ĐẠI HỌC QUỐC GIA HÀ NỘI
TRƯỜNG ĐẠI HỌC CÔNG NGHỆ

BÁO CÁO THỰC TẬP

NGÀNH: CÔNG NGHỆ THÔNG TIN

ĐỀ TÀI: ĐÁNH GIÁ VÀ SO SÁNH HIỆU QUẢ GIỮA DỊCH VỤ AI TỔNG QUÁT VÀ AI CHUYÊN BIỆT CHO LẬP TRÌNH TRONG HỖ TRỢ PHÁT TRIỂN PHẦN MỀM

Giảng viên hướng dẫn: TS Lê Đức Trọng

Sinh viên: Tạ Hải An
Mã sinh viên: 22028242
Lớp: K67I-CS4

Hà Nội, tháng 9 năm 2025

MỤC LỤC

LỜI CẢM ƠN

Trước hết, em xin bày tỏ lòng biết ơn sâu sắc tới **TS Lê Đức Trọng**, giảng viên hướng dẫn, người đã nhiệt tình chỉ bảo, định hướng và hỗ trợ em trong suốt quá trình thực hiện báo cáo thực tập doanh nghiệp. Những ý kiến đóng góp quý báu của thầy đã giúp em hoàn thiện tư duy nghiên cứu cũng như tiếp cận vấn đề một cách khoa học và thực tiễn hơn.

Em cũng xin gửi lời cảm ơn đến **Khoa Công nghệ Thông tin, Trường Đại học Công nghệ – Đại học Quốc gia Hà Nội**, đã tạo điều kiện thuận lợi cho em trong quá trình học tập và thực hiện đề tài.

Mặc dù đã có nhiều cố gắng, song báo cáo khó tránh khỏi những thiếu sót. Em rất mong nhận được sự chỉ dẫn, góp ý của thầy và các thầy cô trong Khoa để hoàn thiện hơn trong những nghiên cứu và công việc sau này.

Hà Nội, tháng 9 năm 2025

Sinh viên thực hiện

Tạ Hải An

I. GIỚI THIỆU CHUNG

a. Giới thiệu công việc

Với học phần Thực tập doanh nghiệp, em lựa chọn một chủ đề có tính thời sự và gắn liền với xu hướng công nghệ hiện nay: **so sánh và đánh giá một số công cụ trí tuệ nhân tạo (AI) trong việc hỗ trợ phát triển phần mềm hiện nay**. Đây là một chủ đề vừa mang ý nghĩa nghiên cứu học thuật, vừa có tính ứng dụng thực tiễn cao trong công việc của lập trình viên.

Công việc chính mà em thực hiện bao gồm bốn giai đoạn:

1. **Thu thập và phân tích tài liệu kỹ thuật** từ các nhà phát triển công cụ, bao gồm tài liệu chính thức, bài viết nghiên cứu, benchmark học thuật (HumanEval, MBPP, CodeXGLUE) và các báo cáo trải nghiệm người dùng.

2. **Thiết kế bộ bài tập kiểm thử thực tế**, bao gồm các tác vụ phổ biến trong lập trình: sinh mã từ mô tả tự nhiên, sửa lỗi (debug), viết test case (unit test), và tái cấu trúc mã (refactor). Đây là các tình huống mà lập trình viên thường xuyên gặp phải, nên có thể phản ánh rõ hiệu quả của AI trong thực tế.

3. **Triển khai chạy thử nghiệm trực tiếp trên các công cụ AI** thuộc hai nhóm:

    - Nhóm AI tổng quát, được phát triển cho mọi tác vụ (General AI): ChatGPT, Gemini, Claude.

    - Nhóm AI được huấn luyện chuyên biệt cho lập trình (Programming AI): GitHub Copilot, Cursor, Windsurf (trước đây là Codeium).

4. **Ghi nhận và phân tích kết quả** dựa trên nhiều tiêu chí: chất lượng mã sinh ra, độ chính xác, khả năng hiểu ngữ cảnh, tốc độ phản hồi, mức độ tương tác và sự tiện lợi trong môi trường phát triển phần mềm (IDE).

Thông qua quá trình này, em không chỉ đánh giá **định lượng năng lực kỹ thuật** của các mô hình AI nền tảng, mà còn làm rõ **trải nghiệm định tính thực tế của lập trình viên** khi sử dụng các công cụ AI khác nhau.

b. Giới thiệu qua bài toán

Bài toán nghiên cứu của em bắt nguồn từ thực tiễn: **AI đang ngày càng được tích hợp vào phát triển phần mềm**, nhưng chưa có nhiều báo cáo mang tính hệ thống, khách quan và thực nghiệm để hỗ trợ lập trình viên lựa chọn công cụ phù hợp với nhu cầu.

