Báo cáo: Hệ thống Đánh giá Mã nguồn Chuyên sâu dựa trên Trí tuệ Nhân tạo – Thiết kế và Lộ trình (Tháng 5 năm 2025)

1. Tóm tắt Báo cáo

Phần này cung cấp một cái nhìn tổng quan, súc tích về hệ thống đánh giá mã nguồn dựa trên trí tuệ nhân tạo (AI) được đề xuất. Hệ thống này mang lại giá trị chiến lược thông qua việc nâng cao đáng kể chất lượng mã nguồn, năng suất của lập trình viên và tính toàn vẹn của kiến trúc phần mềm. Các chức năng cốt lõi bao gồm phân tích ngữ nghĩa sâu dựa trên Mô hình Ngôn ngữ Lớn (LLM), kiến trúc đa tác tử để xử lý các tác vụ chuyên biệt, khả năng quét theo yêu cầu của người dùng cho cả Pull Request (PR) và toàn bộ dự án, tạo ra các giải pháp khắc phục lỗi có tính thực tiễn cao, và tự động hóa việc tạo sơ đồ kiến trúc (sơ đồ lớp và sơ đồ tuần tự) để trực quan hóa các thay đổi và tác động của chúng. Một cách tiếp cận chiến lược được nhấn mạnh là sự phụ thuộc vào mô hình hybrid, kết hợp sức mạnh của Cây Cú pháp Trừu tượng (AST) cho phân tích cấu trúc với khả năng hiểu biết tinh vi của LLM, đồng thời ưu tiên các thành phần mã nguồn mở. Những lợi ích dự kiến, bao gồm chu kỳ phát triển nhanh hơn, giảm thiểu rủi ro một cách chủ động và nợ kỹ thuật ít hơn, sẽ được nêu bật. Cuối cùng, một lộ trình phát triển theo từng giai đoạn sẽ được đề cập ngắn gọn, đảm bảo việc cung cấp giá trị một cách lặp đi lặp lại.

Hệ thống này đại diện cho một sự chuyển dịch từ việc sửa lỗi thụ động sang đảm bảo chất lượng mã nguồn và quản lý sức khỏe kiến trúc một cách chủ động, tận dụng AI để tăng cường chuyên môn của con người. Bằng cách cung cấp phân tích sâu và các giải pháp khả thi sớm (ngay cả đối với các PR đã đóng hoặc quét toàn bộ dự án), hệ thống nhằm mục đích giảm chi phí phát sinh do việc sửa lỗi và quản lý nợ kỹ thuật ở các giai đoạn sau của vòng đời phát triển.[1] Việc triển khai thành công có thể thúc đẩy một văn hóa kỹ thuật chất lượng cao hơn và cho phép các lập trình viên cấp cao tập trung vào các thách thức thiết kế phức tạp hơn thay vì các tác vụ đánh giá thông thường. Yêu cầu của người dùng về một hệ thống "chuyên sâu" "giúp lập trình viên dễ dàng sửa lỗi" và "hiểu cấu trúc dự án" cho thấy sự cần thiết phải vượt ra ngoài việc kiểm tra mã thông thường. Sự kết hợp giữa AST để đảm bảo độ chính xác và LLM để hiểu sâu về ngữ nghĩa là một chủ đề thường thấy trong phân tích mã nâng cao.[6] Cách tiếp cận đa tác tử phù hợp để phân chia các tác vụ phân tích phức tạp thành các đơn vị chuyên biệt, có thể quản lý được. Khung thời gian tháng 5 năm 2025 ngụ ý việc tận dụng các khả năng LLM trưởng thành và các framework mã nguồn mở có sẵn vào thời điểm đó.

2. Giới thiệu

2.1. Bối cảnh Phát triển của Đánh giá Mã nguồn

Các quy trình đánh giá mã nguồn thủ công truyền thống thường gặp nhiều hạn chế, bao gồm tốn thời gian, dễ xảy ra lỗi do con người, thiếu nhất quán, đặc biệt trong các dự án lớn và phức tạp.[8] Sự xuất hiện của Trí tuệ Nhân tạo (AI), đặc biệt là các Mô hình Ngôn ngữ Lớn (LLM), mang lại tiềm năng biến đổi việc đánh giá mã nguồn bằng cách tự động hóa và nâng cao quy trình thông qua việc cung cấp những hiểu biết sâu sắc hơn và phản hồi có tính ứng dụng cao.[12] Tính đến tháng 5 năm 2025, lĩnh vực AI trong phân tích mã nguồn đã có những bước tiến đáng kể, với khả năng lý luận, tạo mã và phát hiện lỗi của LLM ngày càng được cải thiện.[15] Ví dụ, báo cáo AI Index 2025 nhấn mạnh những bước nhảy vọt đáng kể trong khả năng giải quyết các vấn đề về mã hóa của AI.[15]

2.2. Cơ sở Lý luận và Giá trị Chiến lược của Dự án

Nhu cầu về một công cụ đánh giá mã nguồn nội bộ tiên tiến, vượt trội hơn các linter hiện có và các công cụ phân tích tĩnh cơ bản là rất rõ ràng. Một hệ thống như vậy có thể mang lại những lợi ích hữu hình: cải thiện khả năng bảo trì mã nguồn, tăng cường tình trạng bảo mật, đẩy nhanh tốc độ phát triển và hỗ trợ tốt hơn cho việc giới thiệu các lập trình viên mới thông qua phản hồi nhất quán, chất lượng cao.

2.3. Mục tiêu và Mục đích của Hệ thống Đề xuất

- **Mục tiêu chính:** Thiết kế và vạch ra lộ trình phát triển một hệ thống đánh giá mã nguồn đa tác tử, dựa trên LLM, tinh vi, cung cấp phân tích chuyên sâu và thông tin chi tiết hữu ích cho cả PR và toàn bộ dự án.
- **Mục tiêu cụ thể:**
    - Đạt được sự hiểu biết ngữ nghĩa sâu sắc về các thay đổi mã nguồn và ý nghĩa của chúng.
    - Tạo ra các đề xuất giải quyết lỗi chính xác, nhận biết ngữ cảnh và có thể thực hiện được.
    - Trực quan hóa kiến trúc mã nguồn và tác động của các thay đổi thông qua các sơ đồ được tạo tự động.
    - Cho phép đánh giá toàn diện tình trạng dự án, bao gồm nợ kỹ thuật và dự đoán rủi ro.
    - Ưu tiên sử dụng các công nghệ mã nguồn mở và đảm bảo tính mô-đun để có khả năng mở rộng trong tương lai.
    - Đảm bảo hệ thống hoạt động độc lập, tạo báo cáo mà không tích hợp trực tiếp vào các quy trình CI/CD.

2.4. Phạm vi của Báo cáo này

Tài liệu này cung cấp một thiết kế hệ thống chi tiết, các cân nhắc về công nghệ và một lộ trình chiến lược cho việc phát triển hệ thống đánh giá mã nguồn dựa trên AI. Đối tượng mục tiêu là các chuyên gia kỹ thuật, bao gồm các trưởng nhóm kỹ thuật, kiến trúc sư phần mềm và các lập trình viên cấp cao, những người sẽ tham gia vào việc triển khai hoặc đánh giá một hệ thống như vậy.

Sự phức tạp ngày càng tăng của phần mềm và nhu cầu về chu kỳ phát hành nhanh hơn đòi hỏi các công cụ đảm bảo chất lượng thông minh và tự động hơn. Đánh giá mã nguồn dựa trên AI là một bước phát triển tự nhiên. Bằng cách cung cấp phân tích toàn diện, theo yêu cầu của người dùng (trái ngược với việc chỉ kiểm tra qua CI), hệ thống trao quyền cho các lập trình viên chủ động cải thiện chất lượng mã nguồn ở nhiều giai đoạn khác nhau, có khả năng giảm bớt gánh nặng cho các quy trình đánh giá chính thức hơn ở giai đoạn sau. Việc người dùng loại bỏ yêu cầu "tự động quét PR qua webhook" và thay thế bằng "người dùng chủ động lấy thông tin PR" là một yếu tố định hướng kiến trúc quan trọng. Điều này có nghĩa là hệ thống ít tập trung vào việc kiểm soát ngay lập tức mà tập trung hơn vào việc cung cấp phân tích toàn diện, theo yêu cầu. Điều này ảnh hưởng đến thiết kế quy trình làm việc và mô hình tương tác dự kiến. Hệ thống trở thành một công cụ phân tích mà các lập trình viên tham khảo, thay vì một giai đoạn trong quy trình mà họ phải vượt qua. Điều này cũng đơn giản hóa việc triển khai ban đầu bằng cách tách rời khỏi các hệ thống CI/CD.

3. Kiến trúc Hệ thống Cốt lõi

3.1. Sơ đồ Kiến trúc Tổng thể

Kiến trúc tổng thể của hệ thống được thiết kế theo hướng mô-đun, lấy cảm hứng từ microservices hoặc kiến trúc dựa trên thành phần để tạo điều kiện thuận lợi cho việc phát triển, mở rộng và bảo trì độc lập các chức năng khác nhau.

Sơ đồ kiến trúc cấp cao sẽ được trình bày ở đây, minh họa các thành phần chính và luồng dữ liệu giữa chúng. Có thể sử dụng các khái niệm từ mô hình C4 [255] để mô tả ngữ cảnh và các container nếu phù hợp.

Các thành phần chính bao gồm:

- **Lớp Giao diện Người dùng/Tương tác (User Interface/Interaction Layer):** Một công cụ dòng lệnh (CLI) hoặc một giao diện người dùng web đơn giản để người dùng khởi tạo quá trình quét và xem báo cáo.
- **Lớp Điều phối (Orchestration Layer):** Hệ thống Đa Tác tử (Multi-Agent System)

dựa trên LangGraph, quản lý toàn bộ quy trình làm việc.
- **Bộ máy Phân tích Mã nguồn (Code Analysis Engine):** Bao gồm các mô-đun phân tích tĩnh dựa trên AST và phân tích sâu dựa trên LLM/RAG.
- **Bộ máy Tạo Sơ đồ (Diagramming Engine):** Chịu trách nhiệm tạo ra các biểu diễn trực quan về cấu trúc mã nguồn và các thay đổi.
- **Bộ máy Báo cáo (Reporting Engine):** Tổng hợp các phát hiện và tạo ra các báo cáo đánh giá toàn diện.
- **Kho Dữ liệu/Kho Vector (Knowledge Base/Vector Store):** Dành cho ngữ cảnh RAG, lưu trữ các embedding của mã nguồn, tài liệu và có thể cả các mẫu kiến trúc.
- **Kho Cấu hình & Quy tắc (Configuration & Rule Store):** Để lưu trữ các quy tắc phân tích tĩnh tùy chỉnh và cấu hình hệ thống.

Luồng dữ liệu sẽ được minh họa, cho thấy cách thông tin (thông tin PR, mã nguồn, AST, prompt/response của LLM, dữ liệu sơ đồ) di chuyển giữa các thành phần này.

3.2. Framework Đa Tác tử (LangGraph)

Việc lựa chọn LangGraph làm framework điều phối đa tác tử là do khả năng mô hình hóa các quy trình làm việc phức tạp, có trạng thái và có khả năng lặp đi lặp lại, vốn là đặc điểm cố hữu của một quy trình đánh giá mã nguồn toàn diện bao gồm nhiều giai đoạn phân tích và vòng lặp phản hồi.[20] LangGraph cho phép quản lý các tác vụ chạy dài, xử lý lỗi trong các nhánh song song và tích hợp khả năng con người tham gia vào vòng lặp (human-in-the-loop).[23]

Vai trò và Trách nhiệm của các Tác tử:

Hệ thống sẽ bao gồm các tác tử chuyên biệt, mỗi tác tử chịu trách nhiệm về một phần cụ thể của quy trình đánh giá:

- **UserInteractionAgent (Tác tử Tương tác Người dùng):** Xử lý các yêu cầu quét PR/dự án từ người dùng, thu thập các tham số cần thiết (ví dụ: URL kho chứa, ID PR, nhánh).
- **CodeFetcherAgent (Tác tử Thu thập Mã nguồn):** Tương tác với các kho chứa Git (sử dụng các thư viện như go-git cho Go [35], GitPython cho Python [37], JGit cho Java [39], hoặc isomorphic-git cho môi trường Node.js [41]) để truy xuất diff của PR, các tệp cụ thể hoặc toàn bộ nội dung dự án.
- **ASTParsingAgent (Tác tử Phân tích Cú pháp AST):** Sử dụng Tree-sitter [42] để phân tích mã nguồn thành Cây Cú pháp Trừu tượng (AST) cho các ngôn ngữ được hỗ trợ. Quản lý các ngữ pháp ngôn ngữ khác nhau.
- **StaticAnalysisAgent (Tác tử Phân tích Tĩnh):** Thực hiện các kiểm tra dựa trên

quy tắc trên AST. Tác tử này sẽ chứa bộ máy quy tắc tùy chỉnh được xây dựng trên Tree-sitter. Nó có thể xác định lỗi cú pháp, vi phạm kiểu và các anti-pattern cơ bản. Cách tiếp cận này tận dụng phương pháp "AST + bộ máy quy tắc" được thấy trong các công cụ như Kodus [6] và Semgrep.

- **LLMOrchestratorAgent (Tác tử Điều phối LLM):** Quản lý các tương tác với các LLM khác nhau (cả mô hình mã nguồn mở và các nhà cung cấp thương mại như OpenAI, Google Gemini). Chịu trách nhiệm về kỹ thuật prompt, quản lý cửa sổ ngữ cảnh và tổng hợp các phản hồi của LLM. Tác tử này sẽ là trung tâm để cung cấp phân tích ngữ nghĩa sâu.
- **RAGContextAgent (Tác tử Ngữ cảnh RAG):** Xây dựng và truy vấn cơ sở kiến thức RAG. Điều này bao gồm việc chia nhỏ mã nguồn (có thể sử dụng LlamaIndex CodeSplitter [46] hoặc các phương pháp nhận biết AST khác [46]), tạo embedding và truy xuất ngữ cảnh liên quan (đoạn mã, tài liệu, các mẫu đánh giá trước đây) cho LLMOrchestratorAgent.
- **DiagramGenerationAgent (Tác tử Tạo Sơ đồ):** Nhận thông tin cấu trúc (từ ASTParsingAgent và phân tích thay đổi của LLMOrchestratorAgent) để tạo dữ liệu cho sơ đồ lớp và sơ đồ tuần tự. Sử dụng các thư viện như PlantUML [49] hoặc Mermaid.js [54] (hoặc các bộ máy cơ bản của chúng như Graphviz [58]) để render các sơ đồ này.
- **ImpactAnalysisAgent (Tác tử Phân tích Tác động):** Phân tích các thay đổi mã nguồn (diff) và AST để xác định các thành phần và phụ thuộc bị ảnh hưởng, cung cấp thông tin này cho DiagramGenerationAgent và ReportingAgent.
- **SolutionSuggestionAgent (Tác tử Đề xuất Giải pháp):** Phối hợp chặt chẽ với LLMOrchestratorAgent để tinh chỉnh các kết quả đầu ra của LLM thành các giải pháp dễ hiểu, có thể thực hiện được cho các lập trình viên. Tác tử này tập trung vào khía cạnh "cách khắc phục".
- **ProjectScanningAgent (Tác tử Quét Dự án):** Điều phối các lần quét toàn bộ dự án, quản lý ngữ cảnh trên nhiều tệp (có thể sử dụng tóm tắt phân cấp [59] hoặc các kỹ thuật RAG nâng cao [65]) để đánh giá tình trạng tổng thể của dự án, xác định các điểm nóng về nợ kỹ thuật [2] và dự đoán rủi ro.
- **ReportingAgent (Tác tử Báo cáo):** Tổng hợp các phát hiện từ tất cả các tác tử khác thành một báo cáo có cấu trúc, dễ đọc cho con người, bao gồm các sơ đồ đã tạo và điểm số rủi ro. Tác tử này sẽ định dạng đầu ra theo một lược đồ xác định, có thể lấy cảm hứng từ SARIF [67] để có khả năng tương tác.

Giao tiếp và Quản lý Trạng thái:

- Tận dụng khả năng quản lý trạng thái của LangGraph để truyền thông tin giữa các tác tử (ví dụ: AST, dữ liệu diff, phản hồi LLM, dữ liệu sơ đồ).[20]
- Sử dụng checkpointing cho các tác vụ chạy dài như quét toàn bộ dự án để đảm

bảo khả năng tiếp tục và khả năng chịu lỗi.[33]
- Xác định các lược đồ dữ liệu rõ ràng cho giao tiếp giữa các tác tử, có thể sử dụng GraphQL hoặc tương tự để trao đổi dữ liệu có cấu trúc.[70]

3.3. Chiến lược Tích hợp LLM

- **Nhà cung cấp LLM có thể thay thế (Pluggable LLM Providers):** Thiết kế một lớp trừu tượng cho các tương tác LLM, cho phép dễ dàng chuyển đổi giữa các API của OpenAI, Google Gemini và các mô hình mã nguồn mở khác nhau (ví dụ: CodeLlama [71], StarCoder2 [73], DeepSeek Coder V2 [74], Qwen2 [19], Mistral-Large [19], Phi-4 [19], Gemma-2 [19]) được lưu trữ cục bộ hoặc thông qua các API như Hugging Face Inference Endpoints hoặc Ollama.[16] Hệ thống nên cho phép người dùng định cấu hình nhà cung cấp LLM và khóa API ưa thích của họ.[6] Kodus.io đề cập đến việc "Mang theo khóa của riêng bạn" (Bring Your Own Key).[6]
- **Quản lý Cửa sổ Ngữ cảnh (Context Window Management):**
  - Triển khai các chiến lược để xử lý các giới hạn cửa sổ ngữ cảnh của LLM, đặc biệt đối với các codebase lớn hoặc các PR mở rộng.[77]
  - Các kỹ thuật bao gồm:
    - **Chunking (Chia nhỏ):** Chia mã thành các phần nhỏ hơn, dễ quản lý. Ưu tiên chunking dựa trên AST (ví dụ: LlamaIndex CodeSplitter [46]) hơn là chia văn bản đơn giản để duy trì tính toàn vẹn ngữ nghĩa.[46]
    - **Summarization (Tóm tắt):** Tóm tắt phân cấp mã ở các cấp độ khác nhau (hàm, tệp, thư mục, dự án) để tạo ngữ cảnh cô đọng cho LLM.[59]
    - **Retrieval Augmented Generation (RAG):** Tự động tìm nạp các đoạn mã, tài liệu hoặc thông tin kiến trúc liên quan để bổ sung cho prompt (xem mục 3.5.2).
    - **Sliding Window / Overlapping Chunks (Cửa sổ trượt / Chunk chồng chéo):** Xử lý văn bản theo các đoạn chồng chéo để duy trì tính liên tục.[76]
- **Kỹ thuật Prompt Hiệu quả (Effective Prompt Engineering):**
  - Phát triển các mẫu prompt mạnh mẽ, phù hợp cho các tác vụ phân tích mã khác nhau (phát hiện lỗi, đề xuất giải pháp, trích xuất dữ liệu sơ đồ, đánh giá kiến trúc).
  - Kết hợp các đặc trưng rút trích từ AST vào prompt để cung cấp ngữ cảnh có cấu trúc cho LLM, cải thiện độ chính xác và giảm ảo giác (hallucination).[6]
  - Sử dụng kỹ thuật Chain-of-Thought (CoT) để hướng dẫn LLM qua các quy trình suy luận phức tạp.[95]
  - Sử dụng Few-shot prompting với các ví dụ về mã tốt, lỗi phổ biến và các mẫu sửa lỗi mong muốn.[95]

3.4. Thu thập và Tiền xử lý Dữ liệu

- **Truy xuất Thông tin PR/Dự án theo Yêu cầu Người dùng:** UserInteractionAgent sẽ thu thập URL kho chứa, số PR (đối với quét PR), tên nhánh hoặc yêu cầu quét toàn bộ dự án.
- **Phân tích Cú pháp Mã nguồn và Tạo AST:**
  - ASTParsingAgent sẽ sử dụng **Tree-sitter** [42] làm bộ máy phân tích cú pháp cốt lõi do tốc độ, tính mạnh mẽ, khả năng phục hồi lỗi và hỗ trợ đa ngôn ngữ.
  - Duy trì một bộ sưu tập các ngữ pháp Tree-sitter cho các ngôn ngữ được hỗ trợ (ban đầu là Python, Java, Kotlin). XML của Android (AndroidManifest.xml, tệp layout) sẽ yêu cầu một ngữ pháp XML phù hợp (ví dụ: tree-sitter-xml [98]) hoặc các phần mở rộng tùy chỉnh nếu cần cho các không gian tên/thuộc tính dành riêng cho Android.[100]
  - Hệ thống phải xử lý các lỗi phân tích cú pháp một cách linh hoạt, đặc biệt với mã nguồn chưa hoàn chỉnh hoặc có lỗi cú pháp thường thấy trong các PR. Khả năng phục hồi lỗi của Tree-sitter rất hữu ích ở đây.[102]
- **Phân tích Diff:** CodeFetcherAgent sẽ lấy các diff (ví dụ: kết quả từ lệnh git diff). Các thư viện như jsdiff [177] cho JavaScript hoặc difflib của Python [180] có thể được sử dụng để xử lý diff chi tiết nếu cần, mặc dù LLM cũng có thể xử lý văn bản diff thô.

