp. Hệ thống không chỉ tự động hóa quy trình kiểm tra kỹ thuật mà còn giúp sinh viên làm quen với các công cụ CI/CD – vốn là tiêu chuẩn trong quy trình phát triển phần mềm hiện đại.

Ngoài ra, việc tích hợp các công cụ phân tích mã nguồn tự động giúp phát hiện sớm lỗi, đánh giá chất lượng code và theo dõi mức độ đóng góp của từng thành viên trong nhóm cũng sẽ góp phần tăng cường tính minh bạch, công bằng trong đánh giá, đồng thời nâng cao ý thức và kỹ năng làm việc nhóm của sinh viên.

Với những ý nghĩa thực tiễn và tính ứng dụng cao trong giảng dạy cũng như học tập, đề tài "Xây dựng hệ thống theo dõi và đánh giá tự động mã nguồn của sinh viên thông qua GitHub Actions" được lựa chọn nhằm góp phần cải thiện quy trình đào tạo, kiểm tra đánh giá trong lĩnh vực Công nghệ thông tin.

## Mục tiêu

Đề tài được thực hiện với các mục tiêu chính sau:

- Xây dựng hệ thống theo dõi và đánh giá mã nguồn tự động dành cho các dự án phần mềm nhóm của sinh viên, sử dụng GitHub làm nền tảng lưu trữ mã nguồn và công cụ CI/CD để kiểm tra tự động.

- Tích hợp GitHub Actions nhằm tự động hóa quá trình kiểm tra mã nguồn mỗi khi có commit hoặc pull request, bao gồm phân tích chất lượng mã, kiểm tra lỗi, và đánh giá các tiêu chí kỹ thuật.

- Sử dụng SonarCloud để phân tích chất lượng mã nguồn, từ đó cung cấp các chỉ số như mức độ phức tạp, số lượng lỗi, độ bao phủ kiểm thử và các cảnh báo liên quan đến bảo mật hoặc maintainability.

- Khai thác GitHub API để thu thập dữ liệu hoạt động, bao gồm số lần commit, số dòng mã thêm/bớt, và đánh giá mức độ đóng góp của từng thành viên trong nhóm, phục vụ cho việc chấm điểm công bằng và minh bạch.

- Xây dựng giao diện quản trị và dashboard trực quan, cho phép giảng viên dễ dàng theo dõi tiến độ, chất lượng mã nguồn, lịch sử nộp bài và đóng góp cá nhân trong từng dự án nhóm.

- Hỗ trợ quy trình đào tạo hiện đại, giúp sinh viên tiếp cận các công cụ và quy trình phát triển phần mềm chuyên nghiệp như CI/CD, phân tích tĩnh, kiểm thử tự động, qua đó nâng cao kỹ năng và tư duy lập trình.

Thông qua các mục tiêu trên, hệ thống nhằm hướng tới việc nâng cao hiệu quả giảng dạy, giảm tải công việc thủ công cho giảng viên, đồng thời thúc đẩy sinh viên chủ động hơn trong học tập và rèn luyện kỹ năng nghề nghiệp.

## Nội dung nghiên cứu

Nội dung nghiên cứu của đề tài tập trung vào việc xây dựng một hệ thống hỗ trợ theo dõi và đánh giá chất lượng mã nguồn một cách tự động cho các dự án phần mềm làm việc nhóm của sinh viên. Các nội dung cụ thể được triển khai như sau:

***Tìm hiểu cơ sở lý thuyết***

- Tìm hiểu về GitHub, GitHub Actions, SonarCloud và cách tích hợp các công cụ này vào quy trình phát triển phần mềm.

- Khảo sát các chỉ số đánh giá chất lượng mã nguồn (bugs, code smells, vulnerabilities, coverage, maintainability, v.v.).

- Phân tích các công cụ đánh giá mã nguồn tự động hiện có và khả năng áp dụng trong môi trường giáo dục.

***Đề xuất kiến trúc hệ thống***

- Xây dựng kiến trúc tổng quan cho hệ thống theo dõi, trong đó tích hợp GitHub Actions để thu thập dữ liệu và SonarCloud để phân tích mã nguồn.

- Thiết kế cơ sở dữ liệu lưu trữ thông tin nhóm, sinh viên, số lần nộp bài, kết quả phân tích chất lượng mã, v.v.

***Xây dựng hệ thống tự động***

- Phát triển hệ thống cho phép giảng viên tạo môn học, đề tài và quản lý nhóm sinh viên.

- Tích hợp GitHub OAuth để xác thực người dùng.

- Cấu hình GitHub Actions để tự động phân tích mã mỗi khi sinh viên đẩy code lên repository.

- Đồng bộ dữ liệu từ SonarCloud để hiển thị báo cáo đánh giá lên dashboard quản trị.

***Xây dựng dashboard theo dõi và đánh giá***

- Xây dựng giao diện hiển thị trực quan các thông tin như: số lần nộp bài, chất lượng mã, mức độ đóng góp của từng thành viên, tiến độ thực hiện đề tài.

- Phân loại đánh giá theo nhóm, theo sinh viên, theo tiêu chí giảng viên đặt ra.

***Kiểm thử và đánh giá hệ thống***

- Áp dụng thử hệ thống vào một số lớp học phần thực tế để kiểm tra khả năng hoạt động, khả năng mở rộng và mức độ đáp ứng nhu cầu giảng dạy.

- Thu thập phản hồi để phân tích tính hiệu quả và đề xuất cải tiến.

## Đối tượng và phạm vi nghiên

***Đối tượng nghiên cứu***:

- Mã nguồn của sinh viên được lưu trữ và cập nhật trên các kho lưu trữ GitHub.

- Các công cụ hỗ trợ đánh giá tự động như GitHub Actions và SonarCloud.

- Quy trình làm việc nhóm và mức độ đóng góp của từng thành viên trong nhóm.

- Các chỉ số phản ánh chất lượng phần mềm như độ phức tạp, độ bao phủ kiểm thử, số lượng lỗi, v.v.

***Phạm vi nghiên cứu:***

- Hệ thống được xây dựng và áp dụng trong phạm vi các dự án phần mềm của sinh viên, chủ yếu ở bậc đại học chuyên ngành Công nghệ thông tin.

- Việc đánh giá tập trung vào chất lượng mã nguồn và mức độ đóng góp cá nhân, không đi sâu vào đánh giá chức năng nghiệp vụ hay hiệu suất thực thi của phần mềm.

- Công cụ sử dụng giới hạn trong hệ sinh thái GitHub (GitHub API, GitHub Actions) và SonarCloud cho phân tích mã.

- Dữ liệu nghiên cứu chủ yếu lấy từ quá trình sinh viên làm đồ án, bài tập lớn hoặc học phần phát triển phần mềm có làm việc nhóm.

## Phương pháp nghiên cứu

***Nghiên cứu lý thuyết:***

- Quản lý mã nguồn với Git và GitHub.

- Các công cụ tích hợp liên tục (CI/CD), đặc biệt là GitHub Actions.

- Phương pháp đánh giá chất lượng mã nguồn bằng SonarQube và SonarCloud.

- Các mô hình theo dõi đóng góp cá nhân trong nhóm phát triển phần mềm.

- Nghiên cứu các giao diện và API do GitHub và SonarCloud cung cấp để hiểu rõ cơ chế truy xuất và khai thác dữ liệu.

- Tìm hiểu các hệ thống tương tự (nếu có) để rút ra bài học kinh nghiệm, từ đó đề xuất mô hình phù hợp với bối cảnh sinh viên làm việc nhóm.

***Nghiên cứu thực nghiệm:***

- Tiến hành thiết kế, xây dựng và triển khai hệ thống theo dõi tiến độ dự án và đánh giá chất lượng mã nguồn.

- Tích hợp các thành phần chính: GitHub API, GitHub Actions, SonarCloud và giao diện người dùng.

- Ghi nhận kết quả thu được từ các chỉ số: số lần nộp bài, số lượng đóng góp, chất lượng mã nguồn… và phản hồi của người dùng.

- Phân tích dữ liệu thực nghiệm để đánh giá mức độ hiệu quả, khả năng áp dụng thực tiễn và đề xuất hướng cải tiến hệ thống.

# CƠ SỞ LÝ THUYẾT

## Tổng quan về GitHub và làm việc nhóm qua GitHub

GitHub là một nền tảng dịch vụ web hàng đầu thế giới, cung cấp các kho lưu trữ mã nguồn dựa trên hệ thống quản lý phiên bản phân tán Git. Ra đời từ năm 2008, GitHub không chỉ đơn thuần là nơi lưu trữ mã nguồn mà đã phát triển thành một hệ sinh thái toàn diện, tạo ra môi trường lý tưởng cho các lập trình viên trên toàn cầu cùng nhau cộng tác.



Với hơn 100 triệu người dùng và hàng triệu dự án lớn nhỏ, GitHub đóng vai trò là xương sống của nhiều quy trình phát triển phần mềm, từ các dự án mã nguồn mở khổng lồ cho đến các dự án thương mại phức tạp. Nền tảng này cho phép mỗi thành viên trong nhóm làm việc trên các nhánh (branch) độc lập mà không ảnh hưởng đến phiên bản chính của dự án. Khi công việc hoàn thành, họ có thể yêu cầu hợp nhất mã (pull request), nơi các thành viên khác có thể xem xét, thảo luận và đưa ra phản hồi trước khi tích hợp vào nhánh chính.

### Git và GitHub là gì?

Git là một hệ thống quản lý phiên bản (Version Control System – VCS) giúp theo dõi và kiểm soát các thay đổi trong mã nguồn của một dự án. Git hoạt động theo mô hình phân tán, cho phép mỗi người dùng có một bản sao toàn bộ repository và lịch sử commit của dự án.

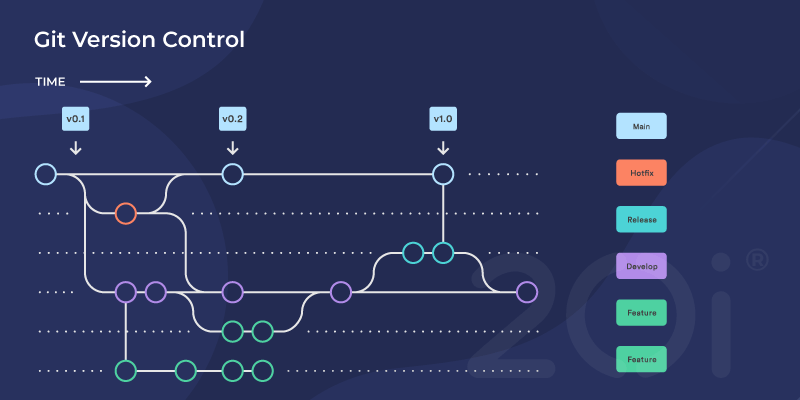
GitHub là một dịch vụ trực tuyến cung cấp giao diện web cho Git, đồng thời tích hợp thêm nhiều tính năng hỗ trợ phát triển phần mềm như quản lý dự án, theo dõi lỗi (issues), kiểm tra chất lượng mã, CI/CD, tài liệu dự án, v.v.

### Các tính năng hỗ trợ làm việc nhóm

Làm việc nhóm qua GitHub được hỗ trợ bởi hàng loạt tính năng sau:

*- Repository (kho lưu trữ mã nguồn):* Mỗi dự án phần mềm được tổ chức dưới dạng một repository – một không gian lưu trữ toàn bộ mã nguồn, tài liệu, lịch sử thay đổi (commit history), các nhánh phát triển (branches), vấn đề phát sinh (issues), và các thiết lập cấu hình. Repository không chỉ là nơi lưu trữ mã, mà còn là trung tâm tương tác giữa các thành viên trong nhóm, nơi diễn ra mọi hoạt động quản lý mã nguồn.

*- Branching (quản lý nhánh):* GitHub cho phép mỗi thành viên trong nhóm tạo nhánh phát triển riêng biệt từ nhánh chính (thường là main hoặc master). Việc phân nhánh giúp các thành viên có thể phát triển tính năng mới, sửa lỗi hoặc thử nghiệm mà không làm ảnh hưởng đến mã ổn định đang tồn tại. Mô hình này khuyến khích lập trình song song và giảm thiểu xung đột mã khi hợp nhất.



*- Pull Request (Yêu cầu hợp nhất mã):* Khi hoàn tất công việc trên một nhánh, thành viên có thể tạo một pull request để đề xuất hợp nhất mã nguồn vào nhánh chính. Đây là bước quan trọng trong quy trình kiểm duyệt mã (code review), nơi các thành viên khác trong nhóm có thể đọc, đánh giá, phản hồi hoặc đề xuất thay đổi trước khi mã được hợp nhất. Pull Request là một công cụ quan trọng nhằm nâng cao chất lượng mã, phát hiện lỗi sớm và tăng cường sự minh bạch trong quy trình phát triển.

A screenshot of a computer

AI-generated content may be incorrect.

*- Issue Tracking (Theo dõi vấn đề):* Hệ thống quản lý Issue trong GitHub cho phép nhóm tạo, phân loại, và theo dõi các lỗi phần mềm, yêu cầu tính năng, hay các nhiệm vụ cụ thể. Các issue có thể được gán cho thành viên, đánh nhãn, và liên kết với pull request để dễ dàng theo dõi tiến độ. Đây là một công cụ hỗ trợ lập kế hoạch và kiểm soát chất lượng dự án hiệu quả.

*- Project Board (Bảng quản lý dự án):* GitHub cung cấp bảng quản lý công việc theo mô hình Kanban, giúp nhóm phát triển hình dung được trạng thái hiện tại của các task (việc cần làm, đang làm, đã hoàn thành). Các issue hoặc pull request có thể được gán vào các cột trạng thái để cập nhật tiến độ làm việc. Điều này đặc biệt hữu ích trong việc tổ chức công việc và theo dõi hiệu suất nhóm.

*- GitHub Actions & CI/CD:* Tính năng GitHub Actions cho phép nhóm phát triển tự động hóa các quy trình như kiểm thử đơn vị (unit testing), kiểm tra chất lượng mã, biên dịch, và triển khai (deployment). Các workflow CI/CD (Continuous Integration/Continuous Deployment) giúp phát hiện lỗi sớm, đảm bảo chất lượng và rút ngắn thời gian phát hành sản phẩm.

*- Wiki & Documentation:* GitHub cung cấp không gian để viết tài liệu hướng dẫn sử dụng, ghi chú kỹ thuật hoặc quy trình phát triển trong dự án thông qua Wiki hoặc các file README, CONTRIBUTING. Việc lưu trữ tài liệu trực tiếp trong repository giúp các thành viên dễ dàng truy cập và cập nhật khi cần thiết.

### Lợi ích khi làm việc nhóm qua GitHub

Việc sử dụng GitHub trong môi trường làm việc nhóm mang lại nhiều lợi ích vượt trội, bao gồm:

*- Tăng cường cộng tác:* GitHub giúp các thành viên trong nhóm làm việc hiệu quả hơn thông qua cơ chế phản hồi, bình luận và kiểm tra mã nguồn.

*- Minh bạch và kiểm soát chất lượng*: Mọi thay đổi đều được ghi lại, giúp đảm bảo tính minh bạch và dễ dàng đánh giá chất lượng đóng góp của từng thành viên.

*- Tích hợp dễ dàng với công cụ CI/CD*: GitHub hỗ trợ tích hợp với nhiều công cụ kiểm thử, triển khai liên tục như GitHub Actions, giúp tự động hóa quy trình phát triển phần mềm.

