Giai đoạn 1: Vision, Architecture & Data Grounding (Dự kiến: 2-3 tuần)

Vai trò/Người	Đầu việc cụ thể	Mô tả chi tiết & Hướng dẫn	Sản phẩm đầu ra & Tiêu chí đánh giá
Hiếu (INTJ)	**1. Chi tiết hóa Kiến trúc 4 Tầng**	~~- Dựa trên mô tả 4 tầng, vẽ lại sơ đồ kiến trúc chi tiết hơn, thể hiện rõ các luồng "trigger" và luồng dữ liệu giữa các module (ví dụ: Anomaly Detector -> SHAP & LLM, Analyst Disposition -> MLOps Loop).~~ ~~- Lựa chọn công nghệ cụ thể cho từng module chính (Tầng 1 - Pub/Sub; Tầng 2 - Vertex AI Pipelines, Vertex AI Endpoints; Tầng 3 - Cloud Run/Functions; Tầng 4 - Vertex AI Training).~~	- **Sơ đồ kiến trúc phiên bản 2.0:** Chi tiết, thể hiện rõ các luồng và công nghệ dự kiến. - **Tài liệu Kiến trúc hệ thống:** Mô tả chức năng và công nghệ cho từng module trong 4 tầng. - **Tiêu chí:** Khả thi về mặt kỹ thuật, logic chặt chẽ, thể hiện đúng tầm nhìn.
	2. Thiết kế Module Phân loại HFACS (Dựa trên Excel)	~~- **Lên kế hoạch kỹ thuật:** Thiết kế một pipeline nhỏ để xử lý file Excel. Bao gồm các bước: Đọc file -> Trích xuất cột "Summary" -> Xây dựng Prompt cho Gemini API để phân loại từng "Summary" theo HFACS Taxonomy.~~ ~~- **Viết code PoC (Proof of Concept) ban đầu:** Code một script Python đơn giản để thực hiện pipeline trên cho toàn bộ file Excel. Mục tiêu là chứng minh AI có thể đọc và phân loại được.~~	- **Kế hoạch kỹ thuật module HFACS:** Một tài liệu ngắn mô tả luồng xử lý và cấu trúc prompt. - **Script Python `hfacs_classifier_poc.py`:** Chạy được và in ra kết quả phân loại HFACS cho vài dòng dữ liệu. - **Tiêu chí:** Chứng minh được tính khả thi của việc dùng LLM để phân loại HFACS từ dữ liệu tường thuật.
	3. Thiết kế khung `simulator.py`	~~- Dựa trên các loại dữ liệu ở Tầng 1 và kịch bản lỗi của Hường, thiết kế các lớp (classes) và hàm (functions) chính cho **module mô phỏng dữ liệu**.~~ - Cần có các hàm để tạo: Telemetry (dữ liệu số), Maintenance Logs (dạng text/JSON), Weather/ATC (dữ liệu bối cảnh), và Narrative Reports (dữ liệu văn	- **Cấu trúc file `simulator.py` (dạng skeleton code):** Các class, hàm, và comment giải thích chức năng, chưa cần implement chi tiết. - **Tiêu chí:** Cấu trúc logic, dễ mở rộng, đáp ứng được nhu cầu dữ liệu của 4 nguồn ở Tầng 1.