3.5. Bộ máy Phân tích Mã nguồn

- **3.5.1. Phân tích Tĩnh và Ngữ nghĩa với AST (Bộ máy Dựa trên Quy tắc):**
  - StaticAnalysisAgent sẽ thực thi một bộ máy quy tắc tùy chỉnh được xây dựng trên các truy vấn Tree-sitter.
  - **Ngôn ngữ Định nghĩa Quy tắc DSL (Rule Definition DSL):** Thiết kế một DSL đơn giản, mang tính khai báo để định nghĩa các quy tắc phân tích mã (ví dụ: dựa trên YAML, tương tự như Semgrep hoặc ast-grep [101]). DSL này cho phép định nghĩa các mẫu để khớp trong AST.[101]
  - **Danh mục Quy tắc:** Tập trung vào các lỗi phổ biến, vi phạm phong cách (ví dụ: Google Style Guides [153], Android Kotlin Style Guide [155], Airbnb JavaScript Style Guide [156]), các anti-pattern, kiểm tra bảo mật cơ bản (ví dụ: thông tin nhạy cảm được mã hóa cứng, các điểm chèn mã phổ biến, lấy cảm hứng từ OWASP Top 10 [194] và CWE Top 25 [195]), và các kiểm tra ban đầu dành riêng cho Android (ví dụ: sử dụng tài nguyên sai cách, các vấn đề phổ biến trong tệp kê khai dựa trên [100]).
  - **Duyệt AST và Khớp Mẫu:** Sử dụng các kỹ thuật duyệt AST hiệu quả và khớp mẫu.[101]
  - Cách tiếp cận này phản ánh cách Kodus sử dụng một bộ máy quy tắc dựa trên AST để cung cấp ngữ cảnh có cấu trúc chính xác cho LLM.[6]
- **3.5.2. Hiểu biết Mã nguồn Chuyên sâu dựa trên LLM:**

    - LLMOrchestratorAgent, với sự hỗ trợ của RAGContextAgent và SolutionSuggestionAgent, sẽ thực hiện phân tích sâu hơn.
    - **Phát hiện Lỗi (Bug Detection):** Xác định các lỗi logic phức tạp [203], các lỗi ngữ nghĩa tinh vi [208], các lỗ hổng bảo mật vượt ra ngoài các mẫu đơn giản [211], các vấn đề đồng thời (data races, deadlocks) [213], và các điểm nghẽn hiệu năng.[217] LLM như GPT-4 đã cho thấy khả năng trong các lĩnh vực này, nhưng vẫn còn hạn chế.[219] Sự giám sát của con người và RAG là rất quan trọng.[219]
    - **Tạo Đề xuất Giải quyết Lỗi có Tính Thực tiễn (Generating Actionable Error Resolution Suggestions):** Hệ thống sẽ tập trung vào việc cung cấp các giải pháp rõ ràng, nhận biết ngữ cảnh và có thể thực hiện được cho các vấn đề được phát hiện, giúp các lập trình viên dễ dàng sửa lỗi.[6] Điều này không chỉ bao gồm việc xác định lỗi mà còn giải thích *tại sao* đó là lỗi và *cách* khắc phục.
    - **Tích hợp Retrieval Augmented Generation (RAG):**
        - RAGContextAgent sẽ tạo và duy trì một cơ sở kiến thức từ mã nguồn của dự án, tài liệu và có thể cả các nguồn bên ngoài (ví dụ: hướng dẫn thực hành tốt nhất, khuyến cáo bảo mật).
        - Các framework như LlamaIndex [46] hoặc khả năng RAG của LangChain [239] sẽ được sử dụng để lập chỉ mục và truy xuất.
        - **RAG dành riêng cho Mã nguồn (Code-Specific RAG):** Các chiến lược RAG cho mã nguồn bao gồm chunking nhận biết AST [46], embedding tóm tắt mã nguồn và sử dụng cơ sở dữ liệu đồ thị để nắm bắt các mối quan hệ trong mã nguồn.[226]
        - **Ngữ cảnh Hybrid Tượng trưng-Nơ-ron (Hybrid Symbolic-Neural Context):** Kết hợp ngữ cảnh có cấu trúc từ phân tích AST (tượng trưng) với ngữ cảnh được truy xuất theo ngữ nghĩa từ RAG (nơ-ron) để cung cấp đầu vào cho LLM.[6] Đây là một yếu tố khác biệt quan trọng cho phân tích nâng cao.
- **3.5.3. Phân tích Kiến trúc và Tạo Sơ đồ:**
    - DiagramGenerationAgent sẽ chịu trách nhiệm cho việc này.
    - **Trích xuất Thông tin cho Sơ đồ:**
        - **Sơ đồ Lớp (Class Diagrams):** Xác định các lớp, thuộc tính, phương thức và mối quan hệ (thừa kế, tập hợp, liên kết) bằng cách phân tích AST từ các tệp Python, Java và Kotlin. Đối với Android, phân tích AndroidManifest.xml để tìm các thành phần [43] và các tệp XML layout để hiểu cấu trúc giao diện người dùng.[101] Các công cụ như IntelliJ IDEA [249] và các trình tạo UML khác [250] cung cấp ví dụ về những gì cần trích xuất.
        - **Sơ đồ Tuần tự (Sequence Diagrams):** Đối với một PR, xác định các hàm bị sửa đổi/thêm vào. Theo dõi các lệnh gọi đi từ các hàm này và các lệnh

gọi đến chúng trong một độ sâu hợp lý, sử dụng phân tích AST và có thể cả RAG để giải quyết các lệnh gọi động hoặc triển khai giao diện. Phạm vi sẽ được giới hạn trong tác động trực tiếp của PR. Có thể cần sự hỗ trợ của LLM để suy ra các chuỗi tương tác từ mã phức tạp hoặc mô tả bằng ngôn ngữ tự nhiên trong thông điệp commit/mô tả PR.[251]

- **Làm nổi bật Thay đổi và Các Thành phần bị Ảnh hưởng:**
  - So sánh AST trước và sau thay đổi trong một PR để xác định các lớp/phương thức/lệnh gọi bị thêm/sửa đổi/xóa.
  - Trực quan hóa những thay đổi này trên sơ đồ (ví dụ: mã hóa màu, chú thích).
  - Theo dõi các phụ thuộc để làm nổi bật các thành phần khác có khả năng bị ảnh hưởng bởi các thay đổi của PR.
- **Công cụ Render Sơ đồ:**
  - Tạo mô tả sơ đồ ở các định dạng như cú pháp PlantUML [49] hoặc Mermaid.js.[54] Các định dạng dựa trên văn bản này có thể dễ dàng được tạo bằng mã.
  - Hệ thống sẽ xuất các mô tả sơ đồ này dưới dạng tệp văn bản, sau đó có thể được render bằng các công cụ bên ngoài hoặc các thành phần xem tích hợp.
  - Xem xét trực quan hóa mô hình C4 cho các khung nhìn kiến trúc lớn hơn nếu việc quét toàn bộ dự án nhằm mục đích cung cấp điều này.[255]

3.6. Mô-đun Báo cáo

- ReportingAgent sẽ tổng hợp tất cả các phát hiện.
- **Cấu trúc của Báo cáo Đánh giá:** Báo cáo phải được cấu trúc tốt, bao gồm:
  - Tóm tắt các phát hiện.
  - Danh sách các vấn đề được phát hiện, được phân loại theo mức độ nghiêm trọng (nghiêm trọng, cao, trung bình, thấp) và loại (lỗi, lỗ hổng, hiệu suất, phong cách).
  - Đối với mỗi vấn đề: mô tả, vị trí (tệp, số dòng), giải pháp đề xuất và lý do/điểm tin cậy của LLM nếu có.
  - Các sơ đồ đã tạo (lớp và tuần tự) với các thay đổi và tác động được đánh dấu.
  - Điểm rủi ro tổng thể cho PR hoặc dự án.
- **Định dạng Đầu ra:** Báo cáo sẽ được tạo ở định dạng dễ đọc cho con người (ví dụ: Markdown, HTML) và có thể ở định dạng máy có thể đọc được như SARIF [67] để tích hợp trong tương lai.
- **Giao diện Người dùng (UI) để Trình bày Đề xuất:** Mặc dù hệ thống là độc lập, bản thân báo cáo chính là giao diện người dùng. Cần xem xét các phương pháp hay nhất để trình bày phản hồi AI.[220] Điều này bao gồm:

- Giải thích rõ ràng về các vấn đề và đề xuất.[266]
- Trực quan hóa các đường dẫn suy luận hoặc sự chú ý của LLM nếu khả thi.[296]
- Hiển thị điểm tin cậy cho các đề xuất.[297]
- Trình bày các đề xuất thay thế nếu có.[297]

3.7. Mô-đun Quét Toàn bộ Dự án

- ProjectScanningAgent sẽ quản lý việc này.
- **Phân tích Cấu trúc Dự án Chuyên sâu:**
  - Tận dụng AST và RAG để xây dựng một mô hình toàn diện của toàn bộ dự án, bao gồm các phụ thuộc giữa các tệp/mô-đun, biểu đồ lệnh gọi và các mẫu kiến trúc.[297]
  - Các kỹ thuật như tóm tắt phân cấp [59] sẽ rất quan trọng để cung cấp ngữ cảnh toàn dự án cho các LLM có cửa sổ ngữ cảnh hạn chế.
  - Cân nhắc các công cụ như CodeQL [150] hoặc Semgrep để lấy cảm hứng về khả năng phân tích toàn bộ chương trình, ngay cả khi chúng ta xây dựng bộ máy dựa trên AST của riêng mình.
- **Dự đoán Rủi ro và Tạo Đề xuất:**
  - Xác định các điểm nóng tiềm ẩn về lỗi, các khu vực có nợ kỹ thuật cao [2], và các vi phạm kiến trúc.[297]
  - Sử dụng LLM kết hợp với các chỉ số mã nguồn (ví dụ: độ phức tạp cyclomatic từ Radon [118], khớp nối, tính gắn kết [310]) và các cảnh báo phân tích tĩnh [311] để dự đoán điểm rủi ro.[178]
  - Tạo các đề xuất tái cấu trúc, cải thiện khả năng bảo trì hoặc giải quyết các rủi ro đã xác định.

Phần này là trung tâm của thiết kế kỹ thuật. Kiến trúc đa tác tử cho phép chuyên môn hóa và khả năng mở rộng. Cách tiếp cận AST+LLM/RAG kết hợp là trọng tâm để đạt được sự hiểu biết sâu sắc về mã nguồn. Việc tạo sơ đồ bổ sung giá trị đáng kể cho việc trực quan hóa tác động. Quét toàn bộ dự án nâng tầm công cụ vượt ra ngoài các đánh giá PR đơn giản. Yêu cầu của người dùng về "chuyên sâu" và "hiểu biết sâu" đòi hỏi phải vượt ra ngoài việc kiểm tra mã ở mức bề mặt. AST cung cấp xương sống cấu trúc. LLM cung cấp sự hiểu biết ngữ nghĩa. RAG cung cấp ngữ cảnh. LangGraph điều phối các tương tác phức tạp này. Tính năng sơ đồ hóa đòi hỏi sự cân nhắc kỹ lưỡng về cách trích xuất dữ liệu liên quan và biểu diễn nó một cách có ý nghĩa, đặc biệt là làm nổi bật các thay đổi. Quét toàn bộ dự án là tính năng phức tạp nhất do kích thước ngữ cảnh và nhu cầu phân tích toàn diện. Việc "không ảnh hưởng đến dự án/CI/CD" có nghĩa là hệ thống này là một công cụ phân tích và báo cáo ngoại tuyến, điều này ảnh hưởng đến cách dữ liệu được tìm nạp và báo cáo được phân phối.

4. Ngăn xếp Công nghệ và Lựa chọn Thành phần Mã Nguồn Mở

4.1. Framework Cốt lõi:

- **Langchain/LangGraph (Python):** Dùng để điều phối đa tác tử.[23] LangGraph được ưu tiên vì khả năng quản lý trạng thái rõ ràng và khả năng đồ thị theo chu kỳ phù hợp với các tương tác tác tử phức tạp.

4.2. Lựa chọn LLM:

- **Mô hình Mã nguồn Mở:**
  - Ưu tiên các mô hình có khả năng mã hóa mạnh mẽ và cửa sổ ngữ cảnh lớn có sẵn vào tháng 5 năm 2025.[17] Ví dụ: DeepSeek Coder V2 [74], dòng CodeLlama [71], StarCoder2 [73], Qwen2 [19], Mistral-Large [19], Phi-4 [19], Llama 3.1/3.3.[17]
  - Cân nhắc: Giấy phép (đảm bảo tính permisive cho việc sử dụng nội bộ và các sửa đổi tiềm năng trong tương lai), hiệu suất, dễ dàng tinh chỉnh/lưu trữ.
  - Framework để tinh chỉnh: Hugging Face TRL [321], Llama Factory [323], Axolotl.[323]
  - Cách tiếp cận của Kodus.io về việc sử dụng AST để cung cấp ngữ cảnh có cấu trúc cho các mô hình GPT [6] có thể được điều chỉnh cho các LLM mã nguồn mở.
- **API LLM Thương mại:**
  - OpenAI (GPT-4, GPT-4o [124], hoặc mới hơn vào tháng 5 năm 2025).
  - Google Gemini (Gemini 1.5 Pro với cửa sổ ngữ cảnh 1M-2M token [18], Gemini Flash [125]).
  - Anthropic Claude (Claude 3.5 Sonnet với cửa sổ ngữ cảnh 200K token [126]).
- **Tối ưu hóa Chi phí LLM:** Các chiến lược như tối ưu hóa prompt, bộ nhớ đệm phản hồi và sử dụng các mô hình nhỏ hơn, dành riêng cho tác vụ khi thích hợp sẽ rất quan trọng, đặc biệt nếu sử dụng API thương mại để phân tích quy mô lớn.[326]

4.3. Phân tích Cú pháp AST:

- **Tree-sitter:** Thư viện phân tích cú pháp cốt lõi.[42]
- **Ngữ pháp Ngôn ngữ:** Cần tìm nguồn hoặc phát triển ngữ pháp Tree-sitter cho Python, Java, Kotlin. Đối với XML của Android (AndroidManifest.xml, tệp XML layout), có thể sử dụng ngữ pháp XML chung như tree-sitter-xml [98], có thể với các truy vấn tùy chỉnh cho các yếu tố dành riêng cho Android.[101]
- Cần xem xét hiệu suất của Tree-sitter trên các tệp/dự án lớn.[102] Phân tích cú pháp tăng dần là một lợi ích chính.[102]

4.4. Thư viện/Công cụ Tạo Sơ đồ (cho việc tạo):

- **PlantUML:** Công cụ tạo sơ đồ UML dựa trên văn bản. Có thể tạo sơ đồ lớp và sơ

đồ tuần tự. Đầu ra có thể là các tệp .puml văn bản.[49]

- **Mermaid.js:** Công cụ tạo sơ đồ và biểu đồ dựa trên JavaScript, nhận các định nghĩa văn bản lấy cảm hứng từ Markdown và tạo sơ đồ. Phù hợp để nhúng vào báo cáo HTML/Markdown.[54]
- **Graphviz (ngôn ngữ DOT):** Bộ máy cơ bản cho nhiều công cụ tạo sơ đồ, bao gồm PlantUML. Có thể được sử dụng trực tiếp nếu cần kiểm soát chi tiết.[255]
- **Thư viện JavaScript (cho giao diện người dùng tương tác tiềm năng):** Cytoscape.js [336] để trực quan hóa đồ thị tương tác nếu xem xét một trình xem báo cáo dựa trên web.