*- Khả năng mở rộng và bảo mật cao*: Hệ thống phân quyền chi tiết và khả năng tích hợp với các giải pháp bảo mật giúp bảo vệ mã nguồn trước các rủi ro an ninh.

### Mô hình làm việc nhóm qua GitHub

Một quy trình làm việc nhóm tiêu chuẩn và hiệu quả với GitHub thường tuân theo mô hình Gitflow hoặc các biến thể đơn giản hơn. Quy trình này giúp đảm bảo mã nguồn luôn ổn định, đồng thời tạo ra một luồng làm việc rõ ràng và có tổ chức cho tất cả các thành viên. Dưới đây là các bước chi tiết trong mô hình này:

*1. Clone Kho Mã Nguồn từ Kho Chính (Main Repository)*: Đây là bước khởi đầu. Mỗi thành viên trong nhóm sẽ sử dụng lệnh git clone để tạo một bản sao cục bộ (local copy) của kho mã nguồn chính về máy tính cá nhân của mình.

*2. Tạo Nhánh Riêng (Feature Branch)*: Trước khi bắt đầu công việc, mỗi thành viên sẽ tạo một nhánh mới từ nhánh chính (thường là *main* hoặc *master*) bằng lệnh *git checkout -b <tên-nhánh-của-bạn>.* Tên nhánh nên mô tả rõ ràng công việc đang thực hiện (ví dụ: *feature/user-authentication* hoặc *bugfix/login-error*).

*3. Commit và Push Mã Nguồn lên Nhánh Cá Nhân*: Khi hoàn thành một phần công việc nhỏ, lập trình viên sẽ thực hiện lệnh git add và git commit để lưu lại các thay đổi. Sau đó, sử dụng lệnh git push để đẩy mã nguồn từ máy tính cá nhân lên nhánh tương ứng trên GitHub.

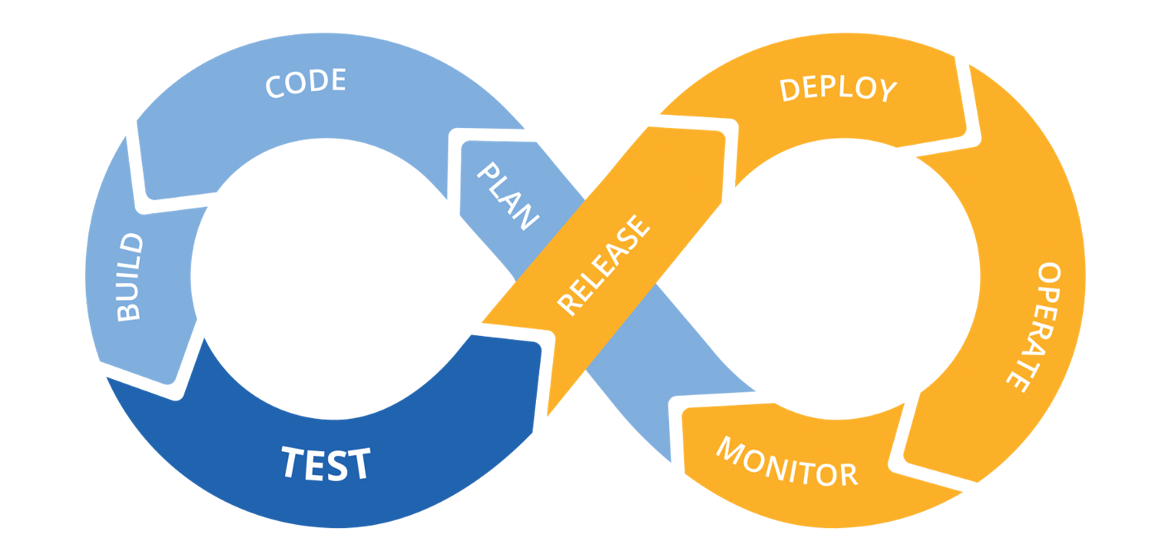
*4. Tạo Pull Request (Yêu cầu hợp nhất mã)*: Khi một tính năng hoặc sửa lỗi đã hoàn thành, lập trình viên sẽ tạo một Pull Request (PR) trên GitHub. PR này là một yêu cầu chính thức để hợp nhất (merge) mã nguồn từ nhánh cá nhân vào nhánh chính. Kèm theo PR, cần có một mô tả chi tiết về những thay đổi đã thực hiện và lý do của chúng.

*5. Xem xét và Phê duyệt Pull Request (Code Review)*: Các thành viên khác trong nhóm (đặc biệt là những người có kinh nghiệm) sẽ xem xét PR, kiểm tra code, phát hiện lỗi, và đưa ra các đề xuất cải thiện. Họ có thể thêm bình luận trực tiếp trên từng dòng code hoặc đưa ra nhận xét chung.

*6. Hợp nhất vào Nhánh Chính (Merge)*: Sau khi PR đã được xem xét kỹ lưỡng và phê duyệt, thành viên phụ trách sẽ hợp nhất (merge) mã nguồn vào nhánh chính. Khi đó, các thay đổi đã được kiểm tra và chấp nhận sẽ trở thành một phần của dự án.

## Tổng quan CI/CD

CI/CD (viết tắt của Continuous Integration / Continuous Delivery hoặc Continuous Deployment) là một phương pháp luận và tập hợp các quy trình tự động hóa nhằm thay đổi hoàn toàn cách các đội phát triển phần mềm làm việc. CI/CD tạo ra một chu trình tự động, giúp rút ngắn khoảng cách giữa việc viết mã và đưa sản phẩm đến tay người dùng, giảm thiểu rủi ro và tăng tốc độ phát triển.



### Continuous Integration (CI) - Tích hợp Liên tục

Continuous Integration (CI) là nền tảng của quy trình CI/CD. Nó tập trung vào việc hợp nhất (merge) các thay đổi mã nguồn từ nhiều lập trình viên vào một nhánh chung (thường là nhánh main hoặc master) một cách thường xuyên và tự động. Mục tiêu cốt lõi của CI là ngăn chặn "lỗi tích hợp", tức là những lỗi phát sinh khi các đoạn mã của các thành viên khác nhau được gộp lại.

Quy trình CI điển hình bao gồm các bước sau:

*- Commit mã nguồn:* Các lập trình viên thường xuyên commit và push các thay đổi nhỏ lên kho mã nguồn chung.

*- Xây dựng tự động (Automated Build):* Một hệ thống CI sẽ tự động phát hiện các thay đổi này và khởi động quá trình xây dựng (build) dự án.

*- Kiểm thử tự động (Automated Testing):* Sau khi build thành công, hệ thống sẽ chạy các bộ kiểm thử tự động, bao gồm unit tests, integration tests và static code analysis. Mục tiêu là đảm bảo rằng mã mới không gây ra lỗi cho các chức năng hiện có của hệ thống.

*- Phản hồi nhanh:* Nếu có bất kỳ bước nào thất bại, hệ thống sẽ gửi thông báo đến nhóm phát triển ngay lập tức để họ có thể sửa lỗi kịp thời.

*Lợi ích của CI và CD đã được đề cập trong phần "CI/CD giúp" ở đoạn bạn cung cấp, nhưng tôi sẽ chi tiết hơn để làm rõ từng phần. Đây là phiên bản mở rộng, tập trung vào lợi ích cụ thể của từng thành phần.*

Lợi ích của Continuous Integration (CI):

*- Phát hiện lỗi sớm:* Đây là lợi ích quan trọng nhất của CI. Bằng cách tự động build và chạy các bài kiểm thử mỗi khi có mã mới được hợp nhất, CI giúp phát hiện các lỗi tích hợp ngay từ đầu, trước khi chúng trở nên phức tạp và khó sửa.

- Cải thiện chất lượng mã nguồn: CI thường được tích hợp với các công cụ phân tích tĩnh (static analysis tools). Các công cụ này tự động kiểm tra mã nguồn theo các tiêu chuẩn đã định, giúp lập trình viên tuân thủ các quy tắc lập trình sạch và tối ưu.

*- Giảm thiểu thời gian và công sức thủ công:* CI tự động hóa các tác vụ lặp đi lặp lại như build và kiểm thử. Điều này giải phóng lập trình viên khỏi các công việc tốn thời gian, cho phép họ tập trung hơn vào việc viết mã và phát triển tính năng.

*- Phản hồi nhanh và liên tục:* Với mỗi lần commit, lập trình viên sẽ nhận được phản hồi ngay lập tức về việc mã của họ có vượt qua các bài kiểm thử hay không. Điều này thúc đẩy một chu trình phản hồi nhanh, giúp nhóm sửa lỗi một cách chủ động và hiệu quả.

### Continuous Delivery (CD) - Phân phối Liên tục

Continuous Delivery (CD) là bước tiếp theo của CI. Nó đảm bảo rằng sau khi mã nguồn đã vượt qua tất cả các bài kiểm thử tự động của CI, sản phẩm phần mềm luôn ở trạng thái sẵn sàng để triển khai. Tuy nhiên, việc triển khai lên môi trường sản xuất (production) vẫn cần một sự can thiệp thủ công, chẳng hạn như một người quản lý dự án sẽ nhấn nút để triển khai phiên bản mới.

Mục tiêu chính của Continuous Delivery là rút ngắn thời gian từ khi mã được viết cho đến khi nó sẵn sàng được phát hành, nhưng vẫn giữ quyền kiểm soát việc triển khai cuối cùng.

Quy trình CD thường được xây dựng trên nền tảng của CI và bao gồm các giai đoạn chính sau:

*- Mã nguồn được tạo và kiểm thử (Build & Test):* Đây là bước cuối cùng của quy trình CI. Sau khi mã nguồn được hợp nhất và các bài kiểm thử đơn vị (unit tests) đã chạy thành công, hệ thống sẽ xây dựng (build) mã nguồn thành một "artifact" có thể triển khai được (ví dụ: file JAR, Docker image, hoặc gói cài đặt). Đảm bảo mã nguồn có thể biên dịch và vượt qua các bài kiểm thử cơ bản, tạo ra một bản dựng (build) sẵn sàng cho các giai đoạn tiếp theo.

*- Kiểm thử tích hợp và staging (Integration & Staging Testing)*: Bản dựng được triển khai tự động lên một môi trường staging (môi trường giả lập môi trường sản xuất). Tại đây, các bài kiểm thử tích hợp (integration tests), kiểm thử end-to-end, và kiểm thử hiệu suất (performance tests) sẽ được chạy. Đây là bước quan trọng để đảm bảo rằng phần mềm hoạt động đúng với các thành phần khác của hệ thống. Kiểm tra và xác minh tính ổn định, hiệu suất của phần mềm trong một môi trường gần giống với môi trường sản xuất nhất có thể.

*- Triển khai thủ công hoặc tự động (Manual or Automated Deployment)*: Sau khi vượt qua tất cả các bài kiểm thử trên môi trường staging, bản dựng được đánh dấu là "đã sẵn sàng để phát hành" (release ready). Đến đây, quy trình CD có thể rẽ nhánh. Triển khai thủ công, một thành viên trong nhóm sẽ xem xét và quyết định khi nào triển khai bản cập nhật lên môi trường sản xuất. Họ chỉ cần nhấn nút hoặc chạy một lệnh để bắt đầu quá trình triển khai. Đây là mô hình Continuous Delivery. Triển khai tự động, bản dựng sẽ được triển khai tự động ngay lập tức lên môi trường sản xuất mà không cần sự can thiệp thủ công. Đây là mô hình Continuous Deployment.

*- Giám sát và Phản hồi (Monitoring & Feedback)*: Sau khi triển khai, hệ thống giám sát (monitoring) sẽ theo dõi hiệu suất, phát hiện lỗi và các vấn đề tiềm ẩn trong môi trường sản xuất. Dữ liệu từ quá trình này sẽ được thu thập và phân tích để cung cấp phản hồi cho nhóm phát triển, giúp họ cải thiện sản phẩm trong các chu kỳ sau. Đảm bảo hệ thống hoạt động ổn định sau khi triển khai và cung cấp thông tin để cải thiện sản phẩm liên tục.

Lợi ích của Continuous Delivery (CD):

*- Tăng tốc độ phát hành (Release velocity):* CD tự động hóa quá trình chuẩn bị phần mềm để phát hành. Điều này cho phép nhóm phát triển phát hành các bản cập nhật thường xuyên hơn và nhanh hơn nhiều so với quy trình thủ công truyền thống.

*- Giảm thiểu rủi ro khi triển khai:* Bằng cách đảm bảo phần mềm luôn ở trạng thái sẵn sàng để triển khai, CD giảm thiểu rủi ro liên quan đến các bản phát hành lớn. Mỗi bản cập nhật nhỏ hơn, dễ kiểm soát hơn, và nếu có lỗi, việc khôi phục cũng nhanh hơn.

*- Tăng tính minh bạch và đáng tin cậy:* Với quy trình CD, mọi người đều biết rằng phiên bản mới đã vượt qua tất cả các bài kiểm thử cần thiết và sẵn sàng để triển khai. Điều này tạo ra sự tin tưởng vào chất lượng sản phẩm và giúp nhóm phát triển tự tin hơn.

*- Tăng khả năng phản ứng với thị trường:* Các bản cập nhật nhỏ, thường xuyên cho phép nhóm phát triển nhanh chóng đưa các tính năng mới và các bản vá lỗi đến người dùng. Điều này giúp doanh nghiệp phản ứng linh hoạt hơn với các yêu cầu của thị trường và phản hồi từ khách hàng.

### Quy trình CI/CD

Quy trình **CI/CD** là một "đường ống dẫn" (pipeline) tự động, giúp tự động hóa toàn bộ chu kỳ phát triển phần mềm. Quy trình này đảm bảo mỗi thay đổi mã nguồn đều được kiểm tra kỹ lưỡng trước khi được đưa vào môi trường sản xuất. Dưới đây là các giai đoạn chính của một quy trình CI/CD điển hình:

Giai đoạn Tích hợp Liên tục (CI - Continuous Integration)

*- Commit Mã nguồn:* Mọi quy trình CI/CD đều bắt đầu khi một lập trình viên commit và push mã nguồn của mình lên kho lưu trữ (repository) chung, thường là trên các nền tảng như GitHub, GitLab, hoặc Bitbucket. Hành động này kích hoạt pipeline CI/CD.

*- Build Mã nguồn:* Hệ thống CI/CD sẽ tự động tải mã nguồn mới nhất và chạy quá trình build (biên dịch mã nguồn, tạo gói ứng dụng, hoặc tạo Docker image). Nếu quá trình build thất bại, hệ thống sẽ thông báo ngay lập tức để lập trình viên sửa lỗi.

*- Kiểm thử tự động (Automated Testing):* Sau khi build thành công, hệ thống sẽ chạy một loạt các bài kiểm thử tự động, bao gồm Unit Tests (kiểm thử các đơn vị mã nguồn nhỏ), Integration Tests (kiểm thử sự tương tác giữa các module), và Static Code Analysis (phân tích tĩnh mã nguồn để tìm lỗi và vi phạm quy chuẩn).

