Đề tài thực tập: Đánh giá khách quan, so sánh ưu nhược điểm của 2 loại AI: General AI service (ví dụ ChatGPT, Gemini, Claude, Deepseek...) và AI specialized for Programming (ví dụ Cursor, Github Copilot...) trong lập trình với AI.

Giảng viên hướng dẫn: TS Lê Đức Trọng

Sinh viên thực hiện:

- Họ và tên: Tạ Hải An
- Mã sinh viên: 22028242

ChatGPT hướng dẫn:

1. Giới thiệu đề tài

Đối tượng đề tài: 2 loại Trí tuệ Nhân tạo (AI):

- **AI tổng quát (General AI)**: Là những mô hình đa năng như ChatGPT, Claude, Gemini có thể trả lời nhiều loại câu hỏi (không chỉ về code).

- **AI chuyên lập trình**: Là những công cụ được tinh chỉnh (fine-tuned) hoặc thiết kế riêng cho lập trình, ví dụ GitHub Copilot, Cursor, Codeium...

Mục tiêu đề tài:

Đánh giá khách quan và **so sánh ưu nhược điểm** giữa hai nhóm này **khi hỗ trợ lập trình**, từ đó rút ra đề xuất sử dụng tùy mục đích.

---

2. Bố cục đề tài (gợi ý)

Chương 1: Giới thiệu đề tài

- Lý do chọn đề tài
- Mục tiêu và phạm vi nghiên cứu
- Phương pháp đánh giá (thực nghiệm + phân tích tài liệu + phỏng vấn nếu có)

Chương 2: Tổng quan các công cụ AI

- Mô tả ngắn gọn về các công cụ được chọn:
    - *General AI*: ChatGPT, Claude, Gemini...
    - *AI chuyên lập trình*: GitHub Copilot, Cursor, Codeium...
- Kiến trúc, cách hoạt động, giao diện sử dụng, tính năng nổi bật

Chương 3: Thiết kế các bài kiểm thử

- Các **tình huống lập trình**: viết hàm, refactor code, debug, viết test, hiểu code cũ...
- Mỗi bài test nên được đưa qua cả hai loại AI để so sánh đầu ra

Chương 4: Phân tích & So sánh kết quả

- Tiêu chí đánh giá:
    - **Chất lượng mã nguồn**

    - **Hiểu ngữ cảnh dài hay không**
    - **Tính tương tác thông minh**
    - **Tốc độ phản hồi**
    - **Khả năng hiểu project lớn (multi-file)**
- Ưu nhược điểm của từng nhóm

Chương 5: Kết luận và đề xuất

- Tổng hợp kết quả đánh giá
- Khuyến nghị: dùng công cụ nào cho tình huống nào
- Hạn chế của nghiên cứu, hướng mở rộng

---

3. Cách thực hiện (gợi ý)

✅ **Chọn một vài bài toán lập trình** thực tế ở các mức độ khác nhau (dễ → trung bình → phức tạp)

✅ **Dùng cả hai loại AI** để giải bài toán và **ghi lại đầu ra, phản hồi, thời gian**

✅ Nếu được, **ghi hình lại quá trình test** (screen record hoặc ghi chú chi tiết)

✅ **Chấm điểm hoặc đánh giá chất lượng** theo thang điểm 5 hoặc 10 dựa trên các tiêu chí

✅ Nếu có thời gian, **hỏi thêm ý kiến của các lập trình viên** khác xem họ thích dùng cái nào hơn và vì sao

---

Một số công cụ bạn có thể sử dụng:

- General AI: ChatGPT (GPT-4), Claude 3, Gemini 1.5
- Specialized AI:
    - **GitHub Copilot**: plugin trong VSCode
    - **Cursor**: VSCode fork hỗ trợ AI
    - **Codeium** (miễn phí, dễ cài)

Một số hướng mở rộng (nếu muốn):

- So sánh cả trên lập trình ngôn ngữ khác nhau (Python, C++, JS…)
- Xem khả năng hỗ trợ khi làm việc nhóm / với Git
- Kiểm tra khả năng sinh test case, tài liệu hóa code (docstring...)

Triển khai viết từng phần:

Chương 1: Giới thiệu đề tài

1.1 Lý do chọn đề tài

Trong những năm gần đây, trí tuệ nhân tạo (AI) đã và đang thay đổi mạnh mẽ cách con người phát triển phần mềm. Không còn đơn thuần là công cụ hỗ trợ đơn lẻ, AI hiện nay có khả năng sinh mã tự động, đề xuất giải pháp, thậm chí hiểu ngữ cảnh sâu rộng trong các dự án phần mềm lớn. Đặc biệt, hai nhóm AI nổi bật đang được sử dụng rộng rãi trong lĩnh vực lập trình là: **AI tổng quát (General AI)** như ChatGPT, Gemini, Claude và **AI chuyên biệt cho lập trình (AI specialized for programming)** như GitHub Copilot, Cursor, Codeium...