Bài toán lớn có thể được mô tả như sau:

- Làm rõ sự khác biệt giữa **AI tổng quát (General AI)** và **AI chuyên biệt cho lập trình (Programming AI)**.
- Đánh giá hiệu quả của từng công cụ thông qua thực nghiệm, cả ở góc độ định lượng (benchmark, độ chính xác) và định tính (trải nghiệm người dùng).
- So sánh ưu điểm, hạn chế, cũng như mức độ phù hợp của mỗi nhóm công cụ.
- Đưa ra các **khuyến nghị sử dụng** tùy thuộc vào ngữ cảnh: học tập, lập trình cá nhân, hay phát triển dự án lớn.

Trong khuôn khổ báo cáo thực tập, em tập trung giải quyết một phần cụ thể của bài toán này:

- **Xây dựng bộ benchmark thử nghiệm** gồm bốn tác vụ chính (Task 1–4) và tham chiếu thêm các bộ benchmark chuẩn (HumanEval, MBPP).
- **Đo lường và phân tích kết quả** thu được từ việc áp dụng thực nghiệm trên các công cụ AI.
- **Liên hệ và đối chiếu** với những nghiên cứu, số liệu đã được công bố để làm cơ sở học thuật cho phân tích.

Phương pháp thực hiện

Để đảm bảo tính khách quan và khả năng ứng dụng, em áp dụng các phương pháp sau:

- **Thiết kế kịch bản kiểm thử thực tế**: các tác vụ được xây dựng sao cho vừa đủ khó để kiểm tra khả năng AI, vừa sát với công việc thường ngày của lập trình viên.
- **Sử dụng API và môi trường lập trình**: các công cụ AI được triển khai và chạy thử trực tiếp trong môi trường phát triển (IDE như VSCode, Cursor) và giao diện chat (ChatGPT, Claude, Gemini).

- **Phân tích kết quả theo bộ tiêu chí định sẵn**, gồm: độ chính xác, tính đầy đủ, khả năng hiểu ngữ cảnh, mức độ diễn giải, tốc độ phản hồi.
- **So sánh đối chiếu với benchmark chuẩn**: kết quả thực nghiệm được tham chiếu với các nghiên cứu có sẵn để tăng độ tin cậy.

Cách tiếp cận này giúp báo cáo không chỉ mang tính thực nghiệm cá nhân, mà còn có cơ sở khoa học vững chắc, kết hợp giữa **ứng dụng thực tế** và **học thuật chuẩn mực**.

II. YÊU CẦU BÀI TOÁN

a. Mô tả chi tiết bài toán

Bài toán của đề tài là **so sánh và đánh giá hai nhóm dịch vụ AI trong hỗ trợ lập trình**, bao gồm:

1) **General AI (AI tổng quát):**
   - Các công cụ tiêu biểu: ChatGPT (GPT-4), Gemini (Gemini 2.5), Claude (Claude 3).
   - Đặc điểm chung: được xây dựng trên các mô hình ngôn ngữ lớn (LLM) mạnh mẽ, có khả năng xử lý đa dạng tác vụ, từ viết văn bản, trả lời câu hỏi cho đến sinh mã lập trình.
   - Ưu điểm: đa năng, giao diện đơn giản (chat-based), phù hợp khi cần giải thích, học tập, hoặc sinh code từ mô tả tự nhiên.
2) **Specialized AI (AI chuyên lập trình)**:
   - Các công cụ tiêu biểu: GitHub Copilot, Cursor, Windsurf (tiền thân là Codeium).
   - Đặc điểm chung: tích hợp trực tiếp trong môi trường phát triển (IDE), được tinh chỉnh tối ưu cho nhiệm vụ lập trình, gợi ý mã theo thời gian thực, hỗ trợ refactor và debug ngay trong codebase.
   - Ưu điểm: tốc độ phản hồi nhanh, gắn liền với workflow của lập trình viên, thuận tiện cho dự án vừa và lớn.