		bản). - Đặc biệt chú ý đến hàm "inject lỗi" để có thể tạo ra các bất thường có chủ đích.	
	4. Hướng dẫn & Thiết lập Môi trường	~~- **Hướng dẫn Hường về phân tích dữ liệu Excel:** Chỉ cho Hường cách nhìn dữ liệu từ góc độ của một kỹ sư AI (cần trích xuất những thông tin gì, làm sạch ra sao). Hướng dẫn Hường phong cách làm việc tư duy với AI, từ cung cấp bối cảnh, ra đề bài, refinement~~ ~~- **Thiết lập Project trên Google Cloud:** Tạo project, kích hoạt các API cần thiết (Vertex AI, Cloud Storage), tạo service account và cung cấp file key cho Hường để chạy script PoC.~~	- **Buổi hướng dẫn về dữ liệu & GCP:** Đảm bảo Hường hiểu cách làm việc và có thể chạy code. - **Môi trường GCP sẵn sàng:** Project được cấu hình cơ bản.
Hường (ENFJ)	**1. Nghiên cứu & Chi tiết hóa HFACS Taxonomy cho Prompt Engineering**	~~- Dựa trên kiến thức và các tài liệu học thuật, xây dựng một bản mô tả chi tiết các cấp độ HFACS, kèm theo các từ khóa (keywords) và ví dụ cụ thể liên quan đến hàng không.~~ ~~- Mục đích là để cung cấp "bối cảnh" cực kỳ chất lượng cho Gemini trong các prompt, giúp nó phân loại chính xác hơn.~~ ~~- Ví dụ: Cấp độ "Unsafe Acts - Skill-Based Errors" có thể có keywords như "forgot to", "misread", "improper technique". -> cái này anh thấy em đang làm rất tốt, có thể chuẩn hóa thành 1 bộ rules của từng loại HFACS -> cải thiện prompt~~	- **Tài liệu "HFACS Taxonomy for AI Prompting":** Một bản hướng dẫn chi tiết, có cấu trúc, sẵn sàng để Hiếu tích hợp vào code. - **Tiêu chí:** Chi tiết, thực tế, tối ưu cho việc "dạy" AI.

		~~- Phân loại thủ công theo hệ thống mà mình đã đặt ra (file excel hơn 240 vụ)~~	
	2. Thiết kế Kịch bản Mô phỏng Đa nguồn	~~- Mở rộng kịch bản lỗi cánh tà: Không chỉ mô tả lỗi, mà còn mô tả các dữ liệu liên quan từ 3 nguồn còn lại sẽ trông như thế nào khi lỗi đó xảy ra.~~ ~~- Ví dụ Kịch bản "Kẹt cánh tà":~~ ~~- Telemetry: Dòng điện motor tăng vọt, vị trí cánh tà không thay đổi.~~ ~~- Maintenance Log: "3 tháng trước, có ghi nhận motor cánh tà bên phải phát ra tiếng động lạ".~~ ~~- Narrative Report: "Phi công báo cáo cảm thấy khó điều khiển khi hạ cánh tà".~~ ~~- Weather/ATC: "Thời tiết bình thường, ATC cho phép hạ cánh".S~~	- **Tài liệu "Multi-Source Simulation Scenarios":** Mô tả ít nhất 3 kịch bản lỗi chi tiết, bao trùm cả 4 nguồn dữ liệu. - **Tiêu chí:** Logic, có tính liên kết giữa các nguồn dữ liệu, thực tế.
	3. Hỗ trợ chạy & Đánh giá PoC Phân loại HFACS	~~- Sau khi Hiếu đưa script `run_classification.py`, Hường sẽ là người chạy script với file key GCP.~~ ~~- **So sánh kết quả của AI** với cột `HFACS_Manual_Classification` mà mình đã làm.~~ ~~- Ghi nhận các trường hợp AI phân loại đúng/sai và đưa ra giả thuyết tại sao sai.~~ ~~- **Hướng dẫn:** Hường không cần hiểu sâu code, chỉ cần biết cách chạy và phân tích kết quả.~~	- **Bảng so sánh kết quả AI vs Manual:** Đánh dấu đúng/sai cho từng trường hợp. - **Báo cáo đánh giá PoC:** Ghi nhận tỷ lệ đúng/sai và các nhận xét ban đầu. - **Tiêu chí:** Hoàn thành việc chạy thử và có đánh giá sơ bộ chất lượng của AI.
Kết hợp & Đánh giá	**Buổi tổng kết Giai đoạn 1**	- Hiếu trình bày kiến trúc chi tiết và kết quả PoC kỹ thuật. - Hường trình bày bản phân tích dữ liệu Excel, "Ground Truth" HFACS, và các kịch bản mô phỏng.	- **Sự liên kết giữa Kiến trúc và Dữ liệu:** Đảm bảo kiến trúc được thiết kế để xử lý tốt các loại dữ liệu thực tế và mô phỏng. - **Kết quả PoC khả quan:** Chứng minh được