4.5. Tương tác Kiểm soát Phiên bản:

- **Python:** GitPython.[37]
- **Java:** JGit.[39]
- **Go (nếu các dịch vụ backend bằng Go):** go-git.[35]
- **Node.js (nếu các dịch vụ backend bằng Node.js):** isomorphic-git [41] hoặc git-client.[343]

4.6. Lưu trữ Dữ liệu (Ngữ cảnh RAG, Dữ liệu Tạm thời, Checkpoint):

- **Cơ sở Dữ liệu Vector:** Dùng cho RAG embeddings. Các lựa chọn mã nguồn mở như Weaviate, Qdrant, Milvus, hoặc PostgreSQL với pgvector.[92] LlamaIndex và LangChain hỗ trợ nhiều kho vector khác nhau.[230]
- **Cơ sở Dữ liệu Quan hệ (cho siêu dữ liệu có cấu trúc, trạng thái tác tử, checkpoint):** PostgreSQL [344] hoặc SQLite cho sự đơn giản nếu quy mô cho phép. Checkpointing của LangGraph có thể sử dụng Postgres, SQLite, Redis, v.v..[24]
- **Cơ sở Dữ liệu NoSQL (tùy chọn, cho các lược đồ dữ liệu linh hoạt):** MongoDB [346], Cassandra [346] nếu có nhu cầu cụ thể, nhưng có thể không cần thiết cho hệ thống ban đầu.

4.7. Backend và Frontend (cho Tương tác Người dùng và Xem Báo cáo):

- **Framework Backend (nếu cần API dịch vụ cho các lần quét do người dùng khởi tạo):**
  - Python: FastAPI [348], Flask.[349]
  - Java: Spring Boot.[349]
  - Node.js: NestJS, Fastify, Express.[350]
  - Go: Gin, Beego, Echo.[352]
- **Framework Frontend (để hiển thị báo cáo nếu nhiều hơn tệp tĩnh):**
  - React, Vue, Angular, Svelte.[354]
  - Để render sơ đồ: Các thư viện tương thích với đầu ra PlantUML/Mermaid.js, hoặc render trực tiếp bằng Cytoscape.js.

- Trình xem Diff: react-diff-viewer [355], jsdiff.[177] Các thư viện tô sáng cú pháp như Prism.js [356] hoặc Highlight.js [220] có thể được tích hợp.

4.8. Tích hợp Công cụ Phân tích Tĩnh (Lấy cảm hứng/Hybrid hóa):

- Trong khi xây dựng các quy tắc dựa trên AST tùy chỉnh, hãy xem xét các mẫu từ các công cụ hiện có như SonarQube/SonarLint [120], PMD [360], Checkstyle [112], ESLint [113], Ktlint [114], Bandit.[115]
- Automated Security Helper (ASH) [361] cung cấp một ví dụ về việc điều phối nhiều máy quét mã nguồn mở.

Ngăn xếp công nghệ phải được lựa chọn cẩn thận để cân bằng giữa hiệu suất, khả năng mở rộng, dễ phát triển và khả năng tương thích giấy phép mã nguồn mở. Việc lựa chọn LLM (cục bộ so với API) sẽ ảnh hưởng đáng kể đến chi phí và quyền riêng tư dữ liệu. Tree-sitter là một lựa chọn mạnh mẽ để phân tích cú pháp do hỗ trợ đa ngôn ngữ và hiệu suất của nó. LangGraph cung cấp sức mạnh cần thiết cho các quy trình làm việc của tác tử. Yêu cầu của người dùng về "tránh phát minh lại bánh xe" và "tối đa hóa mã nguồn mở" có nghĩa là đối với mỗi thành phần, chúng ta phải đánh giá các tùy chọn mã nguồn mở trưởng thành. Các ưu tiên về ngôn ngữ (Python, Java, Kotlin, Android) sẽ quyết định bộ ngữ pháp Tree-sitter ban đầu và có khả năng là lựa chọn ngôn ngữ cho một số thành phần backend nếu các thư viện phân tích dành riêng cho ngôn ngữ được sử dụng nhiều (mặc dù một hệ thống tác tử đa ngôn ngữ là khả thi).

Bảng 1: So sánh các LLM Mã nguồn Mở được chọn cho Phân tích Mã nguồn (Tính đến tháng 5 năm 2025)

Tên Model	Nhà phát triển	Ngày phát hành (ước tính)	Tham số	Cửa sổ Ngữ cảnh (tokens)	Tập trung Dữ liệu Đào tạo Chính	Ngôn ngữ Hỗ trợ (từ model card)	Chỉ số Hiệu suất Chính (ví dụ: HumanEval)	Giấy phép
DeepSeek Coder V2	DeepSeek AI	~Q1 2025	>67B	>128K	Mã nguồn, Lý luận	Đa ngôn ngữ	Cao trên các benchmark mã hóa	Permissive
Llama	Meta	~Q2	70B -	128K	Đa	Đa	Cải	Llama

3.1 / 3.3 (70B+)	AI	2024 - Q1 2025	405B		dạng, bao gồm mã nguồn	ngôn ngữ (mở rộng)	thiện so với Llama 2 trên mã hóa	License
StarCoder2 (15B+)	BigCode	~Q4 2024 - Q1 2025	>15B	>16K	Mã nguồn (The Stack v2)	Đa ngôn ngữ	Mạnh về hoàn thành và tạo mã	Apache 2.0
Qwen2.5 (72B Instruct)	Alibaba	~Q4 2024 - Q1 2025	72B	128K	Đa ngôn ngữ, mã hóa, toán học	>29 ngôn ngữ	Vượt trội về mã hóa, toán học	Permissive
Mistral-Large-Instruct-2407	Mistral AI	~Q3 2024	123B	131K	Lý luận, mã hóa, đa ngôn ngữ	>80 ngôn ngữ	Hiệu suất cao về lý luận, mã hóa	Apache 2.0
Phi-4 (nếu có)	Microsoft	~Q1-Q2 2025	>3.8B	>128K (dự kiến)	Dữ liệu chất lượng cao, mã hóa	Đa ngôn ngữ	Mạnh về lập trình, lý luận	MIT
Gemma-2 (9B IT)	Google	~Q4 2024 - Q1 2025	9B	>8K (có thể mở rộng)	Lý luận, tóm tắt, Q&A	Đa ngôn ngữ	Tối ưu cho thiết bị hạn chế tài nguyên	Gemma License

Lưu ý: Thông tin về các mô hình và ngày phát hành là ước tính dựa trên các xu hướng hiện tại và các thông báo đã biết tính đến thời điểm nghiên cứu cho tháng 5 năm 2025. Các chi tiết cụ thể có thể thay đổi. Giấy phép "Permissive" thường ám chỉ các

giấy phép như MIT, Apache 2.0.

5. Thiết kế Tính năng Chi tiết và Chiến lược Triển khai

5.1. Quy trình Quét PR/Dự án do Người dùng Khởi tạo:

- **Cơ chế Đầu vào:** Một công cụ dòng lệnh (CLI) hoặc một giao diện người dùng web đơn giản nơi người dùng có thể gửi URL kho Git và số PR (để quét PR) hoặc chỉ định một nhánh/thẻ để quét toàn bộ dự án.
- **UserInteractionAgent**: Nhận yêu cầu, xác thực đầu vào.
- **CodeFetcherAgent**: Sao chép kho lưu trữ (hoặc tìm nạp dữ liệu PR cụ thể). Đối với PR, nó tìm nạp diff giữa nhánh PR và nhánh đích. Đối với các PR đã đóng, nó tìm nạp trạng thái tại thời điểm hợp nhất/đóng.
- **Điều phối:** UserInteractionAgent chuyển thông tin mã/diff đã tìm nạp đến LLMOrchestratorAgent và các tác tử phân tích liên quan khác.

5.2. Giải quyết Lỗi dựa trên LLM:

- **StaticAnalysisAgent & LLMOrchestratorAgent**: Sau khi phân tích tĩnh ban đầu, các vấn đề tiềm ẩn được chuyển đến LLM để hiểu sâu hơn và tạo giải pháp.
- **Kỹ thuật Prompt:**
  - Cung cấp cho LLM đoạn mã, ngữ cảnh AST (ví dụ: hàm/lớp cha, các import liên quan [81]), vấn đề được phát hiện (từ phân tích tĩnh hoặc lượt LLM ban đầu) và hướng dẫn cụ thể để tạo giải pháp.
  - Cấu trúc prompt ví dụ: "Lỗi: [Loại lỗi] trong. Mã: [Đoạn mã]. Ngữ cảnh AST:. Ngữ cảnh Dự án (từ RAG): [Các mẫu/tài liệu dự án liên quan]. Đề xuất một bản sửa lỗi kèm theo giải thích và mã."
  - Sử dụng các kỹ thuật như Chain-of-Thought [95] và Few-Shot prompting [95] với các ví dụ về các bản sửa lỗi tốt.
- **RAG cho các Mẫu Giải pháp:** RAGContextAgent truy xuất các ví dụ về các lỗi tương tự đã được sửa từ lịch sử của dự án (nếu có và được lập chỉ mục) hoặc từ một cơ sở kiến thức chung về các phương pháp mã hóa tốt nhất và các bản vá lỗ hổng. Điều này giúp LLM tạo ra các giải pháp hiệu quả và phù hợp hơn với phong cách của dự án.[80]
- **SolutionSuggestionAgent**: Nhận các đề xuất giải pháp thô từ LLM và tinh chỉnh chúng để rõ ràng, dễ thực hiện và súc tích. Có thể liên quan đến việc yêu cầu LLM diễn đạt lại hoặc giải thích thêm.

5.3. Tạo và Đánh dấu Sơ đồ:

- **ASTParsingAgent & ImpactAnalysisAgent**:
  - Đối với **Sơ đồ Lớp**: Duyệt AST của các tệp liên quan (các tệp đã sửa đổi trong

PR và các phụ thuộc trực tiếp của chúng) để xác định định nghĩa lớp, trường, phương thức và mối quan hệ (thừa kế, tổng hợp, liên kết). Đối với Android, phân tích AndroidManifest.xml để tìm các thành phần [43] và các tệp XML layout để hiểu cấu trúc UI.[101]
    - Đối với **Sơ đồ Tuần tự**: Đối với một PR, xác định các hàm bị sửa đổi/thêm vào. Theo dõi các lệnh gọi đi từ các hàm này và các lệnh gọi đến chúng trong một độ sâu hợp lý, sử dụng phân tích AST và có thể cả RAG để giải quyết các lệnh gọi động hoặc triển khai giao diện. Phạm vi sẽ được giới hạn trong tác động trực tiếp của PR. Có thể cần sự hỗ trợ của LLM để suy ra các chuỗi tương tác từ mã phức tạp hoặc mô tả bằng ngôn ngữ tự nhiên trong thông điệp commit/mô tả PR.[251]
- **DiagramGenerationAgent**:
    - Chuyển đổi dữ liệu cấu trúc và tương tác đã trích xuất thành cú pháp PlantUML [49] hoặc Mermaid.js.[54]
    - **Đánh dấu Thay đổi:** So sánh AST trước và sau thay đổi trong một PR. Đánh dấu các lớp/phương thức/lệnh gọi được thêm/sửa đổi/xóa trong cú pháp sơ đồ (ví dụ: màu khác nhau, stereotype <<added>>, <<modified>>).
    - **Các Thành phần bị Ảnh hưởng:** Sử dụng thông tin phụ thuộc để đánh dấu các thành phần bị ảnh hưởng trực tiếp hoặc gián tiếp bởi các thay đổi của PR.

5.4. Quét và Phân tích Toàn bộ Dự án:

- **ProjectScanningAgent**:
    - **Quản lý Ngữ cảnh cho Codebase Lớn:**
        - Sử dụng tóm tắt phân cấp: ASTParsingAgent tạo tóm tắt cho các hàm/lớp, sau đó là tệp, thư mục, cho đến cấp độ dự án. Những tóm tắt này được cung cấp cho LLMOrchestratorAgent để hiểu ở cấp độ cao.[59]
        - Sử dụng các kỹ thuật RAG nâng cao: Xây dựng một chỉ mục vector toàn diện của toàn bộ codebase (được chia nhỏ một cách thông minh bằng AST [46]) và tài liệu dự án. Sử dụng các kỹ thuật như viết lại truy vấn, xếp hạng lại và tìm kiếm kết hợp [65] để truy xuất ngữ cảnh có mục tiêu. LlamaIndex cung cấp các tính năng để truy xuất có cấu trúc và tách rời các chunk truy xuất khỏi các chunk tổng hợp.[224]
    - **Mô hình Dự đoán Rủi ro:**
        - Kết hợp các chỉ số phân tích tĩnh (ví dụ: độ phức tạp cyclomatic từ Radon [118], các "code smell" từ định nghĩa của Martin Fowler [366], các chỉ báo nợ kỹ thuật [2]) với sự hiểu biết ngữ nghĩa từ LLM.
        - Huấn luyện một mô hình nhỏ hơn, chuyên biệt (hoặc sử dụng LLM prompting) để dự đoán điểm rủi ro dựa trên các đặc trưng kết hợp này.[317]
        - Xác định các điểm nóng về lỗi dựa trên tần suất thay đổi (từ lịch sử Git) và

các chỉ số độ phức tạp.
    - **Tạo Đề xuất:** Tạo các đề xuất kiến trúc cấp cao, cơ hội tái cấu trúc và chiến lược giảm nợ kỹ thuật dựa trên cái nhìn toàn diện về dự án.

5.5. Chiến lược Hỗ trợ Đa ngôn ngữ:

- **Giai đoạn 1 (Ngôn ngữ Ưu tiên):** Python, Java, Kotlin (bao gồm các cấu trúc dành riêng cho Android đối với Java/Kotlin). Phát triển ngữ pháp Tree-sitter mạnh mẽ và các quy tắc phân tích AST cho các ngôn ngữ này trước tiên.
- **Giai đoạn 2 (Mở rộng):** Dần dần bổ sung hỗ trợ cho các ngôn ngữ khác dựa trên nhu cầu và tính sẵn có của ngữ pháp. Kiến trúc Tree-sitter tạo điều kiện thuận lợi cho việc thêm các ngôn ngữ mới bằng cách cung cấp các ngữ pháp mới.[42]
- Prompt LLM và ngữ cảnh RAG có thể cần điều chỉnh theo từng ngôn ngữ cụ thể.

Phần này trình bày chi tiết "cách thức" cho từng tính năng chính. Sự tương tác giữa các tác tử là rất quan trọng. Đối với việc giải quyết lỗi dựa trên LLM, chất lượng của ngữ cảnh RAG và kỹ thuật prompt sẽ quyết định sự thành công. Việc tạo sơ đồ đòi hỏi logic ánh xạ từ AST sang sơ đồ một cách mạnh mẽ. Quét toàn bộ dự án là tham vọng nhất và sẽ phụ thuộc nhiều vào các kỹ thuật quản lý ngữ cảnh và RAG nâng cao. Yêu cầu của người dùng về "giải pháp khả thi" có nghĩa là SolutionSuggestionAgent rất quan trọng. "Làm nổi bật các thay đổi và các thành phần bị ảnh hưởng" đối với sơ đồ có nghĩa là ImpactAnalysisAgent và DiagramGenerationAgent phải phối hợp chặt chẽ. "Hiểu biết sâu về cấu trúc dự án" đối với các lần quét toàn bộ chỉ ra sự cần thiết của việc xây dựng ngữ cảnh phức tạp bởi RAGContextAgent và ProjectScanningAgent.

6. Quản lý Dữ liệu, Quyền riêng tư và Bảo mật

- **Xử lý Mã nguồn Độc quyền:**
    - Vì hệ thống được tự lưu trữ (self-hosted), mã nguồn độc quyền mặc định vẫn nằm trong cơ sở hạ tầng của người dùng.[6]
    - Nếu sử dụng API LLM thương mại (OpenAI, Gemini), cần xem xét các chiến lược giảm thiểu dữ liệu hoặc ẩn danh hóa (nếu có thể mà không làm mất ngữ cảnh). Tuy nhiên, để phân tích sâu, việc gửi các đoạn mã thường là cần thiết. Sự đồng ý rõ ràng của người dùng và nhận thức về việc truyền dữ liệu cho LLM của bên thứ ba là điều cần thiết.
    - Ưu tiên sử dụng các LLM mã nguồn mở được lưu trữ cục bộ để đảm bảo quyền riêng tư dữ liệu tối đa, đặc biệt đối với các lần quét toàn bộ dự án.[370]
- **Cách ly Dữ liệu với Cloud LLM:**
    - Nếu sử dụng Cloud LLM, đảm bảo các lệnh gọi API được bảo mật (HTTPS).
    - Xem xét các điều khoản của nhà cung cấp về việc lưu giữ dữ liệu và sử dụng cho việc huấn luyện mô hình. Nhiều nhà cung cấp hiện cung cấp các tùy chọn

không lưu giữ dữ liệu cho các API doanh nghiệp.

- **Bảo mật của Hệ thống Tự Lưu trữ:**
  - Bảo mật máy chủ lưu trữ hệ thống đánh giá (các biện pháp tăng cường bảo mật máy chủ tiêu chuẩn).
  - Quản lý khóa API cho các nhà cung cấp LLM một cách an toàn (ví dụ: sử dụng biến môi trường, công cụ quản lý bí mật).
  - Triển khai kiểm soát truy cập nếu hệ thống có giao diện web hoặc API để khởi tạo quét.
- **Bảo mật Thành phần Mã nguồn Mở:**
  - Thường xuyên quét các phụ thuộc mã nguồn mở để tìm các lỗ hổng đã biết bằng các công cụ như Snyk, OWASP Dependency-Check.[371]

Quyền riêng tư dữ liệu là tối quan trọng đối với một công cụ phân tích mã nguồn độc quyền. Khả năng của kiến trúc trong việc sử dụng LLM cục bộ là một lợi thế đáng kể. Cần có các chính sách rõ ràng và nhận thức của người dùng nếu các LLM đám mây thương mại là một tùy chọn. Yêu cầu của người dùng về một "hệ thống độc lập" phù hợp tốt với việc tự lưu trữ. Tùy chọn sử dụng các nhà cung cấp LLM bên ngoài tạo ra sự đánh đổi giữa khả năng/tiện lợi và quyền riêng tư dữ liệu. Phần này phải giải quyết sự đánh đổi này và đề xuất các phương pháp hay nhất.

7. Cân nhắc về Triển khai và Vận hành

- **Yêu cầu Tự Lưu trữ (Self-Hosting):**
  - **Phần cứng:** Đủ CPU, RAM cho các tác tử, phân tích AST và có thể cả suy luận LLM cục bộ (GPU sẽ cần thiết để lưu trữ LLM cục bộ hiệu quả, điều này có thể tốn kém [374]).
  - **Phần mềm:** Docker để đóng gói được khuyến nghị để triển khai và quản lý phụ thuộc dễ dàng hơn (Kodus cung cấp tệp Docker [6]).
  - Thiết lập cơ sở dữ liệu cho RAG và trạng thái tác tử.
- **Khả năng Mở rộng và Hiệu suất:**
  - **Phân tích AST:** Tree-sitter thường nhanh, nhưng hiệu suất trên các tệp hoặc dự án rất lớn cần được theo dõi.[102]
  - **Suy luận LLM:** Các lệnh gọi API đến LLM thương mại sẽ có độ trễ. Tốc độ suy luận LLM cục bộ phụ thuộc nhiều vào phần cứng. Thực hiện batch request đến LLM khi có thể.
  - **Hệ thống RAG:** Tốc độ truy xuất từ cơ sở dữ liệu vector phụ thuộc vào kích thước và tối ưu hóa.[376]
  - **Xử lý Song song:** LangGraph có thể thực thi các tác vụ tác tử độc lập song song.[21] Thiết kế quy trình làm việc để tối đa hóa tính song song.
- **Giám sát và Bảo trì (MLOps cho LLM):**

- Ghi nhật ký tương tác của tác tử, đầu vào/đầu ra của LLM và lỗi.
- Theo dõi hiệu suất LLM (độ chính xác của đề xuất, tỷ lệ ảo giác) và chi phí.[380]
- Triển khai các cơ chế để cập nhật LLM (nếu được lưu trữ cục bộ) hoặc thích ứng với các thay đổi API từ các nhà cung cấp thương mại.
- Thường xuyên cập nhật ngữ pháp Tree-sitter và các phụ thuộc mã nguồn mở khác.
- Thu thập phản hồi của người dùng về chất lượng đề xuất để tinh chỉnh prompt hoặc ngữ cảnh RAG (con người trong vòng lặp để cải tiến liên tục [220]).

