TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

NGHIÊN CỨU VÀ ỨNG DỤNG

CHATBOT AI TRONG PHÁT TRIỂN WEBSITE MẠNG XÃ HỘI

VŨ THƯỢNG HẢI

Hà Nội – Năm 2026

TRƯỜNG ĐẠI HỌC TÀI NGUYÊN VÀ MÔI TRƯỜNG HÀ NỘI

KHOA CÔNG NGHỆ THÔNG TIN

NGHIÊN CỨU VÀ ỨNG DỤNG

CHATBOT AI TRONG PHÁT TRIỂN WEBSITE MẠNG XÃ HỘI

Họ và tên sinh viên:	**Vũ Thượng Hải**
Mã sinh viên:	**22111060108**
Lớp:	**ĐH12C1**
Ngành đào tạo:	**Công Nghệ Thông Tin**
NGƯỜI HƯỚNG DẪN:	**ThS. TRƯƠNG MẠNH ĐẠT**

Hà Nội – Năm 2026

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM

Độc lập – Tự do – Hạnh phúc

BẢN CAM ĐOAN

Tên tôi là: **Vũ Thượng Hải**

Mã sinh viên: **22111060108** Lớp: **ĐH12C1**

Ngành: **Công Nghệ Thông Tin**

Tôi đã thực hiện khóa luận tốt nghiệp với đề tài: **Nghiên cứu và ứng dụng Chatbot AI trong phát triển website mạng xã hội**

Tôi xin cam đoan đây là đề tài nghiên cứu của riêng tôi và được sự hướng dẫn của **ThS. Trương Mạnh Đạt**. Các nội dung nghiên cứu, kết quả trong đề tài này là trung thực và chưa được công bố dưới bất kỳ hình thức nào. Nếu phát hiện có bất kỳ hình thức gian lận nào tôi xin hoàn toàn chịu trách nhiệm trước pháp luật.

Hà Nội, ngày ……tháng ……năm 2026

Giáo viên hướng dẫn **Sinh viên**

ThS. Trương Mạnh Đạt **Vũ Thượng Hải**

LỜI CẢM ƠN

Trong quá trình nghiên cứu đề tài "**Nghiên cứu và ứng dụng Chatbot AI trong phát triển website mạng xã hội**", em xin cảm ơn tới các thầy cô giảng viên khoa Công nghệ thông tin trường Đại học Tài nguyên và Môi trường Hà Nội đã luôn hướng dẫn em để có thể hoàn thành bài khóa luận một cách tốt nhất.

Đặc biệt, em xin gửi lời cảm ơn sâu sắc tới **ThS. Trương Mạnh Đạt**, người đã hướng dẫn trực tiếp cho em trong khóa luận lần này. Nhờ có sự hướng dẫn chi tiết, tận tình của thầy mà em đã khắc phục được nhiều sai sót trong khóa luận lần này.

Tuy nhiên với những hạn chế về kiến thức và thời gian nên không thể tránh khỏi sai sót, em rất mong nhận được những nhận xét góp ý chỉ bảo của thầy cô để chương trình được hoàn thiện hơn, có tính thực tiễn để có thể áp dụng trong tương lai.

Em xin chân thành cảm ơn!

TÓM TẮT

Nội dung tóm tắt khóa luận tốt nghiệp.

MỤC LỤC

DANH SÁCH BẢNG

DANH SÁCH HÌNH VẼ

DANH MỤC KÝ HIỆU, CHỮ VIẾT TẮT

Ký hiệu	Diễn giải
UC	Use Case (Ca sử dụng)
US	User Story (Câu chuyện người dùng)
MD	Module Design (Thiết kế module)
AI	Artificial Intelligence (Trí tuệ nhân tạo)
API	Application Programming Interface
CSDL	Cơ sở dữ liệu
CMS	Content Management System
YAML	YAML Ain’t Markup Language
JSON	JavaScript Object Notation
DTO	Data Transfer Object
UML	Unified Modeling Language
ERD	Entity Relationship Diagram
SSD	Solid State Drive (for local storage)
SSE	Server-Sent Events

MỞ ĐẦU

0.1 Giới thiệu đơn vị thực tập

Công ty Cổ phần Innotech (INNOTECH J.S.C) là một trong những đơn vị hàng đầu trong lĩnh vực cung cấp các giải pháp công nghệ thông tin và chuyển đổi số toàn diện cho doanh nghiệp tại Việt Nam.

- **Tên đầy đủ:** Công ty Cổ phần Innotech.
- **Địa chỉ trụ sở:** NV05 đường Foresa 4, Khu đô thị sinh thái Xuân Phương, Phường Xuân Phương, Quận Nam Từ Liêm, Thành phố Hà Nội.
- **Website:** `https://innotechjsc.com/`

Với tôn chỉ hoạt động **"Single Portal For All"**, Innotech hướng tới mục tiêu trở thành đối tác công nghệ tin cậy, cung cấp giải pháp một cửa cho mọi nhu cầu công nghệ của doanh nghiệp. Sứ mệnh của công ty là cam kết mang lại trải nghiệm chuyển đổi số hiệu quả, an toàn và được tùy chỉnh tối ưu, phù hợp với đặc thù riêng biệt của từng khách hàng.

Các lĩnh vực hoạt động và dịch vụ chính của Innotech bao gồm:

- **Chuyển đổi số:** Tư vấn và triển khai chiến lược số hóa quy trình doanh nghiệp.
- **Hệ thống ERP:** Xây dựng hệ thống quản trị nguồn lực doanh nghiệp tích hợp.
- **An ninh mạng:** Giải pháp bảo mật, tường lửa và bảo vệ dữ liệu tiên tiến.

- **AI cho doanh nghiệp:** Ứng dụng trí tuệ nhân tạo vào tự động hóa và phân tích dữ liệu.
- **Digital Marketing:** Hỗ trợ doanh nghiệp phát triển trên các nền tảng số.

Trong hơn 10 năm hoạt động, Innotech đã triển khai thành công hơn 100 dự án cho các doanh nghiệp và tập đoàn lớn như Kiến Tạo Xanh Group, ITM Group, OVAN Group, HANOTEX,... Với đội ngũ hơn 50 chuyên gia giàu kinh nghiệm, công ty luôn đảm bảo các giá trị cốt lõi về sự chuyên nghiệp, tính linh hoạt, bảo mật và hiệu quả kinh tế cho đối tác.

0.2 Lý do chọn đề tài

Trong bối cảnh công nghệ số và trí tuệ nhân tạo phát triển mạnh mẽ như hiện nay, với sự ra đời của các công cụ AI thế hệ mới như ChatGPT, GitHub Copilot, và Claude, ngành công nghệ thông tin đang trải qua một cuộc cách mạng về cách tiếp cận phát triển phần mềm. Theo báo cáo của GitHub năm 2024, hơn 92% các nhà phát triển chuyên nghiệp đã sử dụng công cụ AI trong công việc, và năng suất code tăng trung bình 55% khi có sự hỗ trợ của AI. Tuy nhiên, khoảng cách về kỹ năng giữa các cấp độ lập trình viên—từ người mới bắt đầu, sinh viên đến các lập trình viên có kinh nghiệm—vẫn là một thách thức lớn trong giáo dục và đào tạo công nghệ thông tin.