Mục tiêu chính:

- Xác định khả năng của từng nhóm AI trong việc hỗ trợ các tác vụ lập trình cơ bản và nâng cao.
- So sánh ưu điểm, hạn chế của mỗi nhóm, cả về khía cạnh kỹ thuật (chất lượng mã, khả năng hiểu ngữ cảnh) và khía cạnh trải nghiệm người dùng (sự tiện lợi, tốc độ, mức độ tương tác).
- Đưa ra khuyến nghị sử dụng phù hợp tùy theo ngữ cảnh: học tập, nghiên cứu, lập trình cá nhân, hay phát triển phần mềm chuyên nghiệp.

b. Các tác vụ kiểm thử

Để phản ánh nhu cầu thực tế của lập trình viên, đề tài xây dựng **bộ kiểm thử gồm 7 tác vụ phổ biến** trong quá trình phát triển phần mềm:

- **Sinh mã từ yêu cầu tự nhiên**: kiểm tra khả năng của AI trong việc chuyển đổi mô tả ngôn ngữ tự nhiên thành đoạn mã Python.
- **Sửa lỗi (Debugging)**: cung cấp đoạn mã có lỗi cú pháp hoặc lỗi logic, yêu cầu AI phát hiện và sửa.
- **Hiểu và diễn giải code**: đánh giá khả năng đọc, phân tích và giải thích một đoạn mã có sẵn.
- **Refactor & tối ưu**: yêu cầu cải thiện mã nguồn bằng cách đặt tên biến/hàm rõ ràng hơn, giảm độ phức tạp, hoặc tăng hiệu suất.
- **Viết unit test**: cung cấp hàm hoặc đoạn code, yêu cầu AI sinh ra test case bằng pytest.
- **Truy xuất file/project lớn**: kiểm tra khả năng AI hiểu ngữ cảnh nhiều file cùng lúc.
- **Tạo tài liệu/hướng dẫn sử dụng**: yêu cầu AI sinh docstring, README hoặc hướng dẫn ngắn gọn để người dùng dễ tiếp cận.

Bảy tác vụ này bao quát nhiều khía cạnh: từ sinh mã, phân tích, sửa lỗi, tối ưu, cho đến hỗ trợ quản lý dự án lớn và tạo tài liệu. Nhờ đó, việc đánh giá có thể toàn diện, phản ánh được **cả năng lực mô hình AI lẫn tính hữu ích thực tế**.

c. Tiêu chí đánh giá

Mỗi công cụ AI được chấm điểm theo **5 tiêu chí chính**, thang điểm 0–10:

- **Độ chính xác (Accuracy)**: mã sinh ra có đúng logic, chạy được và đáp ứng yêu cầu hay không.
- **Tính đầy đủ (Completeness)**: AI có bao quát toàn bộ yêu cầu hay bỏ sót một phần.
- **Khả năng hiểu ngữ cảnh (Context-awareness)**: khả năng xử lý khi có nhiều hàm, nhiều biến hoặc nhiều file liên quan.
- **Mức độ diễn giải (Explainability)**: AI có giải thích kết quả rõ ràng, dễ hiểu, hữu ích cho người học/lập trình viên.
- **Tốc độ và độ trễ (Responsiveness)**: thời gian phản hồi của công cụ có nhanh và ổn định không.

Thang điểm tổng hợp cho mỗi tác vụ là 50 điểm, tổng điểm toàn bộ công cụ tối đa là 350 điểm (7 tác vụ).

d. Phương pháp tiếp cận

Đề tài sử dụng **hai hướng tiếp cận song song**:

- **Benchmark tự xây dựng** (Task 1–4): phản ánh trải nghiệm trực tiếp khi sử dụng AI trong các tình huống thực tế.
- **Tham chiếu benchmark chuẩn** (HumanEval, MBPP, CodeXGLUE, EvalPlus): bổ sung tính khách quan, đảm bảo kết quả có cơ sở đối chiếu học thuật.

Cách tiếp cận này giúp báo cáo vừa mang tính **ứng dụng thực tiễn**, vừa đảm bảo tính học thuật từ những chuẩn mực nghiên cứu đã được cộng đồng công nhận.

e. Phạm vi triển khai

- **Ngôn ngữ chính**: Python, do phổ biến và được hầu hết các công cụ AI hỗ trợ mạnh mẽ.
- **Công cụ sử dụng**: các phiên bản cập nhật mới của ChatGPT, Gemini, Claude, Copilot, Cursor, Windsurf (tại thời điểm của báo cáo - tháng 8/2025).

- **Quy mô thực hiện**: cá nhân, toàn bộ quy trình từ thiết kế, chạy kiểm thử đến phân tích kết quả được triển khai độc lập, đảm bảo tính chủ động và thống nhất.

III. TÓM TẮT LÝ THUYẾT, GIẢI PHÁP, THUẬT TOÁN

a. Các lý thuyết, giải pháp, thuật toán liên quan

Trong những năm gần đây, sự phát triển của **mô hình ngôn ngữ lớn (LLM – Large Language Models)** dựa trên kiến trúc **Transformer** đã tạo bước ngoặt lớn trong cả nghiên cứu và ứng dụng thực tế. Kiến trúc Transformer (Vaswani et al., 2017) cho phép mô hình học ngữ cảnh dài và mối quan hệ giữa các từ hoặc token trong chuỗi, nhờ đó có khả năng sinh văn bản và mã nguồn mạch lạc, hợp logic. Những thành tựu này đóng một vai trò rất lớn trong lĩnh vực phát triển phần mềm - một trong những lĩnh vực khó nhất của khoa học công nghệ và kỹ thuật.