Giai đoạn Phân phối Liên tục (CD - Continuous Delivery)

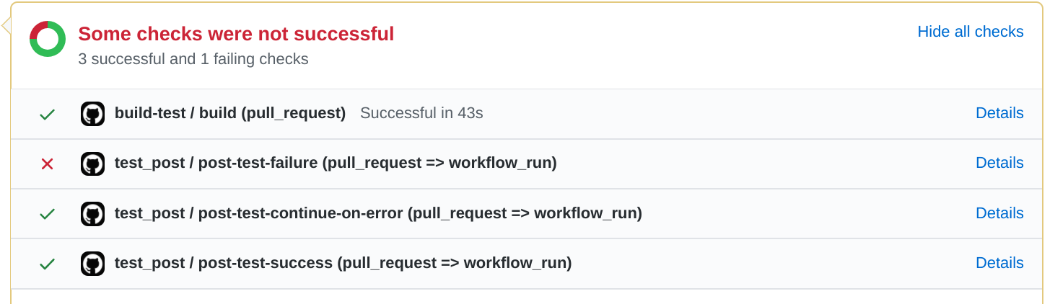
*- Triển khai lên môi trường Staging:* Nếu mã nguồn vượt qua tất cả các bài kiểm thử của giai đoạn CI, nó sẽ tự đ

nhộng được triển khai lên một môi trường giả lập môi trường sản xuất, gọi là staging. Tại đây, các bài kiểm thử cuối cùng như End-to-End Tests và Performance Tests sẽ được thực hiện.

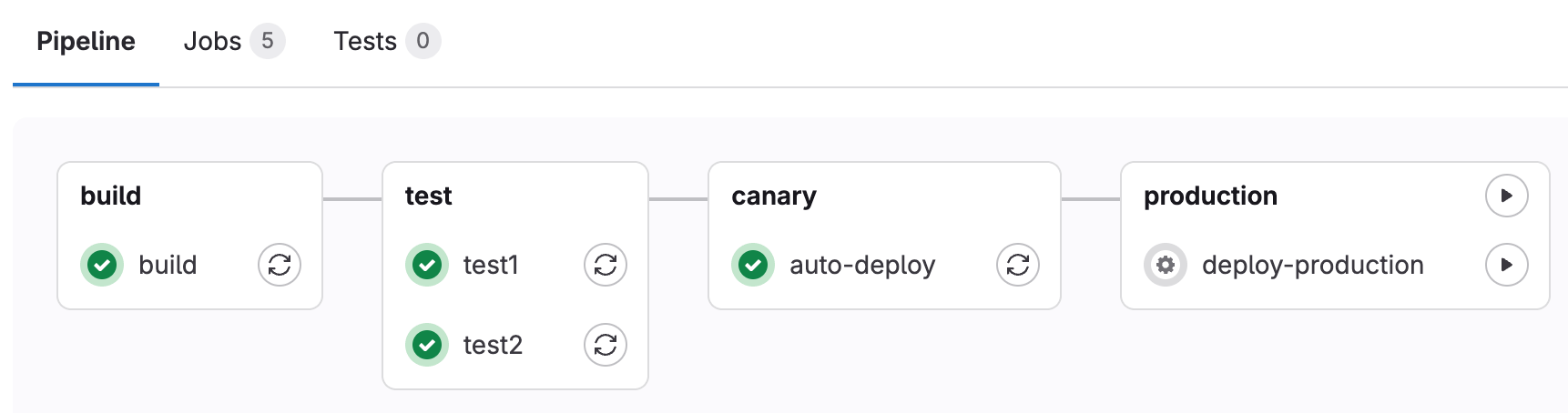
*- Triển khai lên môi trường Production (Tuỳ chọn):* Sau khi mã nguồn đã được xác nhận ổn định trên staging, nó sẽ sẵn sàng để triển khai lên môi trường production (môi trường chạy thật). Trong Continuous Delivery, bước này thường cần sự phê duyệt thủ công. Còn trong Continuous Deployment, bước này sẽ diễn ra tự động.

Việc lựa chọn công cụ phù hợp là rất quan trọng để xây dựng một pipeline CI/CD hiệu quả. Dưới đây là một số công cụ phổ biến được sử dụng rộng rãi hiện nay:

*1. GitHub Actions:* là một nền tảng CI/CD mạnh mẽ được tích hợp sâu vào hệ sinh thái của GitHub, cho phép các nhà phát triển tự động hóa toàn bộ quy trình làm việc ngay trong kho lưu trữ mã nguồn của mình. Khác với các giải pháp CI/CD bên ngoài, GitHub Actions tận dụng cấu trúc của GitHub để cung cấp một trải nghiệm liền mạch và đồng bộ, từ việc quản lý mã nguồn, review code, đến triển khai ứng dụng. Nền tảng này hoạt động dựa trên các tệp tin cấu hình YAML được đặt trong thư mục .github/workflows, nơi các nhà phát triển định nghĩa một chuỗi các sự kiện kích hoạt (triggers), các công việc (jobs) và các bước (steps) cần thực thi. Sự linh hoạt này cho phép xây dựng các workflow phức tạp để tự động hóa các tác vụ từ đơn giản như chạy kiểm thử, phân tích mã nguồn đến các tác vụ phức tạp như xây dựng và triển khai ứng dụng lên các nền tảng đám mây. Một trong những điểm mạnh lớn nhất của GitHub Actions là kho hành động (Action Marketplace), nơi cộng đồng cung cấp hàng ngàn "action" có sẵn, giúp các nhà phát triển dễ dàng tích hợp các công cụ và dịch vụ bên thứ ba mà không cần phải viết mã từ đầu. Điều này không chỉ giúp tiết kiệm thời gian mà còn giảm thiểu đáng kể chi phí và công sức cần thiết để thiết lập một pipeline CI/CD hoàn chỉnh, làm cho GitHub Actions trở thành một lựa chọn lý tưởng cho cả các dự án cá nhân, mã nguồn mở và các dự án thương mại quy mô lớn.



*2. GitLab CI/CD:* là một giải pháp tích hợp liên tục và phân phối liên tục (CI/CD) mạnh mẽ và toàn diện, được tích hợp sẵn trong nền tảng DevOps của GitLab. Khác với các công cụ CI/CD truyền thống, GitLab CI/CD không chỉ là một dịch vụ độc lập mà là một phần không thể thiếu của toàn bộ chu trình phát triển sản phẩm. Nền tảng này được cấu hình thông qua một tệp tin .gitlab-ci.yml duy nhất, cho phép các nhà phát triển định nghĩa các giai đoạn (stages) và công việc (jobs) của pipeline một cách rõ ràng và có cấu trúc. Từ khi một lập trình viên commit mã nguồn, GitLab CI/CD sẽ tự động kích hoạt các runner (máy ảo thực thi công việc) để chạy các tác vụ đã được định nghĩa, bao gồm build, kiểm thử, phân tích mã và triển khai. Một trong những ưu điểm nổi bật của GitLab CI/CD là khả năng hỗ trợ nhiều loại runner khác nhau, từ các runner dùng chung của GitLab đến các runner tự lưu trữ (self-hosted runners), mang lại sự linh hoạt tối đa cho các dự án có yêu cầu đặc thù về môi trường. Điều này giúp các đội ngũ phát triển dễ dàng kiểm soát và quản lý toàn bộ quy trình phát triển sản phẩm, từ quản lý mã nguồn, CI/CD, đến giám sát và bảo mật, tất cả trên một nền tảng duy nhất, từ đó tạo ra một luồng công việc liền mạch và hiệu quả.

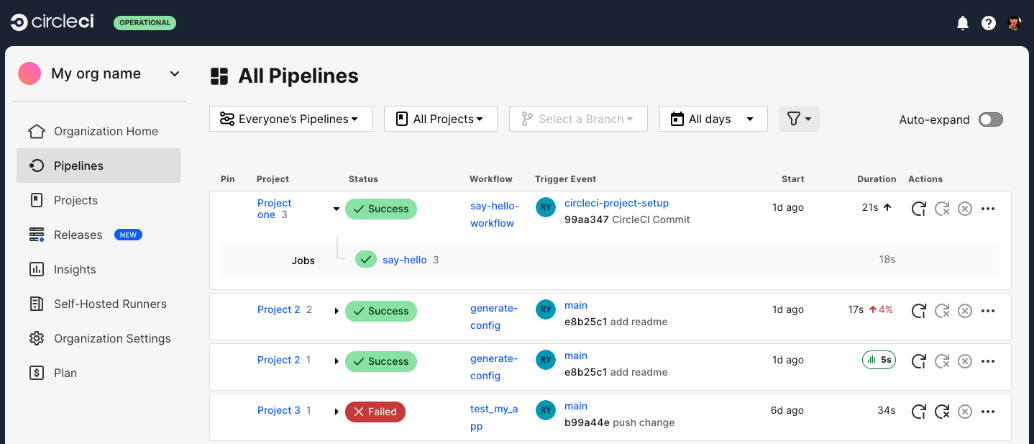


*3. Jenkins:* là một máy chủ tự động hóa mã nguồn mở, lâu đời và phổ biến nhất trong giới công nghệ, được sử dụng rộng rãi để tự động hóa các tác vụ liên quan đến build, kiểm thử và triển khai phần mềm. Với lịch sử phát triển lâu dài, Jenkins đã xây dựng một hệ sinh thái plugin đồ sộ, với hàng ngàn plugin hỗ trợ tích hợp với hầu hết mọi công cụ và dịch vụ trong quy trình phát triển phần mềm, từ các hệ thống quản lý phiên bản (Git), các công cụ build (Maven, Gradle) đến các nền tảng đám mây (AWS, Azure, GCP). Jenkins hoạt động dựa trên mô hình server-agent, trong đó máy chủ Jenkins (Jenkins master) điều phối các tác vụ tới các agent (worker nodes) để thực thi. Sự linh hoạt này cho phép Jenkins mở rộng quy mô một cách hiệu quả, phân tán các tác vụ build và kiểm thử trên nhiều máy chủ để giảm thiểu thời gian chờ đợi. Mặc dù yêu cầu cấu hình và quản lý phức tạp hơn so với các giải pháp CI/CD đám mây hiện đại, Jenkins vẫn là một lựa chọn mạnh mẽ cho các tổ chức có nhu cầu tùy chỉnh cao và muốn kiểm soát hoàn toàn môi trường CI/CD của mình.

A screenshot of a data analysis

AI-generated content may be incorrect.

*4. CircleCI:* là một nền tảng CI/CD dựa trên đám mây được thiết kế để đơn giản hóa quá trình tự động hóa, nổi bật với giao diện trực quan và dễ sử dụng. Với CircleCI, các nhà phát triển có thể nhanh chóng thiết lập một pipeline CI/CD hoàn chỉnh chỉ bằng một tệp tin cấu hình config.yml, giúp tích hợp liền mạch với các kho lưu trữ mã nguồn như GitHub và Bitbucket. CircleCI hỗ trợ đa nền tảng, cung cấp các môi trường chạy linh hoạt (như Docker, Linux, macOS, và Windows) để đáp ứng các nhu cầu khác nhau của dự án. Một trong những tính năng đáng chú ý của CircleCI là khả năng chạy kiểm thử song song, giúp giảm đáng kể thời gian chờ đợi và tăng hiệu quả của pipeline. Ngoài ra, CircleCI còn cung cấp các tính năng quản lý bí mật (secrets management) và các công cụ giám sát hiệu suất, cho phép các đội ngũ phát triển đảm bảo an toàn và theo dõi sát sao quá trình CI/CD. Nhờ vào sự đơn giản, tính linh hoạt và hiệu suất cao, CircleCI đã trở thành một lựa chọn phổ biến cho cả các dự án nhỏ và các doanh nghiệp lớn, giúp các đội ngũ phát triển tập trung vào việc viết mã thay vì quản lý cơ sở hạ tầng.



*5. Travis CI:* là một dịch vụ CI/CD dựa trên đám mây khác, nổi bật với sự đơn giản và dễ dàng tích hợp, đặc biệt với các kho lưu trữ trên GitHub. Được thiết kế ban đầu để hỗ trợ các dự án mã nguồn mở, Travis CI đã nhanh chóng trở thành một công cụ phổ biến cho việc tự động hóa quá trình kiểm thử và build. Travis CI hoạt động dựa trên một tệp tin .travis.yml đơn giản, nơi các nhà phát triển có thể định nghĩa các bước cần thực hiện, từ việc cài đặt môi trường, chạy kiểm thử đến triển khai. Travis CI tích hợp chặt chẽ với GitHub, tự động kích hoạt một pipeline CI/CD mỗi khi có một commit hoặc pull request mới được tạo, cung cấp phản hồi tức thì về chất lượng mã nguồn. Một trong những điểm mạnh của Travis CI là khả năng hỗ trợ nhiều ngôn ngữ lập trình và môi trường khác nhau, giúp các dự án đa dạng có thể tận dụng được lợi ích của nó. Mặc dù có phần đơn giản hơn so với các nền tảng CI/CD khác, Travis CI vẫn là một lựa chọn hiệu quả cho các dự án nhỏ, các dự án mã nguồn mở và các đội ngũ phát triển muốn có một giải pháp CI/CD nhanh chóng và dễ sử dụng.

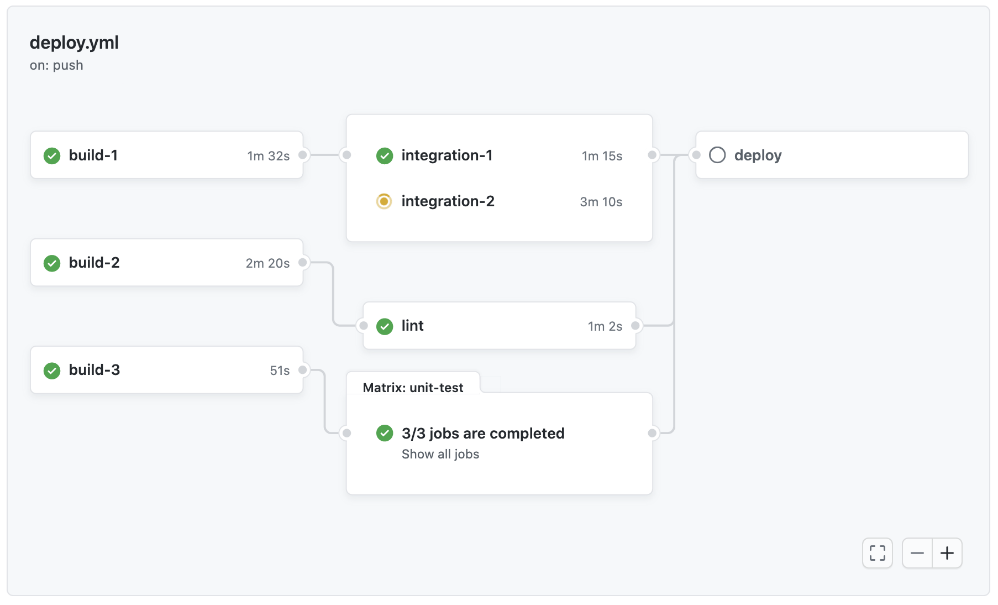


## GitHub Actions

GitHub Actions là một nền tảng tích hợp liên tục/triển khai liên tục (CI/CD) của GitHub, cho phép tự động hóa các tác vụ trong quy trình phát triển phần mềm. Trong dự án CodeFlow, chúng tôi sử dụng GitHub Actions để tự động hóa quá trình kiểm thử, đánh giá chất lượng mã nguồn và đảm bảo tính toàn vẹn của dự án.

GitHub Actions được giới thiệu lần đầu tiên vào tháng 10 năm 2018 tại hội nghị GitHub Universe. Ban đầu, nó được thiết kế để tạo ra các quy trình làm việc tự động hóa dựa trên các sự kiện của kho lưu trữ (repository events) như push hoặc pull\_request, cho phép các nhà phát triển tạo ra các tác vụ tùy chỉnh như gửi thông báo hoặc chạy các script. Đến tháng 11 năm 2019, GitHub Actions được mở rộng và phát triển mạnh mẽ hơn, trở thành một nền tảng CI/CD hoàn chỉnh. Sự thay đổi này đã biến GitHub Actions từ một công cụ tự động hóa đơn giản thành một giải pháp toàn diện cho việc xây dựng, kiểm thử và triển khai phần mềm trực tiếp từ GitHub.