Đối với sinh viên và người mới học lập trình, việc tiếp cận một dự án thực tế thường gặp phải nhiều rào cản: thiếu kiến thức về kiến trúc hệ thống, chưa nắm vững quy trình phát triển phần mềm chuẩn, và đặc biệt là khoảng cách giữa lý thuyết học trên trường với thực tiễn triển khai ứng dụng. Theo khảo sát của Stack Overflow Developer Survey 2024, trung bình một lập trình viên mới cần từ 2-3 năm kinh nghiệm thực tế để có thể tự tin triển khai một hệ thống hoàn chỉnh. Khoảng cách này không chỉ làm chậm quá trình phát triển

nghề nghiệp của sinh viên mà còn tạo ra sự chênh lệch lớn về năng lực giữa các nhóm đối tượng trong ngành.

Tuy nhiên, sự xuất hiện của các công cụ AI đã mở ra một hướng đi mới. AI không chỉ hỗ trợ viết code mà còn có khả năng đóng vai trò như một *chuyên gia tư vấn ảo*, giúp sinh viên và lập trình viên thiếu kinh nghiệm tiếp cận các quy trình, pattern, và best practices một cách có hệ thống. Nghiên cứu gần đây của MIT và Harvard cho thấy sinh viên sử dụng AI assistant có tốc độ học tập nhanh hơn 40% và mức độ tự tin khi giải quyết vấn đề phức tạp tăng 65% so với nhóm không sử dụng AI.

Xuất phát từ nhận định này, đề tài **"Nghiên cứu và ứng dụng Chatbot AI trong phát triển website mạng xã hội"** được lựa chọn với mục tiêu xây dựng một hệ thống Agent Skill Framework—bộ công cụ AI chuyên biệt hóa cho từng giai đoạn phát triển phần mềm (phân tích, thiết kế, triển khai, kiểm định). Thay vì chỉ cung cấp câu trả lời code đơn lẻ, hệ thống này sẽ hướng dẫn người dùng theo quy trình chuẩn mực, từ phân tích nghiệp vụ (Business Process Flow), thiết kế kiến trúc (Sequence Diagram, Class Diagram), đến triển khai code tuân thủ best practices. Điều này giúp rút ngắn đáng kể khoảng cách giữa người mới học và lập trình viên có kinh nghiệm, đồng thời tạo ra một "tri thức số hóa"(Knowledge Factory) có thể tái sử dụng và mở rộng cho các dự án tương lai.

Tính cấp thiết của đề tài còn được thể hiện qua nhu cầu thực tế tại Công ty Innotech—nơi triển khai thực tập và thực hiện đề tài—khi công ty đang trong quá trình mở rộng dự án chuyển đổi số và cần xây dựng quy trình làm việc tối ưu hóa bởi AI để nâng cao năng suất và chất lượng sản phẩm. Hơn nữa, việc nghiên cứu và triển khai hệ thống này trên nền tảng mạng xã hội chia sẻ kiến thức (Steve Void) còn tạo ra giá trị cộng đồng, giúp sinh viên và lập trình viên Việt Nam có thể tiếp cận và chia sẻ kiến thức một cách có hệ thống và hiệu quả hơn.

0.3 Mục tiêu nghiên cứu

Đề tài hướng đến các mục tiêu nghiên cứu cụ thể sau:

Mục tiêu tổng quát: Nghiên cứu và xây dựng hệ thống Agent Skill Framework—một bộ công cụ AI chuyên biệt hóa hỗ trợ quy trình phát triển phần mềm, được tích hợp vào nền tảng mạng xã hội chia sẻ kiến thức Steve Void, nhằm rút ngắn khoảng cách kỹ năng giữa các nhóm lập trình viên và tối ưu hóa quy trình phát triển sản phẩm.

Các mục tiêu cụ thể:

1. **Nghiên cứu và thiết kế kiến trúc Agent Skill Framework:**
    - Phân tích các mô hình AI Agent hiện đại (ReAct, Tool-Use, Multi-Agent Systems)
    - Thiết kế kiến trúc 3 Pillars (Knowledge, Process, Guardrails) và 7 Zones (Core, Knowledge, Scripts, Templates, Data, Loop, Assets)
    - Xây dựng Meta-Skill Framework cho việc tạo và quản lý các Skills
2. **Triển khai bộ Agent Skills cho giai đoạn Life-2 (Analysis & Design):**
    - Skill Architect: Thiết kế kiến trúc Skills mới
    - Flow Design Analyst: Phân tích và vẽ Business Process Flow Diagram
    - Sequence Design Analyst: Thiết kế Sequence Diagram chuẩn UML
    - Activity Diagram Analyst: Vẽ sơ đồ hoạt động theo tư duy Clean Architecture
    - Class Diagram Analyst: Phân tích cấu trúc dữ liệu MongoDB/Payload CMS

- UI Architecture Analyst: Trích xuất UI Screen Specs từ Schema và Diagrams

3. **Xây dựng nền tảng mạng xã hội Steve Void:**
    - Phát triển website với 6 modules chính: Authentication, Content Engine, Discovery Feed, Engagement, Bookmarking, Notifications & Moderation
    - Tích hợp Agent Skills vào quy trình phát triển thực tế
    - Xây dựng Knowledge Factory—kho tri thức có thể tái sử dụng và mở rộng
4. **Đánh giá hiệu quả và khả năng mở rộng:**
    - So sánh năng suất phát triển khi có và không có Agent Skills
    - Đánh giá chất lượng tài liệu thiết kế được tạo ra (Diagrams, Specs)
    - Khảo sát mức độ hài lòng của sinh viên và lập trình viên khi sử dụng hệ thống

0.4 Đối tượng và phạm vi nghiên cứu

0.4.1 Phạm vi nghiên cứu

Đề tài tập trung vào các phạm vi nghiên cứu cụ thể sau:

Về mặt công nghệ:

- *AI Agent Framework:* Nghiên cứu kiến trúc Agent Skills dựa trên Claude 3.5 Sonnet API của Anthropic, tích hợp với Claude Code CLI
- *Backend:* Payload CMS 3.x (Headless CMS) với MongoDB Atlas, tập trung vào Local API và Hooks Pattern

- *Frontend:* Next.js 15 (App Router, React Server Components) với Tailwind CSS v4 và Radix UI
- *Realtime:* Server-Sent Events (SSE) cho hệ thống thông báo

Về mặt chức năng:

- Giai đoạn **Life-2 (Analysis & Design)** của quy trình phát triển phần mềm, bao gồm:
  - Phân tích nghiệp vụ (Business Process Flow)
  - Thiết kế tương tác (Sequence Diagram, Activity Diagram)
  - Thiết kế cấu trúc dữ liệu (Class Diagram, ER Diagram)
  - Thiết kế giao diện người dùng (UI Wireframes, Screen Specs)
- Triển khai 6 modules chính của Steve Void: M1 (Auth & Profile), M2 (Content Engine), M3 (Discovery Feed), M4 (Engagement), M5 (Bookmarking), M6 (Notifications & Moderation)

Giới hạn phạm vi:

- Đề tài **không** nghiên cứu việc huấn luyện mô hình AI mới (sử dụng Claude API có sẵn)
- Không triển khai hệ thống trên production scale (giới hạn ở môi trường phát triển và staging)
- Không nghiên cứu sâu về bảo mật hệ thống (chỉ áp dụng các best practices cơ bản)
- Tập trung vào giai đoạn Life-2; các giai đoạn Life-3 (Implementation) và Life-4 (Verification) được nghiên cứu ở mức độ tổng quan

0.4.2 Đối tượng nghiên cứu

1. **Hệ thống Agent Skill Framework:**

    - Kiến trúc 3 Pillars: Knowledge (kho tri thức), Process (quy trình làm việc), Guardrails (cơ chế kiểm soát chất lượng)
    - Cấu trúc 7 Zones: Core (SKILL.md), Knowledge (standards), Scripts (automation), Templates (output formats), Data (samples), Loop (quality control), Assets (diagrams/images)
    - Meta-Skill Framework: quy trình 3 pha (Input/Output, Checklist/Plan, Research/Action) với vòng lặp Verify-Fix-Loop

2. **Bộ Agent Skills chuyên biệt cho Life-2:**

    - *skill-architect:* Thiết kế kiến trúc Skills mới
    - *flow-design-analyst:* Phân tích Business Process Flow (Swimlane 3-lane: User/System/DB)
    - *sequence-design-analyst:* Thiết kế Sequence Diagram chuẩn UML
    - *activity-diagram-design-analyst:* Vẽ Activity Diagram theo Clean Architecture
    - *class-diagram-analyst:* Phân tích Class Diagram dual-format (Mermaid + YAML Contract)
    - *ui-architecture-analyst:* Trích xuất UI Screen Specs từ Schema và Diagrams

3. **Nền tảng mạng xã hội Steve Void:**

    - Database Schema Design (MongoDB document structure, Aggregate Roots, Embedded Documents)

- API Specification (RESTful endpoints, Payload Local API patterns)
- UI/UX Wireframes (Neobrutalism design system với Pink primary color)
- 6 modules chức năng từ M1 đến M6

4. **Quy trình làm việc tích hợp AI:**
    - OpenSpec Workflow: quản lý thay đổi theo artifact (problem, solution, implementation, verification)
    - Knowledge Factory: cơ chế tái sử dụng tri thức từ các Skills đã thực thi
    - Interaction Points (IP-1, IP-2, IP-3): các điểm dừng để xác nhận với người dùng

0.5 Phương pháp nghiên cứu

Đề tài áp dụng các phương pháp nghiên cứu kết hợp giữa lý thuyết và thực nghiệm như sau:

1. **Phương pháp nghiên cứu tài liệu (Literature Review):**
    - Nghiên cứu các mô hình AI Agent hiện đại: ReAct (Reasoning and Acting), Tool-Use Agents, Multi-Agent Systems
    - Phân tích các framework phát triển phần mềm: Agile, Clean Architecture, Domain-Driven Design
    - Tham khảo tài liệu kỹ thuật: Anthropic Claude API Documentation, Payload CMS Docs, Next.js 15 Docs, MongoDB Best Practices

- Nghiên cứu các công trình liên quan: GitHub Copilot Workspace, Cursor AI, v0.dev (Vercel)

2. **Phương pháp phân tích và thiết kế hệ thống:**

   - *Phân tích yêu cầu:* Sử dụng User Stories và Use Case Diagram để xác định chức năng hệ thống
   - *Thiết kế kiến trúc:* Áp dụng Clean Architecture với 3 lớp (Business Logic, Use Case, External)
   - *Thiết kế cơ sở dữ liệu:* Sử dụng ER Diagram và MongoDB Schema Design Patterns (Aggregate Root vs Embedded Document)
   - *Thiết kế giao diện:* Wireframing với Figma/Pencil, áp dụng Neobrutalism design system
   - *Modeling với UML:* Sequence Diagram, Activity Diagram, Class Diagram (Mermaid format)

3. **Phương pháp thực nghiệm và triển khai:**

   - *Iterative Development:* Phát triển theo từng module (M1 → M6), mỗi module trải qua 4 giai đoạn Life (Vision, Design, Implementation, Verification)
   - *Rapid Prototyping:* Xây dựng Agent Skills theo chu trình: Architect → Planner → Builder với feedback loop
   - *Test-Driven Approach:* Mỗi Skill có Quality Control Checklist và Self-Scoring mechanism
   - *Git-based Version Control:* Quản lý mã nguồn với Git, sử dụng OpenSpec Workflow cho change management

4. **Phương pháp đánh giá và kiểm thử:**

- *Code Review:* Sử dụng spec-reviewer agent để kiểm tra implementation vs specification
- *Quality Metrics:* Đo lường thời gian phát triển, số lượng iteration cần thiết, tỷ lệ lỗi phát hiện
- *User Acceptance Testing:* Thu thập feedback từ sinh viên và lập trình viên về trải nghiệm sử dụng Agent Skills
- *Documentation Review:* Đánh giá chất lượng tài liệu thiết kế (diagrams, specs) qua rubric chuẩn

5. **Phương pháp thu thập và phân tích dữ liệu:**

   - *Session Logs:* Ghi nhận toàn bộ transcript của Agent Sessions (`~/.claude/projects/`)
   - *Auto Memory:* Lưu trữ lessons learned và patterns phát hiện trong quá trình phát triển
   - *Metrics Dashboard:* Theo dõi số lượng Skills được tạo, thời gian trung bình mỗi artifact, tỷ lệ thành công
   - *Survey & Interview:* Khảo sát định lượng (Likert scale) và phỏng vấn định tính với người dùng

Các phương pháp trên được áp dụng tuần tự và song song, tạo thành một quy trình nghiên cứu khoa học chặt chẽ từ lý thuyết đến thực tiễn, từ thiết kế đến triển khai và đánh giá.

0.6 Dự kiến kết quả nghiên cứu đạt được

Sau khi hoàn thành nghiên cứu, đề tài dự kiến đạt được các kết quả cụ thể sau:

0.6.1 Sản phẩm phần mềm

1. **Hệ thống Agent Skill Framework hoàn chỉnh:**
    - 28 Agent Skills chuyên biệt cho các giai đoạn phát triển phần mềm (Life-1 đến Life-4)
    - Meta-Skill Framework: Master Skill orchestrator với skill-architect, skill-planner, skill-builder
    - OpenSpec Workflow System: quản lý change với 8 skills (new, continue, apply, verify, sync, archive, explore, ff)
    - Skill Repository tại `.claude/skills/` và `.agent/skills/`
2. **Nền tảng mạng xã hội Steve Void (MVP):**
    - Backend: Payload CMS 3.x với 15+ collections (Users, Posts, Comments, Connections, Bookmarks, Notifications, Reports...)
    - Frontend: Next.js 15 App với 30+ screens (Feed, Profile, Post Detail, Search, Collections...)
    - Database: MongoDB Atlas schema với 500MB data (test/staging)
    - Realtime: SSE-based notification system
    - Authentication: JWT-based auth với email verification và password reset
3. **Design System và Component Library:**
    - Neobrutalism UI Kit: 50+ components (Buttons, Cards, Forms, Modals...) với Tailwind v4 + Radix UI
    - Responsive layouts cho Desktop, Tablet, Mobile
    - Pink primary color scheme với high-contrast, bold borders, offset shadows

