**[[1]](#endnote-1)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

[LỜI MỞ ĐẦU 3](#_Toc205844551)

[LỜI CẢM ƠN 4](#_Toc205844552)

[NHẬN XÉT 5](#_Toc205844553)

[NHẬN XÉT 8](#_Toc205844554)

[MỤC LỤC 11](#_Toc205844555)

[DANH MỤC CÁC BẢNG, SƠ ĐỒ, HÌNH 16](#_Toc205844556)

[KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT 17](#_Toc205844557)

[CHƯƠNG 1. ĐẶT VẤN ĐỀ 1](#_Toc205844558)

[1.1. Lý do chọn đề tài 1](#_Toc205844559)

[1.2. Mục tiêu 1](#_Toc205844560)

[1.3. Nội dung nghiên cứu 2](#_Toc205844561)

[1.4. Đối tượng và phạm vi nghiên 3](#_Toc205844562)

[1.5. Phương pháp nghiên cứu 4](#_Toc205844563)

[CHƯƠNG 2. CƠ SỞ LÝ THUYẾT 6](#_Toc205844564)

[2.1. Tổng quan về GitHub và làm việc nhóm qua GitHub 6](#_Toc205844565)

[2.1.1 Git và GitHub là gì? 6](#_Toc205844566)

[2.1.2 Các tính năng hỗ trợ làm việc nhóm 7](#_Toc205844567)

[2.1.3 Lợi ích khi làm việc nhóm qua GitHub 8](#_Toc205844568)

[2.1.4 Mô hình làm việc nhóm qua GitHub 9](#_Toc205844569)

[2.2. Tổng quan CI/CD 10](#_Toc205844570)

[2.2.1 Continuous Integration (CI) - Tích hợp Liên tục 10](#_Toc205844571)

[2.2.2 Continuous Delivery (CD) - Phân phối Liên tục 12](#_Toc205844572)

[2.2.3 Quy trình CI/CD 13](#_Toc205844573)

[2.3. GitHub Actions 18](#_Toc205844574)

[2.4. Tổng quan về GitHub API và Webhook 21](#_Toc205844575)

[2.4.1 GitHub API 21](#_Toc205844576)

[2.4.2 Github Webhook 23](#_Toc205844577)

[2.5. Công cụ đánh giá mã nguồn 25](#_Toc205844578)

[2.5.1 Mục tiêu đánh giá mã nguồn 25](#_Toc205844579)

[2.5.2 Phân loại công cụ đánh giá mã nguồn 26](#_Toc205844580)

[2.5.3 Quy trình đánh giá mã nguồn 26](#_Toc205844581)

[2.5.4 Kết hợp công cụ đánh giá mã nguồn với GitHub Actions 28](#_Toc205844582)

[2.6. SonarQube và vai trò trong đánh giá chất lượng mã nguồn 30](#_Toc205844583)

[2.6.1 Giới thiệu về SonarQube 30](#_Toc205844584)

[2.6.2 Các khái niệm và chỉ số cốt lõi 30](#_Toc205844585)

[2.6.3 Quy trình đánh giá và vai trò trong dự án 31](#_Toc205844586)

[2.6.4 Tính hợp SonarQube trong quy trình CI/CD 31](#_Toc205844587)

[2.7. Tích hợp AI vào quy trình đánh giá lập trình 33](#_Toc205844588)

[2.7.1 Kiến trúc và cơ chế hoạt động 33](#_Toc205844589)

[2.7.2 Ứng dụng trong đánh giá mã nguồn 34](#_Toc205844590)

[2.7.3 Các mô hình ngôn ngữ lớn có thể tích hợp 35](#_Toc205844591)

[2.7.4 Tìm hiểu về Gemini trong đánh giá code 35](#_Toc205844592)

[2.8. Tổng quan về hệ thống theo dõi dự án phần mềm của sinh viên 36](#_Toc205844593)

[2.8.1 Mục tiêu chính 37](#_Toc205844594)

[2.8.2 Kiến trúc tổng quan của hệ thống 38](#_Toc205844595)

[2.8.3 Quy trình hoạt động cơ bản của hệ thống 40](#_Toc205844596)

[2.9. Các công nghệ sử dụng trong hệ thống 42](#_Toc205844597)

[2.9.1 Quy trình tạo môn học, đề tài, nhóm 42](#_Toc205844598)

[2.9.2 Quy trình kết nối GitHub, nhận webhook 43](#_Toc205844599)

[2.9.3 Quy trình nộp bài và đánh giá mã nguồn tự động 44](#_Toc205844600)

[2.9.4 Quy trình hiển thị thống kê đóng góp 45](#_Toc205844601)

[2.10. Các công nghệ xây dựng trang web 46](#_Toc205844602)

[2.10.1 Công nghệ Frontend 46](#_Toc205844603)

[2.10.1.1 TypeScript (TS) 46](#_Toc205844604)

[2.10.1.2 JavaScript (JS) 46](#_Toc205844605)

[2.10.1.3 React.js 46](#_Toc205844606)

[2.10.1.4 Next.js 47](#_Toc205844607)

[2.10.1.5 Tailwind CSS 47](#_Toc205844608)

[2.10.1.6 Shadcn/ui 48](#_Toc205844609)

[2.10.1.7 Zustand 48](#_Toc205844610)

[2.10.1.8 Next-i18next 48](#_Toc205844611)

[2.10.2 Công nghệ Backend 49](#_Toc205844612)

[2.10.2.1 Express.js 49](#_Toc205844613)

[2.10.2.2 SQL (Structured Query Language) 49](#_Toc205844614)

[2.10.2.3 Sequelize ORM 49](#_Toc205844615)

[2.10.2.4 Firebase Authentication 50](#_Toc205844616)

[2.10.2.5 Socket.io 50](#_Toc205844617)

[2.10.2.6 JWT (JSON Web Tokens) 51](#_Toc205844618)

[CHƯƠNG 3. HIỆN THỰC HÓA NGHIÊN CỨU 52](#_Toc205844619)

[3.1. Mô tả bài toán thực tế 52](#_Toc205844620)

[3.2. Phân tích yêu cầu người dùng 52](#_Toc205844621)

[3.3. Lựa chọn công nghệ 52](#_Toc205844622)

[3.4. Thiết kế kiến trúc hệ thống tổng thể 52](#_Toc205844623)

[3.5. Thiết kế cơ sở dữ liệu 53](#_Toc205844624)

[3.6. Thiết kế các luồng hoạt động chính của hệ thống 53](#_Toc205844625)

[3.6.1 Quy trình tạo môn học, đề tài, nhóm 53](#_Toc205844626)

[3.6.2 Quy trình kết nối GitHub, nhận webhook 54](#_Toc205844627)

[3.6.3 Quy trình nộp bài và đánh giá mã nguồn tự động 54](#_Toc205844628)

[3.6.4 Quy trình hiển thị thống kê đóng góp 54](#_Toc205844629)

[CHƯƠNG 4. Triển khai hệ thống 55](#_Toc205844630)

[4.1. Thiết lập môi trường phát triển 55](#_Toc205844631)

[4.2. Xây dựng backend với Express và Sequelize 55](#_Toc205844632)

[4.3. Xây dựng các API chức năng chính 55](#_Toc205844633)

[4.3.1 API người dùng và xác thực OAuth 55](#_Toc205844634)

[4.3.2 API môn học, đề tài, nhóm 56](#_Toc205844635)

[4.3.3 API webhook và xử lý commit từ GitHub 56](#_Toc205844636)

[4.3.4 API thống kê đóng góp 56](#_Toc205844637)

[4.4. Thiết kế và xây dựng giao diện bằng Next.js 57](#_Toc205844638)

[4.4.1 Giao diện giảng viên 57](#_Toc205844639)

[4.4.2 Giao diện sinh viên 57](#_Toc205844640)

[4.5. Thiết lập GitHub Actions và tích hợp SonarCloud 57](#_Toc205844641)

[4.6. Tích hợp toàn bộ hệ thống và xử lý dữ liệu thời gian thực 58](#_Toc205844642)

[CHƯƠNG 5. KẾT QUẢ VÀ ĐÁNH GIÁ 59](#_Toc205844643)

[5.1. Kết quả thử nghiệm với tài khoản sinh viên 59](#_Toc205844644)

[5.2. Kết quả thử nghiệm với tài khoản giảng viên 59](#_Toc205844645)

[5.3. Kiểm tra kết quả đánh giá mã nguồn qua GitHub Actions 59](#_Toc205844646)

[5.4. Đánh giá tính chính xác và hiệu quả thống kê đóng góp 59](#_Toc205844647)

[CHƯƠNG 6. KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN 61](#_Toc205844648)

[6.1. Kết luận 61](#_Toc205844649)

[6.2. Hạn chế 61](#_Toc205844650)

[6.3. Hướng phát triển trong tương lai 61](#_Toc205844651)

[PHỤ LỤC 62](#_Toc205844652)

[DANH MỤC TÀI LIỆU THAM KHẢO 63](#_Toc205844653)

DANH MỤC HÌNH

[Hình 2.1. Sơ đồ kiến trúc dựa trên Component của React 47](#_Toc206030684)

[Hình 2.2. Cơ chế hoạt động của Virtual DOM 49](#_Toc206030685)

[Hình 2.3 Sơ đồ quy trình Server-Side Rendering 52](#_Toc206030686)

[Hình 2.4 Cấu trúc tập tin File-based Routing 53](#_Toc206030687)

[Hình 2.5 Kiến trúc của Shadcn/ui 56](#_Toc206030688)

[Hình 2.6 Sơ đồ kiến trúc của Express.js 61](#_Toc206030689)

[Hình 2.7 Sơ đồ kiến trúc của Sequelize ORM 66](#_Toc206030690)

[Hình 2.8 Sơ đồ quy trình xác thực bằng JWT 71](#_Toc206030691)

[Hình 3.1. Lược đồ cơ sở dữ liệu 83](#_Toc206030692)

DANH MỤC Bảng

[Bảng 3.1. Mô tả bảng User 84](#_Toc206030705)

[Bảng 3.2. Mô tả bảng Courses 84](#_Toc206030706)

[Bảng 3.3. Mô tả bảng Topics 85](#_Toc206030707)

[Bảng 3.4 Mô tả bảng Repos 86](#_Toc206030708)

[Bảng 3.5. Mô tả bảng Commits 87](#_Toc206030709)

[Bảng 3.6. Mô tả bảng Pull request 88](#_Toc206030710)

[Bảng 3.7 Mô tả bảng Code analysis 89](#_Toc206030711)

[Bảng 3.8. Mô tả bảng Code analysis metrics 89](#_Toc206030712)

[Bảng 3.9. Mô tả bảng Review ai 90](#_Toc206030713)

[Bảng 3.10. Mô tả bảng Notofications 90](#_Toc206030714)

[Bảng 3.11. Mô tả bảng Topic member 91](#_Toc206030715)

[Bảng 3.12. Mô tả bảng Topic evaluations 92](#_Toc206030716)

[Bảng 3.13. Mô tả bảng Tags 92](#_Toc206030717)

[Bảng 3.14. Mô tả bảng Posts 93](#_Toc206030718)

[Bảng 3.15. Mô tả bảng Comments 93](#_Toc206030719)

[Bảng 3.16 Mô tả bảng User settings 94](#_Toc206030720)

[Bảng 3.17 Mô tả bảng System settings 94](#_Toc206030721)

[Bảng 3.18. Mô tả bảng Course enrollment 95](#_Toc206030722)

[Bảng 3.19. Mô tả bảng Course documents 95](#_Toc206030723)

[Bảng 3.20. Bảng mô tả API đăng ký 96](#_Toc206030724)

[Bảng 3.21. Bảng mô tả API đăng nhập 96](#_Toc206030725)

[Bảng 3.22. Bảng mô tả API đăng nhập GitHub 97](#_Toc206030726)

[Bảng 3.23. Bảng mô tả API lấy thông tin người dùng 97](#_Toc206030727)

[Bảng 3.24 Bảng mô tả API đăng xuất 97](#_Toc206030728)

[Bảng 3.25. Bảng mô tả API tạo môn học mới 98](#_Toc206030729)

[Bảng 3.26. Bảng mô tả API lấy thông tin môn học 99](#_Toc206030730)

[Bảng 3.27. Bảng mô tả API lấy danh sách môn học 99](#_Toc206030731)

[Bảng 3.28. Bảng mô tả API cập nhật môn học 99](#_Toc206030732)

[Bảng 3.29. Bảng mô tả API xóa môn học 100](#_Toc206030733)

[Bảng 3.30. Bảng mô tả API xóa hoàn toàn môn học 100](#_Toc206030734)

[Bảng 3.31. Bảng mô tả API khôi phục môn học 100](#_Toc206030735)

[Bảng 3.32. Bảng mô tả API tạo đề tài mới 100](#_Toc206030736)

[Bảng 3.33. Bảng mô tả API lấy danh sách đề tài 101](#_Toc206030737)

[Bảng 3.34. bảng mô tả API lấy đề tài theo môn học 101](#_Toc206030738)

[Bảng 3.35. Bảng mô tả API cập nhật đề tài 102](#_Toc206030739)

[Bảng 3.36. Bảng mô tả API xóa đề tài 102](#_Toc206030740)

[Bảng 3.37. Bảng mô tả API xóa hoàn toàn đề tài 102](#_Toc206030741)

[Bảng 3.38. Bảng mô tả API khôi phục đề tài 103](#_Toc206030742)

[Bảng 3.39. Bảng mô tả API lấy hoạt động code của môn học 103](#_Toc206030743)

[Bảng 3.40. Bảng mô tả API lấy người đóng góp môn học 103](#_Toc206030744)

[Bảng 3.41. Bảng mô tả API lấy người đóng góp đề tài 103](#_Toc206030745)

[Bảng 3.42. Bảng mô tả API lấy thống kê đề tài 104](#_Toc206030746)

[Bảng 3.43. Bảng mô tả API lấy thông tin repository 104](#_Toc206030747)

[Bảng 3.44. Bảng mô tả API lấy thông tin người dùng GitHub 104](#_Toc206030748)

[Bảng 3.45. Bảng mô tả API lấy thành viên tổ chức 104](#_Toc206030749)

[Bảng 3.46. Bảng mô tả API mời người dùng vào tổ chức 104](#_Toc206030750)

[Bảng 3.47. Bảng mô tả API xử lý webhook commit 105](#_Toc206030751)

[Bảng 3.48. Bảng mô tả API thêm webhook commit 105](#_Toc206030752)

[Bảng 3.49. Bảng mô tả API lấy danh sách repository 105](#_Toc206030753)

[Bảng 3.50. Bảng mô tả API lấy thông tin repository 106](#_Toc206030754)

[Bảng 3.51. Bảng mô tả API lấy repository theo đề tài 106](#_Toc206030755)

[Bảng 3.52. Bảng mô tả API tạo repository mới 106](#_Toc206030756)

[Bảng 3.53. Bảng mô tả API cập nhật repository 107](#_Toc206030757)

[Bảng 3.54. Bảng mô tả API xóa repository 107](#_Toc206030758)

[Bảng 3.55. Bảng mô tả API xóa hoàn toàn repository 107](#_Toc206030759)

[Bảng 3.56. Bảng mô tả API khôi phục repository 107](#_Toc206030760)

[Bảng 3.57. Bảng mô tả API tạo dự án SonarQube 108](#_Toc206030761)

[Bảng 3.58. Bảng mô tả API xóa dự án SonarQube 108](#_Toc206030762)

[Bảng 3.59. Bảng mô tả API lấy metrics SonarQube 108](#_Toc206030763)

KÍ HIỆU CÁC CỤM TỪ VIẾT TẮT

|  |  |
| --- | --- |
| **Từ viết tắt** | **Ý nghĩa** |
| CI |  |
| CD |  |
| LLM |  |
| AI |  |
| API |  |
| UI |  |
| UX |  |
| HTTP |  |
| URL |  |
| ORM |  |
| HTML |  |
| CSS |  |
| SQL |  |

# 

# 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.



Hình 2.1. Logo Github

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.

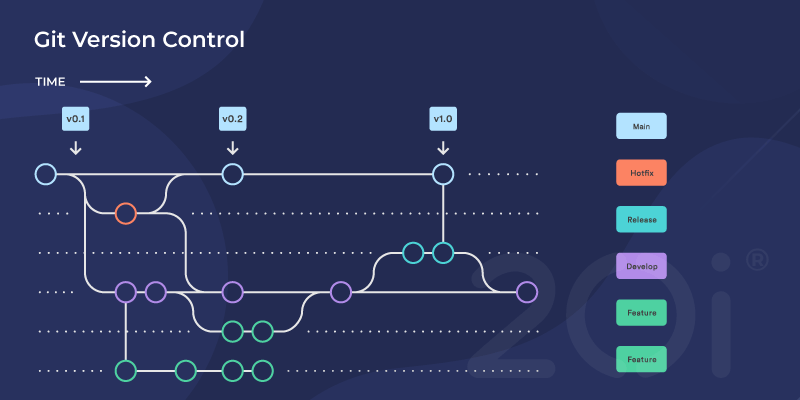
GitHub là một dịch vụ trực tuyến cung cấp giao diện web cho Git, đồng thời tích hợp thêm nhiều tính năng hỗ trợ phát triển phần mềm như quản lý dự án, theo dõi lỗi (issues), kiểm tra chất lượng mã, CI/CD, tài liệu dự án, v.v.

### 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.



Hình 2.2. Sơ đồ phân nhánh làm việc trong github

*- 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.

*- 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.

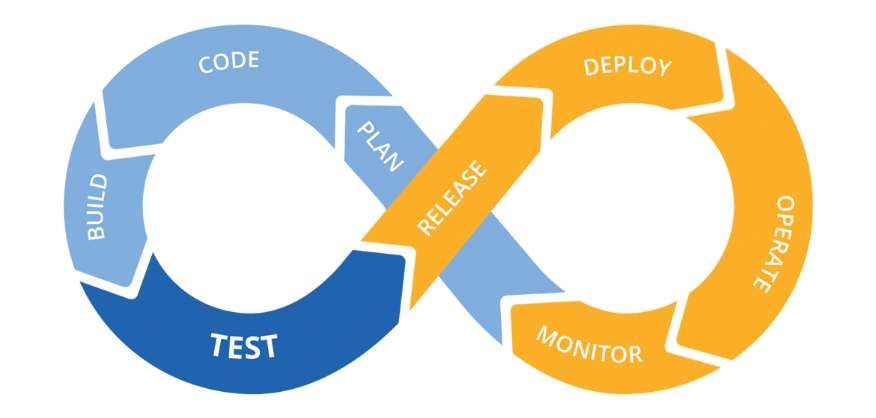
*4. Tạo Pull Request (Yêu cầu hợp nhất mã)*: Khi một tính năng hoặc sửa lỗi đã hoàn thành, lập trình viên sẽ tạo một Pull Request (PR) trên GitHub. PR này là một yêu cầu chính thức để hợp nhất (merge) mã nguồn từ nhánh cá nhân vào nhánh chính. Kèm theo PR, cần có một mô tả chi tiết về những thay đổi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



Hình 2.4. Quy trình CI/CD

### 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.

## 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.

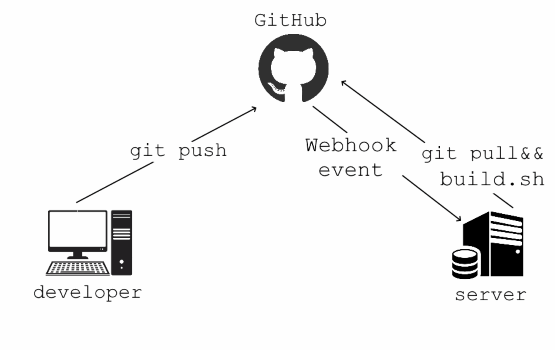
*- Tích hợp và tự động hóa:* Sử dụng API để tương tác với các hệ thống CI/CD, gửi thông báo, hoặc tự động tạo các bản dựng (builds).