Sự ra đời của GitHub Actions đã mang lại một bước đột phá lớn. Nó tích hợp chặt chẽ với hệ sinh thái GitHub, cho phép các đội nhóm phát triển triển khai quy trình CI/CD một cách dễ dàng mà không cần phải sử dụng các dịch vụ bên thứ ba. Điều này giúp đơn giản hóa quy trình phát triển, giảm thiểu chi phí và tăng tốc độ phát triển dự án.



Một workflow trong GitHub Actions thường bao gồm các thành phần chính sau:

*- Event (Sự kiện kích hoạt):* Là các tác nhân hoặc điều kiện khởi chạy workflow. Mỗi workflow có thể được cấu hình để kích hoạt bởi một hoặc nhiều sự kiện. Các sự kiện này có thể xuất phát từ hoạt động của người dùng (ví dụ: push khi đẩy mã nguồn lên kho lưu trữ, pull\_request khi tạo hoặc cập nhật yêu cầu hợp nhất), từ lịch trình định sẵn (schedule sử dụng cú pháp cron), hoặc từ thao tác thủ công (workflow\_dispatch). Event đóng vai trò như “điểm khởi đầu” cho toàn bộ quá trình tự động hóa.

*- Job (Công việc):* Là một nhóm các bước (step) được thực thi trong cùng một môi trường chạy (runner). Mỗi job thường đại diện cho một giai đoạn riêng biệt trong quy trình, ví dụ: biên dịch mã, chạy kiểm thử, hoặc triển khai ứng dụng. Các job mặc định chạy tuần tự, nhưng có thể cấu hình để chạy song song hoặc thiết lập quan hệ phụ thuộc nhằm tối ưu hóa thời gian thực thi.

*- Step (Bước):* Là một tác vụ cụ thể bên trong job, được thực hiện tuần tự theo thứ tự định nghĩa. Một step có thể là một lệnh shell (ví dụ: chạy script, cài đặt thư viện) hoặc gọi một action có sẵn. Step là đơn vị nhỏ nhất của quy trình, đảm bảo mỗi tác vụ được thực hiện rõ ràng và độc lập.

*- Action (Hành động):* Là khối chức năng độc lập, có thể tái sử dụng trong nhiều workflow khác nhau. Action có thể được phát triển bởi chính nhóm dự án hoặc lấy từ GitHub Marketplace, nơi cộng đồng chia sẻ hàng nghìn action cho các mục đích khác nhau như kiểm thử, triển khai, phân tích mã nguồn, gửi thông báo,… Action có thể được xây dựng bằng JavaScript hoặc đóng gói dưới dạng Docker container, cho phép triển khai linh hoạt trên nhiều môi trường.

Lợi ích của GitHub Actions trong CI/CD:

*- Tích hợp trực tiếp và liền mạch với GitHub:* Một trong những ưu điểm lớn nhất của GitHub Actions là việc nó được tích hợp trực tiếp vào kho mã nguồn. Điều này loại bỏ hoàn toàn nhu cầu cài đặt, cấu hình, và quản lý các hệ thống CI/CD bên ngoài như Jenkins hoặc GitLab CI/CD. Mọi thứ đều diễn ra trong cùng một nền tảng GitHub quen thuộc, từ việc quản lý mã nguồn, tạo Pull Request cho đến việc giám sát pipeline CI/CD. Các workflow được cấu hình bằng các tệp tin YAML ngay trong repository, cho phép mã nguồn và quy trình tự động hóa được lưu trữ và quản lý chung, tạo ra một trải nghiệm phát triển liền mạch và đồng bộ.

*- Hỗ trợ đa nền tảng và môi trường linh hoạt:* GitHub Actions cung cấp các môi trường thực thi linh hoạt (runners) trên nhiều hệ điều hành khác nhau, bao gồm Ubuntu, Windows, và macOS. Điều này cho phép các nhà phát triển chạy các tác vụ CI/CD trong môi trường phù hợp nhất với dự án của mình, dù đó là một ứng dụng web trên Linux hay một ứng dụng desktop trên macOS. Hơn nữa, GitHub Actions cũng hỗ trợ các nền tảng tự lưu trữ (self-hosted runners), cho phép các tổ chức chạy workflow trên các máy chủ cá nhân để đáp ứng các yêu cầu đặc thù về bảo mật hoặc tài nguyên.

*- Khả năng mở rộng cao thông qua Action Marketplace:* Sức mạnh của GitHub Actions đến từ kho hành động (Action Marketplace) khổng lồ, nơi cộng đồng phát triển và chia sẻ hàng ngàn "action" có sẵn. Một "action" là một tác vụ tự động hóa có thể tái sử dụng. Thay vì phải viết mã từ đầu cho các công việc phổ biến như cài đặt Node.js, đăng nhập vào AWS, hoặc gửi thông báo, bạn chỉ cần sử dụng các action có sẵn. Điều này không chỉ giúp giảm đáng kể thời gian và công sức để thiết lập pipeline, mà còn thúc đẩy khả năng tái sử dụng và chuẩn hóa quy trình.

*- Tự động hóa toàn diện quy trình phát triển:* GitHub Actions cho phép tự động hóa gần như mọi tác vụ trong quy trình phát triển phần mềm. Một pipeline CI/CD với GitHub Actions có thể được cấu hình để:

+ Build và kiểm thử mã nguồn tự động sau mỗi lần commit.

+ Phân tích mã nguồn để kiểm tra chất lượng và bảo mật.

+ Triển khai (deploy) ứng dụng lên các môi trường staging hoặc production.

+ Gửi thông báo qua Slack, Email hoặc các nền tảng khác khi pipeline thành công hoặc thất bại.

+ Tạo Pull Request hoặc tự động hợp nhất (merge) các nhánh khi đạt đủ điều kiện.

*- Tích hợp bảo mật mạnh mẽ:* GitHub Actions cung cấp các tính năng bảo mật cần thiết để xử lý các thông tin nhạy cảm. Bạn có thể sử dụng GitHub Secrets để lưu trữ các biến môi trường, token API, hoặc các khóa bảo mật một cách an toàn. Các bí mật này chỉ được truy cập bởi các workflow được ủy quyền và không bị lộ ra ngoài mã nguồn. Điều này giúp đảm bảo rằng các thông tin quan trọng luôn được bảo vệ, ngay cả trong các dự án mã nguồn mở.

## Tổng quan về GitHub API và Webhook

### GitHub API

GitHub API là một tập hợp các giao diện lập trình ứng dụng (API) mạnh mẽ, được thiết kế để cho phép các ứng dụng bên thứ ba truy cập, thao tác và tương tác với dữ liệu trên nền tảng GitHub một cách có lập trình. API này cung cấp khả năng điều khiển gần như tất cả các chức năng cốt lõi của GitHub, từ việc quản lý kho mã nguồn, người dùng, cho đến các sự kiện trong quy trình phát triển. Việc tích hợp GitHub API là nền tảng cốt lõi của CodeFlow, giúp hệ thống của chúng tôi không chỉ đọc dữ liệu mà còn chủ động thực hiện các hành động trên GitHub.



GitHub cung cấp nhiều loại API khác nhau, mỗi loại phục vụ một mục đích chuyên biệt:

*- REST API:* Đây là loại API chính, sử dụng kiến trúc RESTful truyền thống với các phương thức HTTP (GET, POST, PUT, DELETE) để thực hiện các thao tác CRUD (Create, Read, Update, Delete) với các tài nguyên. REST API được sử dụng để truy vấn dữ liệu lịch sử, lấy thông tin chi tiết về các commit và pull request, và tạo các bình luận trên mã nguồn. Ví dụ: GET */repos/{owner}/{repo}/commits* để lấy danh sách các commit của một kho mã nguồn.

*- GraphQL API:* Một API hiện đại hơn, cho phép các ứng dụng chỉ truy vấn những trường dữ liệu cụ thể mà chúng cần trong một lần gọi duy nhất. Điều này giúp tối ưu hóa hiệu suất và giảm lượng dữ liệu truyền tải không cần thiết.

*- Git API:* Là một phần của REST API, Git API cho phép tương tác trực tiếp với các đối tượng Git cấp thấp như commits, trees và blobs. API này hữu ích khi cần phân tích sâu về cấu trúc và lịch sử của kho mã nguồn.

Việc sử dụng GitHub API cho phép các nhà phát triển xây dựng các công cụ và dịch vụ tự động hóa mạnh mẽ. Các ứng dụng có thể bao gồm:

*- Thu thập dữ liệu:* Tự động lấy thông tin về các hoạt động của dự án như số lượng commit, pull request, và số liệu về đóng góp cá nhân để phân tích hiệu suất nhóm.

*- Quản lý dự án:* Tự động tạo và cập nhật các vấn đề (issues), gán nhãn (labels), và quản lý các pull request.

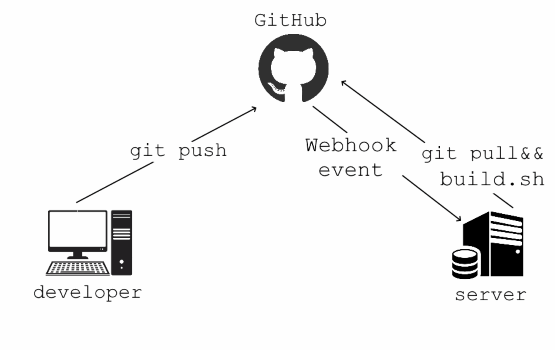
*- Tích hợp và tự động hóa:* Sử dụng API để tương tác với các hệ thống CI/CD, gửi thông báo, hoặc tự động tạo các bản dựng (builds).

Một số ví dụ về các API cụ thể của GitHub:

|  |  |  |  |
| --- | --- | --- | --- |
| Chức năng | API | Phương thức | Mô tả |
| Kho mã nguồn | /repos/{owner}/{repo} | GET | Lấy thông tin chi tiết về một repository. |
|  | /repos/{owner}/{repo}/contributors | GET | Liệt kê danh sách các thành viên đóng góp. |
| Commit | /repos/{owner}/{repo}/commits | GET | Lấy danh sách các commit của repository. |
|  | /repos/{owner}/{repo}/commits/  {commit\_sha} | GET | Lấy thông tin chi tiết một commit. |
| Pull Request | /repos/{owner}/{repo}/pulls | GET | Lấy danh sách các pull request. |
|  | /repos/{owner}/{repo}/pulls/  {pull\_number} | GET | Lấy thông tin chi tiết một pull request. |
|  | /repos/{owner}/{repo}/issues/  {issue\_number}/comments | POST | Thêm bình luận vào pull request. |
| Webhook | /repos/{owner}/{repo}/hooks | GET | Liệt kê các webhook đã cấu hình. |
|  | /repos/{owner}/{repo}/hooks | POST | Tạo một webhook mới cho repository. |

### Github Webhook

GitHub Webhook là một cơ chế cho phép GitHub tự động gửi thông báo và dữ liệu đến một máy chủ hoặc dịch vụ bên ngoài khi có sự kiện nhất định xảy ra trong kho lưu trữ (repository) hoặc tổ chức (organization). Thay vì phải liên tục gửi yêu cầu đến API để kiểm tra (polling), webhook giúp hệ thống bên ngoài nhận thông tin gần như ngay lập tức (near real-time), từ đó phản ứng nhanh chóng với các thay đổi trong mã nguồn hoặc hoạt động dự án.



Webhook hoạt động dựa trên mô hình sự kiện – phản hồi (event-driven). Quy trình diễn ra như sau:

*- Cấu hình webhook:* Người quản trị repository hoặc organization đăng ký webhook trong phần Settings.

*- Xác định URL của endpoint (máy chủ nhận dữ liệu):* Chọn sự kiện muốn theo dõi (ví dụ: push, pull request, issues, releases) hoặc chọn tùy chọn Send me everything để nhận tất cả sự kiện.

*- Sự kiện xảy ra:* Khi sự kiện đã cấu hình xảy ra, GitHub tự động tạo một thông điệp dữ liệu (payload) dưới dạng JSON.

*- Gửi dữ liệu đến endpoint:* GitHub gửi yêu cầu HTTP POST chứa payload đến URL đã đăng ký.Payload bao gồm thông tin chi tiết về sự kiện, người thực hiện, kho lưu trữ liên quan, và các dữ liệu cần thiết để xử lý.

*- Xử lý dữ liệu:* Máy chủ nhận dữ liệu phân tích payload và thực hiện các tác vụ tương ứng, ví dụ: kích hoạt pipeline CI/CD, gửi thông báo đến Slack, cập nhật dashboard tiến độ.

*Payload từ GitHub Webhook thường bao gồm các thành phần chính:*

*- action:* Hành động cụ thể của sự kiện (ví dụ: opened, closed, edited).

*- repository:* Thông tin kho lưu trữ (tên, URL, mô tả, quyền truy cập).

*- sender:* Thông tin về người thực hiện hành động (tên, ID, avatar).

*- event-specific data:* Dữ liệu chi tiết tùy thuộc loại sự kiện, ví dụ: commit mới, nội dung pull request, issue vừa mở,…

Để đảm bảo tính an toàn, GitHub hỗ trợ cơ chế xác minh dữ liệu webhook thông qua secret token:

- Khi cấu hình webhook, người dùng thiết lập một chuỗi bí mật (**secret**).

- GitHub sẽ sử dụng secret này để tạo chữ ký HMAC-SHA256 đính kèm trong header X-Hub-Signature-256 khi gửi yêu cầu.

- Máy chủ nhận dữ liệu sẽ tính toán lại chữ ký dựa trên payload nhận được và secret đã biết, sau đó so sánh để xác thực nguồn gửi.

*Ứng dụng thực tiễn*

- Webhook được ứng dụng rộng rãi trong các hệ thống tự động hóa và tích hợp. Chúng là nền tảng của các hệ thống CI/CD, nơi một commit mới sẽ ngay lập tức kích hoạt một pipeline build và kiểm thử. Webhook cũng được dùng để cập nhật các công cụ quản lý dự án (như Jira, Trello) hoặc gửi thông báo đến các kênh liên lạc (như Slack, Discord) khi có sự kiện quan trọng.

- Sự kết hợp chặt chẽ giữa GitHub API (để tương tác chủ động) và GitHub Webhook (để phản ứng thụ động với các sự kiện) chính là chìa khóa để xây dựng các hệ thống tự động hóa mạnh mẽ và hiệu quả trên nền tảng GitHub.

## Công cụ đánh giá mã nguồn

Công cụ đánh giá mã nguồn (Source Code Analysis Tools) là tập hợp các phần mềm hoặc dịch vụ hỗ trợ lập trình viên và nhóm phát triển phân tích, kiểm tra và đánh giá chất lượng mã nguồn một cách tự động hoặc bán tự động. Các công cụ này giúp phát hiện sớm lỗi cú pháp, lỗi logic, các vấn đề về hiệu suất, bảo mật, khả năng bảo trì và mức độ tuân thủ các tiêu chuẩn lập trình.

Trong bối cảnh phát triển phần mềm hiện đại, đặc biệt là khi áp dụng mô hình DevOps và quy trình Continuous Integration / Continuous Deployment (CI/CD), công cụ đánh giá mã nguồn đóng vai trò then chốt trong việc đảm bảo chất lượng và tính ổn định của sản phẩm trước khi triển khai.