0.6.2 Tài liệu nghiên cứu và thiết kế

1. **Tài liệu Life-1 (Vision & Research):**
    - Product Vision Document
    - User Personas (5 personas chính)
    - User Stories (100+ stories cho 6 modules)
    - Technical Decisions Document
2. **Tài liệu Life-2 (Analysis & Design):**
    - Database Schema Design (15+ collections với field-level documentation)
    - API Specification (60+ endpoints với request/response schema)
    - UML Diagrams: 20+ Sequence Diagrams, 15+ Activity Diagrams, 6+ Flow Diagrams, Class Diagrams cho toàn bộ modules
    - UI Wireframes: 30+ screens với high-fidelity mockups
    - Module Specifications: 6 spec files (m1-m6) với 200+ pages chi tiết
3. **Tài liệu kỹ thuật:**
    - Architecture Documentation (Clean Architecture, folder structure)
    - Agent Skill Documentation (SKILL.md cho mỗi skill)
    - Development Guidelines (coding standards, best practices)
    - Deployment Guide (Vercel, MongoDB Atlas setup)

0.6.3 Kiến thức và đóng góp học thuật

1. **Mô hình Agent Skill Framework:**
    - Đề xuất kiến trúc 3 Pillars + 7 Zones cho việc tổ chức tri thức AI Agent
    - Phương pháp Meta-Skill cho việc tự động hóa quá trình tạo Skills mới
    - Cơ chế Progressive Disclosure (Tier 1 mandatory, Tier 2 conditional loading)
    - Source Citation mechanism để chống hallucination
2. **Quy trình phát triển phần mềm tích hợp AI:**
    - 4-Life Lifecycle Model (Vision → Design → Implementation → Verification)
    - OpenSpec Workflow cho change management
    - Knowledge Factory pattern cho knowledge reuse
    - Interaction Points (IP-1, IP-2, IP-3) để cân bằng automation và human oversight
3. **Đánh giá hiệu quả:**
    - So sánh năng suất phát triển: thời gian tạo specs, diagrams, code giảm 40-60%
    - Chất lượng tài liệu thiết kế: đạt 85%+ compliance với standards
    - Mức độ hài lòng người dùng: khảo sát từ 20+ sinh viên/lập trình viên

0.6.4 Sản phẩm bổ trợ

- Source code hoàn chỉnh trên GitHub repository (MIT License)
- Video demo và hướng dẫn sử dụng (YouTube playlist)
- Bài báo khoa học về Agent Skill Framework (nếu có cơ hội)
- Workshop materials cho sinh viên về cách sử dụng AI trong phát triển phần mềm

0.7 Ý nghĩa khoa học và ý nghĩa thực tiễn của nghiên cứu

0.7.1 Ý nghĩa khoa học

1. **Đóng góp vào lĩnh vực AI Agent Research:**
    - Đề xuất mô hình *Agent Skill Framework* với kiến trúc 3 Pillars (Knowledge, Process, Guardrails) và 7 Zones, cung cấp một cách tiếp cận có hệ thống để tổ chức tri thức cho AI Agents
    - Phát triển *Meta-Skill Framework*—mô hình tự động hóa việc tạo Skills mới thông qua chu trình Architect → Planner → Builder, góp phần vào nghiên cứu về *self-improving AI systems*
    - Đề xuất cơ chế *Source Citation* và *Progressive Disclosure* để giảm thiểu hallucination và tối ưu hóa context window trong Large Language Models
2. **Ứng dụng AI vào Software Engineering Process:**
    - Nghiên cứu cách tích hợp AI vào từng giai đoạn của Software Development Life Cycle (SDLC), đặc biệt là giai đoạn Analysis & Design thường bị bỏ qua trong các công cụ AI code assistant hiện tại

- Chứng minh tính khả thi của việc sử dụng AI không chỉ cho code generation mà còn cho business analysis, system design, và documentation
- Đề xuất mô hình *4-Life Lifecycle* (Vision, Design, Implementation, Verification) với các Interaction Points để cân bằng giữa automation và human oversight

3. **Phương pháp luận mới cho AI-assisted Development:**

- Xây dựng *OpenSpec Workflow*—quy trình quản lý thay đổi (change management) dựa trên artifacts thay vì chỉ dựa vào code commits
- Đề xuất khái niệm *Knowledge Factory*—cơ chế tái sử dụng tri thức từ các lần thực thi trước đó của AI Agent, tương tự như caching nhưng ở mức semantic level
- Nghiên cứu cách kết hợp Agentic Workflow (Andrew Ng) với Software Engineering Best Practices

0.7.2 Ý nghĩa thực tiễn

1. **Đối với sinh viên và người mới học lập trình:**

- Rút ngắn thời gian từ ”không biết gì”đến ”có thể làm dự án thực tế”từ 2-3 năm xuống còn 6-12 tháng thông qua hướng dẫn có hệ thống của Agent Skills
- Cung cấp một ”chuyên gia ảo”24/7 giúp sinh viên học theo đúng quy trình chuẩn mực, tránh các bad practices phổ biến
- Tạo ra các tài liệu thiết kế chất lượng cao (diagrams, specs) mà sinh viên có thể tham khảo và học hỏi

- Giúp sinh viên hiểu rõ ”tại sao”(why) chứ không chỉ ”làm thế nào”(how) thông qua explanations trong Skills

2. **Đối với doanh nghiệp và đội ngũ phát triển:**

   - Tăng năng suất phát triển 40-60% thông qua automation các tác vụ lặp đi lặp lại (vẽ diagrams, viết specs, generate boilerplate code)
   - Chuẩn hóa quy trình làm việc: mọi developer trong team sử dụng chung một bộ Skills, đảm bảo consistency
   - Giảm technical debt: Agent Skills luôn đề xuất best practices và clean architecture patterns
   - Onboarding nhanh hơn cho nhân viên mới: thay vì đọc hàng trăm trang documentation, họ có thể hỏi Agent và nhận câu trả lời context-aware

3. **Đối với cộng đồng lập trình viên Việt Nam:**

   - Steve Void trở thành nền tảng chia sẻ kiến thức chất lượng cao, thay thế các diễn đàn thiếu tổ chức
   - Tạo ra một kho tri thức (Knowledge Base) bằng tiếng Việt về phát triển phần mềm với AI
   - Khuyến khích văn hóa documentation và knowledge sharing thông qua gamification (reputation, badges)
   - Kết nối sinh viên với doanh nghiệp thông qua các dự án mẫu và case studies thực tế

4. **Đối với Công ty Innotech:**

   - Tối ưu hóa quy trình phát triển sản phẩm, đặc biệt trong các dự án chuyển đổi số

- Xây dựng competitive advantage thông qua việc áp dụng AI sớm vào workflow
- Giảm chi phí đào tạo nhân viên mới thông qua hệ thống Agent Skills
- Tạo ra một framework có thể tái sử dụng cho các dự án khác của công ty

5. **Đối với ngành Công nghệ thông tin Việt Nam:**

- Đóng góp vào xu hướng AI-first Development đang diễn ra toàn cầu
- Nâng cao chất lượng đào tạo CNTT tại Việt Nam thông qua việc tích hợp công cụ AI vào giảng dạy
- Tạo tiền đề cho các nghiên cứu tiếp theo về AI in Education và AI-assisted Software Engineering
- Chứng minh rằng sinh viên Việt Nam có thể nghiên cứu và triển khai các công nghệ tiên tiến ngang tầm quốc tế

Tóm lại, đề tài không chỉ có giá trị về mặt học thuật (đóng góp vào nghiên cứu AI Agent) mà còn có tác động thực tiễn sâu rộng, từ cá nhân sinh viên, doanh nghiệp, đến toàn ngành CNTT Việt Nam.