Bảng 2.1. 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.



Hình 2.6.Sơ đồ hoạt động GitHub Webhook

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.

Hình 2.8. Giao diện SonarQube

### 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.

Hình 2.9. Sơ đồ hoạt động tích hợp CI/CD với SonarQube

*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).

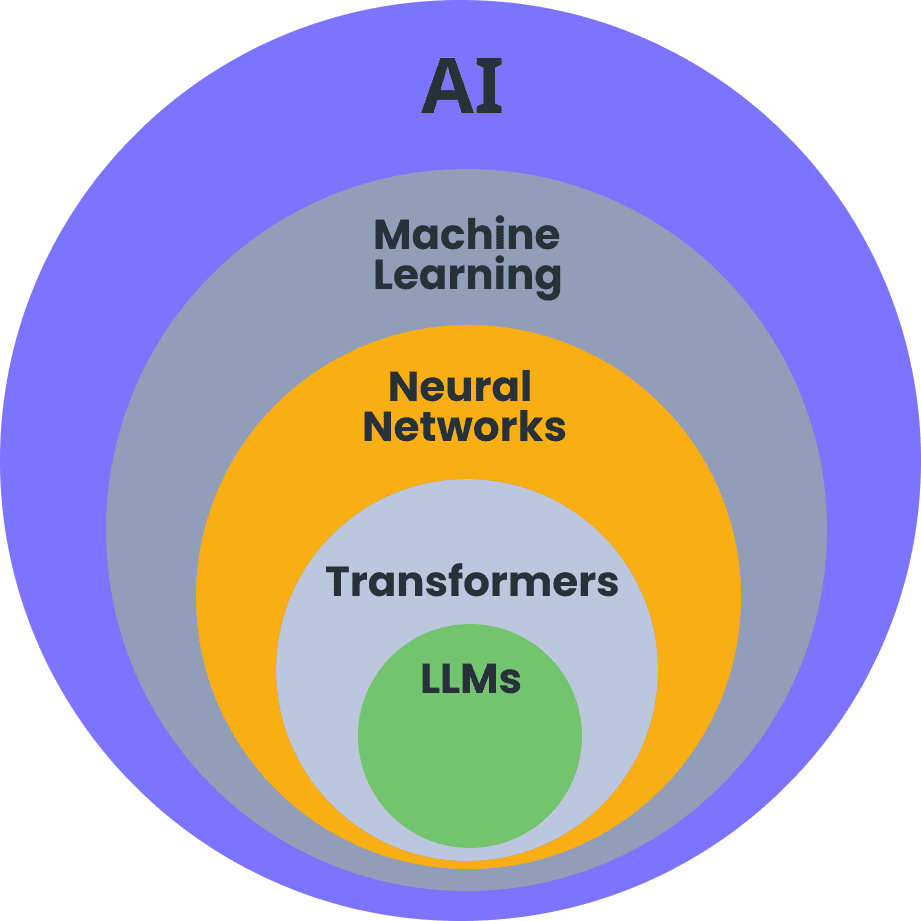
+ Nếu Quality Gate thất bại (fails), tức là mã nguồn mới có vấn đề nghiêm trọng, pipeline sẽ bị dừng lại. Điều này ngăn chặn việc mã kém chất lượng được hợp nhất vào nhánh chính, đảm bảo tính ổn định của dự án.

## 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.



Hình 2.10. LLMs (Mô hình ngôn ngữ lớn)

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.

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)

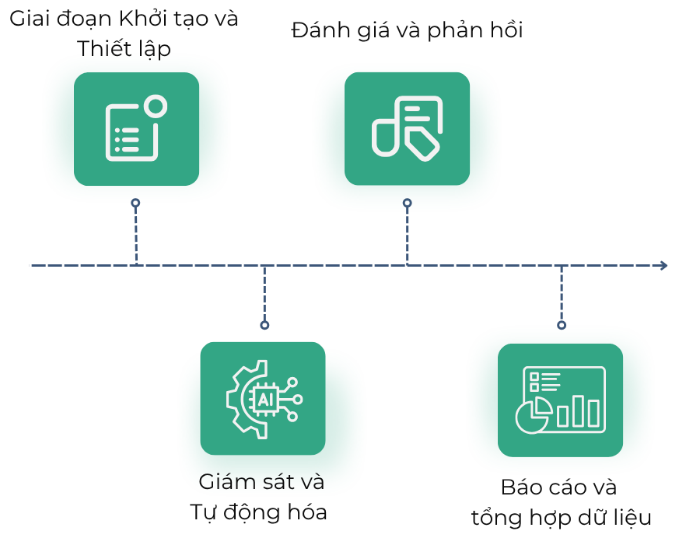
Mô-đun này là nền tảng của hệ thống, quản lý việc lưu trữ và xử lý tất cả các loại dữ liệu.

*- 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:



Hình 2.12.Quy trình hoạt động cơ bản của hệ thống

*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

### Công nghệ Frontend

Trong hệ thống theo dõi và đánh giá dự án phần mềm của sinh viên, tầng Frontend đóng vai trò là giao diện tương tác trực tiếp giữa người dùng (giảng viên, sinh viên) và hệ thống. Việc lựa chọn công nghệ xây dựng giao diện được cân nhắc dựa trên các yếu tố như hiệu suất hiển thị, khả năng mở rộng, trải nghiệm người dùng, và tính dễ bảo trì. Các công nghệ chủ đạo được sử dụng trong phát triển giao diện bao gồm:

#### JavaScript (JS)

JavaScript là ngôn ngữ lập trình thông dịch (interpreted programming language) được phát triển bởi Brendan Eich tại Netscape Communications Corporation vào năm 1995. Đây là ngôn ngữ lập trình đa mô hình (multi-paradigm) hỗ trợ lập trình hướng đối tượng, lập trình hàm, và lập trình thủ tục. JavaScript ban đầu được thiết kế để chạy trên trình duyệt web, nhưng hiện nay đã mở rộng ra nhiều môi trường khác thông qua các runtime như Node.js.

JavaScript sở hữu các đặc điểm nổi bật như: tính động (dynamic typing), khả năng thực thi bất đồng bộ (asynchronous execution), closure, prototype-based inheritance, và event-driven programming. Ngôn ngữ này cũng hỗ trợ các tính năng hiện đại như modules, classes, arrow functions, template literals, và destructuring assignment.

JavaScript đóng một vai trò trung tâm trong hệ sinh thái web hiện đại, được công nhận không chỉ như một ngôn ngữ kịch bản cho trình duyệt mà còn là một nền tảng lập trình toàn diện. Những đặc điểm chính của nó bao gồm khả năng tạo ra các trang web động ở phía client, mở rộng sang phía máy chủ thông qua Node.js, mô hình bất đồng bộ hiệu quả và một cộng đồng lập trình viên cùng hệ sinh thái vô cùng phong phú, làm cho nó trở thành một công nghệ không thể thiếu với vai trò và các đặc điểm chính như:

*- Lập trình client-side (Phía trình duyệt):* JavaScript là công nghệ duy nhất được tất cả các trình duyệt web hỗ trợ một cách nguyên bản. Điều này cho phép các nhà phát triển tạo ra các trang web động, phản ứng với các hành động của người dùng như click chuột, nhập liệu từ bàn phím, và gửi biểu mẫu. JavaScript chịu trách nhiệm cho việc thao tác với DOM (Document Object Model), cập nhật nội dung, cấu trúc và phong cách của một trang web mà không cần tải lại toàn bộ trang. Các framework và thư viện frontend hiện đại như React, Angular và Vue.js đều được xây dựng dựa trên JavaScript.

*- Lập trình server-side (Phía máy chủ):* Với sự ra đời của Node.js vào năm 2009, JavaScript đã mở rộng sang lập trình phía máy chủ. Node.js là một môi trường runtime JavaScript mã nguồn mở và đa nền tảng, cho phép thực thi JavaScript bên ngoài trình duyệt. Điều này cho phép các nhà phát triển sử dụng cùng một ngôn ngữ cho cả frontend và backend, giúp đồng bộ hóa dữ liệu và logic dễ dàng hơn.

*- Mô hình bất đồng bộ (Asynchronous):* JavaScript có một mô hình xử lý bất đồng bộ hiệu quả. Điều này cho phép ứng dụng thực hiện các tác vụ tốn thời gian như gọi API, tải dữ liệu từ cơ sở dữ liệu, hoặc xử lý tệp mà không làm gián đoạn luồng thực thi chính (main thread). Các cơ chế xử lý bất đồng bộ trong JavaScript bao gồm:

+ Callback Functions: Một hàm được truyền làm đối số cho một hàm khác và được gọi lại khi tác vụ hoàn thành.

+ Promises: Cung cấp một cách tiếp cận rõ ràng hơn để xử lý các tác vụ bất đồng bộ, tránh tình trạng "callback hell".

+ Async/await: Cú pháp hiện đại giúp việc viết mã bất đồng bộ trở nên dễ đọc và dễ quản lý hơn, giống như viết mã đồng bộ.

Đoạn mã minh họa cách JavaScript dùng fetch cùng async/await để gọi API lấy danh sách môn học:

export async function fetchCourses() {

const response = await fetch('/api/courses'); // endpoint của backend

if (!response.ok) {

throw new Error('Lỗi khi tải khóa học');

}

return response.json();

}

import React, { useEffect, useState } from 'react';

import { fetchCourses } from './api';

function CourseList() {

const [courses, setCourses] = useState([]);

const [error, setError] = useState(null);

useEffect(() => {

let mounted = true;

fetchCourses()

.then(data => mounted && setCourses(data))

.catch(err => mounted && setError(err.message));

return () => { mounted = false; };

}, []);

if (error) return <div>Lỗi: {error}</div>;

return (

<ul>

{courses.map(c => <li key={c.id}>{c.name}</li>)}

</ul>

);

}

JavaScript có một cộng đồng lập trình viên khổng lồ và một hệ sinh thái mạnh mẽ với hàng triệu thư viện, framework và công cụ có sẵn. npm (Node Package Manager) là kho lưu trữ thư viện lớn nhất thế giới, cung cấp các giải pháp cho hầu hết mọi vấn đề trong phát triển phần mềm, từ xây dựng giao diện người dùng đến quản lý cơ sở dữ liệu.

#### TypeScript (TS)

TypeScript là một ngôn ngữ lập trình được phát triển bởi Microsoft, được xây dựng dựa trên JavaScript với việc bổ sung hệ thống kiểu dữ liệu tĩnh (static typing system). Đây là một superset của JavaScript, có nghĩa là mọi đoạn mã JavaScript hợp lệ đều là TypeScript hợp lệ. TypeScript cung cấp khả năng kiểm tra kiểu dữ liệu tại thời điểm biên dịch, giúp phát hiện sớm các lỗi logic và cải thiện đáng kể chất lượng mã nguồn.

TypeScript không chỉ là một phiên bản mở rộng của JavaScript, mà còn cung cấp một bộ tính năng mạnh mẽ giúp các nhà phát triển xây dựng các ứng dụng có độ tin cậy và khả năng bảo trì cao.

Các tính năng nổi bật của TypeScript bao gồm một hệ thống kiểu dữ liệu mạnh mẽ và đa dạng, khả năng hỗ trợ lập trình hướng đối tượng đầy đủ:

*- Hệ thống kiểu dữ liệu mạnh mẽ và đa dạng:* Đây là đặc điểm nổi bật nhất của TypeScript, cho phép lập trình viên định nghĩa rõ ràng kiểu dữ liệu cho các biến, tham số hàm và giá trị trả về. Hệ thống này bao gồm nhiều loại kiểu dữ liệu khác nhau:

+ Kiểu cơ bản (Primitive Types): Bao gồm string, number, và boolean. Việc sử dụng các kiểu này giúp đảm bảo tính nhất quán của dữ liệu.

+ Array: Cho phép khai báo một mảng chỉ chứa một loại dữ liệu cụ thể, ví dụ: number[] hoặc string[].

+ Tuple: Một mảng có số lượng phần tử cố định và kiểu dữ liệu của mỗi phần tử được xác định trước, ví dụ: [string, number].

+ Union: Cho phép một biến có thể nhận nhiều kiểu dữ liệu khác nhau, ví dụ: string | number.

+ Kiểu Generic: Đây là một tính năng mạnh mẽ giúp tạo ra các thành phần (hàm, lớp) có thể hoạt động với nhiều kiểu dữ liệu khác nhau mà vẫn đảm bảo tính an toàn về kiểu. Ví dụ, một hàm identity có thể hoạt động với bất kỳ kiểu dữ liệu nào mà vẫn giữ được kiểu của giá trị đầu vào.

+ Interface: Định nghĩa một "hợp đồng" về cấu trúc của một đối tượng. Interface giúp đảm bảo rằng các đối tượng tuân thủ một cấu trúc nhất quán.

+ Class: TypeScript hỗ trợ cú pháp Class đầy đủ với tính kế thừa (inheritance) và đa hình (polymorphism). Điều này tạo điều kiện cho việc thiết kế phần mềm hướng đối tượng (OOP) một cách rõ ràng và có cấu trúc.

*- Hỗ trợ các tính năng hiện đại của JavaScript:* Mặc dù TypeScript có những tính năng riêng, nó vẫn luôn cập nhật và hỗ trợ đầy đủ các tính năng mới nhất của JavaScript (ES6+). Điều này cho phép các lập trình viên sử dụng các cú pháp và công nghệ mới nhất mà không phải lo lắng về tính tương thích.

+ Async/Await: Giúp xử lý các tác vụ bất đồng bộ một cách dễ đọc và dễ quản lý hơn.

+ Destructuring: Cho phép gán giá trị từ mảng hoặc đối tượng vào các biến một cách gọn gàng.

+ Spread Operator: Giúp mở rộng các iterable (như mảng hoặc đối tượng) vào nơi mà nhiều đối số (hoặc phần tử) được mong đợi.

Nhờ sự kết hợp giữa hệ thống kiểu dữ liệu mạnh mẽ và các tính năng hiện đại của JavaScript, TypeScript trở thành một lựa chọn lý tưởng cho việc xây dựng các ứng dụng phức tạp, dễ bảo trì và có khả năng mở rộng cao.

Các tính năng nổi bật của TypeScript bao gồm một hệ thống kiểu dữ liệu mạnh mẽ và đa dạng, khả năng hỗ trợ lập trình hướng đối tượng đầy đủ, và sự tương thích hoàn hảo với các tính năng hiện đại của JavaScript. Nhờ đó, TypeScript không chỉ cung cấp một nền tảng vững chắc cho việc xây dựng các ứng dụng phức tạp và dễ bảo trì, mà còn giúp các nhà phát triển nâng cao năng suất và giảm thiểu lỗi trong suốt quá trình phát triển bao gồm:

*- Kiểm tra lỗi ngay khi biên dịch:* Với JavaScript thuần, các lỗi liên quan đến kiểu dữ liệu thường chỉ bộc lộ khi chương trình đang chạy (runtime), gây khó khăn và tốn thời gian cho việc gỡ lỗi. Ngược lại, TypeScript với hệ thống kiểu tĩnh (static typing) sẽ kiểm tra tính nhất quán về kiểu dữ liệu ngay tại thời điểm biên dịch. Nếu có một biến được gán sai kiểu dữ liệu, trình biên dịch sẽ cảnh báo ngay lập tức. Giúp phát hiện và khắc phục lỗi trước khi mã nguồn được triển khai, đảm bảo các thành phần trong ứng dụng tương tác với nhau một cách chính xác.

*- Tăng tính rõ ràng và dễ bảo trì:* Việc khai báo kiểu dữ liệu một cách tường minh cho biến, tham số hàm, và các thuộc tính đối tượng giúp mã nguồn của TypeScript trở nên dễ đọc và dễ hiểu hơn rất nhiều. Tạo ra tài liệu nội bộ, kiểu dữ liệu đóng vai trò như một tài liệu, giúp các lập trình viên mới làm quen với dự án nhanh chóng nắm bắt được mục đích và cách sử dụng của các hàm và đối tượng. Dễ dàng bảo trì và mở rộng, khi cần thay đổi một phần của mã nguồn, các lập trình viên có thể dễ dàng nhận biết tác động của sự thay đổi đó lên các thành phần khác, từ đó giảm thiểu rủi ro và tăng tốc độ bảo trì.

*- Hỗ trợ các tính năng lập trình hướng đối tượng:* TypeScript mang đến một mô hình lập trình hướng đối tượng (OOP) hoàn chỉnh và quen thuộc, giúp các lập trình viên tổ chức mã nguồn một cách có cấu trúc hơn.

+ Class và Interface: Cung cấp cú pháp rõ ràng cho việc định nghĩa lớp (Class) và giao diện (Interface), giúp thiết kế phần mềm có tính mô-đun cao và dễ dàng mở rộng.

+ Generics: Tính năng Generics cho phép tạo ra các thành phần có thể hoạt động với nhiều kiểu dữ liệu khác nhau mà vẫn duy trì tính an toàn về kiểu, giúp mã nguồn linh hoạt hơn.

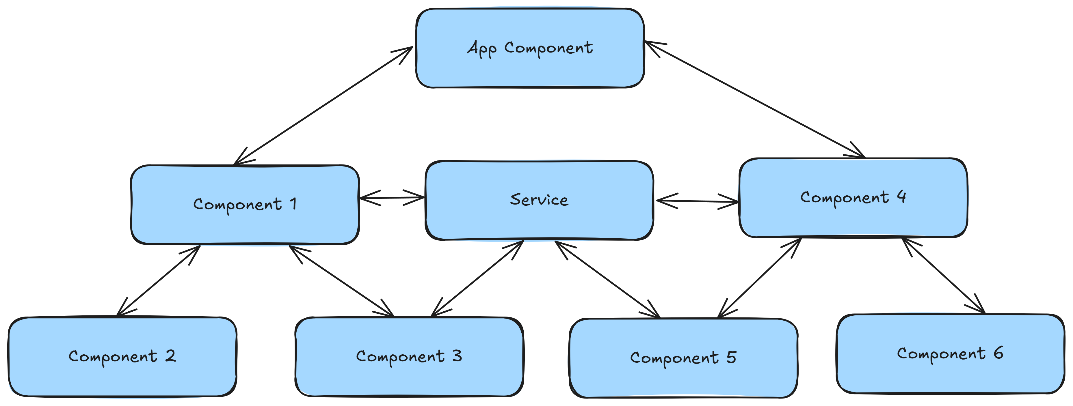
+ Tính kế thừa và đóng gói: Hỗ trợ đầy đủ các nguyên tắc của OOP, cho phép lập trình viên xây dựng các hệ thống lớn một cách nhất quán và có tổ chức.

*- Tích hợp tốt với công cụ phát triển:* TypeScript được xây dựng để hoạt động hiệu quả với các công cụ phát triển hiện đại. Điều này mang lại một trải nghiệm phát triển vượt trội. Gợi ý mã thông minh, dựa trên thông tin kiểu, các IDE như Visual Studio Code có thể cung cấp tính năng gợi ý mã, tự động hoàn thiện và kiểm tra lỗi theo thời gian thực. Tài liệu trực tiếp, khi di chuột qua một hàm hoặc biến, IDE sẽ hiển thị thông tin về kiểu và mô tả, giúp lập trình viên không cần phải chuyển sang một tệp khác để tìm hiểu. Điều này giúp tăng năng suất làm việc và giảm thiểu gián đoạn trong quá trình code.

#### React.js

React.js là một thư viện JavaScript mã nguồn mở được phát triển bởi Facebook (nay là Meta) vào năm 2013, được thiết kế để xây dựng giao diện người dùng (User Interface) một cách hiệu quả và có thể tái sử dụng. React hoạt động dựa trên mô hình Virtual DOM (Document Object Model ảo), cho phép tối ưu hóa hiệu suất bằng cách giảm thiểu việc thao tác trực tiếp trên DOM thực tế.