Trên nền tảng đó, hai hướng ứng dụng AI chính trong lập trình được hình thành:

- **Ứng dụng AI tổng quát (General-purpose AI):**
  Các công cụ như ChatGPT (GPT-4), Gemini (Gemini 2.5), Claude (Claude 3)… đều được huấn luyện trên tập dữ liệu đa dạng, bao gồm cả văn bản ngôn ngữ tự nhiên và mã nguồn. Điểm mạnh của nhóm này:
    - Hiểu ngữ cảnh dài (100k+ tokens).
    - Giao tiếp tự nhiên qua hội thoại.
    - Có thể giải thích, phân tích hoặc sinh mã từ mô tả ngôn ngữ tự nhiên.
- **Ứng dụng AI chuyên biệt cho lập trình (Programming-specialized AI):** Các công cụ như GitHub Copilot, Cursor IDE, Windsurf (SWE-1) được huấn luyện và tối ưu riêng cho lập trình. Điểm mạnh của nhóm này:
    - Tích hợp trực tiếp vào IDE (VSCode, JetBrains, IntelliJ).
    - Gợi ý code inline theo thời gian thực.
    - Hỗ trợ refactor, debug, viết test case, sinh tài liệu trong workflow.

Ngoài ra, cộng đồng nghiên cứu AI đã xây dựng nhiều **benchmark chuẩn** để đánh giá khả năng sinh mã của LLM. Một số tiêu biểu gồm:

- **HumanEval (OpenAI, 2021):** gồm 164 bài toán Python kèm test case. Đây là benchmark phổ biến nhất, dùng để đo độ chính xác khi mô hình chuyển mô tả → mã.
- **MBPP (Google, 2021):** gồm 974 bài toán lập trình cơ bản, đa dạng hơn HumanEval, phản ánh nhiều tình huống gần gũi hơn với lập trình hằng ngày.
- **CodeXGLUE:** tập hợp hơn 10 tập con, bao gồm nhiều tác vụ như code completion, code translation, defect detection. Đây là benchmark toàn diện nhưng phức tạp và khó triển khai đầy đủ.
- **EvalPlus (2023):** bản mở rộng HumanEval & MBPP, bổ sung test case → giúp kết quả đánh giá đáng tin cậy hơn, tránh hiện tượng "ghi nhớ bài cũ" của mô hình.

Phần so sánh chi tiết được thể hiện trong bảng 1 sau đây.

Bảng 1: So sánh một số benchmark chuẩn trong đánh giá AI sinh mã

Benchmark	Quy mô & Nội dung	Mục tiêu chính	Ưu điểm	Hạn chế
HumanEval (OpenAI, 2021)	164 bài toán Python, có sẵn test case	Đánh giá khả năng sinh mã từ mô tả tự nhiên	Đơn giản, phổ biến, dễ so sánh giữa các nghiên cứu	Quy mô nhỏ, ít đa dạng, chỉ có Python
MBPP (Google, 2021)	974 bài toán cơ bản, nhiều chủ đề	Kiểm tra khả năng giải quyết các tác vụ lập trình thường gặp	Quy mô lớn hơn HumanEval, phản ánh lập trình thực tế hơn	Một số bài toán vẫn đơn giản, chưa bao quát bài toán phức tạp

CodeXGLUE	Hơn 10 tập con (code completion, translation, defect detection…)	Đánh giá toàn diện nhiều khía cạnh của AI lập trình	Phủ rộng nhiều tác vụ, hỗ trợ nhiều ngôn ngữ	Rất lớn, khó triển khai toàn bộ trong nghiên cứu cá nhân
EvalPlus (2023)	Mở rộng HumanEval & MBPP với test case bổ sung	Nâng cao độ tin cậy khi đánh giá mô hình sinh mã	Kết quả chính xác hơn, tránh overfitting vào test cũ	Mới, ít công trình ứng dụng rộng rãi, chưa đa dạng ngôn ngữ

b. Cách giải quyết của sinh viên

Trong khuôn khổ đề tài thực tập này, em không chạy trực tiếp toàn bộ các bộ benchmark chuẩn (do hạn chế tài nguyên), mà lựa chọn hướng tiếp cận kết hợp:

- **Tự xây dựng bộ kiểm thử gồm 7 tác vụ** phản ánh sát thực tế công việc lập trình (sinh mã, sửa lỗi, viết test case, giải thích code, refactor, phân tích nhiều file, sinh tài liệu).
- **Tham chiếu kết quả từ benchmark chuẩn** (HumanEval, MBPP, EvalPlus) để bổ sung cơ sở khoa học và đối chiếu kết quả quan sát.

c. Liên hệ và so sánh với các cách đã có

So với hướng tiếp cận thuần học thuật (chạy benchmark trên mô hình nền), cách làm trong báo cáo này có những điểm khác biệt:

- **Tính thực tiễn cao:** Bộ kiểm thử tự thiết kế mô phỏng sát tình huống lập trình thực tế, thể hiện rõ giá trị sử dụng của công cụ AI đối với lập trình viên.