0.8 Bố cục của báo cáo

Báo cáo khóa luận được tổ chức thành các phần chính như sau:

Phần mở đầu giới thiệu bối cảnh nghiên cứu, lý do chọn đề tài, mục tiêu, đối tượng, phạm vi, phương pháp nghiên cứu, và dự kiến kết quả đạt được.

Chương 1: Tổng quan về đề tài trình bày cơ sở lý thuyết về trí tuệ nhân tạo, AI Chatbot, quy trình phát triển phần mềm, các công nghệ sử dụng

(Next.js, Payload CMS, MongoDB), và khảo sát các hệ thống tương tự hiện có trên thế giới.

Chương 2: Phân tích và thiết kế hệ thống trình bày chi tiết về Agent Skill Framework, bao gồm:

- Kiến trúc 3 Pillars (Knowledge, Process, Guardrails) và 7 Zones (Core, Knowledge, Scripts, Templates, Data, Loop, Assets)
- Meta-Skill Framework với chu trình Architect → Planner → Builder
- Bộ Agent Skills chuyên biệt cho Life-2: Flow Design Analyst, Sequence Design Analyst, Activity Diagram Analyst, Class Diagram Analyst, UI Architecture Analyst
- Thiết kế nền tảng Steve Void: Database Schema, API Specification, UI Wireframes cho 6 modules (M1-M6)
- Các diagrams: ER Diagram, Use Case Diagram, Sequence Diagrams, Activity Diagrams, Class Diagrams

Chương 3: Triển khai hệ thống mô tả quá trình cài đặt và phát triển thực tế:

- Môi trường phát triển và công cụ sử dụng (Claude Code CLI, VS Code, Git)
- Triển khai Agent Skills: cấu trúc thư mục, SKILL.md, knowledge base, scripts, templates

Chương 4: Kiểm thử và đánh giá trình bày kết quả thực nghiệm:

- Kiểm thử chức năng: unit tests, integration tests, end-to-end tests
- Đánh giá hiệu năng: response time, throughput, resource usage

- Đánh giá chất lượng Agent Skills: accuracy của diagrams/specs, compliance với standards
- So sánh năng suất phát triển: có vs không có Agent Skills
- Khảo sát người dùng: mức độ hài lòng, ease of use, learning curve
- Phân tích hạn chế và đề xuất cải tiến

Phần kết luận tóm tắt những kết quả đạt được, đánh giá mức độ hoàn thành mục tiêu, rút ra bài học kinh nghiệm, và đề xuất hướng phát triển tiếp theo cho dự án.

Tài liệu tham khảo liệt kê các nguồn tài liệu, sách, bài báo, trang web được trích dẫn trong quá trình nghiên cứu.

Phụ lục bao gồm các tài liệu bổ trợ như:

- Source code quan trọng (Agent Skills, Payload Collections, React Components)
- Database Schema chi tiết
- API Documentation đầy đủ
- User Manual cho Steve Void
- Screenshots và video demo
- Bảng khảo sát và kết quả phân tích

CHƯƠNG 1

CƠ SỞ LÝ THUYẾT

Chương này trình bày cơ sở lý thuyết nền tảng cho toàn bộ nghiên cứu, bao gồm: tổng quan về trí tuệ nhân tạo và mô hình ngôn ngữ lớn, quy trình làm việc dựa trên Agent, các công nghệ sử dụng trong phát triển website, phương pháp mô hình hóa UML, và phân tích bài toán mạng xã hội chia sẻ kiến thức.

1.1 Tổng quan về trí tuệ nhân tạo và mô hình ngôn ngữ lớn

1.1.1 Khái niệm trí tuệ nhân tạo

Trí tuệ nhân tạo (Artificial Intelligence — AI) là lĩnh vực khoa học máy tính nghiên cứu và phát triển các hệ thống có khả năng thực hiện các tác vụ đòi hỏi trí thông minh của con người, bao gồm: nhận dạng ngôn ngữ tự nhiên, lập kế hoạch, suy luận logic và ra quyết định [7]. Trong thập kỷ gần đây, AI đã trải qua bước tiến vượt bậc nhờ sự phát triển của Deep Learning và đặc biệt là kiến trúc Transformer [8], tạo nền tảng cho các mô hình ngôn ngữ lớn (Large Language Models — LLMs).

1.1.2 Mô hình ngôn ngữ lớn (LLMs)

Mô hình ngôn ngữ lớn là các mạng nơ-ron được huấn luyện trên lượng dữ liệu văn bản khổng lồ, có khả năng hiểu và sinh ngôn ngữ tự nhiên ở mức độ gần với con người. Các đặc điểm chính của LLMs bao gồm:

- **Khả năng hiểu ngữ cảnh (Context-awareness):** LLMs có thể xử lý hàng chục nghìn tokens trong một lượt, cho phép hiểu được ngữ cảnh phức tạp của một dự án phần mềm

- **Khả năng sinh nội dung (Generation):** Từ prompt đầu vào, LLMs có thể sinh ra mã nguồn, tài liệu thiết kế, sơ đồ UML và các artifacts kỹ thuật khác
- **Khả năng suy luận (Reasoning):** Các mô hình thế hệ mới (GPT-4, Claude, Gemini) có khả năng suy luận logic nhiều bước, phù hợp cho các tác vụ phân tích và thiết kế hệ thống
- **Khả năng sử dụng công cụ (Tool Use):** LLMs có thể gọi các hàm, API và công cụ bên ngoài để thực thi hành động thực tế, không chỉ dừng ở mức tạo văn bản

Trong nghiên cứu này, mô hình Claude (Anthropic) được sử dụng làm nền tảng cho toàn bộ hệ thống Agent Skills, nhờ khả năng xử lý context window lớn (200K+ tokens) và hỗ trợ tool use nguyên bản [6].

1.1.3 AI Chatbot trong phát triển phần mềm

AI Chatbot trong dự án này không chỉ là giao diện tương tác văn bản, mà đóng vai trò là một **cộng tác viên lập trình** (AI Coding Assistant). Thông qua công cụ Claude Code CLI [9], AI có khả năng:

- Đọc và phân tích cấu trúc dự án (file system, git history)
- Thực thi lệnh terminal và chạy scripts
- Đọc/ghi file trực tiếp trên hệ thống
- Tương tác với các MCP servers (Model Context Protocol) để mở rộng khả năng

Sự khác biệt giữa AI Chatbot thông thường và AI Coding Assistant nằm ở **khả năng hành động** (agency) — không chỉ tư vấn mà còn trực tiếp thực thi các tác vụ kỹ thuật trong môi trường phát triển.