Các khái niệm cốt lõi của React bao gồm: components (các thành phần có thể tái sử dụng), props (properties để truyền dữ liệu giữa các components), state (trạng thái nội bộ của component), lifecycle methods (các phương thức vòng đời), và hooks (các hàm đặc biệt cho functional components). React cũng hỗ trợ JSX (JavaScript XML), một cú pháp mở rộng cho phép viết HTML trong JavaScript.



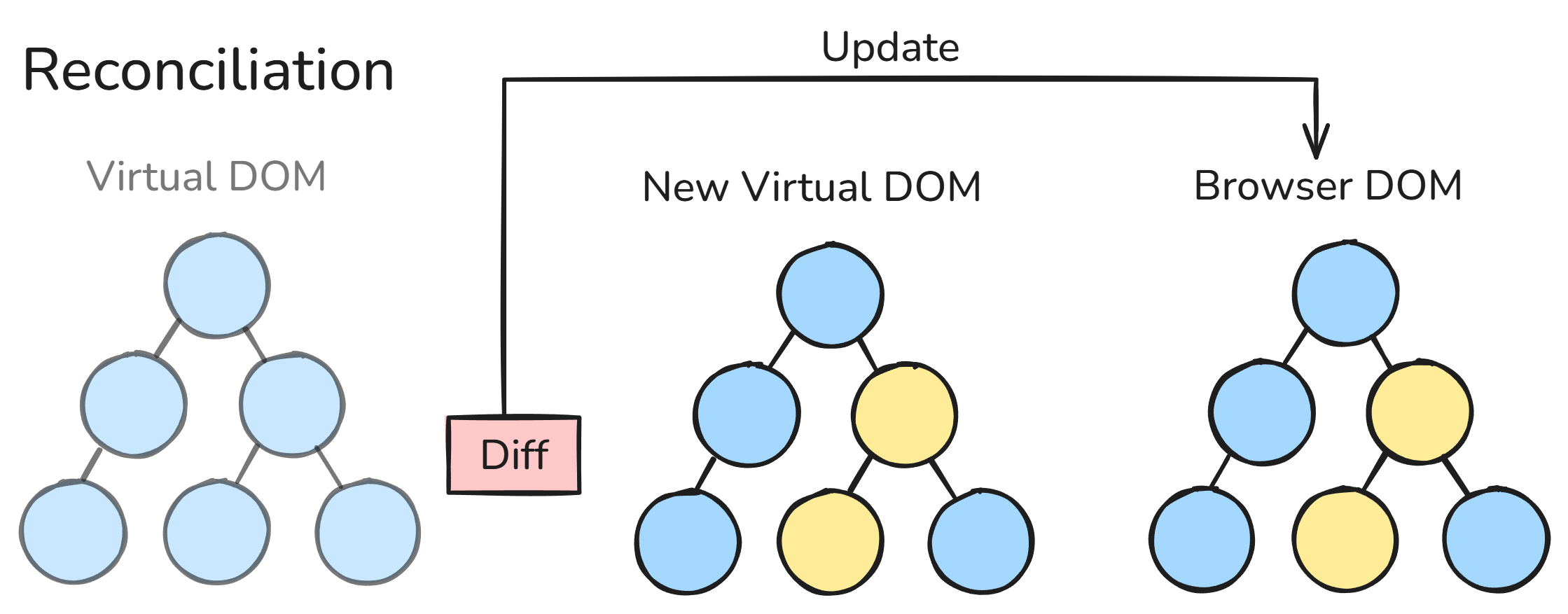
Hình 2.13. Sơ đồ kiến trúc dựa trên Component của React

React.js được xây dựng dựa trên một số nguyên lý cốt lõi, giúp nó trở thành một trong những thư viện phổ biến nhất để xây dựng giao diện người dùng. Những nguyên lý này không chỉ tối ưu hóa hiệu suất mà còn giúp cấu trúc mã nguồn một cách có tổ chức, dễ bảo trì và mở rộng như :

*- UI dưới dạng hàm của trạng thái (UI as a function of state):* Đây là một trong những nguyên lý nền tảng của React. Giao diện người dùng (UI) được xem như một hàm thuần (pure function), với đầu vào là trạng thái (state) của ứng dụng và đầu ra là cấu trúc giao diện tương ứng. Nguyên lý này giúp lập trình viên không cần phải bận tâm đến việc làm thế nào để cập nhật giao diện khi dữ liệu thay đổi. Thay vào đó, họ chỉ cần thay đổi state và React sẽ tự động xử lý việc cập nhật lại giao diện một cách hiệu quả. Điều này làm cho việc phát triển ứng dụng trở nên đơn giản và dễ dự đoán hơn rất nhiều.

*- Kiến trúc dựa trên Component (Component-based architecture):* React khuyến khích tư duy lập trình theo hướng component. Toàn bộ giao diện ứng dụng được chia nhỏ thành các thành phần độc lập, có thể tái sử dụng. Mỗi component bao gồm cả logic (JavaScript) và giao diện (HTML/CSS) của riêng nó. Mỗi component tự quản lý trạng thái và logic của mình, giảm thiểu sự phụ thuộc lẫn nhau giữa các thành phần. Các component như nút bấm, thanh điều hướng, hoặc thẻ sản phẩm có thể được tái sử dụng ở nhiều nơi khác nhau trong ứng dụng, giúp tăng tốc độ phát triển và duy trì tính nhất quán. Việc chia nhỏ ứng dụng thành các component giúp dễ dàng tìm kiếm, gỡ lỗi và bảo trì từng phần một cách độc lập.

*- Virtual DOM:* Để giải quyết vấn đề về hiệu năng khi thao tác trực tiếp với DOM (Document Object Model) thật, React đã giới thiệu khái niệm Virtual DOM.Cơ chế hoạt động, khi trạng thái của ứng dụng thay đổi, React sẽ không cập nhật trực tiếp lên DOM thật. Thay vào đó, nó tạo một bản sao Virtual DOM mới và so sánh với bản cũ.Thuật toán Diffing, react sử dụng một thuật toán so sánh hiệu quả (diffing algorithm) để tìm ra những điểm khác biệt nhỏ nhất giữa hai bản Virtual DOM.Tối ưu hiệu suất, cuối cùng, react chỉ thực hiện các cập nhật cần thiết lên DOM thật, giúp tối ưu hóa hiệu năng, tránh việc render lại toàn bộ giao diện một cách không cần thiết và mang lại trải nghiệm mượt mà cho người dùng.



Hình 2.14. Cơ chế hoạt động của Virtual DOM

*- Luồng dữ liệu một chiều (One-way data flow):* React tuân theo mô hình luồng dữ liệu một chiều, nơi dữ liệu chỉ chảy từ component cha xuống component con thông qua props.Đảm bảo tính dự đoán, dữ liệu chỉ có thể được thay đổi bởi component sở hữu nó (state). Việc này giúp dễ dàng theo dõi luồng dữ liệu và gỡ lỗi khi có vấn đề.Giảm độ phức tạp, mô hình này đơn giản hơn nhiều so với cơ chế ràng buộc hai chiều (two-way binding) của một số framework khác, giúp kiểm soát dữ liệu chặt chẽ hơn và tránh các tác dụng phụ không mong muốn.

*- Giao diện mang tính khai báo (Declarative UI):* Với React, lập trình viên chỉ cần mô tả giao diện mong muốn cho một trạng thái nhất định, thay vì phải viết các bước cụ thể để thao tác với DOM.

Việc lựa chọn React.js làm thư viện phát triển giao diện người dùng mang lại nhiều ưu điểm chiến lược, đặc biệt phù hợp với các dự án phức tạp, đòi hỏi khả năng mở rộng và hiệu suất cao như:

*- Cấu trúc dựa trên Component giúp dễ dàng bảo trì và mở rộng:* React thúc đẩy một kiến trúc dựa trên thành phần (component-based architecture), trong đó toàn bộ giao diện người dùng được chia nhỏ thành các component độc lập, có thể tái sử dụng. Mỗi component đóng gói cả logic và giao diện của riêng nó, tạo ra một hệ thống có tính mô-đun cao. Điều này mang lại các lợi ích sau:

* Tính đóng gói (Encapsulation): Mỗi component hoạt động độc lập, giảm thiểu sự phụ thuộc giữa các thành phần.
* Tính tái sử dụng: Các component như nút bấm, thanh điều hướng, hoặc form nhập liệu có thể được sử dụng lại ở nhiều nơi khác nhau trong ứng dụng, giúp tăng tốc độ phát triển và đảm bảo tính nhất quán của giao diện.
* Dễ bảo trì: Khi cần sửa đổi một component, lập trình viên chỉ cần tập trung vào component đó mà không lo làm ảnh hưởng đến các phần khác của hệ thống.

*- Hiệu năng cao nhờ Virtual DOM:* Một trong những ưu điểm cốt lõi của React là cơ chế Virtual DOM (DOM ảo). Thay vì thao tác trực tiếp với DOM thật của trình duyệt, vốn là một quá trình tốn kém, React sử dụng một bản sao Virtual DOM để tính toán các thay đổi.

* Cơ chế Diffing: Khi có sự thay đổi về trạng thái, React sẽ cập nhật Virtual DOM trước, sau đó sử dụng một thuật toán so sánh hiệu quả (diffing algorithm) để tìm ra những điểm khác biệt.
* Tối ưu hóa cập nhật: Cuối cùng, React chỉ thực hiện các cập nhật cần thiết lên DOM thật, giảm thiểu số lần render lại giao diện, từ đó tối ưu hóa hiệu năng hiển thị và mang lại trải nghiệm mượt mà cho người dùng.

*- Cộng đồng lớn và hệ sinh thái phong phú:* React có một cộng đồng lập trình viên khổng lồ và hệ sinh thái mạnh mẽ, được hỗ trợ bởi Meta và hàng triệu nhà phát triển trên toàn thế giới. Điều này đảm bảo:

* Nguồn tài nguyên dồi dào: Dễ dàng tìm kiếm các tài liệu, hướng dẫn và giải pháp cho mọi vấn đề.
* Hệ sinh thái thư viện đa dạng: Hệ sinh thái của React cung cấp một loạt các thư viện hữu ích cho nhiều mục đích khác nhau, ví dụ: Redux, Zustand, hoặc Recoil, React Query, SWR, React Router.
* Đảm bảo cập nhật liên tục: React luôn được cải tiến và cập nhật để đáp ứng các tiêu chuẩn công nghệ mới nhất.

*- Tích hợp mượt mà với TypeScript:* React tích hợp rất tốt với TypeScript, giúp phát hiện sớm các lỗi liên quan đến kiểu dữ liệu. Điều này mang lại các lợi ích sau:

* Phát hiện lỗi biên dịch sớm: Kết hợp giữa TypeScript và React cho phép kiểm tra lỗi ngay trong quá trình biên dịch, giảm đáng kể các lỗi runtime.
* Tăng tính rõ ràng của code: Việc khai báo kiểu dữ liệu cho props và state giúp code dễ đọc, dễ hiểu và dễ bảo trì hơn, đặc biệt trong các dự án phức tạp.

Ví dụ về cách xây dựng một function component đơn giản trong React để hiển thị danh sách các nhóm dự án. Component GroupList nhận vào một thuộc tính (prop) duy nhất là groups, đây là một mảng đối tượng, mỗi đối tượng đại diện cho một nhóm và chứa thông tin như id, name và danh sách members.

#### Next.js

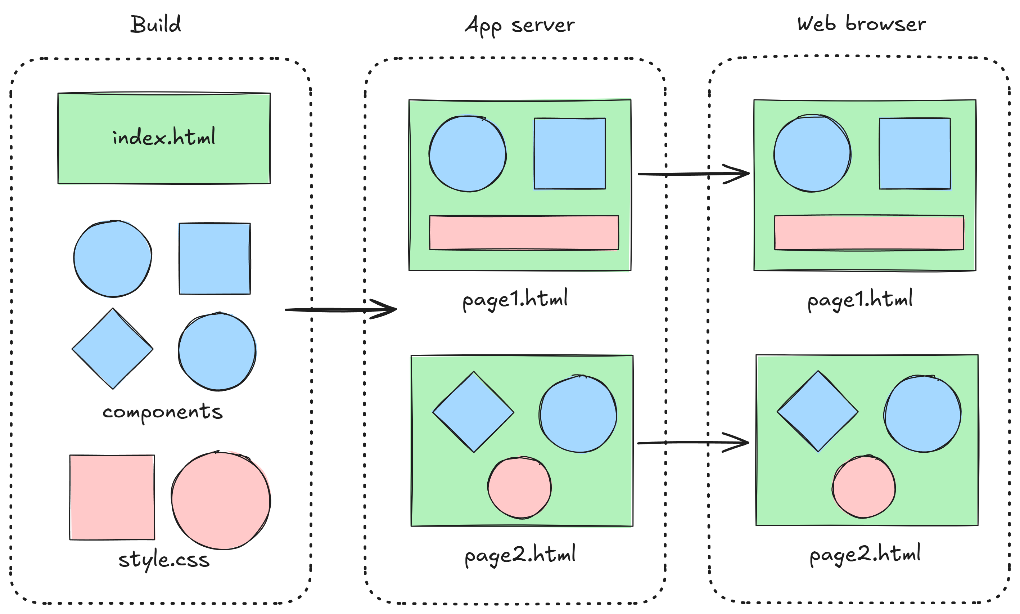
Next.js là một framework phát triển ứng dụng web mã nguồn mở, được xây dựng trên nền tảng React và phát triển bởi Vercel. Mục tiêu của Next.js là khắc phục những hạn chế của React thuần trong việc xử lý tối ưu hóa hiệu năng, SEO và quản lý luồng dữ liệu bằng cách cung cấp một bộ tính năng tích hợp sẵn, từ cơ chế kết xuất nội dung linh hoạt cho đến hệ thống định tuyến và API nội bộ.

Trong khi React chỉ tập trung vào việc xây dựng giao diện người dùng theo hướng Component-Based UI, việc triển khai các cơ chế kết xuất phía máy chủ (Server-Side Rendering – SSR), tạo trang tĩnh (Static Site Generation – SSG), hay tối ưu hóa hình ảnh đều đòi hỏi lập trình viên phải tự cấu hình thủ công hoặc sử dụng thư viện bổ sung. Next.js hợp nhất các tính năng này trong một kiến trúc thống nhất, đồng thời duy trì mô hình phát triển quen thuộc của React, qua đó giảm đáng kể độ phức tạp khi triển khai.

Next.js được lựa chọn cho các dự án web hiện đại nhờ vào các tính năng vượt trội, mang lại hiệu suất cao và trải nghiệm người dùng tối ưu.

*- Các phương thức Render nâng cao (Advanced Rendering Methods):* Next.js cung cấp một cách tiếp cận linh hoạt đối với việc render giao diện, kết hợp giữa render phía máy chủ và render phía trình duyệt để tối ưu hóa hiệu suất.

+ Server-Side Rendering (SSR): Phương pháp này cho phép trang web được render hoàn toàn ở phía máy chủ và gửi mã HTML đã hoàn chỉnh đến trình duyệt. Điều này giúp giảm đáng kể thời gian tải ban đầu và cải thiện điểm số SEO, vì các công cụ tìm kiếm có thể dễ dàng thu thập nội dung.



Hình 2.15 Sơ đồ quy trình Server-Side Rendering

+ Static Site Generation (SSG): Các trang web tĩnh được tạo ra trước khi triển khai (pre-built). Phương pháp này phù hợp với các trang có nội dung ít thay đổi, mang lại tốc độ tải trang gần như tức thì, giảm tải cho máy chủ và tăng cường bảo mật.

+ Incremental Static Regeneration (ISR): Next.js cho phép cập nhật các trang tĩnh đã được tạo sẵn sau khi triển khai, mà không cần phải xây dựng lại toàn bộ trang web. Điều này kết hợp ưu điểm về tốc độ của SSG với khả năng cập nhật dữ liệu theo thời gian thực.

*- Định tuyến dựa trên tệp (File-based Routing):* Next.js đơn giản hóa việc quản lý định tuyến bằng cách tự động tạo các tuyến đường (route) dựa trên cấu trúc thư mục của dự án. Một tệp được đặt trong thư mục app (hoặc pages trong các phiên bản cũ hơn) sẽ tự động trở thành một tuyến đường. Ví dụ, tệp app/about/page.tsx sẽ tương ứng với tuyến đường /about. Điều này giúp quản lý cấu trúc dự án một cách logic và dễ dàng.

*- Tối ưu hóa tự động (Automatic Optimizations):* Next.js tích hợp sẵn các công cụ tối ưu hóa, giảm gánh nặng cho lập trình viên. Thành phần <Image> của Next.js tự động tối ưu hóa kích thước, định dạng (WebP, AVIF), và chất lượng hình ảnh để phù hợp với từng thiết bị, giúp tăng tốc độ tải trang mà không làm giảm chất lượng hình ảnh. Next.js tự động tải và tối ưu phông chữ, ngăn chặn tình trạng CLS (Cumulative Layout Shift), giúp giao diện ổn định hơn trong quá trình tải.

*- Hỗ trợ API Routes:* Next.js cho phép tạo các API endpoints trực tiếp trong cùng một dự án. Bằng cách tạo các tệp trong thư mục app/api, các nhà phát triển có thể xây dựng các API đơn giản để xử lý dữ liệu từ frontend, loại bỏ nhu cầu về một server backend riêng biệt cho các tác vụ nhỏ.

Việc lựa chọn Next.js làm nền tảng phát triển frontend là một quyết định chiến lược, nhằm xây dựng một hệ thống theo dõi dự án không chỉ đáp ứng các yêu cầu về chức năng mà còn vượt trội về hiệu suất, khả năng mở rộng và trải nghiệm người dùng.

- Hiệu suất vượt trội và trải nghiệm người dùng tối ưu: Next.js được thiết kế để giải quyết những vấn đề về hiệu suất mà các ứng dụng React thuần túy thường gặp phải. Các tính năng cốt lõi của Next.js đảm bảo hệ thống có tốc độ tải nhanh và mượt mà. Tối ưu hóa thời gian tải ban đầu, bằng cách sử dụng Server-Side Rendering (SSR) và Static Site Generation (SSG), Next.js render trang web ở phía máy chủ và gửi mã HTML đã hoàn chỉnh đến trình duyệt. Điều này giúp người dùng có thể xem nội dung ngay lập tức, ngay cả khi JavaScript chưa được tải xong, cải thiện đáng kể trải nghiệm ban đầu. Điểm SEO cao hơn, các công cụ tìm kiếm có thể dễ dàng thu thập nội dung từ các trang đã được render sẵn, từ đó cải thiện thứ hạng của trang web. Next.js tự động tối ưu hóa hình ảnh bằng cách nén, định dạng lại và tải theo yêu cầu. Điều này giúp giảm kích thước tệp, tăng tốc độ tải trang mà vẫn giữ được chất lượng hình ảnh.

*- Tăng tốc độ phát triển và đơn giản hóa quy trình:* Next.js cung cấp một môi trường phát triển hiệu quả, giúp lập trình viên tập trung vào logic cốt lõi của ứng dụng thay vì các cấu hình phức tạp. Next.js sử dụng định tuyến dựa trên cấu trúc tệp (file-based routing), loại bỏ nhu cầu phải cấu hình định tuyến thủ công. Điều này làm cho việc quản lý các trang và tuyến đường trở nên đơn giản và dễ hiểu. Khả năng tạo các API endpoint trực tiếp trong dự án cho phép các lập trình viên xây dựng các API đơn giản một cách nhanh chóng, đặc biệt hữu ích cho việc xử lý các tác vụ ở phía backend mà không cần một server riêng biệt.