### Mục tiêu đánh giá mã nguồn

Đánh giá mã nguồn là quá trình phân tích và thẩm định chất lượng của mã dựa trên các tiêu chí kỹ thuật và chuẩn lập trình. Mục tiêu là:

*- Đảm bảo tính đúng đắn và tin cậy của mã nguồn:* Phát hiện các lỗi tiềm ẩn gây gián đoạn hoặc hành vi bất thường, từ đó củng cố độ ổn định của ứng dụng.

*- Nâng cao khả năng bảo trì và mở rộng:* Giúp mã dễ đọc, dễ hiểu và dễ thay đổi. Quá trình này giúp giảm thiểu "nợ kỹ thuật" (technical debt), tức là chi phí phát sinh để sửa chữa các vấn đề chất lượng trong tương lai.

*- Tăng cường bảo mật:* Phát hiện và khắc phục các lỗ hổng bảo mật ngay từ giai đoạn phát triển, trước khi chúng có thể bị kẻ xấu khai thác.

*- Tối ưu hóa chi phí và hiệu suất:* Phát hiện lỗi sớm giúp tiết kiệm đáng kể thời gian và chi phí so với việc sửa lỗi ở giai đoạn cuối của vòng đời phát triển.

### Phân loại công cụ đánh giá mã nguồn

Các công cụ đánh giá mã nguồn có thể được phân loại thành ba nhóm chính, mỗi nhóm có một phương thức hoạt động và mục tiêu riêng:

*- Phân tích tĩnh (Static Code Analysis):* là phương pháp phân tích mã nguồn mà không cần thực thi chương trình. Nó dựa vào cú pháp, cấu trúc mã và một bộ quy tắc lập trình đã định sẵn để tìm kiếm các vấn đề. Quá trình này diễn ra một cách tự động và thường là một phần của quy trình CI/CD. Các vấn đề thường phát hiện như: lỗi cú pháp*,* vấn đề về coding style (quy chuẩn mã hóa), lỗ hổng bảo mật tiềm ẩn, mã dư thừa hoặc phức tạp quá mức…

Ví dụ: SonarQube, ESLint (JavaScript), Checkstyle (Java), Pylint (Python),…

*- Phân tích động (Dynamic Code Analysis):* là phương pháp phân tích mã nguồn bằng cách thực thi chương trình trong một môi trường giả lập hoặc môi trường thực tế. Quá trình này giúp phát hiện các vấn đề chỉ xuất hiện trong quá trình chạy (runtime), đặc biệt là các vấn đề liên quan đến hiệu suất và tương tác với hệ thống.

Ví dụ: Valgrind, JProfiler, AppDynamics.

*- Phân tích hỗn hợp (Hybrid Analysis):* là sự kết hợp giữa cả hai phương pháp phân tích tĩnh và động để tăng độ chính xác và phạm vi phát hiện. Công cụ phân tích hỗn hợp thường sử dụng kết quả từ phân tích tĩnh để hướng dẫn quá trình phân tích động, từ đó tìm kiếm các lỗ hổng bảo mật phức tạp hoặc các lỗi logic tinh vi.

### Quy trình đánh giá mã nguồn

Một công cụ đánh giá mã nguồn hoạt động như một hệ thống tự động hóa, tuân theo một quy trình chặt chẽ để phân tích và cung cấp phản hồi về chất lượng mã. Quy trình này thường bao gồm các giai đoạn sau:

*- Giai đoạn thu thập mã nguồn:* Đây là bước khởi đầu, nơi công cụ tiếp cận và lấy mã nguồn cần phân tích. Quy trình này có thể thực hiện theo hai cách chính:

+ Truy cập cục bộ (Local Access): Công cụ trực tiếp đọc và phân tích mã nguồn từ một thư mục cụ thể trên máy tính hoặc máy chủ.

+ Tích hợp với hệ thống quản lý mã nguồn (VCS Integration): Đây là phương pháp phổ biến nhất trong môi trường làm việc nhóm hiện đại. Công cụ sẽ kết nối với các hệ thống quản lý mã nguồn như GitHub, GitLab hoặc Bitbucket thông qua các API hoặc plugin. Điều này cho phép công cụ tự động lấy mã nguồn mới nhất mỗi khi có một commit hoặc pull request. Việc tích hợp này đảm bảo rằng mọi thay đổi đều được kiểm tra một cách nhất quán và liên tục.

*- Giai đoạn phân tích cú pháp (Parsing):* Sau khi thu thập mã nguồn, công cụ sẽ tiến hành phân tích cú pháp để hiểu cấu trúc của mã. Thay vì đọc mã dưới dạng văn bản thuần, công cụ sẽ chuyển đổi mã nguồn thành một cấu trúc dữ liệu gọi là Cây Cú pháp Trừu tượng (Abstract Syntax Tree - AST).

+ Cấu trúc của AST: AST biểu diễn cấu trúc ngữ pháp của mã nguồn dưới dạng một cây, trong đó các node của cây là các cấu trúc lập trình như hàm, biến, vòng lặp, biểu thức, …

+ Lợi ích của AST: Việc chuyển đổi sang AST giúp công cụ dễ dàng thực hiện các thao tác phân tích phức tạp. Nó có thể duyệt qua cấu trúc cây để kiểm tra logic, tìm kiếm các mẫu mã không phù hợp, hoặc xác định các mối quan hệ giữa các thành phần khác nhau của mã nguồn một cách hiệu quả hơn nhiều so với việc phân tích chuỗi ký tự.

*- Giai đoạn áp dụng quy tắc kiểm tra:* Đây là giai đoạn cốt lõi của quá trình phân tích. Công cụ sẽ so sánh AST của mã nguồn với một bộ quy tắc kiểm tra (ruleset) đã được định sẵn. Bộ quy tắc này được thiết kế để tìm kiếm các vấn đề cụ thể, bao gồm:

+ Quy chuẩn mã hóa (Coding Standards): Các quy tắc về định dạng mã, đặt tên biến, cấu trúc hàm... Ví dụ: PEP8 cho Python, Google Style Guide cho Java.

+ Lỗi và Bug: Các quy tắc để tìm lỗi logic tiềm ẩn, ví dụ như một biến được khai báo nhưng không bao giờ được sử dụng, hoặc một đoạn mã không thể truy cập được.

+ Lỗ hổng bảo mật: Các quy tắc để phát hiện các lỗ hổng phổ biến như SQL injection, XSS, hoặc việc sử dụng các hàm không an toàn.

+ Vấn đề hiệu suất: Các quy tắc để xác định những đoạn mã có thể gây ra hiệu suất kém, chẳng hạn như các vòng lặp lồng nhau quá sâu.

*- Giai đoạn phát hiện và báo cáo:* Khi các quy tắc được áp dụng, công cụ sẽ ghi lại tất cả các vấn đề được tìm thấy. Kết quả này sau đó được tổng hợp và trình bày dưới dạng một báo cáo chi tiết. Báo cáo này thường bao gồm:

+ Thông tin chi tiết về vấn đề: Tên lỗi, mô tả, vị trí chính xác (số dòng) trong mã nguồn.

+ Mức độ nghiêm trọng: Các vấn đề được phân loại theo mức độ nghiêm trọng (ví dụ: Minor, Major, Critical, Blocker).

+ Đề xuất sửa đổi (Fix Suggestion): Nhiều công cụ hiện đại còn cung cấp các gợi ý cụ thể về cách khắc phục vấn đề.

*- Giai đoạn tích hợp vào quy trình CI/CD:* Để tối đa hóa hiệu quả, các công cụ đánh giá mã nguồn được tích hợp chặt chẽ vào quy trình CI/CD.

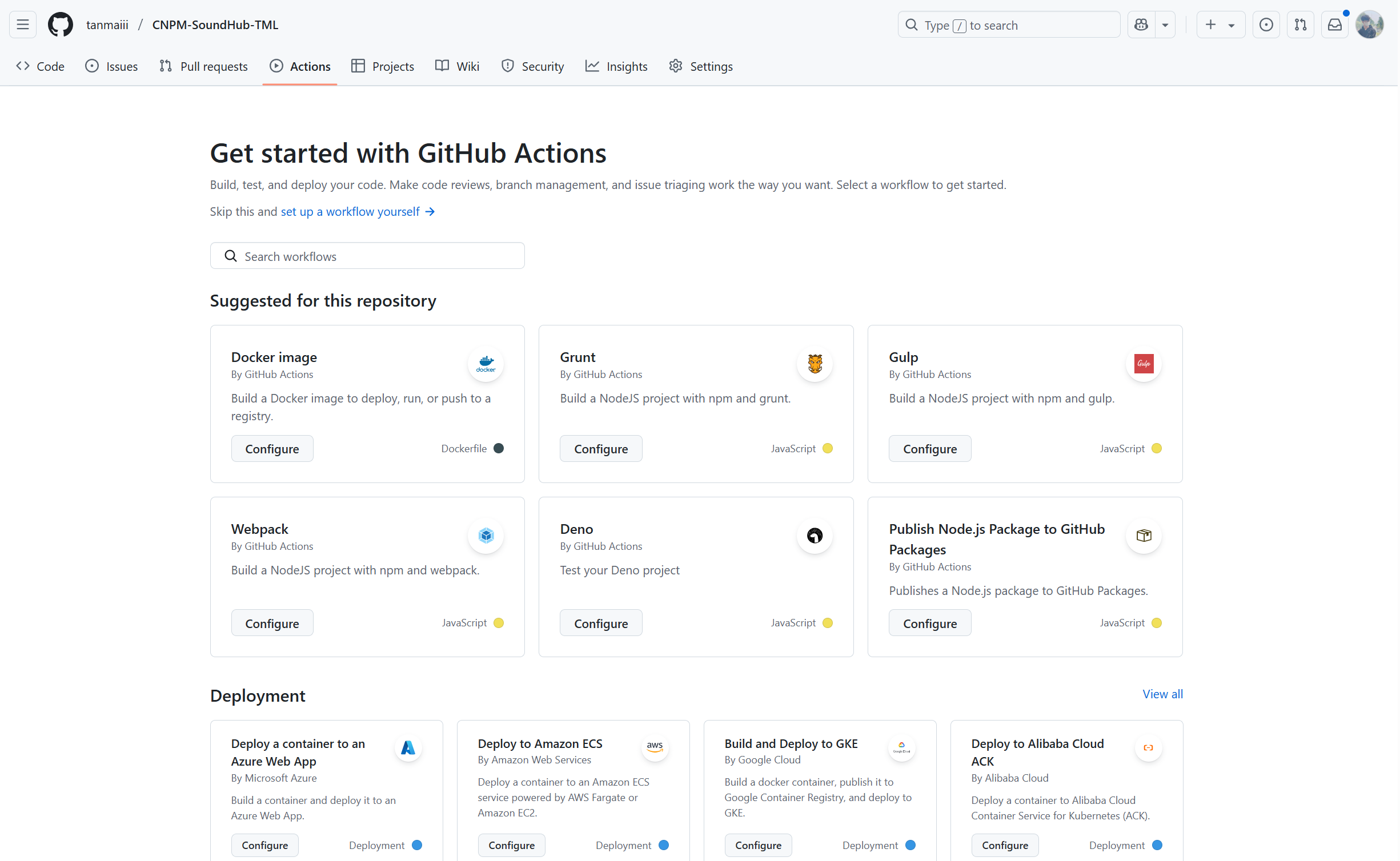
+ Kích hoạt tự động: Công cụ sẽ tự động chạy mỗi khi có một sự kiện quan trọng xảy ra, chẳng hạn như lập trình viên push mã mới hoặc tạo một pull request.

+ Điều kiện chấp nhận (Quality Gate): Hệ thống CI/CD có thể được cấu hình để kiểm tra kết quả báo cáo. Nếu công cụ phát hiện lỗi nghiêm trọng (ví dụ: một lỗi blocker), quy trình build có thể bị dừng lại và ngăn mã lỗi được hợp nhất vào nhánh chính.

+ Phản hồi tức thì: Lập trình viên sẽ nhận được phản hồi ngay lập tức về chất lượng mã của mình, giúp họ sửa lỗi kịp thời trước khi mã được tích hợp sâu hơn vào dự án.

### Kết hợp công cụ đánh giá mã nguồn với GitHub Actions

Việc tích hợp công cụ đánh giá mã nguồn với GitHub Actions là một phương pháp luận cốt lõi trong quy trình CI/CD hiện đại, nhằm tự động hóa toàn bộ quá trình kiểm tra chất lượng mã nguồn ngay khi có thay đổi trong kho lưu trữ. Sự kết hợp này tạo ra một "cổng chất lượng" (quality gate) tự động, giúp phát hiện và xử lý lỗi một cách chủ động, ngăn chặn mã kém chất lượng được hợp nhất vào nhánh chính của dự án.



Một quy trình tích hợp giữa công cụ đánh giá mã nguồn và GitHub Actions thường bao gồm các giai đoạn sau:

*- Cấu hình workflow github actions:* Quy trình bắt đầu bằng việc tạo một tệp tin cấu hình YAML trong thư mục .github/workflows của kho mã nguồn. Tệp tin này định nghĩa các công việc (jobs) và các bước (steps) cần thực hiện. Workflow được cấu hình để tự động kích hoạt bởi các sự kiện trên GitHub. Phổ biến nhất là pull\_request (khi một pull request được mở hoặc cập nhật) hoặc push (khi mã nguồn được đẩy lên một nhánh cụ thể). Sự kiện này đảm bảo rằng mọi thay đổi đều được kiểm tra.

- *Thiết lập và thực thi phân tích:* Trong các bước của workflow, công cụ đánh giá mã nguồn (ví dụ: SonarQube Scanner, ESLint, Pylint) sẽ được cài đặt và cấu hình. Quá trình này có thể thực hiện thông qua các lệnh shell hoặc sử dụng các action có sẵn trên GitHub Marketplace. Sau khi cài đặt, công cụ sẽ được chạy trên mã nguồn mới nhất. Đối với các công cụ phân tích tĩnh, quá trình này bao gồm việc phân tích cú pháp mã nguồn thành cây cú pháp trừu tượng (AST) và so sánh với bộ quy tắc đã định sẵn.

*- Sinh báo cáo và xử lý kết quả:* Kết quả phân tích sẽ được tổng hợp thành một báo cáo chi tiết. Đối với các công cụ như SonarQube, báo cáo này sẽ được gửi đến máy chủ SonarQube Server hoặc SonarCloud để hiển thị trên dashboard. Kết quả kiểm tra cũng được phản hồi trực tiếp trên giao diện của github pull request dưới dạng các "checks" (kiểm tra). Tùy thuộc vào kết quả, check có thể hiển thị trạng thái "passed" (đã qua) hoặc "failed" (thất bại). Đây là một cơ chế quan trọng. Dựa trên các ngưỡng chất lượng đã định, workflow có thể được cấu hình để tự động đánh dấu quá trình kiểm tra là thất bại. Điều này có thể được sử dụng để chặn quá trình hợp nhất mã, yêu cầu lập trình viên phải sửa lỗi trước khi mã được chấp nhận.

Lợi ích khi tích hợp:

*- Phản hồi tức thời và liên tục:* Lập trình viên nhận được kết quả phân tích chỉ vài phút sau khi đẩy mã, cho phép họ sửa lỗi kịp thời.

*- Đảm bảo chất lượng liên tục:* Tự động hóa việc kiểm tra mọi thay đổi, từ đó duy trì một tiêu chuẩn chất lượng nhất quán và cao cho toàn bộ dự án.

*- Nâng cao tính minh bạch:* Kết quả phân tích được hiển thị công khai trên GitHub, giúp mọi thành viên trong nhóm dễ dàng theo dõi và giám sát chất lượng mã.