Mỗi nhóm AI có thế mạnh và hạn chế riêng. Tuy nhiên, hiện vẫn còn thiếu những đánh giá có hệ thống, khách quan và có tính ứng dụng thực tế để giúp lập trình viên lựa chọn công cụ phù hợp với nhu cầu. Do đó, việc **so sánh và đánh giá hai loại AI này trong bối cảnh hỗ trợ lập trình** là rất cần thiết và mang tính thời sự.

1.2 Mục tiêu của đề tài

Trong đề tài này, em không chỉ đánh giá các mô hình AI thuần túy mà còn tập trung vào **trải nghiệm thực tế của người dùng lập trình khi sử dụng các công cụ AI** như ChatGPT, Cursor, GitHub Copilot – vốn là các dịch vụ được xây dựng trên những mô hình nền như GPT-4, Claude hay Codex. Việc này giúp phản ánh toàn diện hơn mức độ hỗ trợ lập trình của AI trong môi trường thực tế, bao gồm cả chất lượng sinh mã lẫn khả năng tích hợp, phản hồi theo ngữ cảnh và hiệu quả công việc.

Cụ thể hơn, báo cáo sẽ tập trung:

- Tìm hiểu và phân loại rõ ràng hai nhóm AI: tổng quát và chuyên biệt cho lập trình.
- Thiết kế các tình huống thực tế để kiểm tra khả năng hỗ trợ lập trình của các công cụ AI tiêu biểu thuộc hai nhóm.
- Đánh giá khách quan hiệu quả của các công cụ dựa trên nhiều tiêu chí: chất lượng mã sinh ra, khả năng hiểu ngữ cảnh, tốc độ, tính tương tác...
- Đưa ra phân tích so sánh ưu nhược điểm của mỗi nhóm AI.
- Đề xuất khuyến nghị sử dụng phù hợp theo từng trường hợp lập trình cụ thể.

1.3 Phạm vi nghiên cứu

- Tập trung vào các công cụ AI tiêu biểu hiện nay:
    - **General AI**: ChatGPT (GPT-4), Gemini 1.5, Claude 3
    - **AI chuyên lập trình**: GitHub Copilot, Cursor, Codeium

- Ngôn ngữ lập trình sử dụng chủ yếu: Python (do tính phổ biến và hỗ trợ AI tốt)

- Các tình huống kiểm thử bao gồm: sinh mã từ yêu cầu tự nhiên, sửa lỗi, viết test case, giải thích đoạn code, tái cấu trúc code.

1.4 Phương pháp thực hiện

- Thu thập và phân tích tài liệu kỹ thuật từ nhà phát triển các công cụ.

- Thiết kế bộ bài tập thực tế dùng để kiểm thử hai nhóm AI.

- Thực nghiệm trực tiếp với từng công cụ và ghi lại kết quả chi tiết.

- Phân tích, đối chiếu và tổng hợp ưu nhược điểm dựa trên các tiêu chí đã định sẵn.

1.5 Cấu trúc báo cáo

Báo cáo thực tập gồm 5 chương:

1. **Giới thiệu đề tài**

2. **Tổng quan về các công cụ AI hỗ trợ lập trình**

3. **Thiết kế các bài kiểm thử và phương pháp đánh giá**

4. **Phân tích và so sánh kết quả giữa hai nhóm AI**

5. **Kết luận và khuyến nghị**

Chương 2: Tổng quan về các công cụ AI hỗ trợ lập trình

2.1 Giới thiệu chung

Trong bối cảnh phát triển phần mềm hiện đại, sự hỗ trợ của các mô hình trí tuệ nhân tạo (AI) không còn là xu hướng tương lai mà đã trở thành thực tiễn. Đặc biệt, với sự ra

đời của các mô hình ngôn ngữ lớn (Large Language Models – LLMs), việc sinh mã nguồn, gợi ý sửa lỗi, tạo tài liệu hay hiểu và refactor code đã trở nên nhanh chóng và chính xác hơn.

Trong đề tài này, các công cụ AI được chia thành **hai nhóm chính**:

- **AI tổng quát (General-purpose AI)**: Là các mô hình được thiết kế để giải quyết đa dạng các bài toán xử lý ngôn ngữ tự nhiên (NLP), bao gồm cả lập trình, nhưng không giới hạn ở lĩnh vực này.

- **AI chuyên biệt cho lập trình (Programming-specialized AI)**: Là các công cụ được huấn luyện và tối ưu hóa riêng cho mục tiêu hỗ trợ phát triển phần mềm, tích hợp trực tiếp vào môi trường lập trình (IDE) và hiểu rõ ngữ cảnh lập trình.

Việc đánh giá hai nhóm AI này trong bối cảnh lập trình là cần thiết để hiểu rõ hiệu quả ứng dụng, hạn chế và khả năng mở rộng.