- Khả năng mở rộng và cấu trúc vững chắc: Đối với một hệ thống được dự định sẽ phát triển theo thời gian, khả năng mở rộng là một yếu tố then chốt. Next.js cung cấp một cấu trúc vững chắc để xây dựng các ứng dụng phức tạp. Khả năng lựa chọn giữa SSR, SSG và ISR cho phép lập trình viên tối ưu hóa từng phần của ứng dụng tùy theo yêu cầu cụ thể, giúp hệ thống có thể mở rộng và đáp ứng nhiều nhu cầu khác nhau. Hỗ trợ bởi một hệ sinh thái mạnh mẽ của React và các thư viện chuyên dụng, giúp dễ dàng tích hợp các công nghệ mới và mở rộng chức năng của hệ thống trong tương lai.

#### Tailwind CSS

Tailwind CSS là một utility-first CSS framework được phát triển bởi Adam Wathan, cung cấp một bộ các class CSS có thể tái sử dụng để xây dựng giao diện người dùng một cách nhanh chóng và nhất quán. Framework này hoạt động dựa trên nguyên tắc "utility-first", có nghĩa là thay vì viết CSS tùy chỉnh, developers sử dụng các class có sẵn để tạo ra các style mong muốn.

Các đặc điểm nổi bật của Tailwind CSS bao gồm: responsive design utilities (các tiện ích thiết kế responsive), color system (hệ thống màu sắc), spacing scale (thang đo khoảng cách), typography utilities (các tiện ích typography), flexbox và grid utilities, và dark mode support. Tailwind CSS cũng cung cấp JIT (Just-In-Time) compiler, cho phép tạo ra CSS tối ưu chỉ cho các class được sử dụng thực tế.

Trong bối cảnh xây dựng một hệ thống theo dõi và đánh giá dự án phần mềm của sinh viên, việc lựa chọn một framework CSS hiệu quả là rất quan trọng. Tailwind CSS với triết lý "utility-first" đã được chọn để giải quyết các yêu cầu về tốc độ phát triển, tính nhất quán và hiệu suất của giao diện người dùng.

*- Tăng tốc độ phát triển giao diện (Rapid UI Development):* Đối với một dự án cần triển khai nhanh chóng các trang giao diện phức tạp như dashboard, biểu đồ thống kê, và bảng báo cáo, Tailwind CSS mang lại lợi thế vượt trội. Lập trình viên không cần phải chuyển đổi liên tục giữa các tệp HTML và CSS. Thay vào đó, họ có thể trực tiếp áp dụng các lớp tiện ích như *flex, grid, p-4, bg-blue-500* vào mã JSX của React để xây dựng giao diện. Điều này giúp tăng tốc độ làm việc đáng kể, cho phép tập trung vào logic ứng dụng hơn là các chi tiết về styling. Tailwind khuyến khích tư duy thiết kế theo hướng "atomic design", nơi giao diện được xây dựng từ các khối nhỏ nhất. Điều này giúp dễ dàng tạo ra các layout và component phức tạp một cách linh hoạt, đáp ứng yêu cầu đa dạng của các trang quản lý.

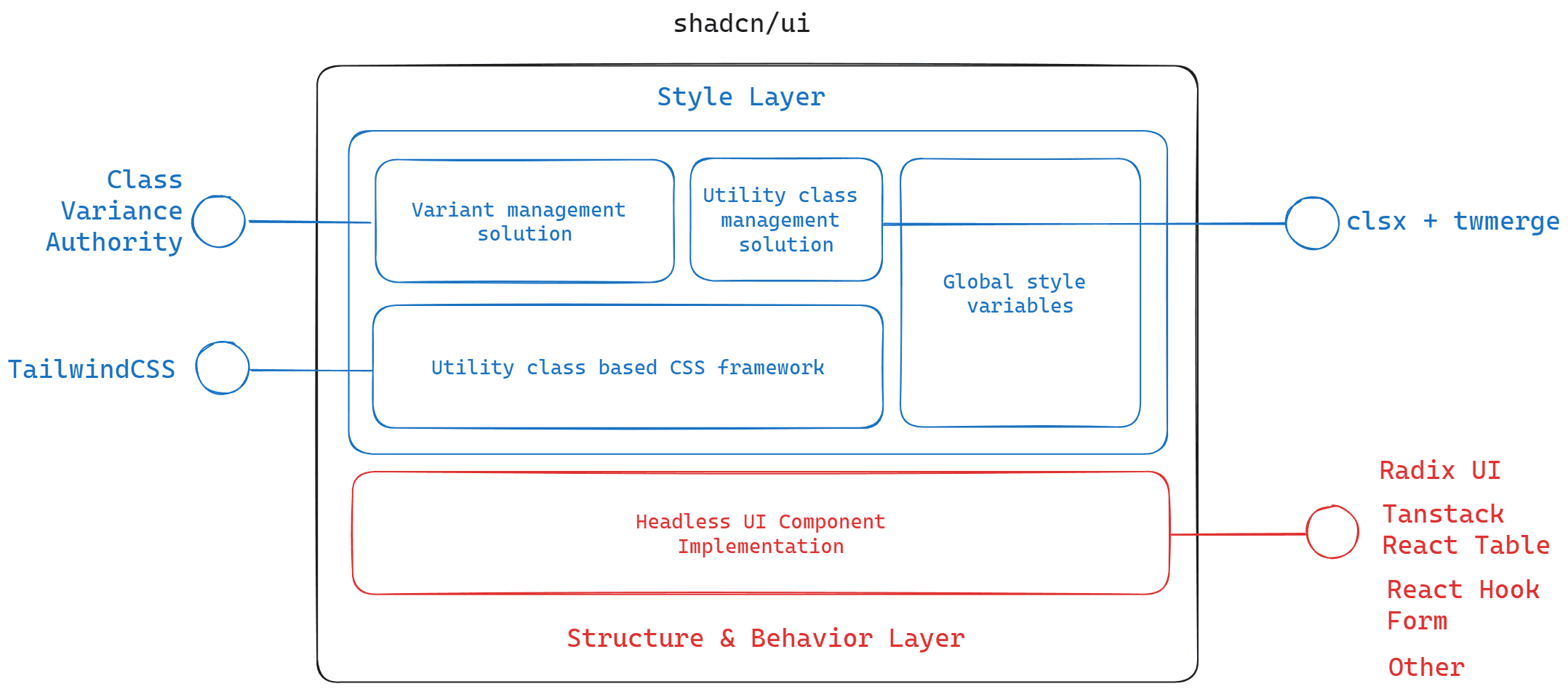
*- Đảm bảo tính nhất quán và dễ bảo trì:* Một hệ thống lớn cần có giao diện đồng nhất để mang lại trải nghiệm người dùng tốt. Tailwind CSS giải quyết vấn đề này một cách hiệu quả. Tailwind cho phép tùy chỉnh các giá trị mặc định như bảng màu, font chữ, khoảng cách trong tệp tailwind.config.js. Khi các giá trị này được định nghĩa, chúng sẽ được sử dụng nhất quán trên toàn bộ ứng dụng, đảm bảo các thành phần như nút bấm, thẻ thông tin, hay biểu đồ đều có chung một phong cách. Phương pháp "utility-first" giúp loại bỏ các lớp CSS tùy chỉnh không cần thiết, tránh tình trạng mã CSS bị trùng lặp và khó quản lý theo thời gian. Khi cần thay đổi, chỉ cần chỉnh sửa trong tệp cấu hình hoặc thay đổi các lớp tiện ích trong HTML.

*- Tối ưu hiệu năng tải trang (Optimized Performance):* Hiệu suất là một yếu tố then chốt, đặc biệt đối với các trang hiển thị dữ liệu lớn. Tailwind CSS góp phần cải thiện hiệu suất tải trang thông qua cơ chế tối ưu hóa. Tailwind CSS sử dụng Just-In-Time (JIT) compiler để chỉ tạo ra các lớp CSS mà bạn thực sự sử dụng trong mã nguồn. Nhờ JIT compiler và công cụ PurgeCSS tích hợp, dung lượng của tệp CSS cuối cùng luôn ở mức tối thiểu, thường chỉ vài KB. Điều này giúp các trang thống kê và báo cáo có thể tải nhanh ngay cả khi dữ liệu lớn, mang lại trải nghiệm mượt mà cho giảng viên và sinh viên.

*- Tích hợp mượt mà với công nghệ hiện đại:* Tailwind CSS được thiết kế để hoạt động tốt với các framework và thư viện frontend hiện đại. Tailwind CSS tương thích hoàn toàn với cú pháp JSX của React và Next.js. Điều này cho phép tạo ra các giao diện động (dynamic UI) một cách dễ dàng, nơi các lớp tiện ích có thể được áp dụng có điều kiện dựa trên dữ liệu từ các API (như GitHub, SonarQube) hoặc trạng thái của hệ thống. Tối ưu hóa với Next.js: Khi kết hợp với Next.js, Tailwind CSS hoạt động hiệu quả trong cả hai môi trường Server-Side Rendering (SSR) và Static Site Generation (SSG), giúp tối ưu hóa hiệu suất render ở cả phía máy chủ và phía trình duyệt.

#### Shadcn/ui

Shadcn/ui là một collection của các component React có thể tái sử dụng, được xây dựng dựa trên Radix UI primitives và Tailwind CSS. Khác với các UI library truyền thống, shadcn/ui không phải là một package được cài đặt thông qua npm, mà là một collection các component có thể copy vào dự án và tùy chỉnh theo nhu cầu cụ thể.



Hình 2.18 Kiến trúc của Shadcn/ui

Các component trong shadcn/ui bao gồm: form components (các component form), navigation components (các component điều hướng), feedback components (các component phản hồi), data display components (các component hiển thị dữ liệu), và overlay components (các component lớp phủ). Mỗi component đều được thiết kế với accessibility (khả năng tiếp cận) cao, responsive design, và dark mode support.

Shadcn/ui là một bộ sưu tập các thành phần giao diện (UI components) được xây dựng trên một triết lý độc đáo, kết hợp sự mạnh mẽ của các thư viện nền tảng với sự linh hoạt của việc sở hữu mã nguồn. Điều này mang lại một cách tiếp cận hiệu quả cho việc phát triển giao diện người dùng, đặc biệt trong các dự án phức tạp và yêu cầu cao về tùy biến.

*- Nền tảng kỹ thuật Tailwind CSS và Radix UI:* Shadcn/ui không phải là một framework độc lập mà là sự kết hợp của hai công nghệ nền tảng:

*+* Tailwind CSS: Đóng vai trò là lớp định kiểu (styling layer). Thay vì sử dụng các file CSS truyền thống, Shadcn/ui áp dụng hệ thống lớp tiện ích (utility classes) của Tailwind CSS. Điều này cho phép người dùng tùy biến nhanh chóng về màu sắc, kích thước, khoảng cách và font chữ của các thành phần một cách trực tiếp trong file JSX.

+ Radix UI: Cung cấp lớp logic và khả năng truy cập (accessibility). Radix UI là một thư viện headless (không có giao diện) chuyên biệt, tập trung vào việc xử lý các hành vi phức tạp của các thành phần như dialog, dropdown menu hoặc date picker, đồng thời đảm bảo tuân thủ các tiêu chuẩn về khả năng truy cập (WAI-ARIA), giúp các thành phần dễ dàng tương tác bằng bàn phím hoặc các thiết bị hỗ trợ.

*- Hướng tiếp cận "Composable":* Các thành phần của Shadcn/ui được thiết kế theo mô hình composition, cho phép kết hợp các thành phần nhỏ để tạo ra các giao diện phức tạp. Ví dụ, một thành phần Card có thể được tạo bằng cách kết hợp CardHeader, CardTitle, CardDescription và CardContent. Điều này giúp mã nguồn có cấu trúc rõ ràng, dễ đọc và dễ bảo trì.

- *Tối ưu hiệu năng và khả năng truy cập:* Shadcn/ui thừa hưởng các ưu điểm về hiệu năng và khả năng truy cập từ các công nghệ nền tảng.Việc không sử dụng các thư viện đóng gói cồng kềnh giúp giảm dung lượng bundle. Kết hợp với Tailwind CSS, Shadcn/ui tối ưu hóa kích thước CSS, đảm bảo thời gian tải trang nhanh. Nhờ Radix UI, các thành phần của Shadcn/ui tuân thủ các chuẩn WAI-ARIA và hỗ trợ các thiết bị hỗ trợ (assistive technologies), đảm bảo tính khả dụng cho mọi nhóm người dùng.

#### Next-i18next

Next-i18next là một giải pháp internationalization cho Next.js applications, được xây dựng dựa trên i18next framework. Công nghệ này cho phép developers xây dựng ứng dụng web đa ngôn ngữ một cách dễ dàng và hiệu quả, hỗ trợ các tính năng như dynamic language switching, pluralization, interpolation, và context-aware translations.

Các tính năng chính của next-i18next bao gồm: language detection (phát hiện ngôn ngữ), namespace management (quản lý không gian tên), fallback languages (ngôn ngữ dự phòng), lazy loading (tải lười), và SEO optimization. next-i18next cũng hỗ trợ các format như JSON, YAML, và custom loaders, cho phép tích hợp với các hệ thống translation management khác nhau.

Next-i18next được thiết kế để giải quyết thách thức của việc xử lý đa ngôn ngữ một cách nhất quán trên cả môi trường render phía máy chủ (Server-Side Rendering - SSR) và phía trình duyệt (Client-Side Rendering - CSR) của Next.js. Thư viện này hoạt động dựa trên một luồng dữ liệu logic, đảm bảo các bản dịch được tải và phân phối hiệu quả tới các component.

*- Tự động phát hiện và chuyển hướng ngôn ngữ:* Khi người dùng truy cập vào ứng dụng lần đầu, Next-i18next sẽ thực hiện các bước sau để xác định ngôn ngữ ưu tiên:

+ Kiểm tra URL: Thư viện sẽ ưu tiên kiểm tra các tham số ngôn ngữ trên URL, ví dụ: /en/about hoặc /vi/about. Đây là phương pháp phổ biến và có lợi cho SEO.

+ Đọc cài đặt trình duyệt: Nếu không có tham số ngôn ngữ trên URL, Next-i18next sẽ đọc các cài đặt ngôn ngữ từ trình duyệt của người dùng thông qua tiêu đề HTTP Accept-Language.

+ Chuyển hướng (Redirection): Dựa trên ngôn ngữ đã phát hiện, Next-i18next sẽ tự động chuyển hướng người dùng đến phiên bản trang web có ngôn ngữ tương ứng.

### Công nghệ Backend

#### Express.js

Express.js là một web application framework nhỏ gọn và linh hoạt cho Node.js, được phát triển bởi TJ Holowaychuk vào năm 2010. Framework này cung cấp một bộ tính năng mạnh mẽ cho việc xây dựng web applications và APIs, bao gồm routing, middleware support, template engines, và static file serving.

Express.js hoạt động dựa trên mô hình middleware, cho phép xử lý request và response thông qua một chuỗi các hàm trung gian. Các tính năng chính bao gồm: routing system (hệ thống định tuyến), middleware functions (các hàm trung gian), template engines (công cụ tạo template), static file serving (phục vụ file tĩnh), và error handling (xử lý lỗi). Framework này cũng hỗ trợ các HTTP methods (GET, POST, PUT, DELETE), query string parsing, và cookie handling.

A diagram of a cloud and a gear

AI-generated content may be incorrect.

Hình 2.19 Sơ đồ kiến trúc của Express.js

Về bản chất, Express.js hoạt động như một lớp trừu tượng (abstraction layer) giúp giảm bớt độ phức tạp khi làm việc trực tiếp với các mô-đun HTTP gốc của Node.js, vốn yêu cầu lập trình viên phải tự quản lý nhiều tác vụ ở mức thấp như phân tích dữ liệu yêu cầu, xử lý header, hay định tuyến URL. Bằng cách cung cấp một bộ API gọn gàng và dễ hiểu, Express.js cho phép nhà phát triển tập trung vào logic nghiệp vụ thay vì xử lý các chi tiết kỹ thuật phức tạp. Mặc dù đơn giản hóa quy trình phát triển, Express.js vẫn giữ nguyên khả năng tùy biến sâu, cho phép truy cập trực tiếp đến các lớp thấp hơn của Node.js khi cần. Các đặc điểm nổi bật có thể được phân tích chi tiết như sau:

*- Hệ thống Routing mạnh mẽ:* cho phép xác định và quản lý các tuyến đường (routes) dựa trên phương thức HTTP và đường dẫn URL. Các tuyến này có thể nhận tham số động trực tiếp từ URL, kết hợp với dữ liệu từ query string hoặc request body để đáp ứng nhiều tình huống xử lý khác nhau. Khả năng tổ chức các tuyến thành module riêng biệt giúp mã nguồn gọn gàng và dễ bảo trì, đặc biệt trong các ứng dụng lớn.

*- Cơ chế xử lý trung gian (Middleware):* Middleware có thể coi như các lớp xử lý trung gian được thực thi tuần tự giữa lúc nhận yêu cầu và gửi phản hồi. Mỗi middleware có thể thao tác, biến đổi hoặc xác thực dữ liệu trước khi chuyển sang bước tiếp theo. Cơ chế này tạo ra một kiến trúc mở rộng linh hoạt, cho phép dễ dàng bổ sung các tính năng như phân tích dữ liệu đầu vào, xác thực người dùng, xử lý lỗi tập trung hoặc tăng cường bảo mật bằng các thư viện như Helmet.

*- Khả năng tích hợp đa dạng:* Express.js có khả năng tích hợp mạnh mẽ với các công nghệ khác trong hệ sinh thái phát triển web. Nó có thể kết nối với hầu hết các hệ quản trị cơ sở dữ liệu thông qua ORM hoặc thư viện chuyên dụng, đồng thời hỗ trợ nhiều hệ thống template engine như Pug, EJS hoặc Handlebars để tạo nội dung HTML động. Khả năng giao tiếp với các dịch vụ và API bên ngoài cũng được đảm bảo nhờ vào việc tận dụng sức mạnh của Node.js trong xử lý bất đồng bộ.

*- Hiệu năng và khả năng mở rộng:* Express.js thừa hưởng kiến trúc non-blocking và event-driven của Node.js, cho phép xử lý một lượng lớn kết nối đồng thời với tốc độ phản hồi cao mà không cần phải tạo luồng mới cho từng yêu cầu. Đặc điểm này giúp nó phù hợp với nhiều loại ứng dụng, từ các hệ thống thời gian thực như chat và trò chơi trực tuyến, cho đến các dịch vụ microservices và API backend phục vụ SPA hoặc ứng dụng di động.

#### MySQL

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ (RDBMS) mã nguồn mở, phổ biến nhất trên thế giới. Được phát triển bởi Oracle Corporation, MySQL sử dụng ngôn ngữ tiêu chuẩn SQL (Structured Query Language) để quản lý, truy vấn và thao tác dữ liệu. MySQL nổi tiếng với độ tin cậy, hiệu suất cao và khả năng dễ dàng tích hợp với các ứng dụng web, đặc biệt là trong các kiến trúc LAMP (Linux, Apache, MySQL, PHP/Python/Perl) và MERN/MEVN (MongoDB/MySQL, Express, React/Vue, Node).

MySQL được sử dụng rộng rãi trong các hệ thống web, ứng dụng doanh nghiệp và hệ thống nhúng nhờ vào tính ổn định, hiệu năng và khả năng mở rộng. Một số tính năng quan trọng gồm:

*- Tính tương thích cao với SQL tiêu chuẩn:* MySQL hỗ trợ phần lớn các cú pháp SQL chuẩn ANSI/ISO, là nền tảng của mọi hệ quản trị cơ sở dữ liệu quan hệ. Điều này mang lại sự linh hoạt cao, giúp các nhà phát triển đã quen thuộc với SQL có thể làm việc hiệu quả với MySQL ngay lập tức. Tính tương thích này cũng tạo điều kiện thuận lợi cho việc chuyển đổi hoặc tích hợp với các hệ thống cơ sở dữ liệu khác khi cần thiết, giảm thiểu chi phí học tập và phát triển.