		- Cả hai cùng review kết quả đánh giá PoC phân loại HFACS, thảo luận **hướng cải thiện prompt/mô hình**. - Thống nhất kế hoạch chi tiết cho Giai đoạn 2 (bắt đầu code các module chính).	tiềm năng của dự án. - **Kế hoạch Giai đoạn 2 rõ ràng và thực tế.**

Bảng Phân Công Chi tiết cho Giai đoạn 2

Vai trò/Người	Đầu việc cụ thể	Mô tả chi tiết & Hướng dẫn	Sản phẩm đầu ra & Tiêu chí đánh giá
Hiếu (INTJ)	**1. Phát triển Module `simulator.py` (Core)**	- Dựa trên thiết kế ở GĐ1, **viết code hoàn chỉnh** cho các hàm tạo dữ liệu mô phỏng. - Tập trung vào việc tạo ra dữ liệu Telemetry (dạng time-series) và hàm `inject_anomaly()` để chèn một bất thường vào một thời điểm cụ thể. - Xây dựng các hàm trả về dữ liệu Maintenance Logs, Weather, Narrative Reports tương ứng với kịch bản được chọn.	- **`simulator.py` phiên bản 1.0:** Một module có thể chạy độc lập, nhận đầu vào là một kịch bản (ví dụ: "flap_jam_scenario") và trả về 4 luồng dữ liệu tương ứng. - **Tiêu chí:** Dữ liệu tạo ra có cấu trúc, logic, và mô phỏng đúng kịch bản lỗi Hường đã thiết kế.
	2. Code Module `Anomaly Detector` (AD)	- Viết một class `AnomalyDetector` trong Python. - Triển khai thuật toán đã chọn ở GĐ1 (ví dụ: Isolation Forest hoặc các quy tắc thống kê nâng cao). - Class này nhận dữ liệu Telemetry time-series làm đầu vào và trả về `True/False` nếu có bất thường, cùng với thời điểm phát hiện.	- **File `anomaly_detector.py`:** Chứa class `AnomalyDetector` có thể huấn luyện (fit) trên dữ liệu bình thường và dự đoán (predict) trên dữ liệu mới. - **Tiêu chí:** Có khả năng phát hiện được lỗi mà `simulator.py` đã chèn vào.
	3. Code Module `Risk Triage Engine` & Tích hợp Luồng Phân tích	- Đây là module "nhạc trưởng". Viết code để: 1. Nhận trigger từ `AnomalyDetector`. 2. **Luồng Kỹ thuật:** Gọi một `ShapExplainer` giả lập (ban đầu chỉ cần in ra các feature quan trọng nhất gây ra bất thường).	- **File `risk_engine.py`:** Chứa logic cốt lõi của việc phân tích và hợp nhất kết quả. - **Tích hợp Gemini API hoàn chỉnh:** Có thể gọi và xử lý kết quả phân loại HFACS một cách ổn định. - **Tiêu chí:** Luồng xử lý từ lúc phát hiện bất

		3. **Luồng Con người:** Lấy `NarrativeReports` từ simulator và gọi Gemini API để phân loại HFACS (tái sử dụng/nâng cấp code PoC GĐ1). 4. **Hợp nhất:** Tổng hợp thông tin từ 2 luồng và quyết định đầu ra.	thường đến lúc có giải thích được tự động hóa.
	4. Code Kịch bản Demo Chính (main.py)	- Viết một file `main.py` để kết nối tất cả các module lại với nhau. - File này sẽ: 1. Gọi `simulator.py` để tạo dữ liệu. 2. Đưa dữ liệu vào `AnomalyDetector`. 3. Nếu có bất thường, kích hoạt `RiskTriageEngine`. 4. In ra kết quả cuối cùng trên console (giải thích kỹ thuật, phân loại HFACS). 5. Mô phỏng phần "Act" bằng cách in ra các hành động đề xuất (ví dụ: "ACTION: Generate Maintenance W/O for Flap System").	- **File `main.py`:** Script chính để chạy toàn bộ demo. - **Tiêu chí:** Chạy từ đầu đến cuối không lỗi, thể hiện được toàn bộ vòng lặp S-D-E-A.
Hường (ENFJ)	**1. Tinh chỉnh & Mở rộng Kịch bản Mô phỏng**	- Làm việc với Hiếu để đảm bảo các kịch bản lỗi ở GĐ1 được `simulator.py` mô phỏng chính xác. - **Nghiên cứu và viết thêm 2-3 kịch bản lỗi mới** cho các hệ thống khác (ví dụ: hệ thống thủy lực, động cơ) để chuẩn bị cho các giai đoạn sau. Giữ nguyên cấu trúc đa nguồn. - Cung cấp các giá trị dữ liệu cụ thể (ví dụ:	- **Tài liệu "Simulation Scenarios v2.0":** Cập nhật và bổ sung các kịch bản mới, chi tiết hơn. - **Input cụ thể cho Hiếu:** Các file CSV hoặc JSON chứa các tham số cho `simulator.py`. - **Tiêu chí:** Kịch bản thực tế, chi tiết, cung cấp đủ thông tin cho Hiếu code.