Việc vận hành một hệ thống dựa trên LLM đòi hỏi các thực tiễn MLOps. Hiệu suất, đặc biệt đối với việc quét toàn bộ dự án, sẽ là một thách thức chính. Chi phí của các lệnh gọi API LLM hoặc cơ sở hạ tầng GPU cục bộ phải được tính đến. Hệ thống "độc lập" và "tự lưu trữ". Điều này ngụ ý người dùng quản lý việc triển khai. Do đó, phần này phải bao gồm các khía cạnh thực tế của việc vận hành một hệ thống như vậy, bao gồm ước tính tài nguyên và chiến lược bảo trì. Hiệu suất của Tree-sitter trên các tệp lớn [330] và giới hạn ngữ cảnh LLM [77] là những yếu tố quan trọng đối với khả năng mở rộng.

8. Lộ trình và Các Cột mốc Dự án

Một cách tiếp cận theo từng giai đoạn là rất quan trọng để quản lý sự phức tạp. Các giai đoạn đầu nên tập trung vào chức năng cốt lõi cho một ngôn ngữ duy nhất để xác thực kiến trúc. Các vòng phản hồi của người dùng là điều cần thiết cho sự thành công lâu dài và cải tiến mô hình. Lộ trình cần phải thực tế. Xây dựng một hệ thống đầy đủ tính năng như mô tả là một công việc quan trọng. Ưu tiên các ngôn ngữ và tính năng cốt lõi trước, sau đó mở rộng, là một chiến lược hợp lý. Sự liên quan của "Tháng 5 năm 2025" có nghĩa là các giai đoạn ban đầu nên nhắm mục tiêu vào các công nghệ và LLM là SOTA hoặc mới nổi vào thời điểm đó.

Bảng 2: Lộ trình Dự án - Các Giai đoạn Chính, Cột mốc và Tính năng

Giai đoạn	Mục tiêu Giai đoạn	Các Cột mốc/Sản phẩm Chính	Tính năng Mục tiêu được Triển khai	Thời gian Ước tính	Ngôn ngữ Chính được Bao phủ
1	Proof-of-Concept Bộ máy Cốt lõi & Ngôn ngữ Đơn lẻ	Thiết lập framework LangGraph. CodeFetcherAgent & ASTParsingA	Quét PR Python cơ bản. Tóm tắt diff. Phát hiện lỗi đơn giản. Báo	3-4 tháng	Python

		gent cho Python. StaticAnalysisAgent cơ bản (vài quy tắc Tree-sitter). Tích hợp 1 LLM OSS. ReportingAgent cơ bản.	cáo Markdown.		
2	Phân tích Nâng cao & Tạo Sơ đồ	Mở rộng bộ quy tắc StaticAnalysisAgent cho Python. RAGContextAgent cho Python. SolutionSuggestionAgent cho Python. DiagramGenerationAgent cho sơ đồ lớp Python. Hỗ trợ Java cơ bản. Tùy chọn API OpenAI/Gemini.	Đề xuất giải pháp khả thi. Sơ đồ lớp cơ bản cho Python & Java PRs.	4-5 tháng	Python, Java
3	Quét Toàn bộ Dự án & Tính năng Nâng cao	ProjectScanningAgent cho Python & Java. Tóm tắt phân cấp/RAG nâng cao. Mô hình dự đoán rủi ro ban đầu. DiagramGenerationAgent	Quét toàn bộ dự án. Thông tin chi tiết về kiến trúc. Hỗ trợ Kotlin/Android.	5-6 tháng	Python, Java, Kotlin, Android

		cho sơ đồ tuần tự. Hỗ trợ Kotlin & phân tích Android cơ bản.			
4	Mở rộng Đa ngôn ngữ, Tối ưu hóa & Phản hồi Người dùng	Mở rộng sang các ngôn ngữ khác. Tối ưu hóa hiệu suất. Cơ chế phản hồi người dùng (RLHF/tinh chỉnh prompt). Cập nhật LLM, ngữ pháp. Kỹ thuật XAI nâng cao.	Hệ thống trưởng thành, đa ngôn ngữ, liên tục cải tiến. Giải thích đề xuất LLM tốt hơn.	Liên tục	Nhiều ngôn ngữ khác

9. Kết luận và Các Cải tiến trong Tương lai

9.1. Tóm tắt Khả năng và Lợi ích của Hệ thống:
Hệ thống được đề xuất giải quyết các yêu cầu của người dùng về một công cụ đánh giá mã nguồn chuyên sâu, dựa trên LLM. Các lợi ích chính bao gồm cải thiện chất lượng mã nguồn, đánh giá nhanh hơn (bằng cách cungPEP báo cáo toàn diện cho người đánh giá là con người), hiểu biết tốt hơn về kiến trúc và xác định rủi ro chủ động. Hệ thống này, với kiến trúc đa tác tử và khả năng phân tích kết hợp AST và LLM/RAG, đại diện cho một bước tiến đáng kể so với các công cụ đánh giá mã nguồn truyền thống.
9.2. Các Hướng Phát triển Tiềm năng trong Tương lai:
Mặc dù hệ thống hiện tại được thiết kế để hoạt động độc lập, các cải tiến trong tương lai có thể tập trung vào việc tích hợp sâu hơn và hỗ trợ chủ động hơn, luôn tuân thủ nguyên tắc "con người trong vòng lặp" cho các quyết định phức tạp.220

- **Tích hợp CI/CD sâu hơn (nếu yêu cầu thay đổi):** Cho phép tự động bình luận trên PR trong các nền tảng như GitHub [219], GitLab [393], Bitbucket [309], hoặc Gerrit.[397]
- **Đề xuất Tái cấu trúc dựa trên AI:** Ngoài việc xác định vấn đề, đề xuất và có khả năng tự động hóa các tác vụ tái cấu trúc.
- **Phân tích Bảo mật Nâng cao:** Kết hợp các kỹ thuật phân tích bảo mật phức tạp hơn, có thể tích hợp với các công cụ SAST chuyên dụng hoặc sử dụng LLM được tinh chỉnh để phát hiện lỗ hổng.[402]

- **Tích hợp IDE thông qua LSP:** Phát triển giao diện Giao thức Máy chủ Ngôn ngữ (LSP) [421] để cung cấp phản hồi thời gian thực trong các IDE.

Kết luận nên nhắc lại giá trị của hệ thống và tiềm năng phát triển trong tương lai, phù hợp với bối cảnh AI và phát triển phần mềm đang phát triển.

10. Phụ lục

A.1. Đặc tả API cho Giao tiếp giữa các Tác tử:

- Xác định các cấu trúc dữ liệu (ví dụ: sử dụng mô hình Pydantic trong Python) cho các thông điệp được truyền giữa các tác tử (ngữ cảnh mã, AST, kết quả LLM, dữ liệu sơ đồ).

A.2. Cân nhắc về Giấy phép Mã nguồn Mở:

- Phân tích chi tiết về AGPL-3.0 [397] và các tác động của nó nếu các công cụ như Kodus [207] được tích hợp trực tiếp hoặc sửa đổi nhiều. Kodus sử dụng giấy phép AGPL-3.0 cho các phần không thuộc phiên bản doanh nghiệp (EE).[207] Các phần EE của Kodus được cấp phép thương mại riêng.[6]
- Xem xét giấy phép của tất cả các thành phần mã nguồn mở được chọn (ngữ pháp Tree-sitter (thường là MIT/Apache), mô hình LLM (Apache 2.0 phổ biến cho các mô hình mở như Llama, Mistral [19]), Langchain (MIT), thư viện tạo sơ đồ, v.v.).
- Hướng dẫn về cách cấu trúc dự án để duy trì sự tuân thủ, đặc biệt nếu các phần của hệ thống có thể được cung cấp dưới dạng dịch vụ hoặc được phân phối.

A.3. Chỉ số Đánh giá:

- **Chất lượng Phân tích Mã:** Precision, Recall, F1-score để phát hiện lỗi; BLEU, ROUGE cho chất lượng đề xuất (so với các bản sửa lỗi của chuyên gia con người).[81]
- **Độ chính xác của Sơ đồ:** Các chỉ số để đánh giá tính đúng đắn của các sơ đồ lớp/tuần tự được tạo ra (ví dụ: sự tương đồng về cấu trúc với các sơ đồ được tạo thủ công, tính đầy đủ).
- **Độ chính xác Dự đoán Rủi ro:** Các chỉ số để đánh giá các dự đoán rủi ro khi quét toàn bộ dự án (ví dụ: mối tương quan với tỷ lệ lỗi thực tế).
- **Hiệu suất Hệ thống:** Độ trễ cho các lần quét PR, thông lượng cho các lần quét toàn bộ dự án, việc sử dụng tài nguyên.
- **Sự hài lòng của Người dùng (nếu giao diện người dùng để tương tác/báo cáo được phát triển):** Phản hồi định tính, tỷ lệ hoàn thành tác vụ.
- Theo dõi sự trôi dạt của LLM và chất lượng đề xuất theo thời gian.[380]

A.4. Prompt Chi tiết cho các Tương tác LLM Chính:

- Ví dụ về các prompt để phát hiện lỗi, tạo giải pháp, tóm tắt mã cho RAG, trích xuất yếu tố sơ đồ và tóm tắt kiến trúc.

Phụ lục cung cấp các chi tiết hỗ trợ quan trọng. Việc cấp phép là một khía cạnh pháp lý không hề đơn giản. Các chỉ số đánh giá là chìa khóa để đo lường sự thành công và hướng dẫn các cải tiến. Các prompt ví dụ cung cấp điểm khởi đầu cụ thể cho việc triển khai.

Bảng 3: Tóm tắt Ngăn xếp Công nghệ Cốt lõi

Loại Thành phần	**Công cụ/Thư viện Mã nguồn Mở được chọn**	**Tính năng Chính Liên quan đến Dự án**	**Giấy phép Chính**	**Lý do Lựa chọn**
Điều phối Tác tử	LangGraph (Python)	Quản lý trạng thái, đồ thị có chu trình, xử lý song song, tích hợp LangChain	MIT	Phù hợp cho các quy trình phức tạp, đa tác tử, có trạng thái.
LLM (Mã nguồn mở)	DeepSeek Coder V2, Llama 3.1, StarCoder2, Qwen2.5 (tùy chọn)	Khả năng mã hóa mạnh, cửa sổ ngữ cảnh lớn, hỗ trợ đa ngôn ngữ	Apache 2.0, Llama License, etc.	Hiệu suất cao trên các tác vụ mã hóa, cộng đồng hỗ trợ mạnh, giấy phép permisive.
LLM (API Thương mại)	OpenAI GPT-4o/GPT-X, Google Gemini 2.5 Pro, Anthropic Claude 3.5 Sonnet	Khả năng lý luận tiên tiến, cửa sổ ngữ cảnh rất lớn, cập nhật thường xuyên	Thương mại	Hiệu suất SOTA, dễ tích hợp API.
Phân tích Cú pháp AST	Tree-sitter	Nhanh, mạnh mẽ, phục hồi lỗi, hỗ trợ đa ngôn ngữ, phân tích cú pháp tăng	MIT	Tiêu chuẩn công nghiệp cho phân tích cú pháp dựa trên AST, nhiều ngữ

		dần		pháp có sẵn.
Tạo Sơ đồ	PlantUML (đầu ra dạng text)	Hỗ trợ UML (lớp, tuần tự), định dạng văn bản dễ tạo tự động	GPL	Phổ biến, hỗ trợ nhiều loại sơ đồ, dễ tích hợp.
	Mermaid.js (đầu ra dạng text)	Hỗ trợ nhiều loại sơ đồ, cú pháp giống Markdown, dễ nhúng	MIT	Linh hoạt, dễ sử dụng cho báo cáo web/Markdown.
Tương tác VCS	GitPython (Python), JGit (Java), go-git (Go)	Tương tác với kho chứa Git để lấy mã nguồn, diff	BSD, EDL, Apache 2.0	Thư viện trưởng thành, phù hợp với ngôn ngữ của các tác tử tương ứng.
Cơ sở dữ liệu Vector (RAG)	Weaviate / Qdrant / Milvus / PostgreSQL+pgvector	Lưu trữ và truy xuất embedding hiệu quả, hỗ trợ tìm kiếm tương đồng	Apache 2.0, etc.	Các lựa chọn mã nguồn mở mạnh mẽ, có thể mở rộng, được hỗ trợ bởi LlamaIndex/LangChain.
Lưu trữ Trạng thái/Checkpoint	PostgreSQL / SQLite / Redis	Lưu trữ trạng thái tác tử, checkpoint cho các tác vụ dài	PostgreSQL License, Public Domain, BSD	Các lựa chọn đáng tin cậy, được hỗ trợ bởi LangGraph.
Backend (nếu có API)	FastAPI (Python)	Hiệu suất cao, dễ phát triển API hiện đại	MIT	Phù hợp cho các dịch vụ dựa trên Python của LangGraph.
Frontend (nếu có UI báo cáo)	React / Vue.js	Framework hiện đại, hệ sinh thái lớn	MIT	Linh hoạt cho việc xây dựng giao diện xem báo cáo tương tác.

Yêu cầu của người dùng về việc "tối đa hóa mã nguồn mở" và "tránh phát minh lại bánh xe" có nghĩa là đối với mỗi thành phần, chúng ta phải đánh giá các tùy chọn mã nguồn mở trưởng thành. Việc lựa chọn công nghệ phải cân nhắc kỹ lưỡng các yếu tố này.

Bảng 4: Phân rã Trách nhiệm Tác tử và Tương tác trong LangGraph

Tên Tác tử	Trách nhiệm Chính	Đầu vào Chính	Đầu ra Chính	Công cụ/Công nghệ Cốt lõi	Tương tác Chính (Với các Tác tử khác)
UserInteractionAgent	Xử lý yêu cầu người dùng, xác thực đầu vào	Thông tin PR/dự án từ người dùng	Tham số quét đã xác thực	CLI/Web UI input parsing	CodeFetcherAgent
CodeFetcherAgent	Truy xuất mã nguồn, diff từ kho Git	URL kho chứa, ID PR, tên nhánh	Dữ liệu mã nguồn/diff	Thư viện Git (GitPython, JGit, etc.)	ASTParsingAgent, LLMOrchestratorAgent
ASTParsingAgent	Phân tích mã nguồn thành AST	Mã nguồn (chuỗi)	Cây AST, thông tin cấu trúc	Tree-sitter, ngữ pháp ngôn ngữ cụ thể	StaticAnalysisAgent, ImpactAnalysisAgent, DiagramGenerationAgent, RAGContextAgent
StaticAnalysisAgent	Thực thi quy tắc phân tích tĩnh trên AST	Cây AST, bộ quy tắc tùy chỉnh	Danh sách vi phạm quy tắc tiềm ẩn	Bộ máy quy tắc Tree-sitter tùy chỉnh (DSL)	LLMOrchestratorAgent (chuyển các phát hiện để phân tích sâu hơn)
LLMOrchestratorAgent	Điều phối tương tác với LLM, quản lý prompt, ngữ	Mã nguồn/diff, ngữ cảnh AST, kết quả	Phản hồi LLM (phân tích, đề xuất)	API LLM (OpenAI, Gemini), LLM mã nguồn	RAGContextAgent, SolutionSuggestionAgen

	cảnh	phân tích tĩnh, truy vấn RAG		mở	t, DiagramGenerationAgent, ProjectScanningAgent
RAGContextAgent	Xây dựng, truy vấn cơ sở kiến thức RAG	Mã nguồn dự án, tài liệu, truy vấn từ tác tử khác	Ngữ cảnh liên quan (đoạn mã, tài liệu)	LlamaIndex/LangChain, Vector DB, mô hình embedding	LLMOrchestratorAgent
DiagramGenerationAgent	Tạo dữ liệu sơ đồ từ phân tích cấu trúc và thay đổi	Thông tin cấu trúc (từ AST), phân tích tác động, dữ liệu LLM	Mô tả sơ đồ (PlantUML/Mermaid)	Thư viện tạo sơ đồ (PlantUML, Mermaid.js)	ReportingAgent
ImpactAnalysisAgent	Phân tích thay đổi mã, xác định thành phần bị ảnh hưởng	Dữ liệu diff, AST trước/sau	Danh sách thành phần bị ảnh hưởng, phạm vi thay đổi	Phân tích diff, duyệt AST	DiagramGenerationAgent, ReportingAgent
SolutionSuggestionAgent	Tinh chỉnh đề xuất giải pháp từ LLM	Phản hồi LLM thô	Đề xuất giải pháp rõ ràng, khả thi	Kỹ thuật prompt, LLM (để diễn giải lại)	ReportingAgent
ProjectScanningAgent	Điều phối quét toàn bộ dự án, quản lý ngữ cảnh lớn	Yêu cầu quét dự án	Báo cáo tình trạng dự án, dự đoán rủi ro, đề xuất kiến trúc	Tóm tắt phân cấp, RAG nâng cao, LLM, phân tích chỉ số mã nguồn	LLMOrchestratorAgent, ReportingAgent
ReportingAgent	Tổng hợp phát hiện, tạo báo cáo cuối cùng	Kết quả từ các tác tử phân tích, dữ liệu sơ đồ	Báo cáo đánh giá (Markdown, HTML, SARIF)	Mẫu báo cáo, thư viện định dạng	Giao diện người dùng/Người dùng cuối

Hệ thống được thiết kế theo kiến trúc đa tác tử. Bảng này làm rõ vai trò của từng tác tử, cách chúng khớp với framework LangGraph, và dữ liệu/công cụ chúng sử dụng, cung cấp một bản thiết kế rõ ràng cho kiến trúc tác tử. Điều này trực tiếp giải quyết yêu cầu "đa tác tử sử dụng Langchain/LangGraph".