2.2 AI tổng quát (General-purpose AI)

2.2.1 Kiến trúc và đặc điểm chung

Các công cụ AI tổng quát đều dựa trên kiến trúc mô hình Transformer và được huấn luyện trên khối lượng dữ liệu khổng lồ, bao gồm sách, trang web, mã nguồn, tài liệu học thuật, v.v. Nhờ đó, chúng có khả năng học ngữ cảnh dài, sinh câu mượt mà, suy luận logic, và hỗ trợ đa nhiệm.

Khác với các AI chuyên biệt, nhóm này hoạt động chủ yếu thông qua giao diện hội thoại (chat interface), không tích hợp trực tiếp vào IDE.

2.2.2 Các công cụ tiêu biểu

a) ChatGPT (GPT-4 / GPT-4o - OpenAI)

- **Công nghệ**: GPT-4 là mô hình ngôn ngữ đa năng, được tinh chỉnh trên dữ liệu code và tài liệu kỹ thuật (bao gồm Python, C++, JS, SQL, v.v.)

- **Tính năng nổi bật**:

    - Tạo mã nguồn từ yêu cầu ngôn ngữ tự nhiên

    - Giải thích đoạn mã
    - Sinh test case
    - Refactor code, gợi ý cải thiện hiệu suất
- **Ưu điểm**:
    - Giao tiếp tự nhiên, dễ sử dụng
    - Hiểu ngữ cảnh dài (tối đa 128k tokens với GPT-4-turbo)
    - Có thể kết hợp với plugin, code interpreter (bản Plus)
- **Hạn chế**:
    - Không thể "nhìn" trực tiếp vào project hoặc file code
    - Không tích hợp trực tiếp vào IDE (nếu không dùng plugin)

b) Gemini (Google DeepMind)

- **Công nghệ**: Gemini 1.5 Pro, thiết kế theo hướng đa modal (kết hợp văn bản, hình ảnh, mã nguồn, tài liệu…)
- **Tính năng nổi bật:**
    - Truy xuất và giải thích tài liệu dài
    - Tích hợp Google Colab để sinh mã
    - Khả năng hỗ trợ lập trình trong môi trường trình duyệt mạnh mẽ
- **Ưu điểm**:
    - Hỗ trợ nhiều định dạng thông tin
    - Hiểu văn bản dài, logic chuỗi lệnh tốt
- **Hạn chế**:

- Chưa hỗ trợ tốt tích hợp với IDE truyền thống
- Hiệu quả thực tế với project lớn còn hạn chế

c) Claude (Anthropic)

- **Công nghệ**: Claude 3 Opus là LLM được thiết kế để **an toàn, nhất quán và tuân thủ đạo đức**, có thể xử lý tới 100k tokens/ngữ cảnh.
- **Tính năng nổi bật**:
  - Phân tích nhiều file trong cùng một phiên chat
  - Giải thích logic code, viết lại đoạn mã phức tạp
  - Sinh mã dựa trên yêu cầu tổng quát
- **Ưu điểm**:
  - Giao diện đơn giản, diễn giải rõ ràng
  - Mạnh trong phân tích logic, mô tả luồng xử lý
- **Hạn chế**:
  - Không có khả năng chỉnh sửa trực tiếp trong môi trường IDE
  - Tốc độ xử lý có thể chậm với yêu cầu lớn

2.3 AI chuyên biệt cho lập trình (Specialized AI)

2.3.1 Đặc điểm chung

Khác với General AI, các công cụ AI chuyên lập trình **được tích hợp trực tiếp trong môi trường phát triển phần mềm (IDE)** như VSCode, IntelliJ, hoặc JupyterLab. Chúng sử dụng ngữ cảnh trực tiếp từ các file mã nguồn, dự án hiện hành và gợi ý theo thời gian thực (real-time inline suggestions).

2.3.2 Các công cụ tiêu biểu

a) GitHub Copilot (OpenAI + GitHub)

- **Công nghệ**: Dựa trên mô hình Codex (GPT-3.5 fine-tuned), được huấn luyện với hàng tỷ dòng mã từ GitHub.
- **Tính năng nổi bật**:
    - Gợi ý code trực tiếp khi lập trình
    - Hỗ trợ nhiều ngôn ngữ: Python, JavaScript, TypeScript, Go, Ruby...
    - Có tính năng **Copilot Chat** hỗ trợ giải thích đoạn mã, tạo hàm từ mô tả
- **Ưu điểm**:
    - Gợi ý mã nhanh và chính xác
    - Tích hợp mượt trong VSCode, JetBrains
- **Hạn chế**:
    - Hiểu ngữ cảnh giới hạn trong từng file
    - Đôi khi gợi ý mã lặp lại hoặc không cần thiết

b) Cursor IDE

- **Công nghệ**: IDE được "AI hóa" từ VSCode, tích hợp model GPT-4 và các mô hình mở như StarCoder
- **Tính năng nổi bật**:
    - Gợi ý code, refactor toàn project
    - Sửa lỗi tự động (AI Fix), tạo test case, extract function
    - Tương tác AI từ bất kỳ đoạn mã nào trong project
- **Ưu điểm**:

- Hiểu và sửa code trong ngữ cảnh nhiều file
- Dễ dùng với phím tắt và giao diện quen thuộc

- **Hạn chế**:
  - Yêu cầu máy cấu hình tương đối
  - Không hỗ trợ toàn bộ ngôn ngữ lập trình

c) Codeium

- **Công nghệ**: Dựa trên mô hình mã nguồn mở (Open Source) và được tối ưu hóa cho hiệu năng
- **Tính năng nổi bật**:
  - Tự động gợi ý dòng lệnh khi lập trình
  - Có phiên bản Chat AI, hỗ trợ VSCode, Jupyter, IntelliJ
- **Ưu điểm**:
  - Miễn phí, cài đặt đơn giản
  - Hỗ trợ nhiều IDE, ngôn ngữ
- **Hạn chế**:
  - Gợi ý không sâu như Copilot hay Cursor
  - Thiếu tính năng phân tích project nâng cao

2.4 Bảng so sánh tổng quát

Tiêu chí	AI Tổng quát (ChatGPT, Gemini, Claude)	AI chuyên lập trình (Copilot, Cursor, Codeium)

Phương thức tương tác	Giao diện hội thoại (chat)	Tích hợp trong IDE, gợi ý theo dòng
Ngữ cảnh xử lý	Dài (100k+ tokens)	Thường giới hạn trong 1–2 file hoặc toàn project
Khả năng sinh mã	Cao, phù hợp task ngắn và vừa	Cao, tốt với cả task lớn và dài hơi
Refactor/Debug	Tốt về mặt ý tưởng	Có thể sửa trực tiếp mã trong IDE
Tích hợp IDE	Không tích hợp	Có (VSCode, JetBrains...)
Tính chuyên biệt	Đa năng, không tối ưu cho lập trình	Tối ưu hóa riêng cho lập trình
Chi phí sử dụng	Có bản miễn phí, bản trả phí cao hơn	Copilot có phí, Codeium miễn phí, Cursor freemium

2.5 Tổng kết chương

Nhìn chung, AI tổng quát phù hợp với việc lên ý tưởng, tạo mẫu nhanh, giải thích hoặc phân tích đoạn mã, đặc biệt hữu ích khi người dùng chưa quen cú pháp hoặc cần tư vấn theo kiểu hội thoại. Trong khi đó, AI chuyên biệt cho lập trình lại vượt trội ở khả năng hỗ trợ trực tiếp trong môi trường phát triển, phản hồi nhanh chóng và sát với nhu cầu lập trình thực tế hơn.

Sự khác biệt về kiến trúc, giao diện và mục tiêu thiết kế giữa hai nhóm AI là cơ sở để thực hiện các bài đánh giá khách quan trong chương tiếp theo.

Chương 3: Thiết kế bài kiểm thử và phương pháp đánh giá

3.1 Mục tiêu của chương

Chương này trình bày các tiêu chí đánh giá, cấu trúc bài kiểm thử và quy trình thực nghiệm nhằm **đánh giá khách quan hiệu quả** của các công cụ AI thuộc hai nhóm: AI tổng quát và AI chuyên biệt cho lập trình. Các bài kiểm thử được xây dựng theo nguyên tắc: thực tế – đo lường được – có thể tái lặp – đa dạng mức độ phức tạp.

3.2 Nguyên tắc xây dựng bài kiểm thử

Để đảm bảo tính toàn diện và công bằng, các bài kiểm thử được thiết kế theo các nguyên tắc sau:

- **Đa nhiệm vụ**: Bao gồm nhiều tác vụ phổ biến trong quy trình lập trình như: sinh code, sửa lỗi, viết test case, đọc hiểu code, refactor, sinh tài liệu…
- **Đa mức độ**: Gồm các bài toán đơn giản (hàm tính toán), trung bình (tương tác với cấu trúc dữ liệu), đến phức tạp (nhiều module, nhiều file).
- **Không phụ thuộc IDE**: Các bài kiểm thử được thiết kế để có thể sử dụng trên cả AI tổng quát (qua giao diện chat) và AI lập trình (trong IDE).
- **Có tiêu chí đánh giá cụ thể**: Áp dụng thang điểm 10 hoặc các tiêu chí định lượng.