*- Hiệu năng tối ưu:* MySQL được thiết kế để xử lý khối lượng lớn dữ liệu và hàng nghìn truy vấn đồng thời một cách hiệu quả. MySQL sử dụng kiến trúc đa luồng để xử lý các truy vấn song song, tận dụng tối đa sức mạnh của bộ vi xử lý đa nhân. Cơ chế này lưu trữ kết quả của các truy vấn thường xuyên, giúp tăng tốc độ phản hồi cho các truy vấn lặp lại mà không cần phải truy cập lại dữ liệu trên đĩa. Khả năng lựa chọn engine phù hợp (như InnoDB cho các tác vụ ghi/đọc cân bằng hoặc MyISAM cho các tác vụ đọc chuyên sâu) cho phép tối ưu hóa hiệu suất dựa trên yêu cầu cụ thể của ứng dụng.

*- Đa dạng công cụ lưu trữ (Storage Engines):* Đây là một trong những tính năng độc đáo và mạnh mẽ nhất của MySQL. Các engine lưu trữ khác nhau cho phép lập trình viên tùy chỉnh hành vi và hiệu suất của từng bảng.InnoDB là engine mặc định và khuyến nghị, hỗ trợ đầy đủ các tính năng của cơ sở dữ liệu quan hệ, đảm bảo tính nguyên tử (atomicity), nhất quán (consistency), độc lập (isolation) và bền vững (durability) của dữ liệu. Duy trì tính toàn vẹn tham chiếu giữa các bảng. MyISAM là engine cũ hơn, phù hợp cho các ứng dụng chỉ cần đọc dữ liệu. MyISAM không hỗ trợ giao dịch nhưng có tốc độ truy vấn đọc nhanh hơn trong một số trường hợp.Với memory lưu trữ dữ liệu trực tiếp trong RAM, mang lại tốc độ truy cập cực nhanh, thích hợp cho các bảng dữ liệu tạm thời hoặc cache.

*- Hệ thống sao lưu và phục hồi dữ liệu mạnh mẽ:* MySQL cung cấp nhiều công cụ và tính năng để sao lưu và phục hồi dữ liệu, đảm bảo tính bền vững và khả năng phục hồi của hệ thống. Các phương pháp sao lưu bao gồm sao lưu vật lý (physical backup) và sao lưu logic (logical backup), giúp người quản trị có thể sao lưu toàn bộ cơ sở dữ liệu hoặc từng phần nhỏ một cách hiệu quả.

*- Quản lý người dùng và phân quyền chi tiết:* MySQL cung cấp một hệ thống bảo mật chặt chẽ, cho phép quản lý người dùng và phân quyền một cách chi tiết. Người quản trị có thể, tạo tài khoản người dùng và xác thực cung cấp các tài khoản với mật khẩu được mã hóa, cấp quyền chi tiết và chỉ định các quyền cụ thể (ví dụ: SELECT, INSERT, UPDATE) cho từng tài khoản trên từng bảng hoặc thậm chí từng cột, đảm bảo rằng mỗi người dùng chỉ có thể thực hiện các hành động được phép.

*- Khả năng mở rộng (Scalability):* MySQL có khả năng mở rộng linh hoạt, phù hợp với các ứng dụng từ quy mô nhỏ đến hệ thống lớn, xử lý hàng tỷ bản ghi. Phân chia cơ sở dữ liệu thành nhiều mảnh nhỏ để lưu trữ trên các máy chủ khác nhau, giúp phân tán tải. Tạo các bản sao của cơ sở dữ liệu để tăng khả năng chịu tải đọc và đảm bảo tính sẵn sàng cao.

Việc lựa chọn MySQL làm hệ quản trị cơ sở dữ liệu mang lại nhiều lợi ích chiến lược, giúp các ứng dụng đạt được hiệu suất cao, độ tin cậy và khả năng mở rộng cần thiết như:

*- Mã nguồn mở và tối ưu chi phí:* MySQL Community Edition là một phiên bản hoàn toàn miễn phí, mang lại lợi ích lớn về mặt chi phí cho các cá nhân, startup và các tổ chức. Điều này không chỉ giúp tiết kiệm chi phí bản quyền ban đầu mà còn giảm chi phí vận hành lâu dài. Sự miễn phí và tính mở của MySQL còn thúc đẩy sự phát triển của một hệ sinh thái các công cụ, thư viện và giải pháp hỗ trợ xung quanh, tạo ra một môi trường phát triển linh hoạt và tiết kiệm.

*- Hiệu suất cao và đáng tin cậy:* MySQL nổi tiếng với khả năng hoạt động ổn định và hiệu quả trong môi trường tải cao. Được thiết kế để xử lý khối lượng lớn truy vấn đồng thời, MySQL sử dụng kiến trúc đa luồng (multi-threaded architecture) để tận dụng tối đa sức mạnh của bộ vi xử lý đa nhân. Các cơ chế tối ưu hiệu suất như bộ nhớ đệm truy vấn (query cache) giúp tăng tốc độ phản hồi cho các truy vấn lặp lại, trong khi sự đa dạng của các engine lưu trữ (như InnoDB) cho phép tối ưu hóa hiệu suất dựa trên yêu cầu cụ thể của từng bảng dữ liệu.

*- Cộng đồng và tài liệu phong phú:* Với hàng triệu người dùng và lập trình viên trên toàn thế giới, MySQL có một cộng đồng hỗ trợ mạnh mẽ và một kho tàng tài liệu khổng lồ. Trang chủ MySQL cung cấp tài liệu chi tiết, hướng dẫn và ví dụ từ cơ bản đến nâng cao. Các diễn đàn, blog, và khóa học trực tuyến giúp việc học và triển khai MySQL trở nên dễ dàng hơn bao giờ hết, đồng thời cung cấp một nguồn tài nguyên quý báu để giải quyết các vấn đề phát sinh.

*- Bảo mật toàn diện:* MySQL cung cấp một hệ thống bảo mật mạnh mẽ, giúp bảo vệ dữ liệu ở nhiều cấp độ khác nhau.Mã hóa dữ liệu, MySQL tích hợp cơ chế mã hóa dữ liệu truyền qua mạng (SSL/TLS) để đảm bảo dữ liệu không bị chặn hoặc đánh cắp.Hệ thống cung cấp các phương thức xác thực mạnh mẽ và khả năng phân quyền chi tiết cho từng tài khoản, giúp người quản trị có thể kiểm soát quyền truy cập của từng người dùng trên từng bảng hoặc thậm chí từng cột dữ liệu.

*- Tích hợp dễ dàng và khả năng nhân bản:* MySQL có khả năng tích hợp mượt mà với hầu hết các ngôn ngữ lập trình và framework phổ biến như PHP, Java, Python, Node.js, .NET.Một trong những tính năng mạnh mẽ nhất của MySQL là khả năng nhân bản dữ liệu, cho phép tạo các bản sao của cơ sở dữ liệu trên nhiều máy chủ. Điều này không chỉ giúp cân bằng tải đọc (read balancing) mà còn đảm bảo tính sẵn sàng cao (High Availability), giúp ứng dụng vẫn hoạt động ngay cả khi máy chủ chính gặp sự cố.

#### Sequelize ORM

Sequelize là một Object-Relational Mapping (ORM) library mạnh mẽ cho Node.js, hỗ trợ nhiều hệ quản trị cơ sở dữ liệu quan hệ khác nhau như PostgreSQL, MySQL, SQLite, và Microsoft SQL Server. ORM này cung cấp một abstraction layer giữa cơ sở dữ liệu và mã nguồn ứng dụng, cho phép developers làm việc với dữ liệu thông qua các object thay vì viết trực tiếp các câu lệnh SQL.

A diagram of a mobile application

AI-generated content may be incorrect.

Hình 2.20 Sơ đồ kiến trúc của Sequelize ORM

Các tính năng nổi bật của Sequelize bao gồm: model definition (định nghĩa model), associations (các mối quan hệ giữa các model), migrations (di chuyển cấu trúc cơ sở dữ liệu), seeders (dữ liệu mẫu), querying (truy vấn dữ liệu), validation (xác thực dữ liệu), và hooks (các hàm được gọi tại các thời điểm cụ thể). Sequelize cũng hỗ trợ transactions, connection pooling, và multiple database connections. Nguyên lý hoạt động và các đặc điểm chính của Sequelize ORM:

*- Chuyển đổi giữa Đối tượng và Dữ liệu quan hệ:* Sequelize thực hiện việc ánh xạ (mapping) hai chiều giữa các đối tượng trong ứng dụng và các hàng/cột trong cơ sở dữ liệu. Quá trình này được thực hiện thông qua việc định nghĩa Models. Mỗi Model là một lớp JavaScript, đại diện cho một bảng trong cơ sở dữ liệu. Các thuộc tính (fields) của Model tương ứng với các cột của bảng, cho phép lập trình viên định nghĩa cấu trúc dữ liệu một cách trực quan. Khi các thao tác như tạo, tìm kiếm, cập nhật hoặc xóa được gọi trên các Model này (ví dụ: User.create()), Sequelize sẽ tự động dịch chúng thành các câu lệnh SQL phù hợp và thực thi trên cơ sở dữ liệu. Cơ chế tự động hóa SQL này không chỉ làm tăng tốc độ phát triển mà còn giảm thiểu đáng kể các lỗi cú pháp và lỗ hổng bảo mật liên quan đến SQL Injection.

*- Quản lý và định nghĩa các mối quan hệ (Associations):* Một trong những tính năng cốt lõi của Sequelize là khả năng quản lý các mối quan hệ giữa các Models một cách hiệu quả. Thay vì phải quản lý các khóa ngoại (foreign keys) một cách thủ công, Sequelize cung cấp một API rõ ràng để định nghĩa các loại quan hệ phổ biến:

+ Một-đến-một (One-to-One): Được thể hiện qua các phương thức như hasOne và belongsTo.

+ Một-đến-nhiều (One-to-Many): Sử dụng hasMany và belongsTo.

+ Nhiều-đến-nhiều (Many-to-Many): Được quản lý bằng phương thức belongsToMany.

*- Hỗ trợ Quản lý Cấu trúc Dữ liệu (Migrations và Seeders):* Sequelize cung cấp một bộ công cụ mạnh mẽ để quản lý vòng đời của cơ sở dữ liệu, từ việc thay đổi cấu trúc cho đến dữ liệu ban đầu. Các tệp kịch bản cho phép các lập trình viên tạo, áp dụng và hoàn tác các thay đổi về cấu trúc cơ sở dữ liệu một cách có hệ thống. Điều này giúp đảm bảo rằng cấu trúc cơ sở dữ liệu luôn được đồng bộ giữa các môi trường phát triển và sản xuất, giải quyết vấn đề "lệch schema" thường gặp trong các dự án lớn.Seeders là các tệp kịch bản được sử dụng để chèn dữ liệu ban đầu hoặc dữ liệu mẫu vào cơ sở dữ liệu. Chức năng này cực kỳ hữu ích cho việc khởi tạo môi trường phát triển, kiểm thử tự động, hoặc điền các dữ liệu tĩnh ban đầu cần thiết cho ứng dụng.

#### Firebase Authentication

Firebase Authentication là một dịch vụ authentication được cung cấp bởi Google, cung cấp các phương thức xác thực người dùng an toàn và dễ sử dụng cho web applications và mobile apps. Dịch vụ này hỗ trợ nhiều phương thức xác thực khác nhau như email/password, phone number, Google, Facebook, Twitter, và các social login providers khác.

Các tính năng nổi bật của Firebase Authentication bao gồm: multiple sign-in methods (nhiều phương thức đăng nhập), user management (quản lý người dùng), security features (các tính năng bảo mật), cross-platform support (hỗ trợ đa nền tảng), và integration với các Firebase services khác. Firebase Authentication cũng cung cấp các tính năng như email verification, password reset, account linking, và custom claims.

Firebase Authentication được xây dựng dựa trên nguyên tắc đơn giản hóa và bảo mật. Dịch vụ này cung cấp một API thống nhất để quản lý các quy trình xác thực, giúp giảm thiểu rủi ro và gánh nặng phát triển với các tính năng và nguyên lý hoạt động như:

*- Hỗ trợ đa dạng các phương thức xác thực:* Một trong những ưu điểm nổi bật của Firebase Authentication là sự linh hoạt trong việc hỗ trợ nhiều phương thức đăng nhập khác nhau.

+ Email và Mật khẩu: Đây là phương thức cơ bản và phổ biến nhất. Firebase xử lý toàn bộ quy trình từ đăng ký, đăng nhập, đến đặt lại mật khẩu một cách an toàn bằng cách sử dụng các cơ chế mã hóa tiêu chuẩn và quản lý token.

+ Đăng nhập liên kết (Federated Identity): Firebase tích hợp dễ dàng với các nhà cung cấp danh tính bên thứ ba như Google, Facebook, Twitter, và GitHub. Điều này cho phép người dùng đăng nhập bằng tài khoản mạng xã hội hiện có của họ, giúp cải thiện trải nghiệm người dùng bằng cách loại bỏ bước tạo tài khoản mới.

+ Đăng nhập ẩn danh: Firebase cũng cung cấp tùy chọn đăng nhập ẩn danh, tạo một tài khoản tạm thời cho người dùng mà không cần thông tin cá nhân. Tài khoản này có thể được nâng cấp thành tài khoản vĩnh viễn sau này, tạo ra một trải nghiệm chuyển đổi mượt mà.

*- Bảo mật và quản lý phiên đăng nhập:* Firebase Authentication được thiết kế với sự ưu tiên cao nhất cho bảo mật. Tất cả thông tin người dùng được mã hóa và lưu trữ an toàn trên các máy chủ của Google.

+ Quản lý phiên (Session Management): Firebase tự động xử lý việc tạo, làm mới và quản lý các token xác thực, đảm bảo rằng phiên đăng nhập của người dùng luôn an toàn và được duy trì một cách hiệu quả. Điều này giúp các nhà phát triển không cần phải lo lắng về việc triển khai các cơ chế quản lý phiên phức tạp.

+ SDK đa nền tảng: Dịch vụ cung cấp các SDK (Software Development Kit) dễ sử dụng cho các nền tảng phổ biến như Web, iOS, và Android. Các SDK này giúp việc tích hợp trở nên đơn giản, nhanh chóng và đồng bộ trên mọi nền tảng.

Với dự án hệ thống theo dõi dự án phần mềm sinh viên, Firebase Authentication đóng một vai trò nền tảng trong việc quản lý quyền truy cập và bảo mật.

*- Xác thực người dùng:* Dịch vụ này được sử dụng để xác thực danh tính của sinh viên và giảng viên, đảm bảo rằng chỉ những người dùng hợp lệ mới có thể truy cập vào các chức năng của hệ thống.

*- Trải nghiệm người dùng liền mạch:* Việc tích hợp tính năng đăng nhập bằng Google giúp đơn giản hóa quy trình đăng nhập cho sinh viên và giảng viên, mang lại trải nghiệm nhanh chóng và thân thiện với người dùng.

*- Phân quyền truy cập:* Thông tin người dùng từ Firebase sau khi được xác thực có thể được sử dụng để phân quyền truy cập. Ví dụ, hệ thống có thể phân biệt giữa sinh viên và giảng viên, từ đó chỉ cho phép giảng viên truy cập vào các báo cáo tổng thể trong khi sinh viên chỉ có thể xem dữ liệu của nhóm mình.

#### Socket.io

Socket.io là một thư viện JavaScript mạnh mẽ được thiết kế để hỗ trợ giao tiếp hai chiều (bidirectional) và thời gian thực (real-time) giữa trình duyệt (client) và máy chủ (server). Thư viện này hoạt động chủ yếu dựa trên giao thức WebSocket nhưng không giới hạn ở WebSocket, vì nó tích hợp thêm các cơ chế dự phòng (fallback mechanisms) như HTTP long-polling hoặc AJAX polling nhằm đảm bảo khả năng tương thích với nhiều môi trường khác nhau, kể cả khi trình duyệt hoặc mạng không hỗ trợ WebSocket thuần túy.

Khi một kết nối được khởi tạo, Socket.io trước tiên sẽ cố gắng sử dụng WebSocket để thiết lập kênh truyền dữ liệu full-duplex (truyền đồng thời hai chiều). Nếu WebSocket không khả dụng, hệ thống sẽ tự động chuyển sang các cơ chế dự phòng mà vẫn duy trì được giao tiếp theo mô hình sự kiện (event-based communication). Điểm đặc biệt của Socket.io là gói gọn toàn bộ quá trình kết nối, duy trì, và khôi phục kết nối trong một API thống nhất, giúp lập trình viên không cần tự xử lý các tình huống phức tạp về mạng với các tính năng nổi bật như:

*- Giao tiếp thời gian thực hai chiều (Real-time Bidirectional Communication):* Đây là tính năng cốt lõi của Socket.io. Kết nối giữa client và server được thiết lập và duy trì liên tục, cho phép cả hai bên có thể gửi và nhận dữ liệu bất cứ lúc nào. Thay vì phải đợi client gửi yêu cầu để server phản hồi, server có thể chủ động "đẩy" (push) dữ liệu đến client ngay khi có thông tin mới. Điều này là nền tảng cho các ứng dụng đòi hỏi sự cập nhật liên tục như chat, thông báo, game online hay bảng điều khiển thời gian thực.

*- Tính linh hoạt và độ bền của kết nối:* Socket.io được thiết kế để hoạt động ổn định trong mọi điều kiện mạng. Nếu kết nối bị ngắt quãng do sự cố mạng, Socket.io sẽ tự động thử kết nối lại một cách thông minh cho đến khi thành công. Điều này giúp duy trì trạng thái của ứng dụng mà không yêu cầu người dùng phải làm mới trang. Socket.io tự động sử dụng công nghệ tốt nhất có sẵn (ưu tiên WebSocket), và chuyển sang các công nghệ fallback như long polling nếu WebSocket không được hỗ trợ, đảm bảo tính tương thích cao với mọi trình duyệt và thiết bị.

- Quản lý luồng dữ liệu hiệu quả: Để quản lý các luồng thông tin trong ứng dụng lớn, Socket.io cung cấp các công cụ mạnh mẽ. Hỗ trợ Phòng (Room Support) là một cơ chế phân chia logic, cho phép các kết nối được gom thành các "phòng". Dữ liệu chỉ được gửi tới các client trong cùng một phòng. Tính năng này rất hữu ích cho các ứng dụng như: nhóm chat, game lobby,… Giúp tổ chức các kênh giao tiếp độc lập, mỗi kênh có một URL riêng (ví dụ: /admin hoặc /notifications). Điều này cho phép phân tách luồng dữ liệu một cách rõ ràng, giúp quản lý code dễ dàng hơn trong các ứng dụng phức tạp.

#### JWT (JSON Web Tokens)

JWT là một open standard (RFC 7519) để truyền thông tin một cách an toàn giữa các parties dưới dạng JSON object. JWT được sử dụng chủ yếu để xác thực và authorization, cho phép servers xác minh identity của users mà không cần lưu trữ session information trên server.

A diagram of a software application

AI-generated content may be incorrect.

Hình 2.21 Sơ đồ quy trình xác thực bằng JWT

Các thành phần của JWT bao gồm: header (chứa metadata về token), payload (chứa claims hoặc statements về user), và signature (để verify authenticity của token). JWT cũng hỗ trợ các tính năng như expiration time, issuer validation, và audience validation, làm cho việc xác thực trở nên an toàn và hiệu quả.