- **Giữ được độ tin cậy học thuật:** Việc tham chiếu benchmark chuẩn giúp kết quả không chỉ dựa trên trải nghiệm chủ quan mà còn gắn với những chỉ số đã được cộng đồng nghiên cứu công nhận.

Như vậy, giải pháp của đề tài đạt được **sự cân bằng** giữa hai khía cạnh:

- Thực tiễn (UX evaluation: đánh giá trải nghiệm người dùng).
- Học thuật (benchmark evaluation: đối chiếu chuẩn nghiên cứu).

IV. MÔ TẢ PHẦN MỀM CÀI ĐẶT

Link mã nguồn Github:

https://github.com/Tahaian22028242/AI_Coding_Benchmark_Kit

a. Giới thiệu chung

Trong quá trình thực tập, để tiến hành đánh giá và so sánh hiệu quả giữa hai nhóm công cụ AI (General AI và Specialized AI for Programming), em đã sử dụng **AI Coding Benchmark Kit** – một bộ công cụ mã nguồn mở viết bằng Python. Bộ công cụ này cho phép tổ chức và chạy các bài kiểm thử chuẩn (benchmark) như **HumanEval**, **MBPP**, đồng thời hỗ trợ xuất kết quả dưới dạng bảng dữ liệu và biểu đồ trực quan.

Mặc dù không phải do em tự phát triển từ đầu, nhưng em đã tiến hành cài đặt, cấu hình môi trường, và tùy chỉnh một số thành phần (chạy subset của **HumanEval**, tích hợp một số API của các LLM như Gemini API và GPT-4o… để gửi yêu cầu và nhận phản hồi) nhằm phục vụ đúng mục tiêu nghiên cứu.

b. Cấu trúc và thành phần chính

Dự án **AI Coding Benchmark Kit** bao gồm một số tệp và thư mục chính sau:

- **ai_benchmark/**: thư mục mẹ của dự án.
- **ai_outputs/:** lưu lại phản hồi của một số dịch vụ AI (ChatGPT, Gemini) khi gửi prompt thủ công.

- **tasks/**: tập các bài toán, vấn đề lập trình mẫu, được sử dụng làm prompt gửi cho AI.
- **tools/**: script công cụ hỗ trợ - xử lý dữ liệu, chạy subset HumanEval, xuất dữ liệu JSON/CSV, vẽ biểu đồ kết quả, v.v.
- **results/**: lưu kết quả chạy benchmark dưới dạng tệp văn bản, ảnh.
- **requirements.txt**: danh sách thư viện cần cài đặt.
- **run_benchmark.py:** mã Python chạy và chấm điểm code do các dịch vụ AI tạo sinh.
- **README.md**: hướng dẫn sử dụng.

c. Quy trình cài đặt và sử dụng

(Chi tiết xem thêm tại file **README.md**)

1) Chuẩn bị môi trường

- Cài đặt Python 3.10+ và pip.
- Tạo môi trường ảo (virtualenv) để quản lý gói.
- Cài đặt các thư viện cần thiết bằng lệnh:

```
pip install -r requirements.txt
```

2) Cấu hình API

- Tạo API key cho các công cụ AI cần so sánh (ví dụ: OpenAI GPT-4o, Google Gemini).
- Lưu key vào biến môi trường hoặc file cấu hình.

3) Chạy kiểm thử

- Sử dụng script có sẵn, ví dụ:

```
python ai_benchmark/tools/humaneval_run_subset.py
```

- Kết quả sẽ được sinh ra trong thư mục **results/**.

4) Phân tích và trực quan hóa kết quả

- Sử dụng script `plot_results.py` để vẽ biểu đồ so sánh.