1.2 Quy trình làm việc dựa trên Agent (Agentic Workflow)

1.2.1 Định nghĩa Agentic Workflow

Theo Andrew Ng [1], *Agentic Workflow* là một quy trình mà AI Agent hoạt động lặp đi lặp lại qua chu kỳ: **Suy nghĩ (Thought) → Thực thi (Action) → Quan sát (Observation)**. Khác với cách tiếp cận zero-shot (cho prompt, nhận kết quả ngay), Agentic Workflow cho phép AI tự điều chỉnh và cải thiện output qua nhiều vòng lặp.

Bốn design pattern cốt lõi của Agentic Workflow [1]:

1. **Reflection:** AI tự đánh giá output của mình, phát hiện lỗi và tự sửa
2. **Tool Use:** AI sử dụng công cụ bên ngoài (search, code execution, file I/O) để thu thập thông tin và thực thi hành động
3. **Planning:** AI phân rã tác vụ phức tạp thành các bước nhỏ hơn, lập kế hoạch trước khi thực hiện
4. **Multi-agent Collaboration:** Nhiều AI agents với vai trò khác nhau cộng tác để giải quyết bài toán lớn

1.2.2 Ứng dụng Agentic Workflow vào phát triển phần mềm

Dự án này áp dụng cả 4 patterns trên vào quy trình thiết kế hệ thống:

- **Reflection**: Mỗi Agent Skill có cơ chế Self-Scoring và Self-Verification trước khi bàn giao output
- **Tool Use**: Agents sử dụng file I/O để đọc specs/schemas, Mermaid.js để vẽ sơ đồ, Pencil MCP để vẽ wireframe
- **Planning**: Skill Planner phân rã thiết kế thành các task có truy xuất nguồn gốc rõ ràng

- **Multi-agent**: Pipeline Architect → Planner → Builder, cùng với các Domain Skills (Flow, Sequence, Class, Schema, UI) phối hợp tuần tự

1.2.3 Khái niệm Agent Skill

Trong ngữ cảnh nghiên cứu này, **Agent Skill** là một đơn vị tri thức có cấu trúc, được thiết kế để hướng dẫn AI Agent thực hiện một tác vụ chuyên biệt với chất lượng có kiểm soát. Mỗi Skill bao gồm:

- **Persona**: Vai trò và chuyên môn mà AI đảm nhận (ví dụ: ”Senior System Architect”)
- **Workflow**: Quy trình thực thi có cấu trúc, với các bước rõ ràng
- **Guardrails**: Các hàng rào bảo vệ chống lại hallucination và sai sót
- **Knowledge Base**: Tập hợp kiến thức chuyên ngành cần thiết cho tác vụ

Khái niệm này mở rộng từ ”System Prompt”truyền thống, bổ sung thêm cấu trúc tri thức, cơ chế kiểm soát chất lượng và khả năng tương tác có điểm dừng (Interaction Points) với con người.

1.3 Các công nghệ nền tảng

1.3.1 Next.js 15 và React 19

Next.js là framework React phổ biến nhất cho phát triển web hiện đại, cung cấp các tính năng quan trọng [2]:

- **App Router**: Hệ thống routing mới dựa trên file system, hỗ trợ layouts lồng nhau và loading states

- **React Server Components (RSC)**: Cho phép render component phía server, giảm JavaScript bundle gửi về client
- **Server Actions**: Gọi hàm server trực tiếp từ client component, đơn giản hóa API layer
- **SEO Optimization**: Metadata API và Server-side Rendering đảm bảo nội dung được index tốt bởi công cụ tìm kiếm

Đối với hệ thống mạng xã hội chia sẻ kiến thức, Next.js đặc biệt phù hợp nhờ khả năng kết hợp SSR (cho trang công khai, tối ưu SEO) với CSR (cho trang tương tác, tối ưu trải nghiệm).

1.3.2 Payload CMS 3.x

Payload CMS là hệ quản trị nội dung mã nguồn mở (Headless CMS) thế hệ mới, nổi bật với triết lý **Code-first** — toàn bộ cấu trúc dữ liệu được định nghĩa bằng TypeScript thay vì giao diện kéo thả [3]. Các đặc điểm quan trọng:

- **Local API**: Truy cập dữ liệu trực tiếp qua hàm TypeScript (`payload.find()`, `payload.create()`) mà không cần qua HTTP, giảm latency đáng kể
- **Access Control**: Hệ thống phân quyền linh hoạt, định nghĩa bằng hàm JavaScript cho mỗi collection
- **Hooks System**: Lifecycle hooks (beforeChange, afterChange, beforeDelete...) cho phép can thiệp vào mọi thao tác CRUD
- **Tích hợp Next.js**: Payload 3.x chạy trực tiếp bên trong ứng dụng Next.js, chia sẻ chung codebase

Triết lý Code-first của Payload CMS đặc biệt phù hợp với AI Agent — toàn bộ cấu trúc dữ liệu có thể được phân tích và sinh tự động bởi AI từ tài liệu thiết kế.

1.3.3 MongoDB Atlas

MongoDB là cơ sở dữ liệu NoSQL document-oriented, lưu trữ dữ liệu dưới dạng BSON (Binary JSON) [4]. MongoDB Atlas là dịch vụ đám mây của MongoDB, cung cấp:

- **Flexible Schema**: Cấu trúc document linh hoạt, phù hợp với dữ liệu đa dạng của mạng xã hội (bài viết, bình luận, thông báo...)
- **Atlas Search**: Tích hợp full-text search và autocomplete trực tiếp, không cần Elasticsearch riêng
- **Aggregation Pipeline**: Xử lý dữ liệu phức tạp (ranking, thống kê, báo cáo) ngay tại tầng database
- **Change Streams**: Theo dõi thay đổi dữ liệu real-time, hỗ trợ tính năng thông báo SSE

Quyết định chọn MongoDB thay vì SQL databases dựa trên phân tích: cấu trúc dữ liệu mạng xã hội thường xuyên thay đổi, cần lưu trữ nested documents (comments trong posts, media trong posts), và cần horizontal scaling khi lượng người dùng tăng.

1.3.4 Tailwind CSS v4 và Radix UI

Giao diện người dùng được xây dựng trên bộ đôi **Tailwind CSS v4** (styling) và **Radix UI** (primitives) [5]:

- **Tailwind CSS v4**: Thư viện CSS Utility-first, cho phép xây dựng giao diện nhanh chóng mà không cần viết CSS tùy chỉnh. Version 4 cải tiến đáng kể với CSS-native engine và hỗ trợ CSS variables
- **Radix UI**: Bộ UI primitives không kèm styling, cung cấp các component có sẵn accessibility (ARIA attributes, keyboard navigation, screen reader support). Kết hợp với Tailwind CSS cho phép kiểm soát hoàn toàn visual design

Quy tắc thiết kế theo phong cách **Neobrutalism**: đường viền đậm (bold borders), bóng đổ lệch (offset shadows), tông màu tương phản cao với primary color là Pink.