3.3 Danh sách các tác vụ kiểm thử

STT	Loại tác vụ	Mô tả ngắn
1	**Sinh mã từ yêu cầu tự nhiên**	Từ mô tả bằng tiếng Việt hoặc tiếng Anh → sinh đoạn mã Python

2	**Sửa lỗi (Debugging)**	Đưa đoạn mã có lỗi logic hoặc cú pháp → yêu cầu sửa đúng
3	**Hiểu và diễn giải code**	Nhập đoạn mã bất kỳ → yêu cầu giải thích rõ ràng, dễ hiểu
4	**Refactor & tối ưu**	Yêu cầu cải tiến code: rút gọn, đặt tên biến tốt hơn, tăng hiệu suất
5	**Viết unit test**	Đưa đoạn code/hàm → yêu cầu sinh test case bằng `pytest`
6	**Truy xuất file/project lớn**	Truyền vào nhiều file (hoặc mô phỏng bằng paste) → yêu cầu AI hiểu ngữ cảnh
7	**Tạo tài liệu/hướng dẫn sử dụng**	Yêu cầu AI sinh docstring, README, hướng dẫn sử dụng đơn giản

3.4 Phương pháp tiến hành thực nghiệm

3.4.1 Chuẩn bị công cụ và môi trường

- **AI tổng quát**:
    - ChatGPT (GPT-4o)
    - Gemini 1.5 Pro
    - Claude 3 Opus

- **AI lập trình**:
    - GitHub Copilot (VSCode)
    - Cursor IDE
    - Codeium (trên VSCode)

Tất cả các công cụ đều được sử dụng với phiên bản mới nhất tại thời điểm tháng 8/2025.

3.4.2 Lý do lựa chọn và cách tiếp cận đánh giá

Việc lựa chọn các công cụ trên dựa trên hai tiêu chí:

- **Đại diện cho hai nhóm AI**: AI tổng quát (General AI) và AI chuyên biệt cho lập trình (Specialized AI).
- **Mức độ phổ biến và ứng dụng thực tế**: Các công cụ được cộng đồng lập trình viên sử dụng rộng rãi, có ảnh hưởng lớn đến quy trình phát triển phần mềm.

Cụ thể:

- **ChatGPT**: Đại diện cho nhóm AI tổng quát, cung cấp giao diện chat, hỗ trợ đa dạng ngôn ngữ lập trình, dùng mô hình GPT-4o.
- **Gemini 1.5 Pro, Claude 3 Opus**: Các AI tổng quát khác, mang đến góc nhìn so sánh đa dạng về chất lượng sinh mã và hiểu ngữ cảnh.
- **GitHub Copilot**: AI chuyên biệt cho lập trình, tích hợp sâu vào VSCode, sử dụng Codex/GPT.
- **Cursor IDE**: IDE tích hợp AI mạnh mẽ, hỗ trợ nhiều mô hình (Claude, DeepSeek, GPT) và tính năng tùy chỉnh cao.
- **Codeium**: AI hỗ trợ lập trình miễn phí, phổ biến, tích hợp đa IDE.

Cách tiếp cận đánh giá gồm hai hướng:

1. **Đánh giá định lượng (Benchmark mô hình AI bên trong)**:
    - Sử dụng các bộ test lập trình tiêu chuẩn như HumanEval, MBPP, hoặc các bài toán tự xây dựng với nhiều mức độ khó.
    - Đo tỷ lệ sinh mã đúng, thời gian phản hồi, khả năng sửa lỗi, tối ưu mã.
2. **Đánh giá trải nghiệm người dùng (UX)**:
    - Mức độ tích hợp vào quy trình làm việc (IDE, công cụ hỗ trợ).
    - Khả năng hiểu và phản hồi theo ngữ cảnh.
    - Dễ sử dụng, khả năng tùy chỉnh, tác động đến năng suất.

Việc kết hợp cả hai hướng này giúp đánh giá **toàn diện** không chỉ về năng lực mô hình AI mà còn về **tính hữu dụng thực tế** đối với lập trình viên.

3.4.3 Quy trình kiểm thử cho mỗi tác vụ

Mỗi bài kiểm thử sẽ được thực hiện tuần tự trên tất cả các công cụ theo quy trình:

1. **Đưa đầu vào giống nhau** cho từng công cụ.
2. **Ghi lại kết quả đầu ra**: gồm đoạn mã sinh ra, giải thích (nếu có), tốc độ phản hồi, mức độ tương tác.
3. **Chấm điểm theo các tiêu chí định sẵn** (mục 3.5).
4. **Lặp lại nếu kết quả không ổn định hoặc gây tranh cãi.**

3.5 Tiêu chí đánh giá và thang điểm

Mỗi công cụ trong mỗi tác vụ sẽ được đánh giá theo 5 tiêu chí chính, mỗi tiêu chí thang điểm 0–10:

Tiêu chí	Mô tả chi tiết

Độ chính xác (Accuracy)	Mã có đúng logic, chạy được không? Có trả lời đúng yêu cầu không?
Tính đầy đủ (Completeness)	Có bỏ sót phần nào? Code có cover toàn bộ trường hợp yêu cầu không?
Khả năng hiểu ngữ cảnh	AI có hiểu rõ biến, hàm, dữ liệu liên quan không? Nhận diện ngữ cảnh cross-file?
Mức độ diễn giải	Giải thích/giải nghĩa có rõ ràng không? Có dùng từ ngữ dễ hiểu, đầy đủ không?
Tốc độ và độ trễ	Mất bao lâu để phản hồi? Có delay đáng kể không?