		nhiệt độ nên tăng từ 80°C lên 120°C trong 5 giây).	
	2. Tối ưu hóa Prompt Engineering cho Gemini	- Đây là nhiệm vụ cốt lõi của Hường trong giai đoạn này. - Dựa trên kết quả PoC GĐ1, **thử nghiệm và tinh chỉnh các prompt** để Gemini phân loại HFACS chính xác hơn. - Thử các kỹ thuật: Zero-shot, Few-shot (cung cấp 1-2 ví dụ trong prompt), Chain-of-Thought (yêu cầu AI giải thích từng bước suy nghĩ). - Chuẩn bị một bộ prompt "chuẩn" cho từng loại phân tích.	- **Tài liệu "Advanced Prompt Engineering Guide":** Ghi lại các phiên bản prompt, kết quả thử nghiệm và prompt cuối cùng được chọn. - **Tiêu chí:** Tỷ lệ phân loại HFACS chính xác (so với "Ground Truth") tăng lên đáng kể. Prompt có cấu trúc, dễ tích hợp vào code.
	3. Thiết kế & Thực hiện Test Cases cho PoC	- Viết một bộ các trường hợp kiểm thử (test cases) cho PoC. - Bao gồm: - **Test Case 1 (Happy path):** Kịch bản có lỗi, hệ thống phải phát hiện và giải thích đúng. - **Test Case 2 (Normal operation):** Kịch bản không có lỗi, hệ thống không được báo động sai. - **Test Case 3 (Edge case):** Một lỗi tinh vi, khó phát hiện hơn.	- **Tài liệu Test Cases:** Mô tả rõ các bước thực hiện, dữ liệu đầu vào và kết quả mong đợi cho từng test case. - **Tiêu chí:** Bao phủ được các trường hợp hoạt động chính của hệ thống.
	4. Thực thi Kiểm thử, Phân tích Kết quả & Mô phỏng "Learn"	- Sau khi Hiếu hoàn thành `main.py`, Hường sẽ là người **chạy các test case** đã thiết kế. - Ghi nhận kết quả thực tế và so sánh với kết quả mong đợi. - Báo cáo lỗi (bugs) cho Hiếu một cách chi tiết.	- **Báo cáo kết quả kiểm thử:** Ghi nhận pass/fail cho từng test case, mô tả chi tiết các lỗi. - **File `feedback_log.txt`:** Chứa các phản hồi từ người dùng mô phỏng.

		- **Mô phỏng vòng lặp "Learn":** Sau khi chạy demo, hệ thống sẽ hỏi "Is this explanation correct? (y/n)". Hường sẽ nhập câu trả lời, và hệ thống sẽ ghi lại phản hồi này vào một file log đơn giản (`feedback_log.txt`).	- **Tiêu chí:** Quá trình kiểm thử được thực hiện đầy đủ, các lỗi được ghi nhận rõ ràng, vòng lặp "Learn" đơn giản được chứng minh.
Kết hợp & Đánh giá	**Buổi Demo & Tổng kết Giai đoạn 2**	- Hiếu và Hường cùng nhau chạy file `main.py` để trình diễn toàn bộ "lát cắt dọc". - Hường trình bày kết quả kiểm thử và các phân tích về độ chính xác của Gemini. - Thảo luận về các khó khăn, các điểm cần cải thiện trong kiến trúc và code. - Chốt lại các mục tiêu và kế hoạch chi tiết cho Giai đoạn 3 (Mở rộng & Đánh giá định lượng).	- **Sản phẩm PoC chạy được:** Một chương trình Python hoàn chỉnh, có thể demo trực tiếp. - **Chứng minh được khái niệm:** Vòng lặp Sense-Detect-Explain-Act-Learn (đơn giản) được hiện thực hóa. - **Kế hoạch Giai đoạn 3 rõ ràng:** Xác định các module cần nâng cấp và các metrics cần đo lường.