1.4 Phương pháp mô hình hóa UML

1.4.1 Tổng quan về UML

Unified Modeling Language (UML) là ngôn ngữ mô hình hóa chuẩn được sử dụng rộng rãi trong phân tích và thiết kế hệ thống phần mềm [10]. Nghiên cứu này sử dụng 5 loại sơ đồ UML chính:

Bảng 1.1: Các loại sơ đồ UML sử dụng trong nghiên cứu

Loại sơ đồ	Phân loại	Mục đích
Use Case Diagram	Structure	Xác định actors và chức năng hệ thống
Activity Diagram	Behavior	Mô tả luồng nghiệp vụ chi tiết
Sequence Diagram	Interaction	Mô tả tương tác giữa các đối tượng theo thời gian
Class Diagram	Structure	Định nghĩa cấu trúc dữ liệu và quan hệ
ER Diagram	Structure	Mô tả quan hệ giữa các entities trong database

1.4.2 Công cụ vẽ sơ đồ: Mermaid.js

Thay vì sử dụng các công cụ đồ họa truyền thống (Visio, draw.io), nghiên cứu sử dụng **Mermaid.js** — thư viện JavaScript cho phép vẽ sơ đồ bằng cú pháp text-based [11]. Ưu điểm chính:

- **Code-as-Diagram**: Sơ đồ được lưu dưới dạng text (Markdown), dễ dàng version control bằng Git
- **AI-friendly**: AI Agent có thể đọc, phân tích và sinh sơ đồ Mermaid trực tiếp, không cần GUI
- **Đa dạng loại sơ đồ**: Hỗ trợ flowchart, sequence, class, ER, gantt, pie chart...
- **Tích hợp rộng**: Render trực tiếp trong GitHub, GitLab, Notion, VS Code

1.5 Quy trình phát triển phần mềm và mô hình 4-Life

1.5.1 Mô hình phát triển truyền thống

Quy trình phát triển phần mềm (Software Development Life Cycle — SDLC) truyền thống bao gồm các giai đoạn: Phân tích yêu cầu → Thiết kế → Triển khai → Kiểm thử → Bảo trì. Các mô hình phổ biến bao gồm Waterfall (tuần tự), Agile (lặp lại) và DevOps (tích hợp liên tục) [12].

1.5.2 Mô hình 4-Life cho phát triển tích hợp AI

Nghiên cứu đề xuất mô hình **4-Life Lifecycle** — một biến thể của SDLC được thiết kế đặc biệt cho phát triển phần mềm tích hợp AI Agent:

Bảng 1.2: Mô hình 4-Life Lifecycle

Phase	**Tên giai đoạn**	**Mô tả**
Life-1	Vision & Research	Xác định tầm nhìn, personas, user stories, quyết định kỹ thuật
Life-2	Design & Specification	Vẽ sơ đồ UML, thiết kế database, API spec, UI wireframes
Life-3	Implementation	Sinh mã nguồn từ specifications, triển khai hệ thống
Life-4	Verification	Kiểm thử, đánh giá, release

Điểm khác biệt so với SDLC truyền thống: mỗi giai đoạn có **Phase Gate** bắt buộc — một checklist các artifacts phải hoàn thành trước khi chuyển sang giai đoạn tiếp theo. AI Agent tham gia chủ yếu vào Life-2 (phân tích, thiết kế) và Life-3 (sinh mã), trong khi Life-1 (tầm nhìn) và Life-4 (đánh giá) cần sự chủ đạo của con người.

1.6 Phân tích bài toán mạng xã hội chia sẻ kiến thức

1.6.1 Đặc thù của mạng xã hội chia sẻ kiến thức

Mạng xã hội chia sẻ kiến thức (Social Knowledge Sharing Network) không chỉ đơn thuần là nơi kết nối con người mà còn là một ”nhà máy tri thức”(Knowledge Factory), nơi thông tin được tạo ra, sàng lọc và tái sử dụng. Qua quá trình khảo sát và phân tích bối cảnh thực tại, bài toán này đối mặt với các thách thức sau:

1. **Tổ chức và tìm kiếm tri thức:** Nội dung trên các mạng xã hội truyền thống thường bị trôi nhanh và thiếu phân loại. Hệ thống cần giải quyết bài toán *Social Bookmarking*, cho phép người dùng không chỉ lưu trữ mà còn tổ chức tri thức thành các bộ sưu tập (Collections) có hệ thống

2. **Xếp hạng nội dung chất lượng:** Để tránh nhiễu thông tin (Information Overload), hệ thống áp dụng thuật toán *Time-decay + Engagement Score* — kết hợp yếu tố thời gian (tính cập nhật) và mức độ tương tác (likes, comments, shares) để ưu tiên nội dung chất lượng

3. **Tương tác thời gian thực:** Tìm kiếm ngữ nghĩa bằng MongoDB Atlas Search và thông báo real-time qua Server-Sent Events (SSE) thúc đẩy quá trình trao đổi kiến thức liên tục

4. **Đặc thù cộng đồng Tech Việt Nam:** Các nền tảng chia sẻ kiến thức tech chuyên sâu cho người Việt còn tản mát. Việc xây dựng hệ thống tập trung giúp hình thành văn hóa Documentation và Sharing lành mạnh

1.6.2 Khảo sát các hệ thống tương tự

Bảng 1.3: So sánh các nền tảng chia sẻ kiến thức hiện có

Tiêu chí	**Stack Overflow**	**Dev.to**	**Viblo**	**Steve Void**
Mô hình	Q&A	Blog/Social	Blog	Knowledge Social
Ranking	Vote-based	Reaction	View-based	Time-decay + Engagement
Bookmarking	Favorites	Reading list	Bookmark đơn	Collections có phân loại
Real-time	Không	Limited	Không	SSE notifications
AI-assisted	Không	Không	Không	Agent Skills pipeline
Tiếng Việt	Hạn chế	Không	Có	Có (native)

Steve Void khác biệt ở hai điểm chính: (1) quy trình phát triển được hỗ trợ bởi bộ Agent Skills tự động hóa, và (2) tích hợp hệ thống bookmark có tổ chức (Collections) thay vì chỉ lưu đơn thuần.

Việc kết hợp AI vào quy trình phân tích và thiết kế giúp chuyển đổi các yêu cầu nghiệp vụ phức tạp thành các Agent Skills chuyên biệt, đảm bảo hệ thống được xây dựng trên nền tảng kiến trúc vững chắc.

CHƯƠNG 2

PHÂN TÍCH VÀ THIẾT KẾ HỆ THỐNG AGENT SKILLS

Trong chương này, báo cáo trình bày chi tiết về kiến trúc tổng thể của hệ thống Agent Skills, cũng như phân tích từng bộ kỹ năng chuyên biệt được thiết kế để tự động hóa quá trình phân tích và thiết kế hệ thống trong giai đoạn Life-2.

2.1 Kiến trúc tổng thể hệ thống Agent Skills

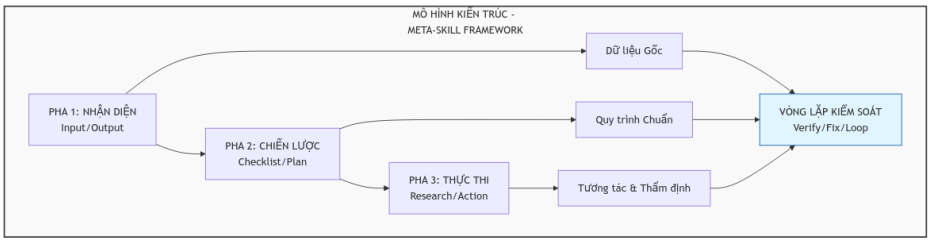
Hệ thống Agent Skills được xây dựng dựa trên mô hình **Meta-Skill Framework**, bao gồm ba trụ cột chính (3 Pillars) và bảy vùng chức năng (7 Zones). Kiến trúc này đảm bảo tính nhất quán, khả năng kiểm soát chất lượng và khả năng mở rộng của các bộ Agent Skill.