JWT được sử dụng như một cơ chế xác thực và phân quyền chính trong hệ thống theo dõi dự án mang lại một số lợi ích và đóng vai trò quan trọng trong dự án:

*- Xác thực không trạng thái (Stateless Authentication):* JWT cho phép máy chủ không cần phải lưu trữ thông tin phiên của người dùng. Mọi thông tin cần thiết (như ID người dùng, vai trò) đều được lưu trữ trong chính token. Điều này giúp hệ thống có khả năng mở rộng tốt hơn, vì bất kỳ máy chủ nào cũng có thể xử lý yêu cầu mà không cần truy cập vào một cơ sở dữ liệu phiên tập trung.

*- Bảo mật và tính toàn vẹn:* Chữ ký điện tử trong JWT đảm bảo rằng token không bị giả mạo. Mặc dù Payload không được mã hóa và có thể đọc được, nhưng không thể thay đổi mà không làm hỏng chữ ký, khiến máy chủ dễ dàng phát hiện.

*- Phân quyền hiệu quả:* JWT cho phép lưu trữ các thông tin phân quyền (ví dụ: user\_role) ngay trong Payload. Backend có thể giải mã token và nhanh chóng xác định quyền truy cập của người dùng mà không cần thực hiện thêm các truy vấn cơ sở dữ liệu.

*- Tính tương thích cao:* JWT là một tiêu chuẩn mở, được hỗ trợ bởi hầu hết các ngôn ngữ lập trình và framework, giúp việc tích hợp vào các hệ thống khác nhau trở nên dễ dàng.

# HIỆN THỰC HÓA NGHIÊN CỨU

## Mô tả bài toán thực tế

Trong bối cảnh giáo dục đại học hiện nay, việc quản lý và đánh giá các dự án phần mềm của sinh viên đang gặp phải nhiều thách thức và hạn chế đáng kể. Các phương pháp quản lý dự án truyền thống dựa trên giấy tờ, email, hoặc các công cụ đơn giản không còn đáp ứng được yêu cầu ngày càng phức tạp của việc phát triển phần mềm hiện đại.

Đặc biệt, trong các môn học liên quan đến phát triển phần mềm, đồ án tốt nghiệp, hoặc các dự án thực hành nhóm, việc theo dõi tiến độ, đánh giá chất lượng mã nguồn, và phân tích đóng góp của từng thành viên đang gặp nhiều khó khăn. Giảng viên và sinh viên phải đối mặt với thách thức trong việc theo dõi tiến độ thực tế của dự án khi thiếu công cụ tự động hóa. Việc cập nhật trạng thái dự án thường được thực hiện thủ công dẫn đến thông tin không chính xác và không kịp thời, đồng thời không có cơ chế cảnh báo khi dự án có nguy cơ chậm tiến độ hoặc gặp vấn đề.

Thêm vào đó, việc đánh giá chất lượng mã nguồn hiện nay chủ yếu dựa trên cảm tính và kinh nghiệm cá nhân của giảng viên, thiếu các tiêu chí đánh giá khách quan và định lượng. Không có công cụ tự động để phân tích và đánh giá mã nguồn theo các chuẩn chất lượng. Bên cạnh đó, quá trình phân tích mức độ đóng góp của từng thành viên cũng gặp nhiều khó khăn khi thiếu dữ liệu chi tiết về hoạt động coding, commit, và review. Việc đánh giá công bằng giữa các thành viên gặp hạn chế do thiếu bằng chứng cụ thể.

Các giải pháp quản lý dự án hiện có trên thị trường như Trello, Asana hoặc Jira mặc dù mạnh mẽ trong môi trường doanh nghiệp nhưng có nhiều hạn chế khi áp dụng vào môi trường giáo dục. Những công cụ này không được thiết kế đặc biệt cho việc quản lý dự án phần mềm sinh viên, thiếu các tính năng tích hợp với hệ thống quản lý học tập (LMS) của trường đại học, và có giao diện cùng quy trình làm việc phức tạp, không phù hợp với nhu cầu đơn giản của sinh viên. Ngoài ra, các công cụ này hạn chế trong việc tích hợp sâu với GitHub, không thể tự động thu thập và phân tích dữ liệu từ repository, cũng như thiếu các công cụ đánh giá chất lượng mã nguồn tự động.

Từ thực trạng trên, nảy sinh nhu cầu cấp thiết về một hệ thống quản lý dự án phần mềm toàn diện. Hệ thống này cần tập trung lưu trữ và quản lý thông tin chi tiết về dự án, nhóm, thành viên, timeline, và milestones. Đồng thời, phải cung cấp giao diện thân thiện cho cả giảng viên và sinh viên, tích hợp GitHub để theo dõi tiến độ dự án tự động, hiển thị dữ liệu qua dashboard trực quan, và hỗ trợ cảnh báo khi có vấn đề.

Ngoài ra, hệ thống cần có khả năng phân tích hoạt động coding của từng thành viên dựa trên dữ liệu từ GitHub repository, đánh giá mức độ đóng góp thông qua các metrics khách quan như số lượng commit, lines of code, hay thời gian coding. Song song đó, cần có công cụ đánh giá chất lượng mã nguồn tự động, phân tích các chỉ số như độ phức tạp (complexity), khả năng bảo trì (maintainability), và độ bao phủ kiểm thử (test coverage), giúp giảng viên đưa ra đánh giá công bằng, minh bạch và chính xác hơn.

Mục tiêu của dự án CodeFlow là phát triển một nền tảng web hiện đại đáp ứng các yêu cầu trên. Nền tảng này sẽ được xây dựng với kiến trúc full-stack hiện đại, tích hợp GitHub API, phát triển các thuật toán phân tích mã nguồn, và cung cấp các công cụ hỗ trợ đánh giá chất lượng code dựa trên chuẩn quốc tế. Bên cạnh đó, dự án hướng tới việc tối ưu trải nghiệm người dùng, hỗ trợ nhiều dự án và nhóm sinh viên đồng thời, và cung cấp báo cáo chi tiết phục vụ cho cả giảng viên lẫn sinh viên.

Ý nghĩa của dự án không chỉ nằm ở khía cạnh kỹ thuật mà còn đóng góp lớn cho giáo dục. Hệ thống sẽ cải thiện chất lượng đào tạo, tăng tính minh bạch và công bằng trong đánh giá, thúc đẩy áp dụng best practices trong phát triển phần mềm, và giảm tải khối lượng công việc thủ công cho giảng viên. Đặc biệt, dự án còn mang giá trị cộng đồng khi được phát triển theo hướng mã nguồn mở, cho phép cộng đồng mở rộng và áp dụng trong nhiều môi trường giáo dục khác nhau.

## Phân tích yêu cầu người dùng

Hệ thống CodeFlow phục vụ ba nhóm người dùng chính trong môi trường giáo dục đại học:

- Giảng viên: Quản lý, hướng dẫn và đánh giá dự án phần mềm của sinh viên, đảm bảo chất lượng đào tạo. Giảng viên cần công cụ toàn diện để theo dõi và đánh giá hiệu quả hoạt động của sinh viên trong quá trình phát triển dự án.

- Sinh viên: Tham gia trực tiếp vào dự án phần mềm, thực hiện coding, hợp tác và hoàn thành dự án. Sinh viên cần môi trường học tập chuyên nghiệp để phát triển kỹ năng lập trình và làm việc nhóm.

- Quản lý: Quản lý tổng thể chương trình đào tạo, giám sát hiệu quả hoạt động giảng dạy. Quản lý cần cái nhìn tổng quan để đưa ra quyết định chiến lược về chương trình đào tạo.

### Yêu cầu của giảng viên

**Quản lý dự án và nhóm:**

*- Tạo lập và quản lý nhiều dự án đồng thời.* Giúp giảng viên quản lý hiệu quả nhiều lớp học và dự án khác nhau.

*- Thiết lập thông tin chi tiết dự án* (tên, mô tả, mục tiêu, thời gian, yêu cầu kỹ thuật). Đảm bảo sinh viên hiểu rõ yêu cầu và mục tiêu của dự án.

*- Quản lý thông tin nhóm sinh viên* (danh sách thành viên, vai trò, thông tin liên hệ). Theo dõi cấu trúc nhóm và phân công trách nhiệm rõ ràng.

*- Thiết lập các giai đoạn phát triển dự án* với mốc thời gian quan trọng. Tạo lộ trình học tập có cấu trúc và đánh giá theo từng giai đoạn.

**Theo dõi tiến độ và trạng thái dự án:**

*- Dashboard tổng quan theo dõi tiến độ tất cả dự án:* Cung cấp cái nhìn tổng thể về tình trạng các dự án đang quản lý.

*- Theo dõi chi tiết từng dự án (số lượng commit, hoạt động coding).* Phát hiện sớm các vấn đề và can thiệp kịp thời.

*- Hệ thống cảnh báo tự động khi dự án chậm tiến độ hoặc gặp vấn đề*. Đảm bảo không bỏ lỡ các dấu hiệu cần hỗ trợ.

**Đánh giá chất lượng mã nguồn:**

*- Công cụ tự động đánh giá chất lượng mã nguồn.* Giảm tải công việc đánh giá thủ công và đảm bảo tính khách quan.

*- Phân tích độ phức tạp, chỉ số bảo trì, độ bao phủ kiểm thử.* Đánh giá toàn diện chất lượng code theo tiêu chuẩn công nghiệp.

*- Báo cáo vấn đề chất lượng mã phổ biến và gợi ý cải thiện:* Hướng dẫn sinh viên cải thiện kỹ năng lập trình.

**Đánh giá đóng góp thành viên:**

*- Đo lường mức độ đóng góp từng thành viên (số commit, dòng mã, thời gian):* Đảm bảo đánh giá công bằng và minh bạch.

*- Theo dõi hoạt động hợp tác (đánh giá mã, pull request, báo cáo vấn đề):* Đánh giá kỹ năng làm việc nhóm và giao tiếp.

*- Báo cáo chi tiết hiệu suất từng thành viên:* Hỗ trợ việc đưa ra quyết định đánh giá cuối khóa.

**Báo cáo và phân tích dữ liệu:**

*- Hệ thống báo cáo toàn diện về tình trạng dự án, tiến độ, chất lượng:* Cung cấp dữ liệu đầy đủ cho việc đánh giá và báo cáo.

*- Xuất dữ liệu ra PDF, Excel, CSV:* Hỗ trợ việc chia sẻ thông tin với các bên liên quan.

*- Phân tích xu hướng:* để cải thiện phương pháp giảng dạy. Rút kinh nghiệm và nâng cao chất lượng đào tạo.

**Giao diện và trải nghiệm người dùng:**

*- Giao diện rõ ràng, dashboard trực quan:* Tăng hiệu quả sử dụng hệ thống và giảm thời gian học tập.

*- Responsive design cho desktop, tablet, mobile*: Đảm bảo truy cập thuận tiện từ mọi thiết bị.

*- Cá nhân hóa giao diện theo nhu cầu:* Tối ưu hóa trải nghiệm người dùng cá nhân.

### Yêu cầu của sinh viên

**Quản lý thông tin cá nhân và dự án:**

*- Xem và cập nhật thông tin cá nhân, hồ sơ, tùy chọn*: Đảm bảo thông tin cá nhân chính xác và cập nhật.

*- Truy cập thông tin chi tiết dự án đang tham gia:* Hiểu rõ yêu cầu và tiến độ dự án.

*- Xem lịch sử tham gia dự án trước đó và hiệu suất:* Theo dõi sự phát triển kỹ năng theo thời gian.

**Theo dõi tiến độ cá nhân và nhóm:**

*- Dashboard cá nhân theo dõi tiến độ công việc.* Tự đánh giá hiệu suất và điều chỉnh kế hoạch làm việc.

*- Xem tiến độ tổng thể nhóm và đóng góp thành viên khác:* Hiểu rõ vai trò và trách nhiệm trong nhóm.

*-Thông báo và nhắc nhở về thời hạn, nhiệm vụ:* Đảm bảo hoàn thành công việc đúng thời hạn.

**Tích hợp với quy trình làm việc GitHub:**

*- Tích hợp liền mạch với GitHub*. Tạo môi trường phát triển chuyên nghiệp và thực tế.

*- Đồng bộ tự động commit, pull request, issue, review.* Giảm thời gian cập nhật thông tin thủ công.

*- Quản lý hoạt động GitHub trực tiếp từ hệ thống.* Tập trung vào việc phát triển thay vì chuyển đổi công cụ.

**Phản hồi và học tập:**

*- Nhận phản hồi về chất lượng mã nguồn và gợi ý cải thiện.* Học hỏi từ lỗi và cải thiện kỹ năng lập trình.

*- Truy cập tài liệu học tập, thực hành tốt, hướng dẫn lập trình.* Nâng cao kiến thức và áp dụng best practices.

*- Theo dõi tiến độ và cải thiện theo thời gian.* Thấy được sự tiến bộ và động lực học tập.

**Hợp tác và giao tiếp:**

*- Công cụ hợp tác với thành viên nhóm (bình luận, thảo luận):* Tăng cường giao tiếp và chia sẻ kiến thức trong nhóm.

*- Tham gia cuộc thảo luận nhóm, chia sẻ ý tưởng:* Phát triển kỹ năng làm việc nhóm và giao tiếp.

*- Truy cập lịch nhóm, tài liệu chia sẻ, công cụ giao tiếp*: Đảm bảo thông tin được chia sẻ hiệu quả.

**Giao diện và khả năng tiếp cận:**

*- Giao diện hiện đại, hấp dẫn với phản hồi trực quan:* Tăng động lực sử dụng và học tập.

*- Thân thiện với thiết bị di động. Học tập mọi lúc, mọi nơi.*

*- Tính năng khả năng tiếp cận cho người dùng khuyết tật:* Đảm bảo tính bao trùm và công bằng.

### Yêu cầu của quản lý

**Quản lý tổng thể chương trình đào tạo:**

*- Dashboard tổng quan tình trạng tất cả dự án trong khoa*: Có cái nhìn toàn diện về hiệu quả chương trình đào tạo.

*- Quản lý thông tin môn học, giảng viên, sinh viên*: Đảm bảo thông tin chính xác và cập nhật.

*- Truy cập chính sách trường, hướng dẫn, tiêu chuẩn*: Đảm bảo tuân thủ quy định và tiêu chuẩn.

**Báo cáo và phân tích dữ liệu tổng thể:**

*- Hệ thống báo cáo toàn diện về hiệu suất học tập, kết quả dự án:* Hỗ trợ việc đưa ra quyết định dựa trên dữ liệu.

*- Công cụ phân tích xu hướng, mẫu hình*: Phát hiện vấn đề và cơ hội cải thiện.

*- Xuất dữ liệu chia sẻ với phòng ban khác, cơ quan kiểm định:* Đáp ứng yêu cầu báo cáo và kiểm định.

**Quản lý người dùng và quyền hạn:**

*- Hệ thống quản lý người dùng (tạo, cập nhật, vô hiệu hóa tài khoản)*:Đảm bảo an toàn và kiểm soát truy cập.

*- Gán và quản lý quyền hạn cho các vai trò khác nhau:* Phân quyền phù hợp với trách nhiệm.

*- Lịch sử kiểm tra và ghi nhật ký cho mục đích tuân thủ:* Đảm bảo tính minh bạch và trách nhiệm giải trình.

**Quản trị hệ thống và bảo trì:**

*- Giám sát hệ thống và kiểm tra tình trạng*: Đảm bảo hệ thống hoạt động ổn định.

*- Quy trình sao lưu và khôi phục dữ liệu:* Bảo vệ dữ liệu quan trọng.

*- Cấu hình hệ thống theo nhu cầu trường học.* Tùy chỉnh hệ thống phù hợp với môi trường cụ thể.

## Lựa chọn công nghệ

Công nghệ được lựa chọn dựa trên yêu cầu chức năng, phi chức năng, tính phù hợp với môi trường giáo dục, khả năng tích hợp, hiệu suất, độ ổn định, cộng đồng hỗ trợ, khả năng mở rộng, dễ học, và chi phí.

### Phía người dùng

- Ngôn ngữ TypeScript: Hệ thống kiểu mạnh, phát hiện lỗi sớm, tương thích JavaScript, hỗ trợ các khái niệm OOP và type checking, phù hợp môi trường giáo dục.

- React: Kiến trúc component, dễ tái sử dụng, quản lý trạng thái tốt, cộng đồng lớn, tài liệu phong phú.

- Next.js: Hỗ trợ SSR, SSG, routing đơn giản, API routes, tối ưu SEO và hiệu suất, giúp sinh viên tiếp cận full-stack.

- Tailwind CSS: Utility-first, phát triển nhanh, giao diện nhất quán, responsive, tối ưu bundle với JIT.

- Shadcn/ui: Component chất lượng cao, dễ tùy biến, hỗ trợ accessibility, responsive, dark mode.

### Phía máy chủ

- Node.js: Dùng chung JavaScript/TypeScript cho cả frontend và backend, non-blocking I/O, cộng đồng lớn.

- Express.js: Đơn giản, linh hoạt, middleware mạnh, dễ xây dựng RESTful API.

- CSDL quan hệ: Đảm bảo tính toàn vẹn, phù hợp dữ liệu phức tạp, hỗ trợ SQL, ACID.

- Sequelize: ORM hỗ trợ nhiều hệ quản trị, TypeScript, quản lý models, associations, migrations.

### Tích hợp & dịch vụ ngoài

- GitHub API: Lấy dữ liệu repository, commits, issues; hỗ trợ webhook real-time.

- Firebase Authentication: Nhiều phương thức đăng nhập, bảo mật cao, dễ tích hợp.

- Docker: Môi trường nhất quán, dễ triển khai, hỗ trợ học về containerization và microservices.

## Thiết kế kiến trúc hệ thống tổng thể

Hệ thống CodeFlow được thiết kế theo kiến trúc microservices với mô hình client-server, sử dụng các công nghệ hiện đại để đảm bảo tính mở rộng, hiệu suất cao và khả năng bảo trì tốt. Kiến trúc tổng thể được chia thành các lớp chức năng rõ ràng, mỗi lớp có trách nhiệm và nhiệm vụ cụ thể, tạo ra một hệ thống có cấu trúc logic và dễ dàng phát triển, triển khai và bảo trì. Hệ thống được thiết kế để hỗ trợ nhiều người dùng đồng thời, xử lý dữ liệu lớn từ GitHub API, và cung cấp trải nghiệm người dùng mượt mà và đáp ứng nhanh.

**Luồng xác thực người dùng**

*Bước 1: Người dùng truy cập hệ thống:* Người dùng mở trình duyệt web và truy cập vào URL của hệ thống CodeFlow. Hệ thống kiểm tra trạng thái đăng nhập hiện tại của người dùng.

*Bước 2: Kiểm tra session:* Nếu người dùng đã đăng nhập trước đó và session vẫn còn hiệu lực, hệ thống sẽ chuyển hướng người dùng đến trang dashboard chính. Nếu không, hệ thống sẽ hiển thị trang đăng nhập.

*Bước 3: Quá trình đăng nhập:* Người dùng nhập thông tin đăng nhập (email/password hoặc sử dụng github login). Hệ thống gửi thông tin đăng nhập đến Firebase Authentication service để xác thực.

*Bước 4: Xác thực và tạo session*: Firebase Authentication xác thực thông tin đăng nhập và trả về JWT token. Hệ thống lưu trữ token và tạo session cho người dùng.

*Bước 5: Chuyển hướng đến dashboard:* Sau khi đăng nhập thành công, người dùng được chuyển hướng đến trang dashboard chính với quyền truy cập phù hợp với role của họ.