Bảng Phân Công chi tiết cho Giai đoạn 3

Vai trò/Người	Đầu việc cụ thể	Mô tả chi tiết & Hướng dẫn	Sản phẩm đầu ra & Tiêu chí đánh giá
Hiếu (INTJ)	**1. Nâng cấp `Anomaly Detector` (AD) lên mô hình ML thực thụ**	- **Lựa chọn & Code các mô hình ML:** Triển khai các mô hình đã chọn như Isolation Forest, One-Class SVM, hoặc một mạng LSTM/TCN đơn giản bằng các thư viện như Scikit-learn, PyTorch/TensorFlow. - **Thiết lập Vertex AI Training:** Viết script để đóng gói code huấn luyện và gửi lên Vertex AI Training Job. Việc này giúp tận dụng sức mạnh của cloud để huấn luyện	- **Code huấn luyện cho 2-3 mô hình AD:** Các file Python có thể chạy được để huấn luyện các mô hình ML/DL. - **Script để chạy Vertex AI Training Job:** Một file shell hoặc Python để tự động hóa việc huấn luyện trên cloud. - **Các file model đã huấn luyện:** Được lưu

		trên bộ dữ liệu lớn. - **Lưu trữ mô hình đã huấn luyện:** Sau khi huấn luyện, lưu file mô hình (ví dụ: `.pkl`, `.h5`) vào Google Cloud Storage.	trữ trên Cloud Storage. - **Tiêu chí:** Quá trình huấn luyện có thể lặp lại, mô hình được lưu trữ một cách có tổ chức.
	2. Triển khai AD dưới dạng API Endpoint	- **Viết code để phục vụ mô hình (serving):** Tạo một ứng dụng web nhỏ (ví dụ: dùng Flask hoặc FastAPI) để tải mô hình đã huấn luyện từ Cloud Storage và cung cấp một endpoint API (ví dụ: `/predict`). - **Triển khai lên Vertex AI Endpoint hoặc Cloud Run:** Đóng gói ứng dụng vào một container (Docker) và triển khai lên GCP để nó có thể nhận request và trả về dự đoán real-time.	- **Code ứng dụng serving:** Chứa logic để tải và gọi mô hình. - **Dockerfile:** Để đóng gói ứng dụng. - **Một API endpoint hoạt động trên GCP:** Có thể gửi dữ liệu Telemetry đến và nhận lại dự đoán bất thường. - **Tiêu chí:** Anomaly Detector có thể được gọi như một dịch vụ độc lập, sẵn sàng tích hợp.
	3. Xây dựng Hệ thống Chạy Thử nghiệm Hàng loạt (Batch Testing System)	- Viết một script chính (`batch_runner.py`) để tự động hóa toàn bộ quá trình đánh giá: 1. Vòng lặp chạy 1000 lần. 2. Trong mỗi lần, gọi `simulator.py` để tạo một kịch bản ngẫu nhiên (có lỗi hoặc không). 3. Gọi API của Anomaly Detector trên GCP. 4. Gọi Gemini API để phân loại HFACS. 5. So sánh kết quả của AI với "sự thật" từ simulator. 6. **Ghi lại kết quả (True Positive, False Positive, False Negative...) vào một file CSV.**	- **Script `batch_runner.py`:** Có khả năng chạy tự động và ghi lại kết quả chi tiết. - **File `results.csv`:** Một file lớn chứa kết quả của hàng ngàn lần chạy thử nghiệm. - **Tiêu chí:** Hệ thống có thể tự động chạy và thu thập dữ liệu hiệu suất một cách đáng tin cậy.
	4. Hướng dẫn & Hỗ trợ kỹ thuật	- Hướng dẫn Hường cách sử dụng các công cụ phân tích dữ liệu (Pandas) và trực quan hóa (Matplotlib/Seaborn) để xử lý file `results.csv`. - Giải thích ý nghĩa của các metrics kỹ thuật và cách chúng được tính toán.	- **Buổi hướng dẫn về phân tích dữ liệu hiệu suất.** - **Các đoạn code mẫu (snippets) cho Pandas và Matplotlib.**