- Kết quả phân tích được lưu dưới dạng CSV, JSON và biểu đồ dạng ảnh PNG để thuận tiện cho việc báo cáo.

d. Vai trò và đóng góp cá nhân

Trong quá trình triển khai benchmark, em đã thực hiện các công việc sau:

- **Cấu hình và tích hợp API**: Thiết lập để chạy song song hai công cụ AI tổng quát (ChatGPT/GPT-4o, Gemini/Gemini 2.5, Claude/Claude 3) và AI chuyên biệt (Copilot, Cursor, Windsurf).
- **Tùy chỉnh tập kiểm thử**: Chạy subset HumanEval thay vì toàn bộ để phù hợp với giới hạn thời gian và hạn mức API (miễn phí).
- **Thu thập và xử lý kết quả**: Xuất kết quả sang CSV, tổng hợp điểm số, và trực quan hóa thành biểu đồ để phục vụ so sánh.
- **Phân tích kết quả**: Từ dữ liệu benchmark, em rút ra nhận xét định tính và định lượng, liên hệ với trải nghiệm thực tế khi sử dụng các công cụ AI.

V. KẾT QUẢ ĐẠT ĐƯỢC, HƯỚNG PHÁT TRIỂN

a. Kết quả đạt được

Sau quá trình thực tập và nghiên cứu đề tài **"ĐÁNH GIÁ, SO SÁNH 2 LOẠI AI: GENERAL AI SERVICE VÀ AI SPECIALIZED FOR PROGRAMMING TRONG LẬP TRÌNH VỚI AI"**, em đã đạt được một số kết quả như sau:

1) **Hoàn thiện bộ benchmark thực nghiệm**
    - Bộ benchmark gồm **7 tác vụ lập trình cơ bản**, phản ánh sát nhu cầu của lập trình viên trong thực tế:
        - Sinh mã từ mô tả tự nhiên.
        - Sửa lỗi (debug).

- Viết unit test.
- Giải thích và diễn giải code.
- Refactor & tối ưu.
- Phân tích nhiều file/project lớn.
- Sinh tài liệu (docstring, README).

- Điểm nổi bật của bộ benchmark này là sự kết hợp giữa **chuẩn học thuật** (HumanEval, MBPP, EvalPlus) và **tình huống thực tế** mà em tự thiết kế.

2) Thực nghiệm trên 6 công cụ AI tiêu biểu

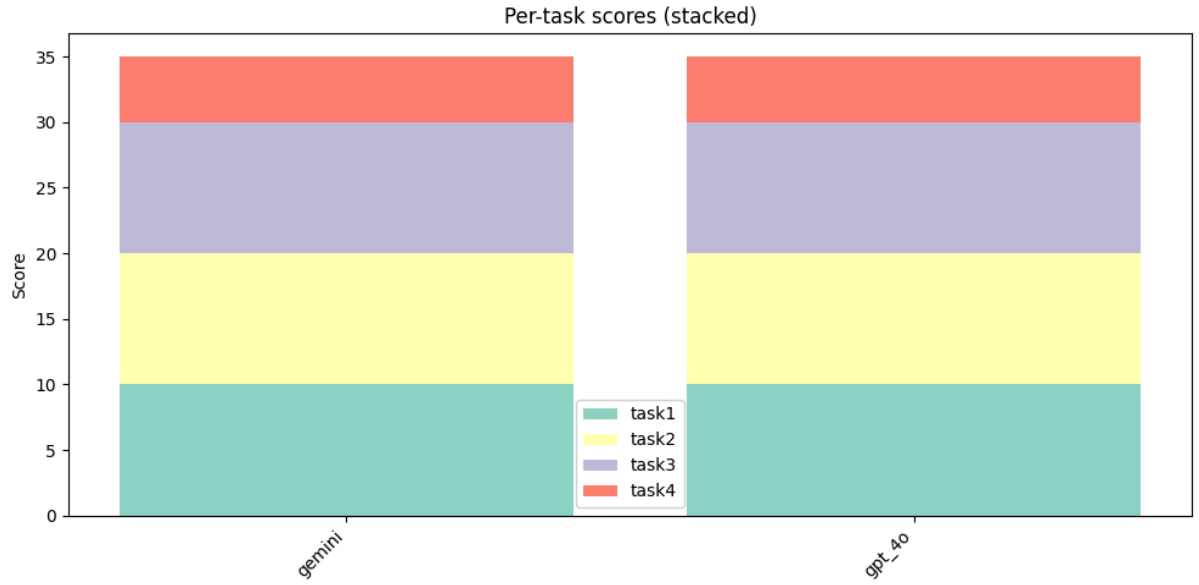
- **Nhóm AI tổng quát:** ChatGPT (GPT-4o), Gemini (Gemini 2.5), Claude (Claude 3).
- **Nhóm AI chuyên biệt:** GitHub Copilot, Cursor IDE, Windsurf (SWE-1).
- Kết quả được tổng hợp theo cả hai hướng: **định tính (trải nghiệm sử dụng)** và **định lượng (benchmark điểm số).**