**Luồng quản lý dự án**

*Bước 1: Tạo dự án mới:* Giảng viên hoặc quản lý tạo dự án mới thông qua giao diện quản lý dự án. Hệ thống yêu cầu nhập thông tin cơ bản về dự án như tên, mô tả, thời gian bắt đầu, thời gian kết thúc, và các yêu cầu kỹ thuật.

*Bước 2: Thiết lập nhóm:* Sau khi tạo dự án, hệ thống cho phép thiết lập nhóm sinh viên tham gia dự án. Giảng viên có thể thêm/sửa/xóa thành viên nhóm và phân công vai trò cho từng thành viên.

*Bước 3: Kết nối GitHub repository:* Hệ thống yêu cầu kết nối với GitHub repository của dự án. Người dùng cung cấp thông tin repository và cấp quyền truy cập cho hệ thống.

*Bước 4: Thu thập dữ liệu ban đầu:* Sau khi kết nối thành công, hệ thống bắt đầu thu thập dữ liệu từ GitHub repository, bao gồm thông tin về commits, pull requests, issues, và các hoạt động khác.

*Bước 5: Theo dõi tiến độ:* Hệ thống liên tục theo dõi và cập nhật tiến độ dự án dựa trên hoạt động thực tế trên GitHub. Dữ liệu được hiển thị trên dashboard với các metrics và biểu đồ trực quan.

**Luồng phân tích chất lượng mã nguồn**

*Bước 1: Thu thập mã nguồn:* Hệ thống thu thập mã nguồn từ GitHub repository thông qua GitHub API. Quá trình này có thể được thực hiện theo lịch trình định sẵn hoặc khi có commit mới.

*Bước 2: Phân tích mã nguồn:* Hệ thống sử dụng các công cụ phân tích mã nguồn để đánh giá chất lượng code, bao gồm phân tích độ phức tạp, maintainability index, test coverage, và adherence to coding standards.

*Bước 3: Tính toán metrics:* Dựa trên kết quả phân tích, hệ thống tính toán các metrics chất lượng như code complexity score, maintainability score, và overall quality score.

*Bước 4: Lưu trữ kết quả:* Kết quả phân tích được lưu trữ vào cơ sở dữ liệu với timestamp và reference đến commit tương ứng.

*Bước 5: Hiển thị báo cáo:* Kết quả phân tích được hiển thị trên dashboard với các biểu đồ và báo cáo chi tiết, cho phép giảng viên và sinh viên theo dõi chất lượng mã nguồn theo thời gian.

**Luồng phân tích đóng góp thành viên**

*Bước 1: Thu thập hoạt động:* Hệ thống thu thập thông tin về hoạt động của từng thành viên từ GitHub, bao gồm số lượng commit, lines of code contributed, time spent coding, và participation in code reviews.

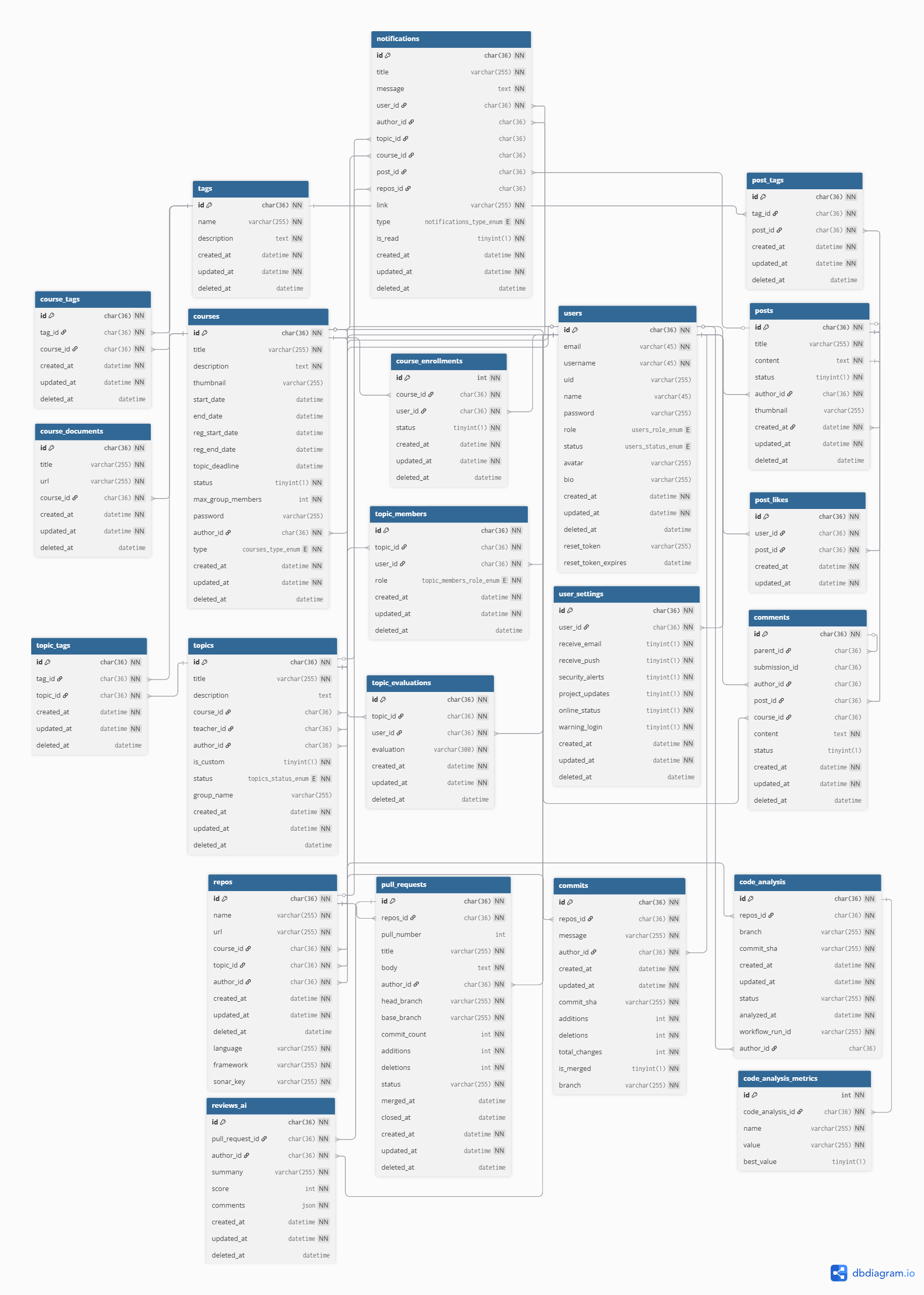
*Bước 2: Tính toán metrics:* Dựa trên dữ liệu thu thập được, hệ thống tính toán các metrics đóng góp cho từng thành viên, bao gồm contribution score, activity level, và collaboration metrics.

*Bước 3: Phân tích xu hướng:* Hệ thống phân tích xu hướng đóng góp của từng thành viên theo thời gian, xác định patterns và changes trong activity level.

*Bước 4: So sánh và đánh giá:* Hệ thống so sánh đóng góp giữa các thành viên trong nhóm và đưa ra đánh giá tương đối về mức độ đóng góp của từng người.

*Bước 5: Tạo báo cáo*: Kết quả phân tích được tổng hợp thành báo cáo chi tiết với các biểu đồ và insights, giúp giảng viên đánh giá công bằng và minh bạch.

## Thiết kế cơ sở dữ liệu



Hình 3.1. Lược đồ cơ sở dữ liệu

- Mô tả bảng USER(Người dùng)

Bảng 3.1. Mô tả bảng User

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã người dùng | varchar(36) | Primary key |
| 2 | email | Địa chỉ email | text | Unique |
| 3 | username | Tên đăng nhập | varchar(255) | Unique |
| 4 | uid | ID được lưu tại Firebase | varchar(255) |  |
| 5 | name | Tên đầy đủ | varchar(255) |  |
| 6 | password | Mật khẩu | varchar(255) |  |
| 7 | role | Vai trò người dùng | enum |  |
| 8 | status | Trạng thái tài khoản | enum |  |
| 9 | avatar\_url | Đường dẫn ảnh đại diện | text |  |
| 10 | bio | Thông tin giới thiệu | text |  |
| 11 | resetToken | Token đặt lại mật khẩu | varchar(255) |  |
| 12 | resetTokenExpires | Thời gian hết hạn token | datetime |  |
| 13 | created\_at | Thời gian tạo | datetime |  |
| 14 | updated\_at | Thời gian cập nhật | datetime |  |
| 15 | deleted\_at | Thời gian xóa | datetime |  |

- Mô tả bảng COURSES (Khóa học):

Bảng 3.2. Mô tả bảng Courses

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã khóa học | varchar(36) | Primary key |
| 2 | title | Tên khóa học | varchar(255) |  |
| 3 | description | Mô tả khóa học | text |  |
| 4 | thumbnail | Ảnh đại diện khóa học | varchar(255) |  |
| 5 | startDate | Ngày bắt đầu | datetime |  |
| 6 | endDate | Ngày kết thúc | datetime |  |
| 7 | regStartDate | Ngày bắt đầu đăng ký | datetime |  |
| 8 | regEndDate | Ngày kết thúc đăng ký | datetime |  |
| 9 | topicDeadline | Hạn chót nộp đề tài | datetime | Nullable |
| 10 | authorId | Mã giảng viên tạo | varchar(36) | Foreign key |
| 11 | status | Trạng thái khóa học | boolean |  |
| 12 | maxGroupMembers | Số thành viên | int |  |
| 13 | type | Loại khóa học | enum |  |
| 14 | password | Mật khẩu tham gia | varchar(255) |  |
| 15 | createdAt | Thời gian tạo | datetime |  |
| 16 | updatedAt | Thời gian cập nhật | datetime |  |
| 17 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng TOPICS (Đề tài):

Bảng 3.3. Mô tả bảng Topics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã đề tài | varchar(36) | Primary key |
| 2 | title | Tên đề tài | varchar(255) |  |
| 3 | description | Mô tả đề tài | text |  |
| 4 | courseId | Mã khóa học | varchar(36) | Foreign key |
| 5 | teacherId | Mã giảng viên | varchar(36) | Foreign key |
| 6 | authorId | Mã người tạo đề tài | varchar(36) | Foreign key |
| 7 | isCustom | Đề tài tùy chỉnh | boolean |  |
| 8 | status | Trạng thái đề tài | enum |  |
| 9 | groupName | Tên nhóm thực hiện | varchar(255) |  |
| 10 | createdAt | Thời gian tạo | datetime |  |
| 11 | updatedAt | Thời gian cập nhật | datetime |  |
| 12 | deletedAt | Thời gian xóa mềm | datetime |  |

 - Mô tả bảng REPOS (Kho lưu trữ mã nguồn)

Bảng 3.4 Mô tả bảng Repos

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã kho lưu trữ | varchar(36) | Primary key |
| 2 | name | Tên kho lưu trữ | varchar(255) |  |
| 3 | url | Đường dẫn GitHub | varchar(255) |  |
| 4 | courseId | Mã khóa học | varchar(36) | Foreign key |
| 5 | topicId | Mã đề tài | varchar(36) | Foreign key |
| 6 | authorId | Mã người tạo | varchar(36) | Foreign key |
| 7 | language | Ngôn ngữ lập trình | varchar(255) |  |
| 8 | framework | Framework sử dụng | varchar(255) |  |
| 9 | sonarKey | Khóa SonarQube | varchar(255) |  |
| 10 | createdAt | Thời gian tạo | datetime |  |
| 11 | updatedAt | Thời gian cập nhật | datetime |  |
| 12 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng COMMITS (Các lần cập nhật mã nguồn)

Bảng 3.5. Mô tả bảng Commits

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã commit | varchar(36) | Primary key |
| 2 | reposId | Mã kho lưu trữ | varchar(36) | Foreign key |
| 3 | commitSha | Mã hash commit | varchar(255) |  |
| 4 | message | Nội dung commit | varchar(255) |  |
| 5 | authorId | Mã tác giả | varchar(36) | Foreign key |
| 6 | additions | Số dòng thêm | int |  |
| 7 | deletions | Số dòng xóa | int |  |
| 8 | totalChanges | Tổng số thay đổi | int |  |
| 9 | isMerged | Đã hợp nhất chưa | boolean |  |
| 10 | branch | Nhánh phát triển | varchar(255) |  |
| 11 | createdAt | Thời gian tạo | datetime |  |
| 12 | updatedAt | Thời gian cập nhật | datetime |  |

- Mô tả bảng PULL\_REQUESTS (Yêu cầu hợp nhất)

Bảng 3.6. Mô tả bảng Pull request

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã pull request | varchar(36) | Primary key |
| 2 | reposId | Mã kho lưu trữ | varchar(36) | Foreign key |
| 3 | pullNumber | Số thứ tự PR | int |  |
| 4 | title | Tiêu đề PR | varchar(255) |  |
| 5 | body | Nội dung PR | text |  |
| 6 | authorId | Mã tác giả | varchar(36) | Foreign key |
| 7 | headBranch | Nhánh nguồn | varchar(255) |  |
| 8 | baseBranch | Nhánh đích | varchar(255) |  |
| 9 | commitCount | Số commit | int |  |
| 10 | additions | Số dòng thêm | int |  |
| 11 | deletions | Số dòng xóa | int |  |
| 12 | status | Trạng thái PR | varchar(255) |  |
| 13 | mergedAt | Thời gian hợp nhất | datetime |  |
| 14 | closedAt | Thời gian đóng | datetime |  |
| 15 | createdAt | Thời gian tạo | datetime |  |
| 16 | updatedAt | Thời gian cập nhật | datetime |  |
| 17 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng CODE\_ANALYSIS (Phân tích mã nguồn)

Bảng 3.7 Mô tả bảng Code analysis

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã phân tích | varchar(36) | Primary key |
| 2 | reposId | Mã kho lưu trữ | varchar(36) | Foreign key |
| 3 | branch | Nhánh phân tích | varchar(255) |  |
| 4 | commitSha | Mã hash commit | varchar(255) |  |
| 5 | status | Trạng thái phân tích | varchar(255) |  |
| 6 | analyzedAt | Thời gian phân tích | datetime |  |
| 7 | workflowRunId | ID workflow | varchar(255) |  |
| 8 | authorId | Mã tác giả | varchar(36) | Foreign key |
| 9 | createdAt | Thời gian tạo | datetime |  |
| 10 | updatedAt | Thời gian cập nhật | datetime |  |

* Mô tả bảng CODE\_ANALYSIS\_METRICS (Chỉ số phân tích mã nguồn)

Bảng 3.8. Mô tả bảng Code analysis metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã chỉ số | varchar(36) | Primary key |
| 2 | codeAnalysisId | Mã phân tích | varchar(36) | Foreign key |
| 3 | metricName | Tên chỉ số | varchar(255) |  |
| 4 | metricValue | Giá trị chỉ số | float |  |
| 5 | metricUnit | Đơn vị đo | varchar(50) |  |
| 6 | createdAt | Thời gian tạo | datetime |  |
| 7 | updatedAt | Thời gian cập nhật | datetime |  |

* Mô tả bảng REVIEWS\_AI (Đánh giá AI)

Bảng 3.9. Mô tả bảng Review ai

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã đánh giá | varchar(36) | Primary key |
| 2 | pullRequestId | Mã pull request | varchar(36) | Foreign key |
| 3 | reviewContent | Nội dung đánh giá | text |  |
| 4 | confidenceScore | Độ tin cậy | float |  |
| 5 | reviewType | Loại đánh giá | varchar(255) |  |
| 6 | createdAt | Thời gian tạo | datetime |  |
| 7 | updatedAt | Thời gian cập nhật | datetime |  |

* Mô tả bảng NOTIFICATIONS (Thông báo)

Bảng 3.10. Mô tả bảng Notofications

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã thông báo | varchar(36) | Primary key |
| 2 | title | Tiêu đề thông báo | varchar(255) |  |
| 3 | message | Nội dung thông báo | text |  |
| 4 | userId | Mã người nhận | varchar(36) | Foreign key |
| 5 | authorId | Mã người gửi | varchar(36) | Foreign key |
| 6 | topicId | Mã đề tài liên quan | varchar(36) | Foreign key |
| 7 | courseId | Mã khóa học liên quan | varchar(36) | Foreign key |
| 8 | postId | Mã bài viết liên quan | varchar(36) | Foreign key |
| 9 | reposId | Mã kho lưu trữ | varchar(36) | Foreign key |
| 10 | link | Đường dẫn liên kết | varchar(255) |  |
| 11 | type | Loại thông báo | enum |  |
| 12 | isRead | Đã đọc chưa | boolean |  |
| 13 | createdAt | Thời gian tạo | datetime |  |
| 14 | updatedAt | Thời gian cập nhật | datetime |  |
| 15 | deletedAt | Thời gian xóa mềm | datetime |  |

* Mô tả bảng TOPIC\_MEMBER (Thành viên đề tài)

Bảng 3.11. Mô tả bảng Topic member

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã thành viên | varchar(36) | Primary key |
| 2 | topicId | Mã đề tài | varchar(36) | Foreign key |
| 3 | userId | Mã người dùng | varchar(36) | Foreign key |
| 4 | role | Vai trò trong nhóm | varchar(255) |  |
| 5 | joinedAt | Thời gian tham gia | datetime |  |
| 6 | leftAt | Thời gian rời nhóm | datetime |  |
| 7 | createdAt | Thời gian tạo | datetime |  |
| 8 | updatedAt | Thời gian cập nhật | datetime |  |
| 9 | deletedAt | Thời gian xóa mềm | datetime |  |

* Mô tả bảng TOPIC\_EVALUATIONS (Đánh giá đề tài)

Bảng 3.12. Mô tả bảng Topic evaluations

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã đánh giá | varchar(36) | Primary key |
| 2 | topicId | Mã đề tài | varchar(36) | Foreign key |
| 3 | evaluatorId | Mã người đánh giá | varchar(36) | Foreign key |
| 4 | score | Điểm đánh giá | float |  |
| 5 | feedback | Nhận xét | text |  |
| 6 | evaluationDate | Ngày đánh giá | datetime |  |
| 7 | createdAt | Thời gian tạo | datetime |  |
| 8 | updatedAt | Thời gian cập nhật | datetime |  |
| 9 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng TAGS (Nhãn)

Bảng 3.13. Mô tả bảng Tags

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã nhãn | varchar(36) | Primary key |
| 2 | name | Tên nhãn | varchar(255) |  |
| 3 | color | Màu sắc nhãn | varchar(7) |  |
| 4 | createdAt | Thời gian tạo | datetime |  |
| 5 | updatedAt | Thời gian cập nhật | datetime |  |
| 6 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng POSTS (Bài viết)

Bảng 3.14. Mô tả bảng Posts

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã bài viết | varchar(36) | Primary key |
| 2 | title | Tiêu đề bài viết | varchar(255) |  |
| 3 | content | Nội dung bài viết | text |  |
| 4 | authorId | Mã tác giả | varchar(36) | Foreign key |
| 5 | topicId | Mã đề tài | varchar(36) | Foreign key |
| 6 | isPinned | Được ghim không | boolean |  |
| 7 | createdAt | Thời gian tạo | datetime |  |
| 8 | updatedAt | Thời gian cập nhật | datetime |  |
| 9 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng COMMENTS (Bình luận)

Bảng 3.15. Mô tả bảng Comments

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã bình luận | varchar(36) | Primary key |
| 2 | content | Nội dung bình luận | text |  |
| 3 | authorId | Mã tác giả | varchar(36) | Foreign key |
| 4 | postId | Mã bài viết | varchar(36) | Foreign key |
| 5 | parentId | Mã bình luận cha | varchar(36) | Foreign key |
| 6 | createdAt | Thời gian tạo | datetime |  |
| 7 | updatedAt | Thời gian cập nhật | datetime |  |
| 8 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng USER\_SETTINGS (Cài đặt người dùng)