		- Cung cấp các đoạn code mẫu để Hường bắt đầu phân tích.	
Hường (ENFJ)	**1. Mở rộng & Phân loại "Ground Truth" cho Simulator**	- Làm việc với Hiếu để đảm bảo `simulator.py` có thể tạo ra nhiều loại kịch bản ngẫu nhiên hơn (nhiều loại lỗi, nhiều mức độ nhiễu khác nhau). - **Quan trọng:** Định nghĩa rõ "sự thật" (ground truth) cho mỗi kịch bản mà simulator tạo ra. Ví dụ: "kịch bản X là một `True Anomaly` và thuộc loại HFACS `Skill-Based Error`". Thông tin này cần được simulator trả về để hệ thống batch testing có thể so sánh.	- **Tài liệu "Simulation Ground Truth Specification":** Mô tả cách simulator sẽ gán nhãn "sự thật" cho dữ liệu nó tạo ra. - **Tiêu chí:** "Sự thật" được định nghĩa rõ ràng, nhất quán và có thể được truy xuất tự động.
	2. Phân tích Định lượng & Trực quan hóa Kết quả	- **Đây là nhiệm vụ cốt lõi của Hường.** Sử dụng thư viện Pandas trong Python để đọc file `results.csv`. - **Tính toán các chỉ số hiệu suất chính:** - Thời gian trung bình để phát hiện (Mean Time to Detect - MTTD). - Tỷ lệ cảnh báo sai (False Positive Rate - FPR). - Tỷ lệ bỏ sót lỗi (False Negative Rate - FNR). - Độ chính xác của việc phân loại HFACS (so sánh kết quả của Gemini với ground truth). - **Trực quan hóa kết quả:** Tạo các biểu đồ (bar chart, line chart, confusion matrix) bằng Matplotlib/Seaborn để minh họa các kết quả này.	- **Jupyter Notebook phân tích kết quả (`analysis.ipynb`):** Chứa code phân tích, tính toán metrics và các biểu đồ. - **Các file ảnh biểu đồ (`.png`):** Sẵn sàng để chèn vào bài báo khoa học. - **Báo cáo phân tích hiệu suất:** Một tài liệu tóm tắt các kết quả chính và ý nghĩa của chúng. - **Tiêu chí:** Phân tích chính xác, biểu đồ rõ ràng, dễ hiểu và đưa ra được các kết luận có ý nghĩa.
	3. Viết phần "Methodology" & "Results" cho Bài báo Khoa học	- Dựa trên các công việc đã làm, bắt đầu viết hai chương quan trọng nhất của bài báo: - **Methodology (Phương pháp luận):** Mô tả chi tiết kiến trúc hệ thống, các mô hình AI đã sử dụng (cả AD và LLM), quy trình huấn luyện, và cách thiết lập hệ thống thử	- **Bản nháp của hai chương "Methodology" và "Results":** Được viết bằng ngôn ngữ học thuật. - **Tiêu chí:** Mô tả rõ ràng, chính xác, logic và có thể tái tạo được (reproducible).