Bảng 2: So sánh định tính qua trải nghiệm sử dụng

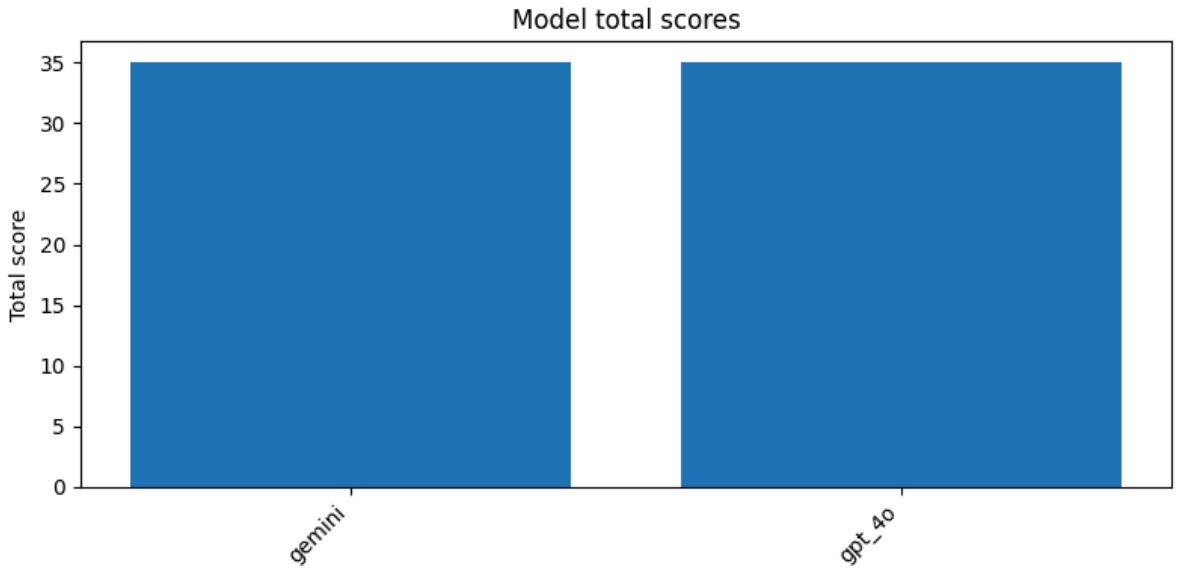
Tiêu chí	**AI tổng quát (ChatGPT, Gemini, Claude)**	**AI chuyên lập trình (Copilot, Cursor, Windsurf)**
Hiểu ngôn ngữ tự nhiên	Xuất sắc, có thể hiểu yêu cầu mơ hồ, hỗ trợ đa ngôn ngữ, giải thích chi tiết, dễ học	Hạn chế hơn, chủ yếu phản hồi tốt với prompt dạng code hoặc comment trong IDE
Gợi ý code	Sinh code đầy đủ từ mô tả chi tiết; mạnh khi viết hàm, module từ đầu	Gợi ý inline nhanh, sát ngữ cảnh đang lập trình; phù hợp tăng năng suất viết code thực tế

Debug & Refactor	Giải thích lỗi rõ ràng, đề xuất nhiều hướng tối ưu	Thao tác trực tiếp trong IDE: sửa lỗi, đổi tên biến, refactor project tự động
Làm việc với nhiều file	Có thể xử lý nếu copy/paste vào chat, nhưng bất tiện	Cursor/Windsurf hỗ trợ hiểu ngữ cảnh xuyên file, tiện lợi hơn nhiều
Sinh tài liệu (docstring, README)	Rất tốt, trình bày mạch lạc, cấu trúc đẹp, dễ hiểu	Sinh docstring nhanh, đơn giản, thường ngắn gọn
Tích hợp trong workflow	Hoạt động qua giao diện chat, không gắn liền IDE	Tích hợp IDE trực tiếp, inline suggestion theo thời gian thực

Một số ảnh biểu đồ trực quan so sánh định lượng:



Hình 1: So sánh điểm đánh giá hoàn thành 4 task đại diện: sinh mã, debug, viết unit test và tối ưu hóa code của 2 công cụ Gemini (Gemini 2.5) và GPT-4o



Hình 1: So sánh điểm giải quyết vấn đề tổng thể của 2 công cụ Gemini (Gemini 2.5) và GPT-4o

Ví dụ minh họa:

- Khi yêu cầu *"Viết hàm kiểm tra số nguyên tố trong Python và giải thích độ phức tạp thời gian"*, GPT-4o trả về lời giải kèm phân tích O(√n), rất hữu ích cho người học. Ngược lại, Copilot chỉ gợi ý code nhanh nhưng không đưa giải thích chi tiết.
- Khi thực hiện *refactor toàn bộ project nhỏ gồm 5 file*, Copilot và Cursor hỗ trợ rename biến xuyên file tự động, trong khi Gemini cần copy thủ công từng đoạn code để chỉnh.