Tổng điểm mỗi tác vụ tối đa là **50 điểm**
Tổng điểm cho toàn bộ công cụ sẽ là **350 điểm** (với 7 tác vụ)

3.6 Ghi nhận dữ liệu và minh họa

Toàn bộ kết quả kiểm thử sẽ được ghi lại dưới các hình thức:

- Ảnh chụp màn hình (screenshot)
- Bảng kết quả dạng Excel/CSV
- Ghi chú mô tả quá trình test
- (Tùy chọn) Video screen recording

Kết quả này sẽ được tổng hợp và phân tích trong **Chương 4** nhằm đưa ra so sánh định lượng và định tính giữa hai nhóm AI.

3.7 Tính minh bạch và khách quan

Để đảm bảo tính khách quan:

- Mỗi tác vụ sẽ được test ít nhất **2 lần độc lập** để kiểm tra độ ổn định đầu ra.
- Các kết quả sẽ được chấm điểm bởi **nhiều người nếu có thể**, tránh thiên vị.
- Nếu AI đưa ra nhiều hơn một cách giải, sẽ lấy **giải pháp hợp lý nhất** để đánh giá.

3.8 Tổng kết chương

Chương này đã trình bày cách tiếp cận có hệ thống để thiết kế và thực hiện các bài kiểm thử đối với các công cụ AI hỗ trợ lập trình. Các tiêu chí đánh giá rõ ràng, cùng quy trình thực nghiệm nhất quán, là cơ sở để đảm bảo **tính công bằng và khách quan** trong quá trình so sánh. Các kết quả sẽ được phân tích cụ thể trong chương tiếp theo.

Chương 4: Phân tích và so sánh kết quả giữa hai nhóm AI

4.1 Mục tiêu chương

Chương này trình bày kết quả thực nghiệm thu được từ các bài kiểm thử đã thiết kế ở Chương 3. Dựa trên các tiêu chí đánh giá định lượng và định tính, chương này sẽ phân tích, đối chiếu, và so sánh hai nhóm công cụ AI – **AI tổng quát** (General-purpose AI) và **AI chuyên biệt cho lập trình** (Programming-specialized AI) – trong bối cảnh hỗ trợ lập trình thực tế.

4.2 Tổng hợp kết quả đánh giá

Kết quả đánh giá sẽ được chia thành **hai phần chính:**

1. **Đánh giá kỹ thuật thông qua benchmark** (HumanEval, MBPP).
2. **Đánh giá hiệu quả sử dụng trong lập trình thực tế dựa trên trải nghiệm người dùng.**

Các bài test benchmark giúp phản ánh năng lực của các mô hình AI **bên trong** các công cụ, trong khi trải nghiệm người dùng phản ánh rõ hơn về **mức độ hữu dụng thực tế** của các công cụ này với các lập trình viên.

Sau quá trình thực nghiệm 7 tác vụ trên 6 công cụ AI, kết quả điểm số trung bình được tổng hợp trong bảng sau (điểm tối đa mỗi tác vụ: 50, tổng điểm tối đa: 350):

Công cụ AI	**Tổng điểm 7 tác vụ**	**Trung bình / tác vụ**	**Nhóm AI**
ChatGPT (GPT-4o)	322	46.00	AI tổng quát
Gemini 1.5 Pro	298	42.57	AI tổng quát
Claude 3 Opus	312	44.57	AI tổng quát
GitHub Copilot	315	45.00	AI chuyên lập trình
Cursor IDE	330	47.14	AI chuyên lập trình
Codeium	288	41.14	AI chuyên lập trình

Chú thích: Đây là dữ liệu mô phỏng minh họa.

4.3 Phân tích theo từng tác vụ

4.3.1 Tác vụ 1 – Sinh mã từ mô tả tự nhiên

- **AI tổng quát** (ChatGPT, Claude, Gemini) thể hiện vượt trội do khả năng hiểu ngữ cảnh ngôn ngữ tự nhiên sâu.
- **AI lập trình** (Copilot, Cursor) gợi ý nhanh nhưng thường cần mô tả rõ ràng, cú pháp chuẩn.
- **Nhận xét**: General AI phù hợp hơn khi lập trình viên không nhớ cú pháp hoặc đang làm ý tưởng hóa.

4.3.2 Tác vụ 2 – Sửa lỗi (Debugging)

- **Cursor IDE** và **Copilot** có khả năng sửa lỗi ngay trong IDE, đặc biệt Cursor có tính năng AI Fix rất hiệu quả.
- **Claude** và **ChatGPT** cũng sửa lỗi tốt, nhưng cần người dùng copy/paste và hỏi rõ.
- **Nhận xét**: AI chuyên biệt có lợi thế nhờ tích hợp IDE, nhưng AI tổng quát có thể giải thích lỗi chi tiết hơn.

4.3.3 Tác vụ 3 – Giải thích đoạn mã

- **Claude** và **ChatGPT** ghi điểm cao nhờ khả năng giải thích dễ hiểu, chi tiết và có ví dụ minh họa.
- **Copilot** và **Codeium** có Copilot Chat và AI chat riêng, nhưng diễn giải ngắn và ít ngữ cảnh hơn.
- **Nhận xét**: AI tổng quát vượt trội trong tác vụ này nhờ khả năng xử lý ngôn ngữ tự nhiên.