		nghiệm hàng loạt. - **Results (Kết quả):** Trình bày các chỉ số hiệu suất đã tính toán và các biểu đồ đã vẽ một cách khoa học.	
	4. Đề xuất Hướng cải thiện	- Dựa trên phân tích kết quả (ví dụ: "FPR cao ở kịch bản thời tiết xấu", "Gemini hay nhầm lẫn giữa hai loại lỗi HFACS X và Y"), đề xuất các hướng cụ thể để cải thiện hệ thống cho giai đoạn sau (ví dụ: "cần thêm dữ liệu thời tiết vào mô hình AD", "cần tinh chỉnh lại prompt cho Gemini về hai loại lỗi này").	- **Một danh sách các đề xuất cải thiện:** Dựa trên dữ liệu phân tích, có tính xây dựng. - **Tiêu chí:** Các đề xuất hợp lý, có bằng chứng từ dữ liệu.
Kết hợp & Đánh giá	**Buổi Review Kết quả Định lượng & Kế hoạch Viết bài**	- Hiếu trình bày lại kiến trúc kỹ thuật đã triển khai trên GCP. - Hường trình bày chi tiết các kết quả phân tích định lượng, các biểu đồ và ý nghĩa của chúng. - Cả hai cùng thảo luận về các kết quả: Hệ thống hoạt động tốt ở đâu? Yếu ở đâu? Tại sao? - Lên kế hoạch chi tiết cho Giai đoạn 4: Hoàn thiện bài báo khoa học.	- **Bộ kết quả định lượng hoàn chỉnh:** Cung cấp bằng chứng khách quan về hiệu suất của hệ thống. - **Bản nháp các chương quan trọng của bài báo:** Nền tảng vững chắc để hoàn thiện. - **Sự hiểu biết sâu sắc về điểm mạnh/yếu của hệ thống:** Giúp phần "Discussion" và "Future Work" trong bài báo trở nên sâu sắc hơn.

Bảng Phân Công chi tiết cho Giai đoạn 4

Vai trò/Người	Đầu việc cụ thể	Mô tả chi tiết & Hướng dẫn	Sản phẩm đầu ra & Tiêu chí đánh giá
Hiếu (INTJ)	**1. Hoàn thiện Tài liệu Kỹ thuật & Sơ đồ Hệ thống**	- Rà soát và hoàn thiện tất cả các sơ đồ kiến trúc (sơ đồ tổng thể, luồng dữ liệu, MLOps pipeline) để đảm bảo chúng rõ ràng, chính xác và có chất lượng cao (dùng cho việc chèn vào bài báo). - Viết các đoạn mô tả kỹ thuật ngắn gọn, chính xác về các thuật toán, công nghệ đã sử dụng (ví dụ: mô tả TCN, cách triển khai Vertex AI Endpoint) để Hường có thể tích hợp vào phần Methodology.	- **Bộ sơ đồ kiến trúc cuối cùng:** Định dạng vector hoặc độ phân giải cao, có chú thích rõ ràng. - **Tài liệu "Technical Descriptions":** Một file chứa các đoạn mô tả kỹ thuật chuẩn xác. - **Tiêu chí:** Chuyên nghiệp, dễ hiểu, sẵn sàng để sử dụng ngay.
	2. Chuẩn bị Code & Dữ liệu để Công bố (Optional but highly recommended)	- Dọn dẹp lại code trong repository GitHub: thêm comment, viết `README.md` chi tiết hướng dẫn cách cài đặt và chạy lại các thử nghiệm. - Chuẩn bị một bộ dữ liệu mô phỏng mẫu và script `batch_runner.py` để người khác có thể tái tạo lại kết quả (reproducibility). Đây là một điểm cộng rất lớn cho các bài báo khoa học.	- **Repository GitHub "sạch sẽ":** Có tài liệu hướng dẫn rõ ràng. - **Một gói dữ liệu và code mẫu:** Để chứng minh tính tái tạo của nghiên cứu. - **Tiêu chí:** Một nhà nghiên cứu khác có thể đọc và hiểu, hoặc thậm chí chạy lại được thử nghiệm.
	3. Review Tổng thể & Phản biện (Devil's Advocate)	- **Đọc và phản biện bản nháp bài báo của Hường từ góc độ kỹ thuật và logic:** Đặt các câu hỏi khó như "Tại sao lại chọn mô hình này mà không phải mô hình kia?", "Kết luận này có thực sự được chứng minh bởi dữ liệu không?", "Hạn chế này có quá lớn không?". - Kiểm tra sự nhất quán giữa các phần, đảm bảo các con số và mô tả kỹ thuật là chính xác tuyệt đối.	- **Các bình luận (comments) và đề xuất chỉnh sửa chi tiết** trên bản nháp bài báo. - **Tiêu chí:** Giúp bài báo trở nên chặt chẽ hơn, lường trước được các câu hỏi từ người phản biện (reviewers) thực tế.