Bảng 3.16 Mô tả bảng User settings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã cài đặt | varchar(36) | Primary key |
| 2 | userId | Mã người dùng | varchar(36) | Foreign key |
| 3 | theme | Giao diện | varchar(50) |  |
| 4 | language | Ngôn ngữ | varchar(10) |  |
| 5 | emailNotifications | Thông báo email | boolean |  |
| 6 | pushNotifications | Thông báo đẩy | boolean |  |
| 7 | createdAt | Thời gian tạo | datetime |  |
| 8 | updatedAt | Thời gian cập nhật | datetime |  |
| 9 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng SYSTEM\_SETTINGS (Cài đặt hệ thống)

Bảng 3.17 Mô tả bảng System settings

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã cài đặt | varchar(36) | Primary key |
| 2 | key | Khóa cài đặt | varchar(255) |  |
| 3 | value | Giá trị cài đặt | text |  |
| 4 | description | Mô tả cài đặt | varchar(255) |  |
| 5 | createdAt | Thời gian tạo | datetime |  |
| 6 | updatedAt | Thời gian cập nhật | datetime |  |
| 7 | deletedAt | Thời gian xóa mềm | datetime |  |

- Mô tả bảng COURSE\_ENROLLMENT (Quan hệ khóa học - đăng ký)

Bảng 3.18. Mô tả bảng Course enrollment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | courseId | Mã khóa học | varchar(36) | Foreign key |
| 2 | userId | Mã người dùng | varchar(36) | Foreign key |
| 3 | enrolledAt | Thời gian đăng ký | datetime |  |

- Mô tả bảng COURSE\_DOCUMENTS (Tài liệu khóa học)

Bảng 3.19. Mô tả bảng Course documents

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Thuộc tính** | **Diễn giải** | **Kiểu dữ liệu** | **Ràng buộc** |
| 1 | id | Mã tài liệu | varchar(36) | Primary key |
| 2 | courseId | Mã khóa học | varchar(36) | Foreign key |
| 3 | title | Tiêu đề tài liệu | varchar(255) |  |
| 4 | url | Đường dẫn tài liệu | varchar(255) |  |
| 5 | createdAt | Thời gian tạo | datetime |  |
| 6 | updatedAt | Thời gian cập nhật | datetime |  |
| 7 | deletedAt | Thời gian xóa mềm | datetime |  |

## Xây dựng các API chức năng chính

### API người dùng và xác thực OAuth

Bảng 3.20. Bảng mô tả API đăng ký

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/auth/signup | |
| **Thuộc tính** | email | Email của người dùng |
| username | Tên đăng nhập |
| password | Mật khẩu |
| name | Tên đầy đủ |
| role | Vai trò người dùng |
| **Diễn giải** | API cho phép đăng ký tài khoản mới vào hệ thống với thông tin cơ bản | |

Bảng 3.21. Bảng mô tả API đăng nhập

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/auth/login | |
| **Thuộc tính** | email | Email của người dùng |
| password | Mật khẩu |
| **Diễn giải** | API cho phép đăng nhập hệ thống bằng email và mật khẩu | |

Bảng 3.22. Bảng mô tả API đăng nhập GitHub

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/auth/loginWithGithub | |
| **Thuộc tính** | access\_token | Access Token hợp lệ của tài khoản GitHub |
| uid | Mã người dùng của tài khoản GitHub |
| email | Email của người dùng |
| name | Tên đầy đủ |
| username | Tên đăng nhập |
| **Diễn giải** | API cho phép đăng nhập hệ thống với dữ liệu OAuth từ GitHub. Nếu người dùng chưa tồn tại, hệ thống sẽ tạo mới thông tin người dùng | |

Bảng 3.23. Bảng mô tả API lấy thông tin người dùng

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/auth/info |
| **Diễn giải** | API cho phép lấy ra thông tin chi tiết của người dùng đang đăng nhập |

Bảng 3.24 Bảng mô tả API đăng xuất

|  |  |
| --- | --- |
| **Phương thức** | **POST** |
| **Endpoint** | {BASE\_URL}/auth/logout |
| **Diễn giải** | API cho phép đăng xuất khỏi hệ thống và vô hiệu hóa token hiện tại |

### API môn học

Bảng 3.25. Bảng mô tả API tạo môn học mới

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/courses | |
| **Thuộc tính** | title | Tên môn học |
| description | Mô tả môn học |
| startDate | Ngày bắt đầu |
| endDate | Ngày kết thúc |
| regStartDate | Ngày bắt đầu đăng ký |
| regEndDate | Ngày kết thúc đăng ký |
| topicDeadline | Hạn chót nộp đề tài |
| maxGroupMembers | Số thành viên tối đa nhóm |
| type | Loại môn học |
| password | Mật khẩu tham gia |
| **Diễn giải** | API cho phép giảng viên tạo môn học mới | |

Bảng 3.26. Bảng mô tả API lấy thông tin môn học

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/courses/:id |
| **Diễn giải** | API lấy thông tin chi tiết của môn học theo ID |

Bảng 3.27. Bảng mô tả API lấy danh sách môn học

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **GET** | |
| **Endpoint** | {BASE\_URL}/courses | |
| **Thuộc tính** | page | Số trang (query parameter) |
| limit | Số lượng item mỗi trang (query parameter) |
| search | Từ khóa tìm kiếm (query parameter) |
| **Diễn giải** | API lấy danh sách tất cả môn học trong hệ thống | |

Bảng 3.28. Bảng mô tả API cập nhật môn học

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **PUT** | |
| **Endpoint** | {BASE\_URL}/courses/:id | |
| **Thuộc tính** | title | Tên môn học |
| description | Mô tả môn học |
| startDate | Ngày bắt đầu |
| endDate | Ngày kết thúc |
| regStartDate | Ngày bắt đầu đăng ký |
| regEndDate | Ngày kết thúc đăng ký |
| topicDeadline | Hạn chót nộp đề tài |
| maxGroupMembers | Số thành viên tối đa nhóm |
| type | Loại môn học |
| password | Mật khẩu tham gia |
| **Diễn giải** | API cho phép giảng viên tạo môn học mới | |

Bảng 3.29. Bảng mô tả API xóa môn học

|  |  |
| --- | --- |
| **Phương thức** | **DELETE** |
| **Endpoint** | {BASE\_URL}/courses/:id |
| **Diễn giải** | API cho phép giảng viên xóa mềm môn học |

Bảng 3.30. Bảng mô tả API xóa hoàn toàn môn học

|  |  |
| --- | --- |
| **Phương thức** | **DELETE** |
| **Endpoint** | {BASE\_URL}/courses/:id/force |
| **Diễn giải** | API cho phép giảng viên xóa hoàn toàn môn học khỏi database |

Bảng 3.31. Bảng mô tả API khôi phục môn học

|  |  |
| --- | --- |
| **Phương thức** | **POST** |
| **Endpoint** | {BASE\_URL}/courses/:id/restore |
| **Diễn giải** | API cho phép giảng viên khôi phục môn học đã bị xóa mềm |

### API quản lý đề tài

Bảng 3.32. Bảng mô tả API tạo đề tài mới

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/topics | |
| **Thuộc tính** | title | Tên đề tài |
| description | Mô tả đề tài |
| courseId | ID môn học |
| teacherId | ID giảng viên hướng dẫn |
| isCustom | Đề tài tùy chỉnh |
| groupName | Tên nhóm thực hiện |
| **Diễn giải** | API cho phép tạo đề tài mới cho môn học | |

Bảng 3.33. Bảng mô tả API lấy danh sách đề tài

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **GET** | |
| **Endpoint** | {BASE\_URL}/topics | |
| **Thuộc tính** | page | Số trang (query parameter) |
| limit | Số lượng item mỗi trang (query parameter) |
| search | Từ khóa tìm kiếm (query parameter) |
| **Diễn giải** | API cho phép admin lấy danh sách tất cả đề tài | |

Bảng 3.34. bảng mô tả API lấy đề tài theo môn học

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **GET** | |
| **Endpoint** | {BASE\_URL}/topics/:courseId/course | |
| **Thuộc tính** | page | Số trang (query parameter) |
| limit | Số lượng item mỗi trang (query parameter) |
| search | Từ khóa tìm kiếm (query parameter) |
| status | Trạng thái đề tài (query parameter) |
| **Diễn giải** | API lấy danh sách đề tài của một môn học cụ thể | |

Bảng 3.35. Bảng mô tả API cập nhật đề tài

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **PUT** | |
| **Endpoint** | {BASE\_URL}/topics/:id | |
| **Thuộc tính** | title | Tên đề tài |
| description | Mô tả đề tài |
| courseId | ID môn học |
| teacherId | ID giảng viên hướng dẫn |
| isCustom | Đề tài tùy chỉnh |
| groupName | Tên nhóm thực hiện |
| **Diễn giải** | API cho phép cập nhật thông tin đề tài | |

Bảng 3.36. Bảng mô tả API xóa đề tài

|  |  |
| --- | --- |
| **Phương thức** | **DELETE** |
| **Endpoint** | {BASE\_URL}/topics/:id |
| **Diễn giải** | API cho phép xóa mềm đề tài |

Bảng 3.37. Bảng mô tả API xóa hoàn toàn đề tài

|  |  |
| --- | --- |
| **Phương thức** | **DELETE** |
| **Endpoint** | {BASE\_URL}/topics/:id/force |
| **Diễn giải** | API cho phép xóa hoàn toàn đề tài khỏi database |

Bảng 3.38. Bảng mô tả API khôi phục đề tài

|  |  |
| --- | --- |
| **Phương thức** | **PUT** |
| **Endpoint** | {BASE\_URL}/topics/:id/restore |
| **Diễn giải** | API cho phép khôi phục đề tài đã bị xóa mềm |

### API Thống kê và phân tích

Bảng 3.39. Bảng mô tả API lấy hoạt động code của môn học

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **GET** | |
| **Endpoint** | {BASE\_URL}/courses/:id/code-activity | |
| **Thuộc tính** | startDate | Ngày bắt đầu (query parameter) |
| endDate | Ngày kết thúc (query parameter) |
| groupBy | Nhóm theo (day/week/month) (query parameter) |
| **Diễn giải** | API lấy thống kê hoạt động code của môn học theo thời gian | |

Bảng 3.40. Bảng mô tả API lấy người đóng góp môn học

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/courses/:id/contributors |
| **Diễn giải** | API lấy danh sách người đóng góp code trong môn học |

Bảng 3.41. Bảng mô tả API lấy người đóng góp đề tài

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/topics/:id/contributors |
| **Diễn giải** | API lấy danh sách người đóng góp code trong đề tài |

Bảng 3.42. Bảng mô tả API lấy thống kê đề tài

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/topics/:id/stats |
| **Diễn giải** | API lấy thống kê tổng quan của đề tài (số commit, pull request, v.v.) |

### API webhook và GitHub

Bảng 3.43. Bảng mô tả API lấy thông tin repository

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/github/repos/:repoName |
| **Diễn giải** | API lấy thông tin chi tiết của repository GitHub theo tên |

Bảng 3.44. Bảng mô tả API lấy thông tin người dùng GitHub

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/github/user/:username |
| **Diễn giải** | API lấy thông tin người dùng GitHub theo username |

Bảng 3.45. Bảng mô tả API lấy thành viên tổ chức

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/github/orgs/members |
| **Diễn giải** | API lấy danh sách thành viên của tổ chức GitHub |

Bảng 3.46. Bảng mô tả API mời người dùng vào tổ chức

|  |  |
| --- | --- |
| **Phương thức** | **POST** |
| **Endpoint** | {BASE\_URL}/github/orgs/invite-user/:username |
| **Diễn giải** | API mời người dùng tham gia tổ chức GitHub |

Bảng 3.47. Bảng mô tả API xử lý webhook commit

|  |  |
| --- | --- |
| **Phương thức** | **POST** |
| **Endpoint** | {BASE\_URL}/github/webhook |
| **Diễn giải** | API xử lý webhook từ GitHub khi có commit mới |

Bảng 3.48. Bảng mô tả API thêm webhook commit

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/github/add-webhook | |
| **Thuộc tính** | repoName | Tên repository |
| repoUrl | URL repository |
| webhookUrl | URL webhook |
| events | Các sự kiện cần lắng nghe |
| **Diễn giải** | API thêm webhook cho repository để theo dõi commit | |

### API Quản lý Repository

Bảng 3.49. Bảng mô tả API lấy danh sách repository

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **GET** | |
| **Endpoint** | {BASE\_URL}/repos | |
| **Thuộc tính** | page | Số trang (query parameter) |
| limit | Số lượng item mỗi trang (query parameter) |
| search | Từ khóa tìm kiếm (query parameter) |
| **Diễn giải** | API lấy danh sách tất cả repository trong hệ thống | |

Bảng 3.50. Bảng mô tả API lấy thông tin repository

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/repos/:id |
| **Diễn giải** | API lấy thông tin chi tiết của repository theo ID |

Bảng 3.51. Bảng mô tả API lấy repository theo đề tài

|  |  |
| --- | --- |
| **Phương thức** | **GET** |
| **Endpoint** | {BASE\_URL}/repos/topic/:id |
| **Diễn giải** | API lấy repository của một đề tài cụ thể |

Bảng 3.52. Bảng mô tả API tạo repository mới

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/repos | |
| **Thuộc tính** | name | Tên repository |
| url | URL repository GitHub |
| courseId | ID môn học |
| topicId | ID đề tài |
| authorId | ID người tạo |
| language | Ngôn ngữ lập trình |
| framework | Framework sử dụng |
| sonarKey | Khóa SonarQube |
| **Diễn giải** | API tạo repository mới và liên kết với đề tài | |

Bảng 3.53. Bảng mô tả API cập nhật repository

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **PUT** | |
| **Endpoint** | {BASE\_URL}/repos/:id | |
| **Thuộc tính** | name | Tên repository |
| url | URL repository GitHub |
| language | Ngôn ngữ lập trình |
| framework | Framework sử dụng |
| sonarKey | Khóa SonarQube |
| **Diễn giải** | API cập nhật thông tin repository | |

Bảng 3.54. Bảng mô tả API xóa repository

|  |  |
| --- | --- |
| **Phương thức** | **DELETE** |
| **Endpoint** | {BASE\_URL}/repos/:id |
| **Diễn giải** | API xóa mềm repository |

Bảng 3.55. Bảng mô tả API xóa hoàn toàn repository

|  |  |
| --- | --- |
| **Phương thức** | **DELETE** |
| **Endpoint** | {BASE\_URL}/repos/:id/force |
| **Diễn giải** | API xóa hoàn toàn repository khỏi database |

Bảng 3.56. Bảng mô tả API khôi phục repository

|  |  |
| --- | --- |
| **Phương thức** | **PUT** |
| **Endpoint** | {BASE\_URL}/repos/:id/restore |
| **Diễn giải** | API khôi phục repository đã bị xóa mềm |

### API Tích hợp SonarQube

Bảng 3.57. Bảng mô tả API tạo dự án SonarQube

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **POST** | |
| **Endpoint** | {BASE\_URL}/sonar | |
| **Thuộc tính** | name | Tên dự án |
| key | Khóa dự án |
| organization | Tổ chức |
| **Diễn giải** | API tạo dự án mới trên SonarQube | |

Bảng 3.58. Bảng mô tả API xóa dự án SonarQube

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **DELETE** | |
| **Endpoint** | {BASE\_URL}/sonar | |
| **Thuộc tính** | name | Tên dự án |
| key | Khóa dự án |
| organization | Tổ chức |
| **Diễn giải** | API xóa dự án khỏi SonarQube | |

Bảng 3.59. Bảng mô tả API lấy metrics SonarQube

|  |  |  |
| --- | --- | --- |
| **Phương thức** | **GET** | |
| **Endpoint** | {BASE\_URL}/sonar/:name | |
| **Thuộc tính** | metrics | Danh sách metrics cần lấy (query parameter) |
| component | Component cần phân tích (query parameter) |
| **Diễn giải** | API lấy các metrics chất lượng mã nguồn từ SonarQube | |

### Một số API khác

Ngoài các API đã được trình bày chi tiết ở trên, hệ thống CodeFlow còn bao gồm các nhóm API bổ sung quan trọng khác để hỗ trợ đầy đủ chức năng của hệ thống:

API Quản lý tệp tin (File Management): Nhóm API này cung cấp chức năng upload và quản lý tệp tin trong hệ thống, bao gồm upload tệp tin với middleware xử lý file, và hiển thị avatar người dùng. Các API này đảm bảo việc lưu trữ và truy xuất tệp tin một cách an toàn và hiệu quả.

API Quản lý bài viết và tương tác (Posts & Interactions): Nhóm API này hỗ trợ hệ thống thảo luận và chia sẻ thông tin, bao gồm tạo, cập nhật, xóa bài viết, quản lý like/unlike, và lấy bình luận theo bài viết. Các API này tạo ra môi trường giao tiếp và học tập cộng tác giữa sinh viên và giảng viên.

API Quản lý nhãn (Tags Management): Nhóm API này cho phép quản lý hệ thống phân loại và gắn nhãn cho các đối tượng trong hệ thống như môn học, đề tài, và bài viết. Các API này hỗ trợ việc tìm kiếm và phân loại nội dung một cách hiệu quả.

API Quản lý bình luận (Comments Management): Nhóm API này cung cấp chức năng bình luận cho các đối tượng trong hệ thống, bao gồm tạo, cập nhật, xóa bình luận với hỗ trợ bình luận đa cấp (nested comments). Các API này tăng cường tính tương tác và thảo luận trong hệ thống.

API Quản lý thông báo (Notifications): Nhóm API này hỗ trợ hệ thống thông báo real-time, bao gồm tạo thông báo, lấy thông báo theo người dùng, đánh dấu đã đọc, và quản lý trạng thái thông báo. Các API này đảm bảo người dùng luôn được cập nhật về các hoạt động quan trọng trong hệ thống.

API Tìm kiếm toàn cục (Global Search): Nhóm API này cung cấp chức năng tìm kiếm tổng hợp trên toàn bộ hệ thống, cho phép tìm kiếm môn học, đề tài, bài viết, và người dùng với các bộ lọc và phân trang. API này giúp người dùng nhanh chóng tìm kiếm thông tin cần thiết.

API Tích hợp AI Gemini: Nhóm API này tích hợp với Google Gemini AI để cung cấp các tính năng thông minh như tạo nội dung, phân tích văn bản, và hỗ trợ học tập. API này mở rộng khả năng của hệ thống với các tính năng AI hiện đại.

## Thiết lập workflow cho GitHub Actions và tích hợp SonarCloud

### Thiết lập tệp cấu hình dự án

- Cấu hình sonar-project.properties

sonar.organization=${organization}

sonar.projectKey=${projectKey}

sonar.sources=.

sonar.exclusions=node\_modules/\*\*, dist/\*\*, build/\*\*, coverage/\*\*

### Thiết lập tệp wokflow

- Node.js / JavaScript / TypeScript

name: Node.js CI with SonarCloud

on:

push:

branches: [ main ]

pull\_request:

branches: [ main ]

jobs:

build:

runs-on: ubuntu-latest

steps:

- uses: actions/checkout@v4

with:

fetch-depth: 0

- name: Use Node.js

uses: actions/setup-node@v4

with:

node-version: 18

- run: npm ci

- run: npm run build

- run: npm run test -- --coverage

- name: SonarCloud Scan

uses: SonarSource/sonarcloud-github-action@v2

env:

GITHUB\_TOKEN: ${{ secrets.GITHUB\_TOKEN }}

SONAR\_TOKEN: ${{ secrets.SONAR\_TOKEN }}

# KẾT QUẢ VÀ ĐÁNH GIÁ

### Giao diện giảng viên

### Giao diện sinh viên

# 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

1. [↑](#endnote-ref-1)