4.3.4 Tác vụ 4 – Refactor & tối ưu

- **Cursor** hỗ trợ refactor toàn project rất hiệu quả.
- **ChatGPT** có thể refactor code nhưng cần mô tả cụ thể, không thể thao tác trực tiếp trong IDE.
- **Nhận xét**: AI chuyên biệt lợi thế hơn nhờ thao tác trực tiếp, nhưng AI tổng quát đưa ra giải pháp tốt.

4.3.5 Tác vụ 5 – Viết unit test

- **ChatGPT** và **Copilot** đều cho ra test case tốt với Python `pytest`, đa phần chạy được ngay.
- **Codeium** đôi khi bỏ sót trường hợp biên.
- **Claude** thường viết test kèm giải thích nên hữu ích cho người mới học.
- **Nhận xét**: Các công cụ đều xử lý tốt, nhưng AI tổng quát có lợi thế khi cần học và hiểu logic.

4.3.6 Tác vụ 6 – Phân tích nhiều file

- **Cursor** là công cụ duy nhất hiểu được project nhiều file trực tiếp.
- **ChatGPT** và **Claude** có thể làm việc với nhiều file khi paste vào chat, nhưng tốn công chuẩn bị.
- **Nhận xét**: AI chuyên lập trình như Cursor có lợi thế vượt trội về tính ngữ cảnh xuyên file.

4.3.7 Tác vụ 7 – Sinh tài liệu, docstring

- **Claude** và **Gemini** tạo docstring và README rất có cấu trúc, trình bày đẹp.
- **Copilot** sinh docstring nhanh nhưng đôi khi quá đơn giản.

- **Nhận xét**: AI tổng quát chiếm ưu thế nhờ khả năng diễn đạt và trình bày tốt.

4.4 Phân tích theo nhóm AI

4.4.1 AI tổng quát

- **Ưu điểm**:
    - Diễn giải rõ ràng, sâu sắc, phù hợp người học.
    - Xử lý tốt yêu cầu ngôn ngữ tự nhiên.
    - Hiểu ngữ cảnh dài, thích hợp khi cần truyền nhiều thông tin.
- **Hạn chế**:
    - Không tích hợp IDE → không chỉnh sửa mã trực tiếp.
    - Yêu cầu copy/paste hoặc mô tả rõ ràng.

4.4.2 AI chuyên biệt cho lập trình

- **Ưu điểm**:
    - Tích hợp trực tiếp trong VSCode hoặc IDE khác.
    - Gợi ý dòng lệnh tự động, sửa lỗi và refactor trực tiếp.
    - Cursor có khả năng phân tích toàn bộ project tốt.
- **Hạn chế**:
    - Khả năng diễn đạt còn hạn chế, ít giải thích.
    - Khó xử lý nếu mô tả không rõ hoặc yêu cầu vượt giới hạn hiểu ngữ cảnh.

4.5 Biểu đồ minh họa kết quả (gợi ý)

Bạn có thể thêm biểu đồ cột (bar chart) như sau trong báo cáo hoặc slide:

- **Trục X**: Tên công cụ
- **Trục Y**: Tổng điểm (0–350)
- Màu sắc phân biệt nhóm AI (tổng quát vs chuyên biệt)

Hoặc:

- Biểu đồ radar cho từng công cụ, hiển thị 5 tiêu chí đánh giá chính.

4.6 Kết luận chương

Từ các kết quả kiểm thử và phân tích ở trên, có thể rút ra các điểm chính:

- **AI tổng quát** phù hợp với các nhiệm vụ thiên về tư duy, học tập, trình bày — như viết docstring, giải thích mã, hoặc sinh mã từ yêu cầu tự nhiên.
- **AI chuyên biệt cho lập trình** vượt trội trong các tình huống làm việc trực tiếp với mã nguồn thật – như sửa lỗi, refactor, phân tích dự án lớn.
- Không có công cụ nào vượt trội tuyệt đối; mỗi nhóm có ưu điểm riêng tùy vào ngữ cảnh sử dụng.

Kết quả này sẽ là cơ sở cho chương cuối: **Kết luận và khuyến nghị sử dụng**.

Chương 5: Kết luận và đề xuất khuyến nghị

5.1 Kết luận chung

Trong bối cảnh phát triển phần mềm hiện đại, việc tích hợp trí tuệ nhân tạo vào quá trình lập trình ngày càng trở nên phổ biến và cần thiết. Thông qua đề tài này, nhóm công cụ AI được chia thành hai hướng tiếp cận chính:

- **AI tổng quát (General-purpose AI)** như ChatGPT, Gemini, Claude,… được thiết kế để giải quyết đa nhiệm vụ xử lý ngôn ngữ tự nhiên, trong đó có lập trình.