Hường (ENFJ)	**1. Dẫn dắt và Hoàn thiện Toàn bộ Bài báo**	- **Đây là nhiệm vụ cốt lõi của Hường.** Dựa trên các bản nháp từ GĐ3, viết và kết nối tất cả các phần còn lại: - **Abstract (Tóm tắt):** Viết cuối cùng, tóm gọn toàn bộ nghiên cứu. - **Introduction (Giới thiệu):** Nêu vấn đề, tầm quan trọng, và tóm tắt đóng góp của bài báo. - **Related Work (Tổng quan tài liệu):** So sánh với các nghiên cứu trước. - **Discussion (Thảo luận):** Diễn giải ý nghĩa sâu hơn của kết quả. Tại sao hệ thống hoạt động tốt/chưa tốt? Ý nghĩa thực tiễn là gì? - **Limitations & Future Work (Hạn chế & Hướng phát triển):** Thẳng thắn nêu các hạn chế và đề xuất các hướng nghiên cứu tiếp theo. - **Conclusion (Kết luận):** Tóm tắt lại các đóng góp chính.	- **Bản thảo hoàn chỉnh lần 1 của bài báo khoa học.** - **Tiêu chí:** Mạch lạc, logic, tuân thủ cấu trúc của một bài báo khoa học chuẩn, văn phong trôi chảy.
	2. Tích hợp Sơ đồ và Biểu đồ	- Chèn tất cả các sơ đồ kiến trúc từ Hiếu và các biểu đồ kết quả từ GĐ3 vào đúng vị trí trong bài báo. - Viết chú thích (captions) rõ ràng, giải thích ý nghĩa của từng hình ảnh, biểu đồ.	- **Bài báo có đầy đủ hình ảnh, biểu đồ, bảng biểu** với chú thích chuyên nghiệp. - **Tiêu chí:** Hình ảnh minh họa hiệu quả cho nội dung văn bản.
	3. Định dạng & Rà soát Ngữ pháp, Chính tả	- Định dạng bài báo theo một template cụ thể (ví dụ: IEEE, ACM, hoặc của tạp chí/hội nghị mục tiêu). - Chỉnh sửa References (Tài liệu tham khảo). - Sử dụng các công cụ như Grammarly để kiểm tra và sửa lỗi ngữ pháp, chính tả, đảm bảo văn phong học thuật chuyên nghiệp.	- **Bản thảo hoàn chỉnh lần 2:** Đã được định dạng và rà soát lỗi ngôn ngữ. - **Tiêu chí:** Sạch sẽ, chuyên nghiệp, không còn lỗi chính tả/ngữ pháp cơ bản.

	4. Chuẩn bị cho việc Nộp bài	- Nghiên cứu và lựa chọn các hội nghị hoặc tạp chí khoa học phù hợp với chủ đề của bài báo. - Chuẩn bị các tài liệu đi kèm nếu cần (ví dụ: cover letter).	- **Danh sách các địa điểm nộp bài tiềm năng.** - **Bản thảo cuối cùng, sẵn sàng để nộp.** - **Tiêu chí:** Hoàn thành mọi thứ cần thiết cho quá trình nộp bài.
Kết hợp & Đánh giá	**Buổi Review Cuối cùng & Ăn mừng**	- Hiếu và Hường cùng nhau đọc lại toàn bộ bài báo một lần cuối. - Chốt lại phiên bản cuối cùng để nộp. - Cùng nhìn lại toàn bộ quá trình từ Giai đoạn 1 đến 4, ghi nhận những gì đã học được và thành quả đã đạt được. - **Ăn mừng!** Đây là một cột mốc quan trọng.	- **Bài báo khoa học phiên bản cuối cùng (Final Version):** Cả hai đều hài lòng và tự hào về sản phẩm. - **Sự sẵn sàng để bắt đầu một dự án mới hoặc phát triển tiếp dự án này.**