Bảng 5: Tạo Sơ đồ Kiến trúc - Trích xuất và Ánh xạ Dữ liệu

Loại Sơ đồ	Cấu trúc Mã nguồn (Ví dụ)	Loại Node AST (Tree-sitter)	Thông tin Trích xuất	Ánh xạ Phần tử Sơ đồ (Ví dụ PlantUML)
Sơ đồ Lớp	Lớp Java public class User { private String name; public String getName() {} }	class_declaration, field_declaration, method_declaration, identifier, type_identifier	Tên lớp, tên thuộc tính, kiểu thuộc tính, tên phương thức, kiểu trả về, tham số, phạm vi truy cập (public, private)	class User { -name: String \n +getName(): String }
Sơ đồ Lớp	Hàm Python def calculate_total(items: list) -> float:	function_definition, identifier, parameters, typed_parameter, type	Tên hàm, tham số, kiểu tham số, kiểu trả về	(Có thể biểu diễn như một lớp tiện ích nếu cần trực quan hóa các hàm độc lập) class Utils { +calculate_total(items: list): float }
Sơ đồ Lớp	Quan hệ thừa kế Kotlin open class Vehicle, class Car : Vehicle()	class_declaration (với open), superclass	Tên lớp cha, tên lớp con	`Vehicle <
Sơ đồ Lớp	Thành phần Android trong AndroidManifest.xml <activity android:name=".MainActivity"/>	element_node, attribute_node (sử dụng ngữ pháp XML)	Tên thành phần (activity), thuộc tính (name)	component MainActivity <<Activity>>

Sơ đồ Tuần tự	Lệnh gọi phương thức service.process(data) trong một hàm của PR	call_expression, member_expression, identifier, arguments	Đối tượng gọi (service), phương thức được gọi (process), tham số (data), trình tự gọi	Caller -> Service : process(data)
Sơ đồ Tuần tự	Tạo đối tượng new UserRepo()	object_creation_expression, type_identifier	Lớp được khởi tạo	Caller -> UserRepo : <<create>>

Việc tạo sơ đồ là một tính năng chính. Bảng này sẽ trình bày chi tiết cách các cấu trúc mã cụ thể, được xác định thông qua AST, được ánh xạ tới các yếu tố trong ngôn ngữ sơ đồ đã chọn (ví dụ: PlantUML). Điều này cung cấp một kế hoạch cụ thể cho DiagramGenerationAgent.

Bảng 6: Các Kỹ thuật Prompting LLM cho Tác vụ Đánh giá Mã nguồn

Tác vụ	**Kỹ thuật Prompting**	**Thành phần Prompt Chính**	**Ví dụ Đoạn Prompt (Khái niệm)**
Phát hiện Lỗi	Few-shot, Chain-of-Thought, AST-Augmented	Đoạn mã có lỗi, thông báo lỗi (nếu có), đoạn AST liên quan, ví dụ về lỗi tương tự và cách sửa	"Đoạn mã Python sau có lỗi tiềm ẩn: <code>. Lỗi tương tự trong <code> đã được sửa thành <code>. Phân tích và chỉ ra lỗi trong đoạn mã đầu tiên, giải thích lý do."
Tạo Giải pháp	RAG-Augmented, Chain-of-Thought, AST-Augmented	Mã lỗi, thông báo lỗi, ngữ cảnh AST, ngữ cảnh RAG (mẫu sửa lỗi tương tự, tài liệu API)	"Lỗi NullPointerException xảy ra ở file:line trong đoạn mã Java: <code>. Ngữ cảnh AST: ``. Tài liệu liên quan (RAG): [docs]. Đề xuất giải pháp chi tiết và đoạn mã sửa lỗi."

Tóm tắt Mã (cho RAG/Hiểu biết)	Zero-shot, AST-Augmented	Toàn bộ hàm/lớp	"Tóm tắt chức năng chính, đầu vào, đầu ra và các phụ thuộc quan trọng của lớp Java sau: <code>. Thông tin AST: ``."
Trích xuất Phần tử Sơ đồ	Zero-shot, Role-playing	Đoạn mã, loại sơ đồ (lớp/tuần tự)	"Với vai trò là một công cụ phân tích kiến trúc, hãy xác định tất cả các lớp, thuộc tính, phương thức và mối quan hệ (thừa kế, sử dụng) trong đoạn mã Java sau để tạo sơ đồ lớp: <code>."
Phân tích Tác động Thay đổi (PR)	RAG-Augmented, AST-Augmented (diff)	Diff của PR, ngữ cảnh AST của các tệp bị thay đổi, ngữ cảnh RAG (thông tin phụ thuộc dự án)	"Phân tích diff sau: <code>. Các tệp bị ảnh hưởng có cấu trúc AST: ``. Dựa trên kiến thức về dự án (RAG): [dependency info], liệt kê các thành phần có thể bị ảnh hưởng bởi thay đổi này."

Tương tác LLM hiệu quả là rất quan trọng. Bảng này sẽ cung cấp các ví dụ và chiến lược cụ thể để tạo prompt cho các tác vụ phân tích khác nhau, kết hợp các phương pháp hay nhất như sử dụng ngữ cảnh bắt nguồn từ AST.[81] Điều này làm cho khía cạnh "dựa trên LLM" trở nên hữu hình hơn.

Hệ thống được thiết kế để hoạt động như một thực thể độc lập, tập trung vào việc tạo báo cáo chi tiết thay vì can thiệp trực tiếp vào quy trình làm việc của dự án hoặc CI/CD. Điều này đảm bảo rằng hệ thống không gây ảnh hưởng đến môi trường phát triển hiện tại và cung cấp một lớp phân tích bổ sung mà các nhà phát triển có thể chủ động sử dụng. Việc ưu tiên các giải pháp mã nguồn mở và tránh "phát minh lại bánh xe" là một nguyên tắc cốt lõi trong thiết kế này, nhằm tối đa hóa việc sử dụng các công nghệ đã được kiểm chứng và cộng đồng hỗ trợ mạnh mẽ.

Works cited

1. Exploring the Role of Large Language Models in Automated Code Review and Software Quality Enhancement - ijirset, accessed May 25, 2025, https://www.ijirset.com/upload/2023/september/4_Exploring.pdf
2. Are LLMs machines for generating technical debt? : r/brdev - Reddit, accessed May 25, 2025, https://www.reddit.com/r/brdev/comments/1j83o0j/llms_s%C3%A3o_m%C3%A1quinas_de_gerar_d%C3%A9bito_t%C3%A9cnico/?tl=en
3. Technical debt part 2: LLMs, AI and the new Frontier - Blar | Codebase-Aware Intelligence, accessed May 25, 2025, https://blar.io/blog/technical-debt-part-2-llms-ai-and-the-new-frontier
4. Combating Toxic Language: A Review of LLM-Based Strategies for Software Engineering, accessed May 25, 2025, https://arxiv.org/html/2504.15439v1
5. Identifying Technical Debt and Its Types Across Diverse Software Projects Issues - arXiv, accessed May 25, 2025, https://arxiv.org/html/2408.09128
6. kodustech/kodus-ai: Open source AI code reviews — just like your senior dev would do. - GitHub, accessed May 25, 2025, https://github.com/kodustech/kodus-ai
7. Kodus – An open source AI code review engine (AST and LLW, less noise), accessed May 25, 2025, https://dev.to/kodus/kodus-an-open-source-ai-code-review-engine-ast-and-llw-less-noise-3726
8. Top 10 Code Review Tools in 2024 - Swimm, accessed May 25, 2025, https://swimm.io/learn/software-development/top-10-code-review-tools-in-2024
9. What are the most important features for code review tools? - GitLab, accessed May 25, 2025, https://about.gitlab.com/topics/version-control/what-are-best-code-review-tools-features/
10. The Best 9 Code Review Tools for Developers in 2025 - Axify, accessed May 25, 2025, https://axify.io/blog/code-review-tools
11. [2503.12374] Unveiling Pitfalls: Understanding Why AI-driven Code Agents Fail at GitHub Issue Resolution - arXiv, accessed May 25, 2025, https://arxiv.org/abs/2503.12374
12. Artificial Intelligence Index Report 2025 - AWS, accessed May 25, 2025, https://hai-production.s3.amazonaws.com/files/hai_ai_index_report_2025.pdf
13. BitsAI-CR: Automated Code Review via LLM in Practice - arXiv, accessed May 25, 2025, https://arxiv.org/html/2501.15134v1
14. AI Code Review and the Best AI Code Review Tools in 2025 - Qodo, accessed May 25, 2025, https://www.qodo.ai/learn/code-review/ai/
15. AI software development has proven its worth but is risky - The Register, accessed May 25, 2025, https://www.theregister.com/2025/05/01/ai_software_development_productivity_revolution/
16. pdaicode/awesome-LLMs-finetuning - GitHub, accessed May 25, 2025,

https://github.com/pdaicode/awesome-LLMs-finetuning
17. Top 9 Large Language Models as of May 2025 | Shakudo, accessed May 25, 2025, https://www.shakudo.io/blog/top-9-large-language-models
18. Best 44 Large Language Models (LLMs) in 2025 - Exploding Topics, accessed May 25, 2025, https://explodingtopics.com/blog/list-of-llms
19. Top 7 Open-Source LLMs in 2025 - KDnuggets, accessed May 25, 2025, https://www.kdnuggets.com/top-7-open-source-llms-in-2025
20. Complete Guide to Building LangChain Agents with the LangGraph ..., accessed May 25, 2025, https://www.getzep.com/ai-agents/langchain-agents-langgraph
21. Building a Massively Parallel LangGraph System: Orchestrating Thousands of Simultaneous Workflows with Batch APIs : r/LangChain - Reddit, accessed May 25, 2025, https://www.reddit.com/r/LangChain/comments/1ixqfos/building_a_massively_parallel_langgraph_system/
22. LangGraph's Multi-Agent Systems - Overview, accessed May 25, 2025, https://langchain-ai.github.io/langgraph/concepts/multi_agent/
23. Build a Multi-Agent System with LangGraph and Mistral on AWS, accessed May 25, 2025, https://aws.amazon.com/blogs/machine-learning/build-a-multi-agent-system-with-langgraph-and-mistral-on-aws/
24. LangGraph: Architecting Advanced Multi-Agent Workflows for Enterprise AI Solutions, accessed May 25, 2025, https://www.royalcyber.com/blogs/ai-ml/langgraph-multi-agent-workflows-enterprise-ai/
25. LangGraph vs AutoGen vs CrewAI for Multi-Agent Workflows - Amplework Software, accessed May 25, 2025, https://www.amplework.com/blog/langgraph-vs-autogen-vs-crewai-multi-agent-framework/
26. Example - Trace and Evaluate LangGraph Agents - Langfuse, accessed May 25, 2025, https://langfuse.com/docs/integrations/langchain/example-langgraph-agents
27. LangGraph Tutorial: Parallel Tool Execution - Unit 2.3 Exercise 4 - AI Product Engineer, accessed May 25, 2025, https://aiproduct.engineer/tutorials/langgraph-tutorial-parallel-tool-execution-unit-23-exercise-4
28. AI Agent Workflows: A Complete Guide on Whether to Build With LangGraph or LangChain, accessed May 25, 2025, https://towardsdatascience.com/ai-agent-workflows-a-complete-guide-on-whether-to-build-with-langgraph-or-langchain-117025509fa0/
29. LangGraph Tutorial: Error Handling Patterns - Unit 2.3 Exercise 6 - AI Product Engineer, accessed May 25, 2025, https://aiproduct.engineer/tutorials/langgraph-tutorial-error-handling-patterns-unit-23-exercise-6
30. Building Multi-Agent LLMs – Traditional Approach vs LangGraph - Encora, accessed May 25, 2025,

https://www.encora.com/insights/building-multi-agent-llms-traditional-approach-vs-langgraph
31. LangGraph Tutorial: Result Aggregation Patterns - Unit 2.3 Exercise 5 - AI Product Engineer, accessed May 25, 2025, https://aiproduct.engineer/tutorials/langgraph-tutorial-result-aggregation-patterns-unit-23-exercise-5
32. Zero to One: Learning Agentic Patterns - Philschmid, accessed May 25, 2025, https://www.philschmid.de/agentic-pattern
33. Use a LangGraph agent | Generative AI on Vertex AI - Google Cloud, accessed May 25, 2025, https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/use/langgraph
34. Develop a LangGraph agent | Generative AI on Vertex AI - Google Cloud, accessed May 25, 2025, https://cloud.google.com/vertex-ai/generative-ai/docs/agent-engine/develop/langgraph
35. A highly extensible Git implementation in pure Go. - GitHub, accessed May 25, 2025, https://github.com/go-git/go-git
36. A2.4 Appendix B: Embedding Git in your Applications - go-git, accessed May 25, 2025, https://git-scm.com/book/ms/v2/Appendix-B:-Embedding-Git-in-your-Applications-go-git
37. gonzalo123/ia_git: Leveraging AI to perform Code Audits based on Git Commit Diff with Python and LangChain - GitHub, accessed May 25, 2025, https://github.com/gonzalo123/ia_git
38. GitPython - PyPI, accessed May 25, 2025, https://pypi.org/project/GitPython/0.3.2/
39. JGit, the Java implementation of git - GitHub, accessed May 25, 2025, https://github.com/eclipse-jgit/jgit
40. A Guide to JGit | Baeldung, accessed May 25, 2025, https://www.baeldung.com/jgit
41. isomorphic-git/isomorphic-git: A pure JavaScript implementation of git for node and browsers! - GitHub, accessed May 25, 2025, https://github.com/isomorphic-git/isomorphic-git
42. Core Concepts in ast-grep's Pattern, accessed May 25, 2025, https://ast-grep.github.io/advanced/core-concepts.html
43. Tree-sitter: Introduction, accessed May 25, 2025, https://tree-sitter.github.io/
44. LLMs alone weren't cutting it for code reviews, so we built a better way (open source), accessed May 25, 2025, https://www.reddit.com/r/developersIndia/comments/1jqmgud/llms_alone_werent_cutting_it_for_code_reviews_so/
45. We open sourced our AI code reviewer : r/EngineeringManagers - Reddit, accessed May 25, 2025, https://www.reddit.com/r/EngineeringManagers/comments/1jqgv38/we_open_sourced_our_ai_code_reviewer/

46. Project02 - CS 486-686, accessed May 25, 2025, https://cs486-686-s25.cs.usfca.edu/assignments/project02
47. Code - LlamaIndex, accessed May 25, 2025, https://docs.llamaindex.ai/en/stable/api_reference/node_parsers/code/
48. LlamaIndex: Automatic Knowledge Transfer (KT) Generation for Code Bases, accessed May 25, 2025, https://www.llamaindex.ai/blog/llamaindex-automatic-knowledge-transfer-kt-generation-for-code-bases-f3d91f21b7af
49. PlantUML, accessed May 25, 2025, https://plantuml.com/
50. Component diagram - PlantUML.com, accessed May 25, 2025, https://plantuml.com/component-diagram
51. Quick Guide to PlantUML: Diagrams, Syntax & Best Practices | Miro, accessed May 25, 2025, https://miro.com/diagramming/what-is-plantuml/
52. PlantUML.com, accessed May 25, 2025, https://plantuml.com/class-diagram
53. PlantUML.com, accessed May 25, 2025, https://plantuml.com/sequence-diagram
54. Mermaid.js: A Complete Guide - Swimm, accessed May 25, 2025, https://swimm.io/learn/mermaid-js/mermaid-js-a-complete-guide
55. Conquering System Design Diagrams: My Shift to Mermaid.js - DEV Community, accessed May 25, 2025, https://dev.to/rusydy/conquering-system-design-diagrams-my-shift-to-mermaidjs-5bol
56. Class diagrams | Mermaid, accessed May 25, 2025, https://mermaid.js.org/syntax/classDiagram.html
57. Sequence diagrams | Mermaid, accessed May 25, 2025, https://mermaid.js.org/syntax/sequenceDiagram.html
58. Documentation | Graphviz, accessed May 25, 2025, https://www.graphviz.org/documentation/
59. Building Long-Term memories using hierarchical summarization - Pieces for developers, accessed May 25, 2025, https://pieces.app/blog/hierarchical-summarization
60. Repository-Level Code Understanding by LLMs via Hierarchical Summarization: Improving Code Search and Bug Localization - ResearchGate, accessed May 25, 2025, https://www.researchgate.net/publication/391739021_Repository-Level_Code_Understanding_by_LLMs_via_Hierarchical_Summarization_Improving_Code_Search_and_Bug_Localization
61. [Literature Review] Hierarchical Repository-Level Code Summarization for Business Applications Using Local LLMs - Moonlight, accessed May 25, 2025, https://www.themoonlight.io/en/review/hierarchical-repository-level-code-summarization-for-business-applications-using-local-llms
62. Hierarchical Repository-Level Code Summarization for Business Applications Using Local LLMs - arXiv, accessed May 25, 2025, https://arxiv.org/html/2501.07857v1
63. Hierarchical Graph-Based Code Summarization for Enhanced Context Retrieval - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.08975v1