*- Tiết kiệm thời gian và công sức:* Giảm thiểu đáng kể thời gian và công sức cần thiết cho việc kiểm tra mã thủ công và debug.

## SonarQube và vai trò trong đánh giá chất lượng mã nguồn

### Giới thiệu về SonarQube

SonarQube là một nền tảng mã nguồn mở được sử dụng để phân tích và đánh giá chất lượng mã nguồn liên tục (Continuous Code Quality). Nó đóng vai trò là một "người gác cổng" chất lượng, giúp các nhà phát triển và nhóm dự án duy trì các tiêu chuẩn mã nguồn cao, phát hiện sớm các lỗi, lỗ hổng bảo mật và code smells trước khi chúng gây ra vấn đề nghiêm trọng. SonarQube tích hợp vào quy trình phát triển, quét mã nguồn tự động và cung cấp một dashboard trực quan với các chỉ số chi tiết.

A screenshot of a computer

AI-generated content may be incorrect.

### Các khái niệm và chỉ số cốt lõi

SonarQube đánh giá chất lượng mã nguồn dựa trên một loạt các chỉ số và khái niệm cốt lõi, được chia thành các danh mục chính như sau:

*- Độ tin cậy (Reliability):* Đánh giá mức độ ổn định của ứng dụng và các lỗi tiềm ẩn có thể gây ra sự cố. Các chỉ số bao gồm Bugs (lỗi), Reliability Rating (xếp hạng độ tin cậy) và Reliability Remediation Effort (thời gian khắc phục lỗi).

*- Bảo mật (Security):* Phân tích các lỗ hổng bảo mật trong mã nguồn có thể bị kẻ xấu khai thác. Các chỉ số bao gồm Vulnerabilities (lỗ hổng), Security Rating (xếp hạng bảo mật) và Security Remediation Effort (thời gian khắc phục bảo mật). Ngoài ra, SonarQube còn xác định các Security Hotspots (điểm nóng bảo mật) cần được xem xét thủ công.

*- Bảo trì (Maintainability):* Đánh giá mức độ dễ dàng để đọc, hiểu và thay đổi mã nguồn. Các chỉ số bao gồm Code Smells (các vấn đề về thiết kế, cấu trúc), Maintainability Rating (xếp hạng bảo trì) và Technical Debt (nợ kỹ thuật), ước tính thời gian cần để khắc phục tất cả các code smells.

*- Độ bao phủ của kiểm thử (Test Coverage):* Đo lường tỷ lệ phần trăm các dòng mã được bao phủ bởi các bài kiểm thử tự động (unit tests). Một chỉ số quan trọng khác là Duplicated Lines (số dòng mã trùng lặp), cho thấy khả năng tái sử dụng mã kém.

### Quy trình đánh giá và vai trò trong dự án

Quy trình đánh giá với SonarQube thường được tự động hóa bằng cách tích hợp vào pipeline CI/CD.

*- Tích hợp tự động:* SonarQube Scanner được cấu hình trong các công cụ CI/CD như GitHub Actions. Khi một lập trình viên push mã nguồn, scanner sẽ tự động kích hoạt để quét và phân tích.

*- Hiển thị kết quả:* Kết quả phân tích được gửi về SonarQube Server và hiển thị trên một dashboard trực quan, nơi các nhà phát triển có thể xem chi tiết về các lỗi, lỗ hổng và "mùi mã" được tìm thấy.

*- Áp dụng Cổng chất lượng (Quality Gate):* Đây là một tính năng then chốt của SonarQube. Quality Gate cho phép định nghĩa các ngưỡng chất lượng tối thiểu mà mã nguồn phải đạt được để được coi là chấp nhận được. Ví dụ, một dự án có thể bị đánh trượt nếu độ bao phủ kiểm thử dưới 80% hoặc có bất kỳ lỗi nghiêm trọng nào. Cơ chế này đảm bảo rằng chỉ những mã nguồn đạt tiêu chuẩn mới được phép hợp nhất (merge) vào nhánh chính.

### Tính hợp SonarQube trong quy trình CI/CD

Một quy trình tích hợp liên tục (CI) với SonarQube Server được xây dựng để tự động hóa việc phân tích chất lượng mã nguồn ngay trong chu trình phát triển. Quy trình này thường bao gồm các bước sau: ([SonarQube analysis overview | Documentation](https://docs.sonarsource.com/sonarqube-server/10.8/analyzing-source-code/analysis-overview/))

A diagram of a process flow

AI-generated content may be incorrect.

*1. Đẩy thay đổi mã nguồn (Push Changes):* Một lập trình viên hoàn thành công việc và đẩy các thay đổi lên một nhánh trên kho mã nguồn từ xa (remote repository), ví dụ như trên GitHub hoặc GitLab.

*2. Kích hoạt Pipeline CI (Trigger CI Pipeline):* Sự kiện "push" này sẽ kích hoạt pipeline CI/CD. Việc kích hoạt có thể diễn ra theo hai cách chính:

- Sử dụng Webhook: Hệ thống quản lý mã nguồn (SCM) như GitHub sẽ gửi một webhook đến công cụ CI/CD (ví dụ: Jenkins, GitHub Actions) để thông báo rằng có một sự kiện đã xảy ra.

- Polling: Công cụ CI/CD định kỳ kiểm tra kho mã nguồn để phát hiện các thay đổi mới.

*3. Clone và Checkout Repository:* Pipeline sẽ clone kho mã nguồn từ xa về máy chủ CI/CD (CI/CD host) và chuyển đến nhánh làm việc cụ thể. Tại bước này, toàn bộ mã nguồn và siêu dữ liệu (metadata) của SCM được sao chép vào môi trường cục bộ.

*4. Xây dựng mã nguồn (Build Code):* Đối với các ngôn ngữ lập trình được biên dịch (như Java, C#), pipeline sẽ chạy các lệnh build để tạo ra các artifact (ví dụ: file .jar, .war, .exe). Bước này đảm bảo mã nguồn có thể biên dịch thành công trước khi phân tích.

*5. Phân tích mã nguồn với SonarQube Scanner:* Pipeline sẽ thực thi SonarQube Scanner phù hợp với ngôn ngữ lập trình của dự án. Scanner này sẽ tiến hành phân tích mã nguồn một cách tĩnh (static analysis) để tìm kiếm các lỗi, lỗ hổng bảo mật, và "mùi" mã.

*6. Gửi và Xử lý kết quả phân tích (Send Analysis Results):* Sau khi quá trình quét hoàn tất, scanner sẽ gửi tất cả kết quả phân tích đến SonarQube Server. Server sẽ tiếp nhận dữ liệu này, xử lý và tính toán các chỉ số chất lượng, sau đó lưu trữ kết quả vào cơ sở dữ liệu.

*7. Nhận kết quả từ Quality Gate (Receive Quality Gate Result):* Đây là một bước tùy chọn nhưng rất quan trọng trong quy trình CI/CD hiện đại. SonarQube Server sẽ tính toán kết quả của Quality Gate (cổng chất lượng) dựa trên các ngưỡng đã định trước. Sau đó, Server gửi kết quả này về lại pipeline.

*8. Điều kiện tiếp tục hoặc dừng Pipeline (Pipeline Continuation):*

*+* Nếu Quality Gate thành công (succeeds), tức là mã nguồn mới đáp ứng các tiêu chuẩn chất lượng, pipeline sẽ tiếp tục các bước tiếp theo (ví dụ: chạy kiểm thử end-to-end, triển khai).

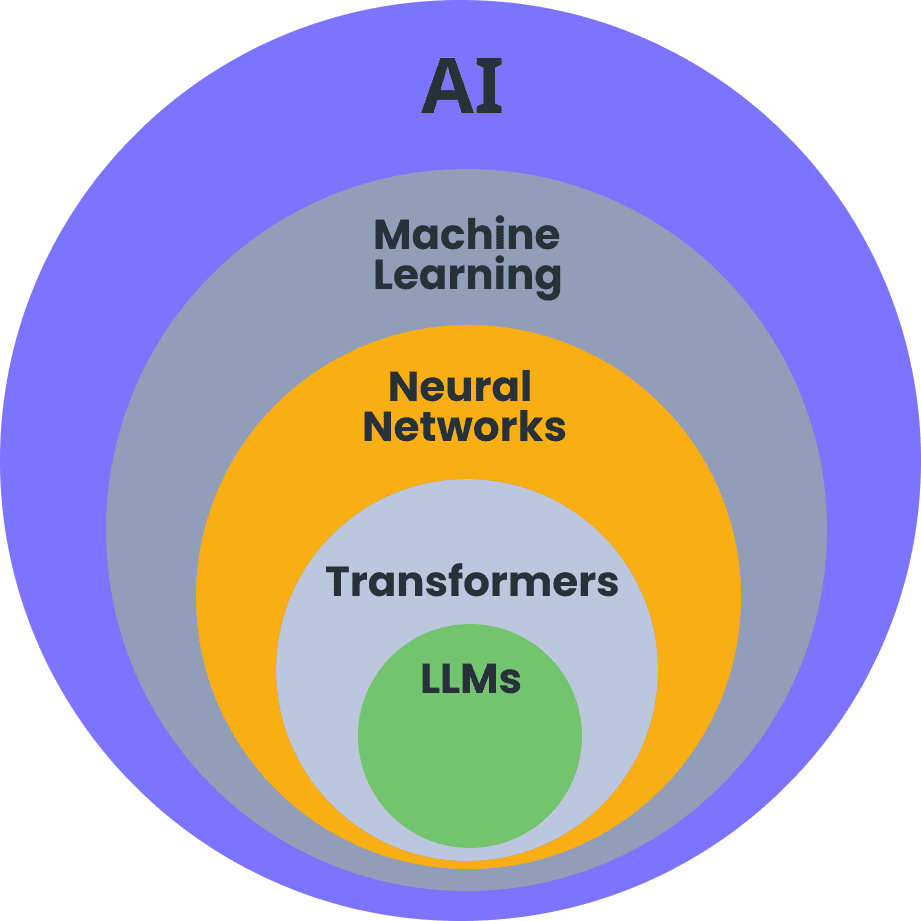
+ Nếu Quality Gate thất bại (fails), tức là mã nguồn mới có vấn đề nghiêm trọng, pipeline sẽ bị dừng lại. Điều này ngăn chặn việc mã kém chất lượng được hợp nhất vào nhánh chính, đảm bảo tính ổn định của dự án.

## Tích hợp AI vào quy trình đánh giá lập trình

Với sự phát triển mạnh mẽ của Trí tuệ nhân tạo (AI), đặc biệt là các mô hình ngôn ngữ lớn (Large Language Models - LLMs), việc đánh giá mã nguồn đã vượt ra khỏi giới hạn của các công cụ phân tích tĩnh truyền thống. Các mô hình này không chỉ có khả năng nhận diện các lỗi cú pháp mà còn hiểu được ngữ cảnh, logic nghiệp vụ của mã nguồn để đưa ra các nhận xét sâu sắc và toàn diện hơn.

### Kiến trúc và cơ chế hoạt động

Kiến trúc cốt lõi của LLM là Transformer, được giới thiệu vào năm 2017. Điểm nổi bật của kiến trúc này là cơ chế "self-attention". Cơ chế này cho phép mô hình xử lý toàn bộ một chuỗi đầu vào cùng một lúc, thay vì xử lý tuần tự, từ đó hiểu được mối quan hệ và ngữ cảnh giữa các token ở xa nhau trong một chuỗi. Điều này giúp LLM có khả năng phân tích và tạo ra các câu trả lời mạch lạc, có tính logic cao.



Quá trình phát triển một LLM thường bao gồm hai giai đoạn chính:

*- Pre-training (Huấn luyện sơ bộ):* Mô hình được huấn luyện trên một lượng lớn dữ liệu không gắn nhãn để học các mẫu ngôn ngữ tổng quát. Giai đoạn này giúp mô hình nắm bắt được tri thức phổ quát và các quy tắc cơ bản của ngôn ngữ.

*- Fine-tuning (Tinh chỉnh):* Sau khi được huấn luyện sơ bộ, mô hình được tinh chỉnh trên các tập dữ liệu nhỏ hơn, có gắn nhãn, để thực hiện các tác vụ chuyên biệt như trả lời câu hỏi, tóm tắt văn bản hoặc phân tích mã nguồn.

### Ứng dụng trong đánh giá mã nguồn

Khả năng hiểu ngữ cảnh và logic của LLM đã mở ra một phương pháp tiếp cận mới trong việc đánh giá mã nguồn, vượt qua những giới hạn của các công cụ phân tích tĩnh truyền thống.

*- Phân tích ngữ cảnh và logic:* Thay vì chỉ dựa vào các quy tắc lập trình cứng nhắc, LLM có thể phân tích mã nguồn trong bối cảnh của toàn bộ dự án hoặc một chức năng cụ thể. Điều này cho phép mô hình phát hiện các lỗi logic phức tạp, các vấn đề về thiết kế kiến trúc hoặc các trường hợp ngoại lệ hiếm gặp mà các công cụ tĩnh khó tìm ra.

*- Đánh giá chất lượng và tuân thủ thực tiễn tốt nhất:* LLM có thể so sánh mã nguồn với các tiêu chuẩn và thực tiễn tốt nhất mà nó đã học được từ hàng triệu dự án mã nguồn mở. Mô hình có thể đưa ra các nhận xét chi tiết về tính dễ đọc, cấu trúc code, cách đặt tên biến và đề xuất các phương pháp lập trình hiệu quả hơn.

*- Hỗ trợ bảo mật:* LLM được huấn luyện để nhận diện các mẫu tấn công và lỗ hổng bảo mật phổ biến. Nó có thể xác định các đoạn mã có nguy cơ cao, ví dụ như việc sử dụng các hàm không an toàn hoặc xử lý thông tin nhạy cảm không đúng cách, và đưa ra cảnh báo chính xác hơn.

*- Tạo phản hồi mang tính giáo dục:* Một trong những ưu điểm lớn nhất là khả năng của LLM trong việc tạo ra các phản hồi bằng ngôn ngữ tự nhiên. Phản hồi này không chỉ chỉ ra vấn đề mà còn giải thích rõ ràng nguyên nhân, đưa ra các ví dụ mã nguồn đã được sửa và giải thích tại sao cách sửa đó tốt hơn. Điều này biến quá trình đánh giá mã thành một công cụ học tập và hướng dẫn hiệu quả cho các lập trình viên.

### Các mô hình ngôn ngữ lớn có thể tích hợp

Các LLM được huấn luyện trên một lượng lớn dữ liệu văn bản và mã nguồn, cho phép chúng hiểu, xử lý và tạo ra văn bản tự nhiên, đồng thời phân tích cấu trúc mã nguồn một cách tinh vi. Một số mô hình phổ biến có thể được tích hợp vào quy trình đánh giá mã nguồn bao gồm:

*- GPT (Generative Pre-trained Transformer):* Nổi bật với khả năng hiểu và tạo ra văn bản tự nhiên có độ phức tạp cao. Các phiên bản như GPT-4 có thể phân tích mã nguồn một cách chi tiết, đưa ra các nhận xét sâu sắc về kiến trúc, bảo mật và hiệu suất.

*- Code Llama*: Là một mô hình ngôn ngữ được thiết kế chuyên biệt cho việc tạo và phân tích mã nguồn. Code Llama đặc biệt hiệu quả trong các tác vụ liên quan đến lập trình, như hoàn thành mã, gỡ lỗi và giải thích code.