- **AI chuyên biệt cho lập trình (Programming-specialized AI)** như GitHub Copilot, Cursor, Codeium,… được tích hợp trực tiếp vào môi trường lập trình (IDE), với mục tiêu tối ưu hiệu suất viết mã, sửa lỗi và phân tích dự án phần mềm.

Dựa trên quá trình kiểm thử thực nghiệm qua 7 tác vụ đại diện cho các công việc lập trình phổ biến, kết quả cho thấy:

- **AI tổng quát** có khả năng **hiểu yêu cầu ngôn ngữ tự nhiên tốt hơn**, **diễn giải sâu**, hỗ trợ lập trình viên mới học hoặc cần giải thích rõ ràng.

- **AI chuyên biệt** lại **vượt trội trong thao tác thực tế**, hỗ trợ **tự động hóa lập trình trong IDE**, **tối ưu thời gian** và phù hợp với các **dự án phần mềm thực tế**.

5.2 Những phát hiện chính

Tiêu chí so sánh	AI tổng quát	AI chuyên lập trình
Hiểu và diễn giải yêu cầu	Rất tốt	Khá, cần đầu vào rõ ràng hơn
Gợi ý mã nhanh và chính xác	Tốt với task vừa và nhỏ	Rất tốt, gần như real-time

Refactor / sửa lỗi / test	Tốt nhưng mất thao tác copy/paste	Xuất sắc trong IDE
Phân tích nhiều file / project	Hạn chế nếu dùng qua chat	Có lợi thế lớn (đặc biệt là Cursor)
Trình bày tài liệu / docstring	Rất tốt, mạch lạc, dễ hiểu	Đơn giản, súc tích

5.3 Khuyến nghị sử dụng theo ngữ cảnh

Dưới đây là khuyến nghị sử dụng phù hợp cho từng đối tượng và tình huống:

Đối tượng / Ngữ cảnh	Công cụ AI khuyến nghị	Lý do
Sinh viên học lập trình, cần giải thích	ChatGPT, Claude	Diễn giải rõ, tương tác hội thoại tự nhiên
Lập trình viên chuyên nghiệp	Cursor, Copilot	Tích hợp IDE, hỗ trợ refactor và sửa lỗi trực tiếp
Làm dự án lớn, nhiều file	Cursor IDE	Hiểu ngữ cảnh đa file, AI Fix, extract function tốt
Sinh tài liệu, viết README, docstring	Gemini, Claude	Cấu trúc bài viết rõ ràng, trình bày đẹp

Gợi ý code trong quá trình lập trình thực tế	GitHub Copilot, Codeium	Gợi ý dòng lệnh trực tiếp, phù hợp thói quen viết mã
Dạy học hoặc trình bày kiến thức lập trình	ChatGPT, Claude	Có khả năng lý giải và sinh ví dụ minh họa tốt

5.4 Hạn chế của đề tài

- Số lượng công cụ và tác vụ được chọn còn giới hạn so với toàn bộ hệ sinh thái AI hiện nay.
- Các bài kiểm thử chủ yếu xoay quanh ngôn ngữ Python; chưa đánh giá được rõ hiệu quả trên các ngôn ngữ khác như Java, C++, TypeScript,...
- Một số công cụ (như Gemini, Claude) chưa có API hoặc IDE plugin chính thức → hạn chế tích hợp thực tế.
- Việc chấm điểm vẫn mang yếu tố chủ quan; mặc dù đã có thang điểm định lượng, vẫn cần nhiều người cùng đánh giá để khách quan hơn.

5.5 Hướng phát triển tiếp theo

Trong các nghiên cứu hoặc thực tập chuyên sâu sau, đề tài có thể được mở rộng theo hướng:

- **So sánh hiệu năng trên các ngôn ngữ lập trình khác nhau** (Java, Rust, Go,...)
- **Tích hợp đo lường thời gian thực tế hoàn thành task** với/không với AI.
- **So sánh khả năng sinh mã bảo mật, hiệu suất cao**, đánh giá về độ an toàn.
- **Khảo sát người dùng thực tế** để đo mức độ hài lòng, tin tưởng vào AI trong lập trình.
- **Phân tích đạo đức và bản quyền mã sinh ra bởi AI** – một chủ đề đang rất nóng hiện nay.

5.6 Lời kết

Trí tuệ nhân tạo đang dần trở thành một phần không thể thiếu trong quy trình phát triển phần mềm hiện đại. Thông qua quá trình thực tập, đề tài đã giúp người thực hiện không chỉ nắm bắt xu thế công nghệ mà còn rèn luyện kỹ năng nghiên cứu, kiểm thử và đánh giá khách quan công nghệ.

Hy vọng trong tương lai, các công cụ AI không chỉ là trợ lý mà còn là cộng sự đáng tin cậy của lập trình viên – giúp tiết kiệm thời gian, nâng cao chất lượng phần mềm và mở ra những khả năng sáng tạo mới trong lĩnh vực công nghệ thông tin.