64. A Prompt Learning Framework for Source Code Summarization - arXiv, accessed May 25, 2025, https://arxiv.org/html/2312.16066v2
65. Top 13 Advanced RAG Techniques for Your Next Project - Analytics Vidhya, accessed May 25, 2025, https://www.analyticsvidhya.com/blog/2025/04/advanced-rag-techniques/
66. Advanced RAG Techniques: What They Are & How to Use Them - FalkorDB, accessed May 25, 2025, https://www.falkordb.com/blog/advanced-rag/
67. SARIF support for code scanning - GitHub Docs, accessed May 25, 2025, https://docs.github.com/en/code-security/code-scanning/integrating-with-code-scanning/sarif-support-for-code-scanning
68. accessed January 1, 1970, https://docs.github.com/en/enterprise-cloud@latest/code-security/code-scanning/introduction-to-code-scanning/about-sarif-output-for-code-scanning
69. Add persistence - GitHub Pages, accessed May 25, 2025, https://langchain-ai.github.io/langgraph/how-tos/persistence/
70. accessed January 1, 1970, https://www.apollographql.com/blog/backend/understanding-graphql-the-role-of-the-schema-and-type-system/
71. codellama - Ollama, accessed May 25, 2025, https://ollama.com/library/codellama
72. accessed January 1, 1970, https://huggingface.co/collections/codellama/codellama-meta-6565d89175c279b3137970e3
73. accessed January 1, 1970, https://huggingface.co/bigcode/starcoder2
74. accessed January 1, 1970, https://www.deepseek.com/en/blog/deepseek-coder-v2
75. Mixtral of experts | Mistral AI, accessed May 25, 2025, https://mistral.ai/news/mixtral-of-experts/
76. Context Window (LLMs) - Klu.ai, accessed May 25, 2025, https://klu.ai/glossary/context-window
77. LLM Context Windows: Why They Matter and 5 Solutions for Context Limits - Kolena, accessed May 25, 2025, https://www.kolena.com/guides/llm-context-windows-why-they-matter-and-5-solutions-for-context-limits/
78. Context Window Development Considerations | FredPope.com - Official Website of Fred Pope, accessed May 25, 2025, https://www.fredpope.com/blog/development/context-window-development-considerations
79. LLM Context Windows: Basics, Examples & Prompting Best Practices - Swimm, accessed May 25, 2025, https://swimm.io/learn/large-language-models/llm-context-windows-basics-examples-and-prompting-best-practices
80. Large Language Models (LLMs) for Source Code Analysis: applications, models and datasets - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.17502v1
81. LLM generated code snippet merging into existing using ASTs : r/theprimeagen - Reddit, accessed May 25, 2025,

https://www.reddit.com/r/theprimeagen/comments/1idtjp2/llm_generated_code_snippet_merging_into_existing/
82. Lessons from 27 Months Building LLM Coding Agents | Tribe AI, accessed May 25, 2025, https://www.tribe.ai/applied-ai/lessons-from-27-months-building-llm-coding-agents
83. Has anyone tried using efficient techniques? : r/RooCode - Reddit, accessed May 25, 2025, https://www.reddit.com/r/RooCode/comments/1iozd2s/has_anyone_tried_using_efficient_techniques/
84. RAG or Fine-tuning? A Comparative Study on LCMs-based Code Completion in Industry, accessed May 25, 2025, https://arxiv.org/html/2505.15179v1
85. Enhancing LLM Code Generation: A Systematic Evaluation of Multi-Agent Collaboration and Runtime Debugging for Improved Accuracy, Reliability, and Latency - arXiv, accessed May 25, 2025, https://arxiv.org/html/2505.02133v1
86. Transform Your Search Learn Retrieval Augmented Generation RAG - AST Consulting, accessed May 25, 2025, https://astconsulting.in/retrieval-augmented-generation-rag/transform-search-learn-rag
87. ReadMe.LLM: A Framework to Help LLMs Understand Your Library - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.09798v2
88. SnipGen: A Mining Repository Framework for Evaluating LLMs for Code - arXiv, accessed May 25, 2025, https://arxiv.org/pdf/2502.07046?
89. RAG vs Fine-tuning vs Prompt Engineering: Everything You Need to Know | InterSystems, accessed May 25, 2025, https://www.intersystems.com/resources/rag-vs-fine-tuning-vs-prompt-engineering-everything-you-need-to-know/
90. Unlocking the Potential of Generative AI through Neuro-Symbolic Architectures – Benefits and Limitations - arXiv, accessed May 25, 2025, https://arxiv.org/html/2502.11269v1
91. Synergizing RAG and Reasoning: A Systematic Review - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.15909v1
92. 11 Game-Changing Open-Source AI Tools Every Dev Should Know About - n8n Blog, accessed May 25, 2025, https://blog.n8n.io/open-source-ai-tools/
93. Enhancing LLM-based Code Translation in Repository Context via Triple Knowledge-Augmented - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.18305v1
94. Automating Software Development Processes Through Multi-Agent Systems - Diva, accessed May 25, 2025, https://mau.diva-portal.org/smash/get/diva2:1871606/FULLTEXT02.pdf
95. Enhancing RAG Systems with Advanced Prompting Techniques - Oracle Blogs, accessed May 25, 2025, https://blogs.oracle.com/ai-and-datascience/post/enhancing-rag-with-advanced-prompting
96. Automated Visualization Code Synthesis via Multi-Path Reasoning and

Feedback-Driven Optimization - arXiv, accessed May 25, 2025, https://arxiv.org/html/2502.11140v2
97. [2410.09268] One Step at a Time: Combining LLMs and Static Analysis to Generate Next-Step Hints for Programming Tasks - arXiv, accessed May 25, 2025, https://arxiv.org/abs/2410.09268
98. tree-sitter-xml - NPM, accessed May 25, 2025, https://www.npmjs.com/package/%40tree-sitter-grammars%2Ftree-sitter-xml
99. tree-sitter-xml - io.github.itsaky - Maven Central - Sonatype, accessed May 25, 2025, https://central.sonatype.com/artifact/io.github.itsaky/tree-sitter-xml
100. GitHub Copilot · Your AI pair programmer · GitHub, accessed May 25, 2025, https://github.com/features/copilot
101. Unraveling Tree-Sitter Queries: Your Guide to Code Analysis Magic - DEV Community, accessed May 25, 2025, https://dev.to/shrsv/unraveling-tree-sitter-queries-your-guide-to-code-analysis-magic-41il
102. Incremental Parsing Using Tree-sitter - Strumenta - Federico Tomassetti, accessed May 25, 2025, https://tomassetti.me/incremental-parsing-using-tree-sitter/
103. AlDanial/cloc: cloc counts blank lines, comment lines, and ... - GitHub, accessed May 25, 2025, https://github.com/AlDanial/cloc
104. Creating Parsers - Tree-sitter, accessed May 25, 2025, https://tree-sitter.github.io/tree-sitter/creating-parsers/
105. AndroidIDEOfficial/tree-sitter-xml: Tree Sitter grammar for Android XML files - GitHub, accessed May 25, 2025, https://github.com/AndroidIDEOfficial/tree-sitter-xml
106. Tree-Sitter: From Code to Syntax-Tree - DEV Community, accessed May 25, 2025, https://dev.to/shailendra53/tree-sitter-from-code-to-syntax-tree-45op
107. accessed January 1, 1970, https://github.com/tree-sitter/tree-sitter-xml
108. Using Parsers - Tree-sitter, accessed May 25, 2025, https://tree-sitter.github.io/tree-sitter/using-parsers#query-syntax
109. Decompiler - Visual Studio Marketplace, accessed May 25, 2025, https://marketplace.visualstudio.com/items?itemName=tintinweb.vscode-decompiler
110. Configure build variants | Android Studio | Android Developers, accessed May 25, 2025, https://developer.android.com/build/build-variants
111. arxiv.org, accessed May 25, 2025, https://arxiv.org/abs/2312.14337
112. Writing Checks – checkstyle, accessed May 25, 2025, https://checkstyle.sourceforge.io/writingchecks.html
113. Custom Rules - ESLint - Pluggable JavaScript Linter, accessed May 25, 2025, https://eslint.org/docs/latest/developer-guide/working-with-rules
114. Ktlint: Features, accessed May 25, 2025, https://ktlint.github.io/#creating-a-ruleset
115. Test Plugins — Bandit documentation, accessed May 25, 2025, https://bandit.readthedocs.io/en/latest/plugins/index.html
116. www.graphviz.org, accessed May 25, 2025,

https://www.graphviz.org/pdf/dotguide.pdf
117. accessed January 1, 1970, https://www.structurizr.com/help/getting-started-java
118. Introduction to Code Metrics — Radon 4.1.0 documentation, accessed May 25, 2025, https://radon.readthedocs.io/en/latest/intro.html
119. accessed January 1, 1970, https://lizard.ws/
120. www.sonarsource.com, accessed May 25, 2025, https://www.sonarsource.com/docs/CognitiveComplexity.pdf
121. accessed January 1, 1970, https://www.ibm.com/docs/en/rdfi/9.7.0?topic=reporting-understanding-halstead-metrics
122. accessed January 1, 1970, https://www.fast.ai/posts/2023-09-06-what-is-rlhf.html
123. Aligning language models to follow instructions | OpenAI, accessed May 25, 2025, https://openai.com/research/instruction-following
124. Hello GPT-4o | OpenAI, accessed May 25, 2025, https://openai.com/index/hello-gpt-4o/
125. Gemini Flash - Google DeepMind, accessed May 25, 2025, https://deepmind.google/technologies/gemini/flash/
126. Introducing Claude 3.5 Sonnet \ Anthropic, accessed May 25, 2025, https://www.anthropic.com/news/claude-3-5-sonnet
127. Cody | AI coding assistant from Sourcegraph, accessed May 25, 2025, https://about.sourcegraph.com/cody
128. accessed January 1, 1970, https://about.gitlab.com/solutions/ai/
129. Generative AI Assistant for Software Development – Amazon Q ..., accessed May 25, 2025, https://aws.amazon.com/codewhisperer/features/
130. accessed January 1, 1970, https://www.tabnine.com/product
131. accessed January 1, 1970, https://www.cacher.io/blog/best-ai-code-review-tools
132. accessed January 1, 1970, https://www.softwaretestinghelp.com/ai-code-review-tools/
133. accessed January 1, 1970, https://thenewstack.io/ai-code-generation-tools-the-next-generation-of-developer-productivity/
134. accessed January 1, 1970, https://www.infoworld.com/article/3704330/the-best-ai-powered-coding-tools.html
135. accessed January 1, 1970, https://techcrunch.com/category/artificial-intelligence/
136. accessed January 1, 1970, https://venturebeat.com/category/ai/
137. Artificial intelligence | MIT News | Massachusetts Institute of ..., accessed May 25, 2025, https://news.mit.edu/topic/artificial-intelligence2
138. All Posts | SAIL Blog - Stanford AI Lab, accessed May 25, 2025, https://ai.stanford.edu/blog/
139. accessed January 1, 1970, https://research.google/blog/ai/

140. accessed January 1, 1970, https://www.meta.ai/research/
141. Microsoft Search, Assistant and Intelligence - Microsoft Research, accessed May 25, 2025, https://www.microsoft.com/en-us/research/group/msai/
142. The Batch | DeepLearning.AI | AI News & Insights, accessed May 25, 2025, https://www.deeplearning.ai/the-batch/
143. Import AI: Issue 1: GANs, ML bias, and a neural net Benjamin Franklin, accessed May 25, 2025, https://jack-clark.net/import-ai/
144. AI & ML – O'Reilly, accessed May 25, 2025, https://www.oreilly.com/radar/topics/ai-ml/
145. accessed January 1, 1970, https://www.gartner.com/en/research/technology-trends/what-s-new-in-the-2024-gartner-hype-cycle-for-emerging-technologies
146. Two Minute Papers - YouTube, accessed May 25, 2025, https://www.youtube.com/@TwoMinutePapers
147. Lex Fridman Podcast - Lex Fridman, accessed May 25, 2025, https://lexfridman.com/podcast/
148. Latent.Space | swyx & Alessio | Substack, accessed May 25, 2025, https://www.latent.space/
149. Blog | Security Trends, Secure Coding and Application Security ..., accessed May 25, 2025, https://www.semgrep.dev/blog/
150. accessed January 1, 1970, https://codeql.github.com/blog/
151. Kernel index [LWN.net], accessed May 25, 2025, https://lwn.net/Kernel/Index/
152. Linux kernel coding style — The Linux Kernel documentation, accessed May 25, 2025, https://www.kernel.org/doc/html/latest/process/coding-style.html
153. Google Java Style Guide, accessed May 25, 2025, https://google.github.io/styleguide/javaguide.html
154. Google Python Style Guide, accessed May 25, 2025, https://google.github.io/styleguide/pyguide.html
155. Kotlin style guide | Android Developers, accessed May 25, 2025, https://developer.android.com/kotlin/style-guide
156. airbnb/javascript: JavaScript Style Guide - GitHub, accessed May 25, 2025, https://github.com/airbnb/javascript
157. Conventional Commits, accessed May 25, 2025, https://www.conventionalcommits.org/en/v1.0.0/
158. accessed January 1, 1970, https://www.sonarsource.com/solutions/technical-debt/
159. Design Stamina Hypothesis - Martin Fowler, accessed May 25, 2025, https://martinfowler.com/bliki/DesignStaminaHypothesis.html
160. refactoring - Martin Fowler, accessed May 25, 2025, https://martinfowler.com/tags/refactoring.html
161. "Key words for use in RFCs to Indicate Requirement Levels ... - IETF, accessed May 25, 2025, https://www.rfc-editor.org/rfc/rfc2119.txt
162. Static Analysis Results Interchange Format (SARIF) Version 2.1.0 ..., accessed May 25, 2025, https://docs.oasis-open.org/sarif/sarif/v2.1.0/sarif-v2.1.0.html
163. accessed January 1, 1970, https://www.youtube.com/watch?v=UwD_TfsBe0M

164. accessed January 1, 1970, https://www.youtube.com/watch?v=L3Z1Qd_4x0s
165. accessed January 1, 1970, https://www.youtube.com/watch?v=ABS-C_A0E0s
166. accessed January 1, 1970, https://www.youtube.com/watch?v=hSj_P6k2t0Q
167. "Clean" Code, Horrible Performance - YouTube, accessed May 25, 2025, https://www.youtube.com/watch?v=tD5NrevFtbU
168. accessed January 1, 1970, https://www.youtube.com/watch?v=u46k0j2N-0E
169. accessed January 1, 1970, https://www.youtube.com/watch?v=EIV9L6K83AQ
170. accessed January 1, 1970, https://www.youtube.com/watch?v=eN2qVdIlpX4
171. accessed January 1, 1970, https://www.youtube.com/watch?v=0mOzk30p0YI
172. accessed January 1, 1970, https://www.youtube.com/watch?v=d_3N5zDR2iU
173. accessed January 1, 1970, https://www.youtube.com/watch?v=VIFgFA5pYhI
174. accessed January 1, 1970, https://www.youtube.com/watch?v=R2fm00YfKvg
175. accessed January 1, 1970, https://www.youtube.com/watch?v=yPSHqsU73PU
176. accessed January 1, 1970, https://www.youtube.com/watch_videos?video_ids=UwD_TfsBe0M,L3Z1Qd_4x0s,ABS-C_A0E0s,hSj_P6k2t0Q,tD5NrevFtbU,u46k0j2N-0E,EIV9L6K83AQ,eN2qVdIlpX4,0mOzk30p0YI,d_3N5zDR2iU,VIFgFA5pYhI,R2fm00YfKvg,yPSHqsU73PU
177. kpdecker/jsdiff: A javascript text differencing implementation. - GitHub, accessed May 25, 2025, https://github.com/kpdecker/jsdiff
178. Top 10 Examples of diff code in Javascript - CloudDefense.AI, accessed May 25, 2025, https://www.clouddefense.ai/code/javascript/example/diff
179. react-diff-viewer-continued vs react-diff-viewer vs react-diff-view | React Diff Libraries Comparison - NPM Compare, accessed May 25, 2025, https://npm-compare.com/react-diff-viewer-continued,react-diff-viewer,react-diff-view
180. difflib — Helpers for computing deltas — Python 3.13.3 documentation, accessed May 25, 2025, https://docs.python.org/3/library/difflib.html
181. Side-by-side comparison of strings in Python | Towards Data Science, accessed May 25, 2025, https://towardsdatascience.com/side-by-side-comparison-of-strings-in-python-b9491ac858/
182. Master Treesitter SQL: Proven Strategies for Seamless Integration - Kodezi Blog, accessed May 25, 2025, https://blog.kodezi.com/master-treesitter-sql-proven-strategies-for-seamless-integration/
183. Writing a TLA⁺ tree-sitter grammar - - Andrew Helwer, accessed May 25, 2025, https://ahelwer.ca/post/2023-01-11-tree-sitter-tlaplus
184. ast-grep - gist/GitHub, accessed May 25, 2025, https://gist.github.com/eightHundreds/70c9ec82c2b7ba7140dc2cfaa311e8d1
185. Diving into Tree-Sitter: Parsing Code with Python Like a Pro - DEV Community, accessed May 25, 2025, https://dev.to/shrsv/diving-into-tree-sitter-parsing-code-with-python-like-a-pro-17h8
186. HANDOUT - 2024-10-18 - LLMS, ASTs and DSLs - mnml's vault - Obsidian Publish, accessed May 25, 2025,

https://publish.obsidian.md/manuel/Writing/Presentation/2024-10-18+-+LLMs+for+DSLs/HANDOUT+-+2024-10-18+-+LLMS%2C+ASTs+and+DSLs
187. Frequently Asked Questions | ast-grep, accessed May 25, 2025, https://ast-grep.github.io/advanced/faq.html
188. Tinkering with Tree-Sitter Using Go - DEV Community, accessed May 25, 2025, https://dev.to/shrsv/tinkering-with-tree-sitter-using-go-4d8n
189. Guide to writing your first Tree-sitter grammar - GitHub Gist, accessed May 25, 2025, https://gist.github.com/Jolg42/096977cf0566e599a2ac937808b221fc
190. For those who don't already know, this is built on tree-sitter (https://tree-sit... | Hacker News, accessed May 25, 2025, https://news.ycombinator.com/item?id=39779595
191. Need a simple Go AST example for my DSL : r/golang - Reddit, accessed May 25, 2025, https://www.reddit.com/r/golang/comments/1i10d7f/need_a_simple_go_ast_example_for_my_dsl/
192. TechTalk 6/28/17 - Domain-Specific Languages (DSLs) and Code Generation - YouTube, accessed May 25, 2025, https://www.youtube.com/watch?v=QP12czKqHMo
193. Dossier: A tree-sitter based multi-language source code and docstring parser : r/rust - Reddit, accessed May 25, 2025, https://www.reddit.com/r/rust/comments/1980y0j/dossier_a_treesitter_based_multilanguage_source/
194. OWASP Top Ten | OWASP Foundation, accessed May 25, 2025, https://owasp.org/www-project-top-ten/
195. CWE Top 25 Most Dangerous Software Weaknesses – 2023 - CWE, accessed May 25, 2025, https://cwe.mitre.org/top25/archive/2023/2023_cwe_top25.html
196. Source Code Analysis Tools | OWASP Foundation, accessed May 25, 2025, https://www.owasp.org/index.php/Source_Code_Analysis_Tools
197. accessed January 1, 1970, https://source.android.com/docs/core/tests/codelines/linters
198. Enable app optimization | Android Studio | Android Developers, accessed May 25, 2025, https://developer.android.com/studio/build/shrink-code
199. Working with .resx Files Programmatically - .NET | Microsoft Learn, accessed May 25, 2025, https://learn.microsoft.com/en-us/dotnet/core/extensions/work-with-resx-files-programmatically
200. Access app-specific files | App data and files - Android Developers, accessed May 25, 2025, https://developer.android.com/training/data-storage/app-specific
201. Let's create a Tree-sitter grammar - Jonas Hietala, accessed May 25, 2025, https://www.jonashietala.se/blog/2024/03/19/lets_create_a_tree-sitter_grammar
202. Tree-sitter-java - Colab, accessed May 25, 2025, https://colab.research.google.com/drive/1MfTbOMrZnQ_FkC65CCJyUE4v21u5pJ4G?usp=sharing
203. Bugdar: AI-Augmented Secure Code Review for GitHub Pull Requests - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.17302v1