*- Gemini:* Đây là một mô hình đa phương thức tiên tiến được phát triển bởi Google, có khả năng xử lý đồng thời văn bản, hình ảnh, video và âm thanh. Trong ngữ cảnh đánh giá mã nguồn, Gemini có thể phân tích mã nguồn một cách toàn diện, nhận diện các lỗi phức tạp và đưa ra phản hồi mang tính giáo dục cao.

### Tìm hiểu về Gemini trong đánh giá code

Gemini là một mô hình ngôn ngữ lớn (LLM) tiên tiến được phát triển bởi Google, ra mắt lần đầu vào tháng 12 năm 2023. Đây là mô hình đa phương thức (multimodal) nổi bật, có khả năng xử lý và kết hợp nhiều loại dữ liệu khác nhau cùng lúc, bao gồm văn bản, mã nguồn, hình ảnh, video và âm thanh. Khả năng này giúp Gemini vượt xa các LLM truyền thống, mang lại một phương pháp tiếp cận toàn diện và thông minh hơn cho việc đánh giá mã nguồn.

A blue and purple logo

AI-generated content may be incorrect.

Lý do nên sử dụng Gemini cho đánh giá mã nguồn:

*- Hiểu ngữ cảnh đa chiều:* Gemini không chỉ đọc code mà còn có thể tham chiếu đến các tài liệu, sơ đồ hoặc hình ảnh liên quan. Ví dụ, nó có thể nhận một đoạn mã giao diện người dùng (UI) cùng với một hình ảnh thiết kế, từ đó đánh giá xem mã có khớp với thiết kế hay không. Khả năng này giúp Gemini đưa ra các nhận xét chính xác hơn và có tính ứng dụng cao hơn. Bằng cách phân tích đồng thời mã nguồn và mô tả chức năng bằng văn bản, Gemini có thể hiểu rõ hơn về logic nghiệp vụ của ứng dụng. Điều này giúp nó phát hiện các lỗi logic phức tạp hoặc các trường hợp ngoại lệ hiếm gặp mà các công cụ phân tích tĩnh khó tìm ra.

*- Phản hồi thông minh và mang tính giáo dục:* Gemini có thể không chỉ chỉ ra lỗi mà còn đề xuất các cách khắc phục bằng mã nguồn và giải thích rõ ràng lý do tại sao cách khắc phục đó là tốt nhất. Các phản hồi của Gemini có thể được tạo ra bằng ngôn ngữ tự nhiên, giúp các lập trình viên, đặc biệt là sinh viên, dễ dàng tiếp thu và học hỏi. Điều này biến quá trình đánh giá mã thành một công cụ học tập hiệu quả.

*- Tích hợp linh hoạt và mạnh mẽ:* Gemini có thể phân tích mã nguồn để tìm kiếm các điểm nghẽn về hiệu suất, đề xuất các thuật toán hiệu quả hơn hoặc các cấu trúc dữ liệu phù hợp hơn. Bằng cách phân tích các mẫu tấn công đã biết, Gemini có thể xác định các lỗ hổng bảo mật trong mã nguồn, đặc biệt là các lỗ hổng liên quan đến logic ứng dụng.

## Tổng quan về hệ thống theo dõi dự án phần mềm của sinh viên

Một hệ thống theo dõi dự án phần mềm của sinh viên là một nền tảng được thiết kế để tự động hóa quá trình quản lý, giám sát và đánh giá các dự án lập trình. Mục tiêu cốt lõi của hệ thống này là cung cấp một công cụ khách quan và hiệu quả, giúp giảng viên và sinh viên nắm bắt được tiến độ, chất lượng mã nguồn và sự đóng góp của từng thành viên trong nhóm. Hệ thống này kết hợp các công nghệ quản lý mã nguồn, tự động hóa quy trình CI/CD và trí tuệ nhân tạo để tạo ra một môi trường học tập và làm việc chuyên nghiệp.

### Mục tiêu chính

*- Tự động hóa quá trình giám sát:* Hệ thống được thiết kế để tự động thu thập và tổng hợp dữ liệu từ các nền tảng quản lý mã nguồn như GitHub, GitLab hoặc Bitbucket. Thông qua cơ chế tích hợp API và webhook, mọi hoạt động của nhóm dự án — bao gồm số lượng commit, nội dung thay đổi mã, thời điểm cập nhật, số lượng pull request, và trạng thái xử lý lỗi — đều được ghi nhận và lưu trữ trong thời gian thực. Điều này giúp giảng viên hoặc người hướng dẫn không phải thực hiện việc kiểm tra thủ công từng kho mã nguồn, tiết kiệm đáng kể thời gian và công sức, đồng thời giảm nguy cơ bỏ sót thông tin quan trọng.

*- Đánh giá khách quan:* Hệ thống cung cấp bộ chỉ số định lượng rõ ràng về tiến độ và chất lượng của dự án. Chất lượng mã nguồn được phân tích thông qua các công cụ đánh giá như SonarQube, đưa ra các thông số như code smell, technical debt, độ phức tạp của hàm (cyclomatic complexity), và mức độ bao phủ kiểm thử (test coverage). Tiến độ dự án được đánh giá dựa trên biểu đồ burndown hoặc velocity chart, còn mức độ đóng góp của từng thành viên được đo bằng số lượng và giá trị thay đổi (diff size), tỷ lệ nhiệm vụ hoàn thành, và mức độ tham gia vào quá trình code review. Nhờ đó, quá trình đánh giá trở nên minh bạch, công bằng và có cơ sở dữ liệu để đối chiếu.

*- Hỗ trợ sinh viên học tập:* Bên cạnh chức năng giám sát và đánh giá, hệ thống còn đóng vai trò là một công cụ huấn luyện (training tool). Bằng cách tích hợp các mô hình trí tuệ nhân tạo (AI) và mô hình ngôn ngữ lớn (LLM) như GPT hoặc Gemini, hệ thống có thể phân tích mã nguồn, nhận diện lỗi lập trình, gợi ý giải pháp khắc phục và đưa ra các khuyến nghị về tối ưu hóa hiệu suất hoặc cải thiện cấu trúc. Các phản hồi này được cung cấp gần như tức thời, giúp sinh viên kịp thời sửa lỗi, hiểu nguyên nhân vấn đề, và tiếp cận các thực tiễn lập trình tốt (best practices). Điều này đặc biệt hữu ích trong môi trường đào tạo, nơi việc phản hồi nhanh và kịp thời có tác động mạnh đến quá trình học tập.

*- Tăng cường minh bạch và trách nhiệm nhóm:* Hệ thống tạo ra một môi trường làm việc nhóm minh bạch bằng cách hiển thị rõ ràng tiến độ, nhiệm vụ và đóng góp của từng thành viên. Mỗi thành viên có thể dễ dàng theo dõi hoạt động của người khác, nhận thông báo khi có commit mới, pull request, hoặc khi một nhiệm vụ đã hoàn thành. Việc công khai dữ liệu này giúp nâng cao tinh thần trách nhiệm cá nhân, giảm thiểu tình trạng người tham gia nhóm nhưng không đóng góp, đồng thời khuyến khích sự hợp tác và hỗ trợ lẫn nhau để đạt mục tiêu chung.

### Kiến trúc tổng quan của hệ thống

Hệ thống được thiết kế dựa trên một kiến trúc tích hợp đa tầng (multi-layered integrated architecture). Kiến trúc này cho phép đồng bộ hóa dữ liệu một cách hiệu quả từ các nền tảng phát triển, thực hiện phân tích chất lượng mã nguồn tự động, và cung cấp phản hồi thông qua một giao diện người dùng trực quan. Hệ thống được cấu thành từ bốn mô-đun chính, mỗi mô-đun đảm nhận một vai trò chuyên biệt và tương tác chặt chẽ với các mô-đun còn lại.

1. Giao diện quản lý

Đây là lớp trình bày của hệ thống, đóng vai trò là điểm tương tác trực tiếp với người dùng cuối, bao gồm giảng viên và sinh viên. Giao diện này được xây dựng dưới dạng một ứng dụng web đa nền tảng, đảm bảo tính khả dụng trên các thiết bị khác nhau.

*- Chức năng cho Giảng viên:* Giảng viên có quyền quản trị để tạo, chỉnh sửa và quản lý các đơn vị học phần (course units), các đề tài (projects) và cấu trúc nhóm sinh viên. Giao diện cung cấp các báo cáo tổng quan và các biểu đồ thống kê định lượng về tiến độ dự án, chất lượng mã nguồn, và mức độ đóng góp cá nhân (individual contribution), hỗ trợ quá trình đánh giá khách quan và minh bạch.

*- Chức năng cho Sinh viên:* Sinh viên sử dụng giao diện để đăng ký tham gia đề tài, liên kết kho mã nguồn GitHub của nhóm, theo dõi các chỉ số chất lượng mã nguồn, và tiếp nhận các phản hồi tự động.

*- Đặc điểm kỹ thuật:* Hệ thống sử dụng cơ chế xác thực OAuth2 thông qua GitHub để đồng bộ danh tính người dùng và quyền truy cập, đảm bảo an toàn và tính toàn vẹn của dữ liệu.

2. Mô-đun tích hợp GitHub

Mô-đun này hoạt động như một lớp trừu tượng hóa, đóng vai trò là cầu nối giữa hệ thống và nền tảng GitHub. Nó đảm bảo việc thu thập và cập nhật dữ liệu một cách chính xác và theo thời gian thực.

*- GitHub REST API:* Mô-đun sử dụng GitHub REST API để thực hiện các truy vấn dữ liệu định kỳ hoặc theo yêu cầu. Các truy vấn này nhằm thu thập thông tin về cấu trúc kho mã nguồn, danh sách thành viên, lịch sử commit, và các Pull Request.

*- GitHub Webhook:* Đây là cơ chế cốt lõi để đảm bảo tính thời gian thực. Hệ thống lắng nghe các sự kiện cụ thể trên GitHub như push, pull\_request, hoặc issue update. Khi một sự kiện xảy ra, GitHub sẽ gửi một payload (gói dữ liệu) đến một endpoint (địa chỉ) đã được cấu hình sẵn trong hệ thống. Điều này kích hoạt ngay lập tức các quy trình xử lý dữ liệu và đánh giá tự động.

*- Ưu điểm:* Cơ chế này giúp giảm thiểu độ trễ dữ liệu, đồng bộ hóa giữa hoạt động thực tế trên GitHub và trạng thái hiển thị trên hệ thống quản lý, đồng thời giảm tải cho các yêu cầu truy vấn API liên tục (polling).

3. Hệ thống đánh giá mã nguồn tự động

Mô-đun này chịu trách nhiệm phân tích chất lượng mã nguồn, sử dụng một phương pháp tiếp cận kết hợp giữa phân tích tĩnh truyền thống và trí tuệ nhân tạo.

*- Phân tích tĩnh (Static Code Analysis):* Sử dụng các công cụ chuyên biệt như SonarQube để phân tích mã nguồn mà không cần thực thi chương trình. Đánh giá các chỉ số quan trọng như: Bugs, Code Smells, Technical Debt và Test Coverage.

*- Phân tích chuyên sâu bằng Trí tuệ nhân tạo (AI):* Tích hợp các Mô hình Ngôn ngữ Lớn (LLM) như Google Gemini hoặc GPT-4.AI có khả năng phân tích ngữ cảnh và logic của mã nguồn để đưa ra nhận xét thông minh, đề xuất cải thiện về cấu trúc, tối ưu hiệu suất, và khắc phục các lỗ hổng bảo mật. Phản hồi từ AI có thể được tự động đính kèm vào các Pull Request dưới dạng bình luận, cung cấp phản hồi tức thời và mang tính giáo dục cho sinh viên.

4. Hệ thống lưu trữ và xử lý dữ liệu (Data Storage & Processing Layer)

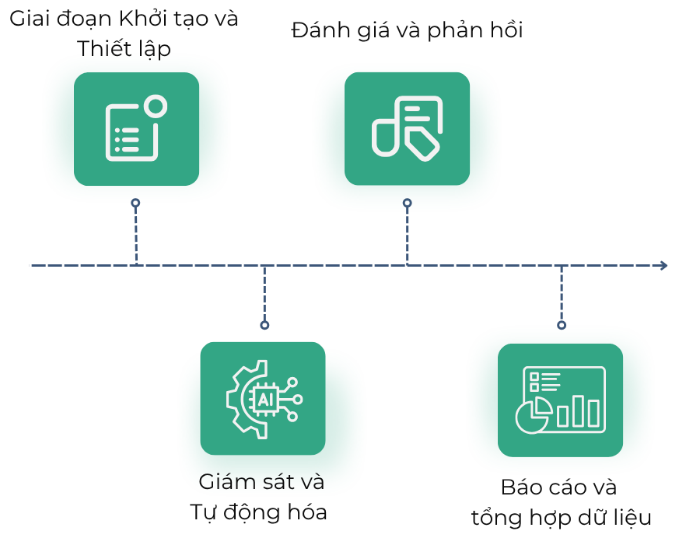
Mô-đun này là nền tảng của hệ thống, quản lý việc lưu trữ và xử lý tất cả các loại dữ liệu.

*- Lưu trữ:* Sử dụng một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) để lưu trữ dữ liệu có cấu trúc như thông tin người dùng, nhóm, đề tài. Các loại dữ liệu phi cấu trúc như log hoạt động hoặc kết quả phân tích AI có thể được lưu trữ trong cơ sở dữ liệu phi quan hệ (NoSQL).

*- Xử lý:* Lớp này thực hiện các tác vụ tiền xử lý, tổng hợp dữ liệu từ các nguồn khác nhau (GitHub, SonarQube) để tạo ra các báo cáo tổng hợp. Nó cũng chịu trách nhiệm sinh các biểu đồ thống kê trực quan về tiến độ, số lượng commit, và chất lượng mã nguồn theo thời gian, đảm bảo hệ thống có khả năng mở rộng và xử lý dữ liệu lớn một cách hiệu quả.

### Quy trình hoạt động cơ bản của hệ thống

Quy trình hoạt động của hệ thống được thiết kế để tự động hóa toàn bộ vòng đời của một dự án, từ khi bắt đầu cho đến khi kết thúc, đảm bảo việc giám sát và đánh giá diễn ra một cách liên tục và hiệu quả. Luồng hoạt động này có thể được chia thành các giai đoạn chính sau:



*Giai đoạn 1: Giai đoạn Khởi tạo và Thiết lập:* Đây là giai đoạn đầu tiên, nơi giảng viên và sinh viên thiết lập nền tảng cho dự án.

- Khởi tạo dự án (Project Initialization): Giảng viên sử dụng giao diện quản lý để tạo một môn học mới, định nghĩa các đề tài, và thiết lập các tiêu chí đánh giá.

- Đăng ký và Liên kết (Registration & Linking): Sinh viên sử dụng tài khoản GitHub để đăng nhập vào hệ thống thông qua cơ chế OAuth2. Sau khi tạo nhóm, họ sẽ liên kết kho mã nguồn GitHub của nhóm với hệ thống. Việc này sẽ cho phép hệ thống truy cập và lắng nghe các sự kiện từ kho mã nguồn.

*Giai đoạn 2: Giám sát và Tự động hóa:* Giai đoạn này là cốt lõi của hệ thống, nơi các quy trình tự động được kích hoạt để theo dõi và phân tích hoạt động của dự án theo thời gian thực.

- Lắng nghe sự kiện GitHub (GitHub Event Listening): Hệ thống sử dụng GitHub Webhook để lắng nghe các sự kiện quan trọng như push (đẩy mã), pull\_request (yêu cầu hợp nhất mã), và pull\_request\_review. Khi một sự kiện xảy ra, GitHub sẽ ngay lập tức gửi một yêu cầu POST đến một endpoint của hệ thống, kèm theo một payload chứa thông tin chi tiết về sự kiện đó.