3) Tổng hợp ưu nhược điểm

- **General AI:** mạnh ở khả năng hiểu ngôn ngữ tự nhiên, diễn giải chi tiết, hữu ích cho học tập và nghiên cứu.
- **Specialized AI:** mạnh ở tốc độ và tính tích hợp IDE, giúp tiết kiệm thời gian, tối ưu cho workflow lập trình chuyên nghiệp.

b. Kiến thức và kỹ năng thu thập được

Trong quá trình thực hiện đề tài, em đã rèn luyện và thu được:

- **Kiến thức chuyên môn:**
  - Hiểu sâu hơn về sự khác biệt giữa AI tổng quát và AI chuyên lập trình.

    - Làm quen và nắm được cách vận hành benchmark khoa học: HumanEval, MBPP, EvalPlus, CodeXGLUE.
    - Biết cách thiết kế, tổ chức và thực hiện một bộ kiểm thử AI có tính thực tiễn.

- **Kỹ năng thực hành:**
    - Nắm được cách tích hợp, cài đặt và sử dụng các công cụ AI lập trình phổ biến.
    - Rèn luyện kỹ năng so sánh, phân tích kết quả cả định tính và định lượng.
    - Phát triển kỹ năng viết báo cáo học thuật và trình bày kết quả nghiên cứu.
- **Kỹ năng mềm:**
    - Quản lý thời gian và và tài nguyên API (vì quota hạn chế). đọc và tự nghiên cứu tài liệu kỹ thuật.
    - Tư duy phản biện, đánh giá khách quan các công cụ công nghệ, không phụ thuộc hoàn toàn vào kết quả AI.

c. Hướng phát triển tiếp theo

Để mở rộng và hoàn thiện hơn đề tài, trong tương lai em sẽ phát triển theo các hướng sau:

Hướng kỹ thuật:

- Mở rộng phạm vi thử nghiệm sang nhiều ngôn ngữ khác (Java, C++, Rust, Go, TypeScript).
- Đánh giá thêm các tác vụ nâng cao: code security, performance optimization.
- Đo lường thời gian thực tế hoàn thành task *có AI* và *không có AI* để lượng hóa hiệu quả.

Hướng nghiên cứu & ứng dụng:

- Khảo sát trải nghiệm người dùng (lập trình viên chuyên nghiệp và sinh viên).
- Phân tích sâu về **vấn đề bản quyền và đạo đức** trong code sinh ra bởi AI.
- Nghiên cứu mô hình workflow lai: kết hợp General AI để phân tích, Specialized AI để triển khai, từ đó tận dụng ưu điểm song song của cả hai.

WEBSITE VÀ TÀI LIỆU THAM KHẢO

[1] OpenAI. *Introducing GPT-4*. OpenAI, 2023. Truy cập tại: https://openai.com/research/gpt-4

[2] OpenAI. *OpenAI API Documentation*. Truy cập tại: https://platform.openai.com/docs

[3] Google DeepMind. *Gemini: A Family of Highly Capable Multimodal Models*. Google DeepMind, 2023. Truy cập tại: https://deepmind.google/technologies/gemini

[4] Anthropic. *Introducing Claude*. Anthropic, 2023. Truy cập tại: https://www.anthropic.com/claude

[5] GitHub. *GitHub Copilot Documentation*. Truy cập tại: https://docs.github.com/copilot

[6] Cursor. *Cursor – The AI Code Editor*. Truy cập tại: https://cursor.sh

[7] Windsurf. *Windsurf - The best AI for coding*. Truy cập tại: https://windsurf.com/

[8] Chen, M. et al. *Evaluating Large Language Models Trained on Code*. arXiv:2107.03374 [cs.CL], 2021. (HumanEval benchmark)

[9] Austin, J. et al. *Program Synthesis with Large Language Models*. arXiv:2108.07732 [cs.LG], 2021. (MBPP benchmark)

[10] Lu, S. et al. *CodeXGLUE: A Machine Learning Benchmark Dataset for Code Understanding and Generation*. arXiv:2102.04664 [cs.SE], 2021.

[11] Li, J. et al. *EvalPlus: Better Evaluation for Code Generation Benchmarks*. arXiv:2305.01210 [cs.SE], 2023.

Ý kiến đánh giá:

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

Điểm số: ……. Điểm chữ: ……

Hà Nội, ngày tháng năm 20 .

Giảng viên đánh giá

(Ký, ghi rõ họ tên)