204. AI Code Review in Practice: ELEKS' Expert Evaluating GitHub Copilot, accessed May 25, 2025, https://eleks.com/expert-opinion/evaluating-github-copilot-code-review/
205. Rethinking Code Review Workflows with LLM Assistance: An Empirical Study - arXiv, accessed May 25, 2025, https://arxiv.org/html/2505.16339v1
206. RepairAgent: An Autonomous, LLM-Based Agent for Program Repair - arXiv, accessed May 25, 2025, https://arxiv.org/html/2403.17134v2
207. kodus-ai/license.md at main · kodustech/kodus-ai · GitHub, accessed May 25, 2025, https://github.com/kodustech/kodus-ai/blob/main/license.md
208. arXiv:2503.14281v1 [cs.CR] 18 Mar 2025, accessed May 25, 2025, https://arxiv.org/pdf/2503.14281
209. XOXO: Stealthy Cross-Origin Context Poisoning Attacks against AI Coding Assistants - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.14281v2
210. How Accurately Do Large Language Models Understand Code? - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.04372v1
211. LLMpatronous: Harnessing the Power of LLMs For Vulnerability Detection - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.18423v1
212. CISO Guide: Penetration Testing for Large Language Models (LLMs) - BreachLock, accessed May 25, 2025, https://www.breachlock.com/resources/reports/ciso-guide-penetration-testing-for-large-language-models-llms/
213. Assessing Large Language Models in Comprehending and Verifying Concurrent Programs across Memory Models - arXiv, accessed May 25, 2025, https://arxiv.org/pdf/2501.14326
214. DR.FIX: Automatically Fixing Data Races at Industry Scale - arXiv, accessed May 25, 2025, https://arxiv.org/pdf/2504.15637
215. Uncovering the Challenges: A Study of Corner Cases in Bug-Inducing Commits, accessed May 25, 2025, https://www.computer.org/csdl/proceedings-article/saner/2025/351000a639/26TIuJsTF1S
216. Data Race Detection Using Large Language Models - arXiv, accessed May 25, 2025, https://arxiv.org/pdf/2308.07505
217. [2405.13565] AI-Assisted Assessment of Coding Practices in Modern Code Review - arXiv, accessed May 25, 2025, https://arxiv.org/abs/2405.13565
218. Security of Language Models for Code: A Systematic Literature Review - arXiv, accessed May 25, 2025, https://arxiv.org/html/2410.15631v2
219. arxiv.org, accessed May 25, 2025, https://arxiv.org/html/2504.20814v1
220. AI code review implementation and best practices - Graphite, accessed May 25, 2025, https://graphite.dev/guides/ai-code-review-implementation-best-practices
221. The Impact of AI on Code Review Processes - Zencoder, accessed May 25, 2025, https://zencoder.ai/blog/ai-advancements-in-code-review
222. Evaluating Human-AI Collaboration: A Review and Methodological Framework - arXiv, accessed May 25, 2025, https://arxiv.org/html/2407.19098v2
223. [Question]: Build advanced RAG · Issue #18589 · run-llama/llama_index -

GitHub, accessed May 25, 2025, https://github.com/run-llama/llama_index/issues/18589
224. Denis2054/RAG-Driven-Generative-AI: This repository provides programs to build Retrieval Augmented Generation (RAG) code for Generative AI with LlamaIndex, Deep Lake, and Pinecone leveraging the power of OpenAI and Hugging Face models for generation and evaluation. - GitHub, accessed May 25, 2025, https://github.com/Denis2054/RAG-Driven-Generative-AI
225. Building Performant RAG Applications for Production - LlamaIndex, accessed May 25, 2025, https://docs.llamaindex.ai/en/stable/optimizing/production_rag/
226. Code retrieval for RAG : r/Rag - Reddit, accessed May 25, 2025, https://www.reddit.com/r/Rag/comments/1gabsik/code_retrieval_for_rag/
227. Improving LlamaIndex's Code Chunker by Cleaning Tree-Sitter CSTs, accessed May 25, 2025, https://docs.sweep.dev/blogs/chunking-improvements
228. 11 LLMs for Explainable Artificial Intelligence: Automating Natural Language Explanations of Predictive Analytics Models, accessed May 25, 2025, https://lamethods.org/book2/chapters/ch11-llmsxai/ch11-llmsxai.html
229. Building an Enterprise Knowledge RAG Platform with LlamaIndex and Amazon Bedrock, accessed May 25, 2025, https://community.aws/content/2vMeX91f1Cb4rcDm4tBPzYJmRtl/building-an-enterprise-knowledge-rag-platform-with-llamaindex-and-amazon-bedrock
230. Llamaindex RAG Tutorial - IBM, accessed May 25, 2025, https://www.ibm.com/think/tutorials/llamaindex-rag
231. I built a RAG Chatbot that Understands Your Codebase (LlamaIndex + Nebius AI) - Reddit, accessed May 25, 2025, https://www.reddit.com/r/Rag/comments/1jvfbct/i_built_a_rag_chatbot_that_understands_your/
232. accessed January 1, 1970, https://docs.llamaindex.ai/en/stable/understanding/putting_it_all_together/fullstack_app_guide.html
233. accessed January 1, 1970, https://docs.llamaindex.ai/en/stable/module_guides/deploying/agents/full_agent_deployment_example.html
234. accessed January 1, 1970, https://www.youtube.com/watch?v=6317BHt0T-E
235. accessed January 1, 1970, https://www.youtube.com/watch?v=TRqHsPLk90A
236. accessed January 1, 1970, https://www.youtube.com/watch?v=10oMInVq3mo
237. accessed January 1, 1970, https://www.youtube.com/watch?v=IUEWcR7T3yQ
238. accessed January 1, 1970, https://www.youtube.com/watch?v=9qqEU5f6K0E
239. langchain-ai/rag-from-scratch - GitHub, accessed May 25, 2025, https://github.com/langchain-ai/rag-from-scratch
240. How to get your RAG application to return sources | 🦜 LangChain, accessed May 25, 2025, https://python.langchain.com/docs/how_to/qa_sources/
241. A Practical Guide to Building Local RAG Applications with LangChain - MachineLearningMastery.com, accessed May 25, 2025, https://machinelearningmastery.com/a-practical-guide-to-building-local-rag-applications-with-langchain/

242. Exploring LangChain: A Practical Approach to Language Models and Retrieval-Augmented Generation (RAG) - CODE Magazine, accessed May 25, 2025, https://www.codemag.com/Article/2501051/Exploring-LangChain-A-Practical-Approach-to-Language-Models-and-Retrieval-Augmented-Generation-RAG
243. How-to guides | 🦜 LangChain, accessed May 25, 2025, https://python.langchain.com/v0.2/docs/how_to/#qa-with-rag
244. accessed January 1, 1970, https://www.youtube.com/watch?v=h434g12ayXU
245. accessed January 1, 1970, https://www.youtube.com/watch?v=RoTzCH4Q_2c
246. accessed January 1, 1970, https://www.youtube.com/watch?v=E2c9yV32nK8
247. accessed January 1, 1970, https://www.youtube.com/watch?v=RoTzCH4Q_2c&list=PLfaIDFEXuae2LXbO1_PKyVJiQ23ZztA0x&index=1
248. Human-inspired Episodic Memory for Infinite Context LLMs - OpenReview, accessed May 25, 2025, https://openreview.net/forum?id=BI2int5SAC
249. UML class diagrams | IntelliJ IDEA Documentation - JetBrains, accessed May 25, 2025, https://www.jetbrains.com/help/idea/class-diagram.html
250. Tools to Generate UML Diagrams from Source Code? : r/softwaredevelopment - Reddit, accessed May 25, 2025, https://www.reddit.com/r/softwaredevelopment/comments/1jblnfo/tools_to_generate_uml_diagrams_from_source_code/
251. Unified Modeling Language Code Generation from Diagram Images Using Multimodal Large Language Models - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.12293v1
252. Building Custom Tooling with LLMs - Martin Fowler, accessed May 25, 2025, https://www.martinfowler.com/articles/exploring-gen-ai/16-building-custom-tooling-with-llms.html
253. AI Sequence Diagram Generator - Eraser IO, accessed May 25, 2025, https://www.eraser.io/ai/sequence-diagram-generator
254. Extracting in Sequence diagram - MagicDraw 19.0 LTR - No Magic Documentation, accessed May 25, 2025, https://docs.nomagic.com/display/MD190/Extracting+in+Sequence+diagram
255. 11 best open source tools for Software Architects | Cerbos, accessed May 25, 2025, https://www.cerbos.dev/blog/best-open-source-tools-software-architects
256. Code Visualization: 4 Types of Diagrams and 5 Useful Tools - CodeSee, accessed May 25, 2025, https://www.codesee.io/learning-center/code-visualization
257. Code Visualization: How to Turn Complex Code Into Diagrams | Lucidchart Blog, accessed May 25, 2025, https://www.lucidchart.com/blog/visualize-code-documentation
258. C4 model tools, accessed May 25, 2025, https://c4model.tools/
259. C4 Model Tools: Key Features and 7 Tools to Consider - CodeSee, accessed May 25, 2025, https://www.codesee.io/learning-center/c4-model-tools
260. Tooling | C4 model, accessed May 25, 2025, https://c4model.com/tooling
261. accessed January 1, 1970,

https://www.answer.ai/posts/2024-03-12-c4-context-window.html
262. accessed January 1, 1970, https://www.youtube.com/watch?v=hYXGRsl9L6A
263. accessed January 1, 1970, https://www.youtube.com/watch?v=rQj33bY9N-g
264. accessed January 1, 1970, https://www.youtube.com/watch?v=xkoM2GqrC5A
265. Top 12 Code Review Tools To Ace Code Quality [2025] - Qodo, accessed May 25, 2025, https://www.qodo.ai/learn/code-review/tools/
266. Trusting AI: does uncertainty visualization affect decision-making? - Frontiers, accessed May 25, 2025, https://www.frontiersin.org/journals/computer-science/articles/10.3389/fcomp.2025.1464348/full
267. Research News - Electrical Engineering and Computer Science, accessed May 25, 2025, https://eecs.engin.umich.edu/category/research
268. Prompt Augmentation: UX Design Patterns for Better AI Prompting - UX Tigers, accessed May 25, 2025, https://www.uxtigers.com/post/prompt-augmentation
269. Aided Prompt Understanding: UX Design Patterns to Build AI Prompting Skills - UX Tigers, accessed May 25, 2025, https://www.uxtigers.com/post/prompt-understanding
270. Understanding User Mental Models in AI-Driven Code Completion Tools: Insights from an Elicitation Study - arXiv, accessed May 25, 2025, https://arxiv.org/html/2502.02194v1
271. A Formative Study to Explore the Design of Generative UI Tools to Support UX Practitioners and Beyond - arXiv, accessed May 25, 2025, https://arxiv.org/html/2501.13145v1
272. Emerging Developer Patterns for the AI Era | Andreessen Horowitz, accessed May 25, 2025, https://a16z.com/nine-emerging-developer-patterns-for-the-ai-era/
273. Top 6 Examples of AI Guidelines in Design Systems – Blog – Supernova.io, accessed May 25, 2025, https://www.supernova.io/blog/top-6-examples-of-ai-guidelines-in-design-systems?ref=weekly.ui-patterns.com
274. AI in UI Design: Current Tools and Applications - UXPin, accessed May 25, 2025, https://www.uxpin.com/studio/blog/ai-in-ui-design-current-tools-and-applications/
275. Compositional Structures as Substrates for Human-AI Co-creation Environment: A Design Approach and A Case Study - Foundation Interface Lab - University of California San Diego, accessed May 25, 2025, https://hci.ucsd.edu/papers/videorigami.pdf
276. Computer Science Apr 2024 - arXiv, accessed May 25, 2025, http://arxiv.org/list/cs/2024-04?skip=2990&show=2000
277. ACM in the News Archive - Association for Computing Machinery, accessed May 25, 2025, https://www.acm.org/media-center/acm-in-the-news-archive
278. AI and UI Design - CHI '24, accessed May 25, 2025, https://programs.sigchi.org/chi/2024/program/session/156158
279. Concerns and Challenges of AI Tools in the UI/UX Design Process: A

Cross-Sectional Survey - YouTube, accessed May 25, 2025, https://www.youtube.com/watch?v=9w-qlH3qXm8
280. Data Analysis and Visualization with Generative AI: Tools - Guides - Georgetown University, accessed May 25, 2025, https://guides.library.georgetown.edu/c.php?g=1438029&p=10678251
281. The Unbeatable Duo: How Human in the Loop AI Elevates Automation - Workflow86, accessed May 25, 2025, https://www.workflow86.com/blog/the-unbeatable-duo-how-human-in-the-loop-ai-elevates-automation
282. Integrating AI into your code review workflow - Graphite, accessed May 25, 2025, https://graphite.dev/guides/integrating-ai-code-review-workflow
283. 15 Best AI Tools for Developers To Improve Workflow in 2025 - Index.dev, accessed May 25, 2025, https://www.index.dev/blog/ai-tools-developer-workflow
284. Best Practices for Using AI in Software Development 2025 - Leanware, accessed May 25, 2025, https://www.leanware.co/insights/best-practices-ai-software-development
285. How to review code written by AI - Graphite, accessed May 25, 2025, https://graphite.dev/guides/how-to-review-code-written-by-ai
286. I Tried the Top AI Tools for UI/UX Design: Here's What I Learned - DesignRush, accessed May 25, 2025, https://www.designrush.com/agency/ui-ux-design/trends/ux-ui-ai-tools
287. LLMs for Explainable AI: A Comprehensive Survey - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.00125v1
288. Explainable Artificial Intelligence Techniques for Software Development Lifecycle: A Phase-specific Survey - arXiv, accessed May 25, 2025, https://arxiv.org/pdf/2505.07058
289. Explainable Artificial Intelligence Techniques for Software Development Lifecycle: A Phase-specific Survey - ResearchGate, accessed May 25, 2025, https://www.researchgate.net/publication/391676961_Explainable_Artificial_Intelligence_Techniques_for_Software_Development_Lifecycle_A_Phase-specific_Survey
290. Unveiling Explainable AI in Healthcare: Current Trends, Challenges, and Future Directions, accessed May 25, 2025, https://www.medrxiv.org/content/10.1101/2024.08.10.24311735v2.full-text
291. [2503.03979] ReasonGraph: Visualisation of Reasoning Paths - arXiv, accessed May 25, 2025, https://arxiv.org/abs/2503.03979
292. All events - cis.mpg.de |, accessed May 25, 2025, https://www.cis.mpg.de/all-events/
293. Top 5 AI Tools for Data Visualization to Consider in 2025 - ThoughtSpot, accessed May 25, 2025, https://www.thoughtspot.com/data-trends/ai/ai-tools-for-data-visualization
294. Enhancing Code LLMs with Reinforcement Learning in Code Generation: A Survey - arXiv, accessed May 25, 2025, https://arxiv.org/html/2412.20367v1
295. On Large Language Model Continual Unlearning - OpenReview, accessed May 25, 2025, https://openreview.net/forum?id=Essg9kb4yx

296. ReasonGraph: Visualisation of Reasoning Paths - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.03979v1
297. Build and operate an effective architecture review board - AWS, accessed May 25, 2025, https://aws.amazon.com/blogs/architecture/build-and-operate-an-effective-architecture-review-board/
298. arxiv.org, accessed May 25, 2025, https://arxiv.org/abs/2504.04334
299. (PDF) The Future of AI-Driven Software Engineering - ResearchGate, accessed May 25, 2025, https://www.researchgate.net/publication/388371819_The_Future_of_AI-Driven_Software_Engineering
300. Enhancing Large Language Models in Long Code Translation through Instrumentation and Program State Alignment - arXiv, accessed May 25, 2025, https://www.arxiv.org/pdf/2504.02017
301. (PDF) SoK: Frontier AI's Impact on the Cybersecurity Landscape - ResearchGate, accessed May 25, 2025, https://www.researchgate.net/publication/390601417_SoK_Frontier_AI's_Impact_on_the_Cybersecurity_Landscape
302. Securing Smart Contracts Against Business Logic Flaws - kth .diva, accessed May 25, 2025, http://kth.diva-portal.org/smash/get/diva2:1956747/FULLTEXT01.pdf
303. Property-Based ASTs - TU Delft Repository, accessed May 25, 2025, https://repository.tudelft.nl/file/File_dd3a46ca-c387-4dd7-9a1f-3645343a5054
304. About code scanning with CodeQL - GitHub Docs, accessed May 25, 2025, https://docs.github.com/en/code-security/code-scanning/introduction-to-code-scanning/about-code-scanning-with-codeql
305. Preparing your code for CodeQL analysis - GitHub Docs, accessed May 25, 2025, https://docs.github.com/en/code-security/codeql-cli/getting-started-with-the-codeql-cli/preparing-your-code-for-codeql-analysis
306. About CodeQL — CodeQL, accessed May 25, 2025, https://codeql.github.com/docs/codeql-overview/about-codeql/
307. accessed January 1, 1970, https://codeql.github.com/docs/writing-codeql-queries/defining-a-path-problem-in-codeql/
308. accessed January 1, 1970, https://about.gitlab.com/solutions/code-review/
309. Bitbucket code review: Merge with confidence | Atlassian, accessed May 25, 2025, https://www.atlassian.com/software/bitbucket/features/code-review
310. accessed January 1, 1970, https://www.pluralsight.com/blog/software-development/understanding-coupling-cohesion
311. Automated Secure Code Review at Scale Using Static Analysis and Generative AI.md, accessed May 25, 2025, https://github.com/247arjun/ai-secure-code-review/blob/main/Automated%20Secure%20Code%20Review%20at%20Scale%20Using%20Static%20Analysis%20and%20Generative%20AI.md