- Kích hoạt quy trình đánh giá tự động: Sau khi nhận được webhook, hệ thống sẽ tự động kích hoạt một quy trình xử lý. Quy trình này bao gồm việc lấy mã nguồn mới nhất hoặc các thay đổi (diff) của pull request. Mã nguồn này sau đó được chuyển đến các công cụ đánh giá.

*Giai đoạn 3: Đánh giá và phản hồi:* Đây là giai đoạn mà các công cụ đánh giá sẽ thực hiện chức năng của mình để phân tích mã nguồn.

- Phân tích tĩnh với SonarQube: Hệ thống sẽ chạy SonarQube Scanner trên mã nguồn để thực hiện phân tích tĩnh. SonarQube sẽ đánh giá các chỉ số như lỗi, "mùi mã", nợ kỹ thuật và độ bao phủ kiểm thử. Kết quả được gửi về SonarQube Server và hiển thị trên dashboard.

- Phân tích sâu với AI (Gemini): Đối với mỗi pull request, hệ thống sẽ trích xuất các thay đổi (diff) và gửi đến API của Google Gemini kèm theo một prompt (lời nhắc) yêu cầu đánh giá. Gemini sẽ phân tích mã nguồn, phát hiện các lỗi logic, lỗ hổng bảo mật, và đề xuất các cách tối ưu hiệu suất hoặc cải thiện khả năng đọc.

- Phản hồi tức thì: Kết quả đánh giá từ AI sẽ được hệ thống xử lý. Sau đó, một bình luận (comment) chứa các nhận xét chi tiết và mang tính giáo dục sẽ được tự động thêm vào pull request trên GitHub. Điều này giúp sinh viên nhận được phản hồi ngay lập tức và sửa lỗi kịp thời.

*Giai đoạn 3: Báo cáo và tổng hợp dữ liệu:* Kết quả từ các giai đoạn trước được lưu trữ và tổng hợp để tạo ra các báo cáo trực quan.

- Lưu trữ dữ liệu: Mọi kết quả phân tích, lịch sử commit, bình luận của AI, và các chỉ số từ SonarQube đều được lưu trữ trong cơ sở dữ liệu của hệ thống.

- Tạo báo cáo và dashboard: Hệ thống sử dụng dữ liệu đã lưu trữ để tạo ra các báo cáo định kỳ và dashboard theo thời gian thực. Các báo cáo này bao gồm: Biểu đồ tiến độ dự án (số lượng commit, pull request theo thời gian), biểu đồ đóng góp cá nhân (số lượng dòng mã thay đổi, số lượng commit), báo cáo chất lượng mã nguồn (điểm Quality Gate, xu hướng lỗi và "mùi mã").

- Hỗ trợ đánh giá: Các báo cáo này cung cấp cho giảng viên một cái nhìn tổng quan và khách quan về hoạt động của từng nhóm và từng thành viên, từ đó hỗ trợ việc đánh giá cuối kỳ một cách công bằng và minh bạch.

## Các công nghệ sử dụng trong hệ thống

### Quy trình tạo môn học, đề tài, nhóm

Quy trình tạo môn học, đề tài và nhóm đóng vai trò nền tảng trong hệ thống theo dõi và đánh giá dự án phần mềm, bởi đây là giai đoạn khởi tạo cấu trúc tổ chức của toàn bộ hoạt động học tập và cộng tác. Mục tiêu chính của quy trình này là xác định phạm vi môn học, phân bổ nhiệm vụ, và thiết lập các nhóm làm việc nhằm tạo điều kiện thuận lợi cho việc quản lý, giám sát và đánh giá sau này.

Trước hết, giảng viên đóng vai trò là người khởi tạo môi trường làm việc bằng cách định nghĩa các thông số cơ bản của môn học, bao gồm tên, mã định danh, thời gian triển khai và các thông tin mô tả. Trên cơ sở đó, các đề tài được xây dựng, kèm theo mô tả nhiệm vụ, yêu cầu kỹ thuật và các tiêu chí đánh giá tổng quát. Mỗi đề tài được coi như một đơn vị công việc độc lập, tạo tiền đề cho sinh viên lựa chọn và triển khai.

Tiếp theo, sinh viên tiến hành đăng ký tham gia vào các đề tài thông qua hệ thống. Việc tổ chức nhóm làm việc có thể được thực hiện theo hai hình thức: sinh viên tự thành lập nhóm dựa trên sự đồng thuận giữa các thành viên, hoặc được giảng viên chỉ định. Quá trình này nhằm đảm bảo rằng mỗi nhóm có cơ cấu nhân sự hợp lý, đủ năng lực để hoàn thành nhiệm vụ được giao.

Sau khi nhóm được xác lập, bước liên kết với kho lưu trữ mã nguồn trên nền tảng GitHub được thực hiện. Ở giai đoạn này, hệ thống không chỉ ghi nhận thông tin về kho mã nguồn mà còn thiết lập cơ chế kết nối cho phép theo dõi tiến độ và hoạt động của nhóm trong suốt quá trình triển khai. Về mặt lý thuyết, kết nối này dựa trên việc sử dụng các giao diện lập trình ứng dụng (API) và cơ chế thông báo sự kiện (Webhook) của GitHub, giúp hệ thống cập nhật dữ liệu theo thời gian gần như thực.

Như vậy, quy trình tạo môn học, đề tài và nhóm không chỉ đơn thuần là bước khởi động, mà còn là bước định hình cấu trúc, luồng công việc và cơ chế giám sát, tạo nền tảng cho các giai đoạn tiếp theo như giám sát tiến độ, đánh giá chất lượng mã nguồn và tổng hợp báo cáo.

### Quy trình kết nối GitHub, nhận webhook

Quy trình kết nối GitHub và thiết lập Webhook là một thành phần quan trọng trong hệ thống theo dõi và đánh giá dự án phần mềm, nhằm bảo đảm việc đồng bộ dữ liệu giữa nền tảng quản lý mã nguồn và hệ thống giám sát được thực hiện liên tục và kịp thời.

Về mặt khái niệm, kết nối GitHub được thực hiện thông qua việc sử dụng Giao diện lập trình ứng dụng (API) của GitHub để truy cập vào thông tin kho mã nguồn, danh sách thành viên, lịch sử hoạt động và các sự kiện phát sinh trong quá trình phát triển phần mềm. Cơ chế này cho phép hệ thống thu thập dữ liệu một cách tự động, thay vì yêu cầu thao tác thủ công từ người dùng.

Sau khi thiết lập kết nối, hệ thống tiến hành cấu hình Webhook, một cơ chế thông báo theo sự kiện mà GitHub cung cấp. Webhook cho phép GitHub gửi thông tin trực tiếp đến hệ thống mỗi khi xảy ra một hành động cụ thể, chẳng hạn như việc đẩy mã nguồn (push), tạo yêu cầu hợp nhất (pull request), hoặc bình luận trong quá trình xem xét mã. Nhờ đó, hệ thống có thể phản ứng gần như tức thời với những thay đổi, cập nhật các báo cáo tiến độ, kích hoạt quá trình phân tích mã nguồn, hoặc gửi phản hồi tự động cho nhóm phát triển.

Việc kết hợp giữa API và Webhook mang lại hai lợi ích bổ sung: API đảm bảo khả năng truy xuất dữ liệu có hệ thống, phục vụ các báo cáo tổng hợp định kỳ; trong khi Webhook cung cấp luồng dữ liệu thời gian thực, hỗ trợ cơ chế giám sát liên tục. Sự kết hợp này giúp hệ thống duy trì tính chính xác và kịp thời của thông tin, đồng thời giảm thiểu độ trễ trong việc phản hồi và xử lý sự kiện.

Như vậy, quy trình kết nối GitHub và nhận Webhook đóng vai trò như “mạch máu dữ liệu” của toàn bộ hệ thống, bảo đảm rằng mọi hoạt động phát triển của sinh viên đều được ghi nhận, phân tích và phản hồi một cách có hệ thống và minh bạch.

### Quy trình nộp bài và đánh giá mã nguồn tự động

Quy trình nộp bài và đánh giá mã nguồn tự động là một cơ chế cốt lõi trong hệ thống, cho phép giảng viên và sinh viên nhận được phản hồi nhanh chóng, khách quan và nhất quán về chất lượng sản phẩm phần mềm. Cơ chế này vận hành dựa trên nguyên lý tích hợp giữa nền tảng quản lý mã nguồn (ví dụ GitHub), các công cụ phân tích tĩnh, và mô hình trí tuệ nhân tạo chuyên biệt cho lập trình.

Về cơ bản, quá trình bắt đầu khi sinh viên hoặc nhóm phát triển thực hiện thao tác đẩy mã nguồn (push) hoặc gửi yêu cầu hợp nhất (pull request) lên kho mã nguồn GitHub đã được kết nối. Sự kiện này được Webhook kích hoạt và gửi thông tin tới máy chủ của hệ thống. Máy chủ tiếp nhận dữ liệu, đồng bộ phiên bản mã nguồn mới nhất và tiến hành chuỗi xử lý tự động.

Bước tiếp theo là phân tích chất lượng mã nguồn, thường được thực hiện bởi các công cụ phân tích tĩnh như SonarQube. Công cụ này áp dụng tập hợp các quy tắc và chỉ số đánh giá (ví dụ: số lượng lỗi, mức độ phức tạp của mã, tỷ lệ lặp lại, nợ kỹ thuật) để đưa ra báo cáo định lượng. Song song, hệ thống có thể sử dụng mô hình ngôn ngữ lớn (LLM) như Google Gemini để thực hiện đánh giá chuyên sâu ở mức ngữ nghĩa và ngữ cảnh, chẳng hạn phát hiện các vấn đề về kiến trúc, bảo mật, hoặc hiệu năng, đồng thời cung cấp giải thích chi tiết và gợi ý cải tiến.

Kết quả đánh giá sau đó được tích hợp trở lại vào quy trình làm việc của nhóm. Các phản hồi có thể xuất hiện trực tiếp trong phần thảo luận của pull request, hoặc hiển thị trên bảng điều khiển của hệ thống. Điều này cho phép sinh viên nhanh chóng nhận diện vấn đề, áp dụng chỉnh sửa và nộp lại phiên bản cải tiến mà không cần chờ đợi đánh giá thủ công.

Cơ chế nộp bài và đánh giá mã nguồn tự động mang lại ba lợi ích chính: (1) tăng tốc độ phản hồi nhờ tự động hóa hoàn toàn; (2) bảo đảm tính nhất quán nhờ sử dụng cùng một bộ tiêu chuẩn và công cụ cho tất cả các nhóm; (3) hỗ trợ học tập thông qua phản hồi mang tính giải thích, giúp sinh viên hiểu rõ nguyên nhân và cách khắc phục lỗi. Đây là một yếu tố then chốt để xây dựng mô hình giảng dạy lập trình hiện đại, gắn kết giữa thực hành và đánh giá theo hướng liên tục.

### Quy trình hiển thị thống kê đóng góp

Quy trình hiển thị thống kê đóng góp trong hệ thống nhằm cung cấp cái nhìn toàn diện và minh bạch về mức độ tham gia của từng thành viên trong nhóm dự án. Đây là một thành phần quan trọng hỗ trợ giảng viên trong công tác đánh giá, đồng thời giúp sinh viên tự nhận thức về hiệu suất làm việc của bản thân và nhóm.

Cơ chế vận hành bắt đầu từ việc thu thập dữ liệu hoạt động từ kho mã nguồn GitHub thông qua API và Webhook. Các sự kiện như commit, pull request, review, hoặc issue comment được ghi nhận và lưu trữ vào cơ sở dữ liệu của hệ thống. Thông tin này không chỉ bao gồm nội dung thay đổi, mà còn chứa siêu dữ liệu (metadata) như thời gian thực hiện, người thực hiện, và phạm vi ảnh hưởng của thay đổi.

Sau giai đoạn thu thập, hệ thống tiến hành xử lý và tổng hợp dữ liệu. Quá trình này có thể bao gồm phân loại loại hoạt động (ví dụ: đóng góp mã mới, sửa lỗi, viết tài liệu), tính toán các chỉ số định lượng như số dòng mã được bổ sung/xóa, số lượng commit hợp lệ, hoặc tần suất tham gia. Để đảm bảo công bằng, hệ thống có thể áp dụng các thuật toán loại bỏ nhiễu, chẳng hạn bỏ qua những thay đổi không đáng kể hoặc các commit tự động.

Kết quả được trình bày trên giao diện thống kê trực quan, dưới dạng biểu đồ cột, biểu đồ tròn hoặc đường thời gian, cho phép so sánh mức độ đóng góp giữa các thành viên và theo dõi xu hướng làm việc của nhóm theo thời gian. Ngoài ra, các chỉ số này có thể được tích hợp vào bảng đánh giá tổng hợp của dự án, giúp giảng viên đưa ra quyết định chấm điểm dựa trên cả chất lượng và khối lượng công việc.

Việc hiển thị thống kê đóng góp không chỉ phục vụ mục đích đánh giá, mà còn đóng vai trò như một công cụ điều phối nhóm. Sinh viên có thể quan sát mức độ tham gia của đồng đội, từ đó chủ động điều chỉnh phân công công việc, đảm bảo tiến độ và cân bằng khối lượng giữa các thành viên. Điều này góp phần hình thành môi trường làm việc minh bạch, thúc đẩy tinh thần trách nhiệm và hợp tác trong nhóm.

## Các công nghệ xây dựng trang web

### Ngôn ngữ HTML

### Bảng định kiểu CSS

### Ngôn ngữ Javascript

### NextJs

### TailwindCSS

### Shadcn/ui

### TypeScript

### Node.js

### Express

### Socket.IO

### MySQL

# HIỆN THỰC HÓA NGHIÊN CỨU

## Mô tả bài toán thực tế

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Phân tích yêu cầu người dùng

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Lựa chọn công nghệ

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Thiết kế kiến trúc hệ thống tổng thể

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Thiết kế cơ sở dữ liệu

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Thiết kế các luồng hoạt động chính của hệ thống

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### Quy trình tạo môn học, đề tài, nhóm

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### Quy trình kết nối GitHub, nhận webhook

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### Quy trình nộp bài và đánh giá mã nguồn tự động

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### Quy trình hiển thị thống kê đóng góp

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

# Triển khai hệ thống

## Thiết lập môi trường phát triển

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Xây dựng backend với Express và Sequelize

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Xây dựng các API chức năng chính

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### API người dùng và xác thực OAuth

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### API môn học, đề tài, nhóm

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### API webhook và xử lý commit từ GitHub

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### API thống kê đóng góp

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Thiết kế và xây dựng giao diện bằng Next.js

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### Giao diện giảng viên

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

### Giao diện sinh viên

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Thiết lập GitHub Actions và tích hợp SonarCloud

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Tích hợp toàn bộ hệ thống và xử lý dữ liệu thời gian thực

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

# KẾT QUẢ VÀ ĐÁNH GIÁ

## Kết quả thử nghiệm với tài khoản sinh viên

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Kết quả thử nghiệm với tài khoản giảng viên

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Kiểm tra kết quả đánh giá mã nguồn qua GitHub Actions

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Đánh giá tính chính xác và hiệu quả thống kê đóng góp

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

# KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

## Kết luận

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Hạn chế

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

## Hướng phát triển trong tương lai

**Lorem Ipsum** is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

PHỤ LỤC

DANH MỤC TÀI LIỆU THAM KHẢO