312. Feature Importance in the Context of Traditional and Just-In-Time Software Defect Prediction Models - Scholarship@Western, accessed May 25, 2025, https://ir.lib.uwo.ca/cgi/viewcontent.cgi?article=1652&context=electricalpub
313. Interpretable Software Defect Prediction from Project Effort and Static Code Metrics - MDPI, accessed May 25, 2025, https://www.mdpi.com/2073-431X/13/2/52
314. LARGE LANGUAGE MODELS - Innovation for Cool Earth Forum (ICEF), accessed May 25, 2025, https://www.icef.go.jp/wp-content/themes/icef_new/pdf/roadmap/2024/11_ICEF%202.0%20LLMs_stand%20alone.pdf
315. socialfoundations/folktexts: Evaluate uncertainty, calibration, accuracy, and fairness of LLMs on real-world survey data! - GitHub, accessed May 25, 2025, https://github.com/socialfoundations/folktexts
316. TemryL/LLM-Disease-Risk-Benchmark - GitHub, accessed May 25, 2025, https://github.com/TemryL/LLM-Disease-Risk-Benchmark/
317. Reasoning with LLMs for Zero-Shot Vulnerability Detection - arXiv, accessed May 25, 2025, https://arxiv.org/html/2503.17885v1
318. Risk Assessment Framework for Code LLMs via Leveraging Internal States - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.14640v1
319. OWASP LLM Top 10: How it Applies to Code Generation | Learn Article - Sonar, accessed May 25, 2025, https://www.sonarsource.com/learn/owasp-llm-code-generation/
320. LLM evaluation metrics and methods - Evidently AI, accessed May 25, 2025, https://www.evidentlyai.com/llm-guide/llm-evaluation-metrics
321. TRL - Transformer Reinforcement Learning - Hugging Face, accessed May 25, 2025, https://huggingface.co/docs/trl/index
322. huggingface/trl: Train transformer language models with reinforcement learning. - GitHub, accessed May 25, 2025, https://github.com/huggingface/trl
323. LLM Fine-Tuning Tools: Best Picks for ML Tasks in 2025 | Label Your Data, accessed May 25, 2025, https://labelyourdata.com/articles/llm-fine-tuning/top-llm-tools-for-fine-tuning
324. accessed January 1, 1970, https://ai.google.dev/gemini-api/docs/models/gemini-1-5-pro-faq
325. accessed January 1, 1970, https://www.anthropic.com/news/claude-2-1-200k
326. How to Monitor Your LLM API Costs and Cut Spending by 90% - Helicone, accessed May 25, 2025, https://www.helicone.ai/blog/monitor-and-optimize-llm-costs
327. Optimizing LLM Performance and Cost: Squeezing Every Drop of Value - ZenML Blog, accessed May 25, 2025, https://www.zenml.io/blog/optimizing-llm-performance-and-cost-squeezing-every-drop-of-value
328. What Does It Cost to Build an AI System in 2025? A Practical Look at LLM Pricing, accessed May 25, 2025, https://www.businesswaretech.com/blog/what-does-it-cost-to-build-an-ai-system-in-2025-a-practical-look-at-llm-pricing

329. Understanding the cost of Large Language Models (LLMs) - TensorOps, accessed May 25, 2025, https://www.tensorops.ai/post/understanding-the-cost-of-large-language-models-llms
330. Tree-sitter slow on big files, yet. Am I the only one using this little ..., accessed May 25, 2025, https://www.reddit.com/r/neovim/comments/1fy7jln/treesitter_slow_on_big_files_yet_am_i_the_only/
331. State of Julia grammar for tree-sitter - General Usage, accessed May 25, 2025, https://discourse.julialang.org/t/state-of-julia-grammar-for-tree-sitter/74573
332. Hierarchical Context Pruning: Optimizing Real-World Code Completion with Repository-Level Pretrained Code LLMs - AAAI Publications, accessed May 25, 2025, https://ojs.aaai.org/index.php/AAAI/article/view/34782/36937
333. TreeSitter - the holy grail of parsing source code - Symflower, accessed May 25, 2025, https://symflower.com/en/company/blog/2023/parsing-code-with-tree-sitter/
334. tree-sitter size limitation (fails if code is >32kb) - Stack Overflow, accessed May 25, 2025, https://stackoverflow.com/questions/79507130/tree-sitter-size-limitation-fails-if-code-is-32kb
335. Behavior with giant source files · Issue #222 · tree-sitter/tree-sitter - GitHub, accessed May 25, 2025, https://github.com/tree-sitter/tree-sitter/issues/222
336. 18 Top JavaScript Graph Visualization Libraries to Use in 2025 - Monterail, accessed May 25, 2025, https://www.monterail.com/blog/javascript-libraries-data-visualization
337. Cytoscape.js, accessed May 25, 2025, https://js.cytoscape.org/
338. Javascript Graph Visualization | Tom Sawyer Software, accessed May 25, 2025, https://blog.tomsawyer.com/javascript-graph-visualization
339. Top 10 JavaScript Libraries for Knowledge Graph Visualization - Focal, accessed May 25, 2025, https://www.getfocal.co/post/top-10-javascript-libraries-for-knowledge-graph-visualization
340. Loading and using JSON for Cytoscape.js - Stack Overflow, accessed May 25, 2025, https://stackoverflow.com/questions/40317668/loading-and-using-json-for-cytoscape-js
341. Make Cytoscape.js Node a Link - Stack Overflow, accessed May 25, 2025, https://stackoverflow.com/questions/41108991/make-cytoscape-js-node-a-link
342. 20. Linkout — Cytoscape User Manual 3.10.3 documentation, accessed May 25, 2025, https://manual.cytoscape.org/en/stable/Linkout.html
343. git-client - NPM, accessed May 25, 2025, https://www.npmjs.com/package/git-client
344. Top 7+ SQL Open-Source Database Types in 2025: Complete List - Aloa, accessed May 25, 2025, https://aloa.co/blog/sql-open-source-database

345. The 10 best databases of 2025: features and latest changes - Baremon, accessed May 25, 2025, https://www.baremon.eu/10-best-databases-of-2025/
346. Best NoSQL Databases 2025 - TrustRadius, accessed May 25, 2025, https://www.trustradius.com/categories/nosql-databases
347. NoSQL databases: Types, use cases, and 8 databases to try in 2025 - NetApp Instaclustr, accessed May 25, 2025, https://www.instaclustr.com/education/nosql-database/nosql-databases-types-use-cases-and-8-databases-to-try/
348. Tech Stack for LLM Application Development – Complete Guide - Prismetric, accessed May 25, 2025, https://www.prismetric.com/tech-stack-for-llm-application-development/
349. www.google.com, accessed May 25, 2025, https://www.google.com/search?q=best+open+source+backend+frameworks+2025+Java
350. 5ly.co, accessed May 25, 2025, https://5ly.co/blog/best-backend-frameworks/#:~:text=Key%20Takeaways%3A%20the%20Best%20Backend%20Frameworks%20in%202025&text=%E2%9C%85%20Top%20frameworks%20this%20year,%2C%20and%20long%2Dterm%20maintainability.
351. Top Node.js frameworks to learn in 2025 - Reddit, accessed May 25, 2025, https://www.reddit.com/r/node/comments/1kc0dbg/top_nodejs_frameworks_to_learn_in_2025/
352. List of Best Golang Web Frameworks of 2025 - Bacancy Technology, accessed May 25, 2025, https://www.bacancytechnology.com/blog/golang-web-frameworks
353. Best Backend Frameworks for Web Development in 2025 - Hostman, accessed May 25, 2025, https://hostman.com/blog/best-backend-frameworks-for-web-development-in-2025/
354. Top 7 Frontend Frameworks to Use in 2025: Pro Advice - Developer Roadmaps, accessed May 25, 2025, https://roadmap.sh/frontend/frameworks
355. Review Board: It's a bright day for code review!, accessed May 25, 2025, https://www.reviewboard.org/
356. Diff Highlight - Prism.js, accessed May 25, 2025, https://prismjs.com/plugins/diff-highlight/
357. Prism, accessed May 25, 2025, https://prismjs.com/
358. accessed January 1, 1970, https://www.sonarsource.com/products/sonarqube/product-overview/
359. Free linter in your AI IDE Tool & Real-Time Software for Code with ..., accessed May 25, 2025, https://www.sonarsource.com/products/sonarlint/
360. Making rulesets | PMD Source Code Analyzer, accessed May 25, 2025, https://docs.pmd-code.org/latest/pmd_userdocs_making_rulesets.html
361. awslabs/automated-security-helper - GitHub, accessed May 25, 2025, https://github.com/awslabs/automated-security-helper
362. Running ASH in a CI environment - ASH - Automated Security Helper,

accessed May 25, 2025, https://awslabs.github.io/automated-security-helper/tutorials/running-ash-in-ci/
363. Apache License 2.0 - awslabs/automated-security-helper - GitHub, accessed May 25, 2025, https://github.com/awslabs/automated-security-helper/blob/main/LICENSE
364. ASH - Automated Security Helper, accessed May 25, 2025, https://awslabs.github.io/automated-security-helper/
365. RAG LLMs are Not Safer: A Safety Analysis of Retrieval-Augmented Generation for Large Language Models - arXiv, accessed May 25, 2025, https://arxiv.org/html/2504.18041v1
366. Code Smell - Martin Fowler, accessed May 25, 2025, https://martinfowler.com/bliki/CodeSmell.html
367. accessed January 1, 1970, https://martinfowler.com/articles/technical-debt.html
368. accessed January 1, 1970, https://ieeexplore.ieee.org/document/7424404
369. accessed January 1, 1970, https://codescene.com/blog/prioritize-technical-debt-with-code-complexity-and-hotspots/
370. Securing Proprietary Code in the Age of AI Coding Assistants - RBA Consulting, accessed May 25, 2025, https://www.rbaconsulting.com/blog/securing-proprietary-code-in-the-age-of-ai-coding-assistants/
371. Static code analysis explained & best tools - Snyk, accessed May 25, 2025, https://snyk.io/articles/open-source-static-code-analysis/
372. Source Code Analysis Tools - OWASP Foundation, accessed May 25, 2025, https://owasp.org/www-community/Source_Code_Analysis_Tools
373. accessed January 1, 1970, https://snyk.io/learn/application-security/sast-security/
374. LLM Implementation and Maintenance Costs for Businesses: A Detailed Breakdown, accessed May 25, 2025, https://inero-software.com/llm-implementation-and-maintenance-costs-for-businesses-a-detailed-breakdown/
375. How Much Does it Cost to Build an AI System? - ProjectPro, accessed May 25, 2025, https://www.projectpro.io/article/cost-of-ai/1087
376. Best Practices for Production-Scale RAG Systems — An ... - Orkes, accessed May 25, 2025, https://orkes.io/blog/rag-best-practices/
377. How to achieve better performance with RAG ? : r/LangChain - Reddit, accessed May 25, 2025, https://www.reddit.com/r/LangChain/comments/1is53f2/how_to_achieve_better_performance_with_rag/
378. Mastering RAG: How To Architect An Enterprise RAG System - Galileo AI, accessed May 25, 2025, https://galileo.ai/blog/mastering-rag-how-to-architect-an-enterprise-rag-system
379. Cody vs. Cursor: Choosing the Right AI Code Assistant for Your Development Workflow, accessed May 25, 2025,

https://www.devtoolsacademy.com/blog/cody-vs-cursor-choosing-the-right-ai-code-assistant-for-your-development-workflow/
380. Core Components of an AI Evaluation System - Walturn, accessed May 25, 2025, https://www.walturn.com/insights/core-components-of-an-ai-evaluation-system
381. Model Drift: Best Practices to Improve ML Model Performance - Encord, accessed May 25, 2025, https://encord.com/blog/model-drift-best-practices/
382. Essential Work Samples for Evaluating AI Model Performance Monitoring Skills - Yardstick, accessed May 25, 2025, https://www.yardstick.team/work-samples/essential-work-samples-for-evaluating-ai-model-performance-monitoring-skills
383. Navigating the Risks of AI-Generated Code: A Guide for Business Leaders, accessed May 25, 2025, https://thecoderegistry.com/navigating-the-risks-of-ai-generated-code-a-guide-for-business-leaders/
384. How to Integrate Collaboration Tools into Design Workflows - UXPin, accessed May 25, 2025, https://www.uxpin.com/studio/blog/how-to-integrate-collaboration-tools-into-design-workflows/
385. Security Risks Of AI In Software Development: What You Need To Know - OpsMx, accessed May 25, 2025, https://www.opsmx.com/blog/security-risks-of-ai-in-software-development-what-you-need-to-know/
386. GitHub Code Review · GitHub, accessed May 25, 2025, https://github.com/features/code-review
387. What is AI code generation? - GitHub, accessed May 25, 2025, https://github.com/resources/articles/ai/what-is-ai-code-generation
388. Configuring automatic code review by Copilot - GitHub Docs, accessed May 25, 2025, https://docs.github.com/en/copilot/using-github-copilot/code-review/configuring-automatic-code-review-by-copilot
389. Code Review with ChatGPT · Actions · GitHub Marketplace, accessed May 25, 2025, https://github.com/marketplace/actions/code-review-with-chatgpt
390. Using GitHub Copilot code review, accessed May 25, 2025, https://docs.github.com/copilot/using-github-copilot/code-review/using-copilot-code-review
391. GitHub Copilot code review in GitHub.com (private preview) - GitHub Changelog, accessed May 25, 2025, https://github.blog/changelog/2024-10-29-github-copilot-code-review-in-github-com-private-preview/
392. Junfeng Yang, Systems, Computer Science Professor, Columbia University, accessed May 25, 2025, http://www.cs.columbia.edu/~junfeng/publications.html
393. Code Review Guidelines - GitLab Docs, accessed May 25, 2025, https://docs.gitlab.com/development/code_review/
394. GitLab architecture overview | GitLab Docs, accessed May 25, 2025,

https://docs.gitlab.com/ee/development/architecture.html
395. doc/development/fe_guide/architecture.md · master - GitLab, accessed May 25, 2025, https://gitlab.com/gitlab-org/gitlab/-/blob/master/doc/development/fe_guide/architecture.md
396. GitLab - Code Review Automation with GenAI - Google Codelabs, accessed May 25, 2025, https://codelabs.developers.google.com/genai-for-dev-gitlab-code-review
397. System Design - Gerrit Code Review, accessed May 25, 2025, https://gerrit-review.googlesource.com/Documentation/dev-design.html
398. What is Gerrit and use cases of Gerrit ? - DevOpsSchool.com, accessed May 25, 2025, https://www.devopsschool.com/blog/what-is-gerrit-and-use-cases-of-gerrit/
399. Official documentation about "Gerrit Change Number" - Stack Overflow, accessed May 25, 2025, https://stackoverflow.com/questions/49264185/official-documentation-about-gerrit-change-number
400. Gerrit Code Review - /plugins/ REST API, accessed May 25, 2025, https://gerrit.openafs.org/Documentation/rest-api-plugins.html
401. Code Review with Phabricator - an open source, sof... - SAP Community, accessed May 25, 2025, https://community.sap.com/t5/application-development-and-automation-blog-posts/code-review-with-phabricator-an-open-source-software-engineering-platform/ba-p/13095585
402. AI Code Reviews - GitHub, accessed May 25, 2025, https://github.com/resources/articles/ai/ai-code-reviews
403. AI Security Tools: The Open-Source Toolkit - Wiz, accessed May 25, 2025, https://www.wiz.io/academy/ai-security-tools
404. arxiv.org, accessed May 25, 2025, https://arxiv.org/abs/2308.07921
405. accessed January 1, 1970, https://testing.googleblog.com/2023/11/scaling-test-input-generation-with-llms.html
406. accessed January 1, 1970, https://www.semgrep.dev/blog/2023/semgrep-secrets/
407. arxiv.org, accessed May 25, 2025, https://arxiv.org/abs/2401.01337
408. arxiv.org, accessed May 25, 2025, https://arxiv.org/abs/2305.15334
409. accessed January 1, 1970, https://betterprogramming.pub/code-review-with-chatgpt-exploring-the-benefits-and-drawbacks-1a48a9c101c0
410. accessed January 1, 1970, https://thenewstack.io/can-ai-really-help-developers-review-code/
411. accessed January 1, 1970, https://www.synopsys.com/blogs/software-security/sast-vs-dast-vs-sca.html
412. accessed January 1, 1970, https://www.mend.io/resources/blog/top-10-sast-tools/

413. accessed January 1, 1970, https://www.infoworld.com/article/3688682/how-to-automate-code-reviews-with-ai.html
414. accessed January 1, 1970, https://aws.amazon.com/blogs/devops/automating-code-reviews-and-recommendations-using-amazon-codeguru-reviewer-and-amazon-codewhisperer/
415. Find Expensive Code – Amazon CodeGuru Features – AWS, accessed May 25, 2025, https://aws.amazon.com/codeguru/features/
416. accessed January 1, 1970, https://www.jetbrains.com/help/idea/static-code-analysis.html
417. accessed January 1, 1970, https://www.veracode.com/security/static-analysis-sast
418. accessed January 1, 1970, https://www.aicrowd.com/challenges/meta-comprehensive-code-analysis-challenge-codereview
419. accessed January 1, 1970, https://paperswithcode.com/task/automated-program-repair
420. Vulnerability Detection | Papers With Code, accessed May 25, 2025, https://paperswithcode.com/task/vulnerability-detection
421. Language Server Protocol - Wikipedia, accessed May 25, 2025, https://en.wikipedia.org/wiki/Language_Server_Protocol
422. What is the difference between LSP and Linting? : r/neovim - Reddit, accessed May 25, 2025, https://www.reddit.com/r/neovim/comments/1amo8l6/what_is_the_difference_between_lsp_and_linting/
423. Hands-On LSP Tutorial: Building a Custom Auto-Complete - Prefab.cloud, accessed May 25, 2025, https://prefab.cloud/blog/lsp-language-server-from-zero-to-completion/
424. What are good tactics for implementing a language server?, accessed May 25, 2025, https://langdev.stackexchange.com/questions/594/what-are-good-tactics-for-implementing-a-language-server
425. Language Server Protocol Specification - 3.17 - Microsoft Open ..., accessed May 25, 2025, https://microsoft.github.io/language-server-protocol/specifications/lsp/3.17/specification/
426. Programmatic Language Features | Visual Studio Code Extension API, accessed May 25, 2025, https://code.visualstudio.com/api/language-extensions/programmatic-language-features
427. GNU Affero General Public License - GNU Project - Free Software Foundation, accessed May 25, 2025, https://www.gnu.org/licenses/agpl-3.0.html
428. GNU Affero General Public License - GNU Project - Free Software ..., accessed May 25, 2025, https://www.gnu.org/licenses/agpl-3.0.en.html
429. accessed January 1, 1970,

https://raw.githubusercontent.com/kodustech/kodus-ai/main/license.md
430. Pricing - Kodus, accessed May 25, 2025, https://kodus.io/en/pricing/
431. Does AGPL-3.0 require open-sourcing the derivatives if the original work is open-source?, accessed May 25, 2025, https://opensource.stackexchange.com/questions/15159/does-agpl-3-0-require-open-sourcing-the-derivatives-if-the-original-work-is-open
432. Question regarding the AGPL-3.0 license · Issue #2129 - GitHub, accessed May 25, 2025, https://github.com/ultralytics/ultralytics/issues/2129
433. Licenses - GNU Project - Free Software Foundation, accessed May 25, 2025, https://www.gnu.org/licenses/licenses.html
434. GNU Affero General Public License v3 (AGPL-3.0) Explained in Plain English - TLDRLegal, accessed May 25, 2025, https://www.tldrlegal.com/license/gnu-affero-general-public-license-v3-agpl-3-0
435. Is an AGPL License the Right Choice for Your Open Source Projects? - Snyk, accessed May 25, 2025, https://snyk.io/articles/agpl-license/
436. AGPL3 License · GitHub - Gist, accessed May 25, 2025, https://gist.github.com/balupton/492562
437. GNU Affero General Public License - Wikipedia, accessed May 25, 2025, https://en.wikipedia.org/wiki/GNU_Affero_General_Public_License
438. kodus-ai/license_ee.md at main - GitHub, accessed May 25, 2025, https://github.com/kodustech/kodus-ai/blob/main/license_ee.md