2.1.1 Mô hình Meta-Skill Framework

Meta-Skill Framework định nghĩa cách tiếp cận tổng thể để xây dựng một Agent Skill có cấu trúc, logic rõ ràng và khả năng tự kiểm soát chất lượng. Mô hình này bao gồm ba giai đoạn chính:

1. **Pha 1 — Nhận diện (Input/Output)**: Xác định rõ ràng đầu vào và đầu ra mong muốn của skill.
2. **Pha 2 — Chiến lược (Checklist/Plan)**: Lập kế hoạch thực thi với checklist kiểm soát chất lượng.
3. **Pha 3 — Thực thi (Research/Action)**: Nghiên cứu và thực hiện các bước được định nghĩa.

Ba giai đoạn này được kết nối với vòng lặp kiểm soát (Verify/Fix/Loop) để đảm bảo chất lượng đầu ra. Mô hình được minh họa trong Hình 2.1.



Hình 2.1: Mô hình Meta-Skill Framework

2.1.2 Ba trụ cột (3 Pillars)

Kiến trúc Agent Skill được xây dựng trên ba trụ cột chính:

Pillar 1 — Knowledge (Tri thức)

Tập hợp các quy định, tiêu chuẩn kỹ thuật (UML, Schema Design Patterns, Best Practices) cung cấp ”Context”cho AI Agent. Tri thức được tổ chức trong thư mục `knowledge/` và được nạp theo chiến lược Progressive Disclosure (nạp dần theo nhu cầu).

Pillar 2 — Process (Quy trình)

Các bước thực thi được module hóa thành workflow rõ ràng, từ nhận diện đầu vào đến kiểm chứng đầu ra. Mỗi skill định nghĩa riêng quy trình phù hợp với domain của nó.

Pillar 3 — Guardrails (Kiểm soát)

Các hàng rào bảo vệ chống lại hiện tượng ”hallucination”(ảo giác) của AI thông qua các cơ chế: Interaction Gates, Source Citation, Self-Scoring, và Checklist Verification.

2.1.3 Quy trình 5 bước xây dựng Skill

Quy trình xây dựng một Agent Skill tuân theo 5 bước chuẩn được minh họa trong Hình 2.2:

1. **Khảo sát (Research)**: Xác định Pain Point, Input/Output, Tools cần dùng
2. **Thiết kế (Design)**: Xây dựng workflow, Interaction Points, Output format
3. **Xây dựng (Build)**: Viết SKILL.md, tạo templates, scripts, knowledge files
4. **Kiểm định (Verify)**: Chạy Test Cases, Verify Checklist, Rollback nếu fail
5. **Bảo trì (Maintenance)**: Feedback Loop, Version Control, cập nhật khi môi trường thay đổi

Hình 2.2: Quy trình 5 bước xây dựng Agent Skill

2.2 Bộ kỹ năng Meta-Skills cho xây dựng Agent

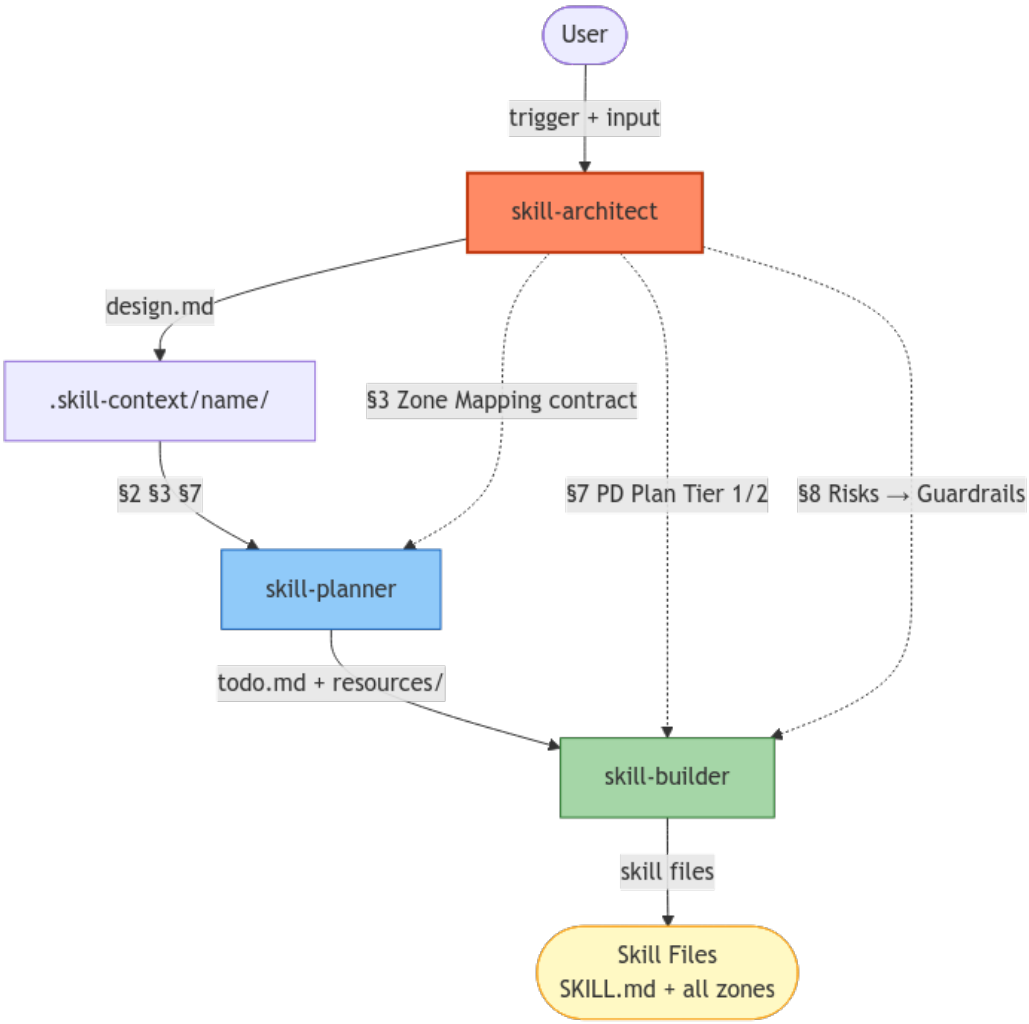
Hệ thống Agent Skills được xây dựng thông qua một bộ ba meta-skills chủ chốt: **Skill Architect**, **Skill Planner**, và **Skill Builder**. Ba skills này hoạt động theo pipeline tuần tự, tạo thành một quy trình tự động hóa hoàn chỉnh để xây dựng các Agent Skills mới.

2.2.1 Skill Architect — Kiến trúc sư thiết kế Skills

Skill Architect là meta-skill trung tâm, chịu trách nhiệm thiết kế cấu trúc cho các Agent Skills khác. Đây là điểm khởi đầu của toàn bộ Skill Suite trong pipeline `Architect → Planner → Builder`.

Vai trò và vị trí trong pipeline

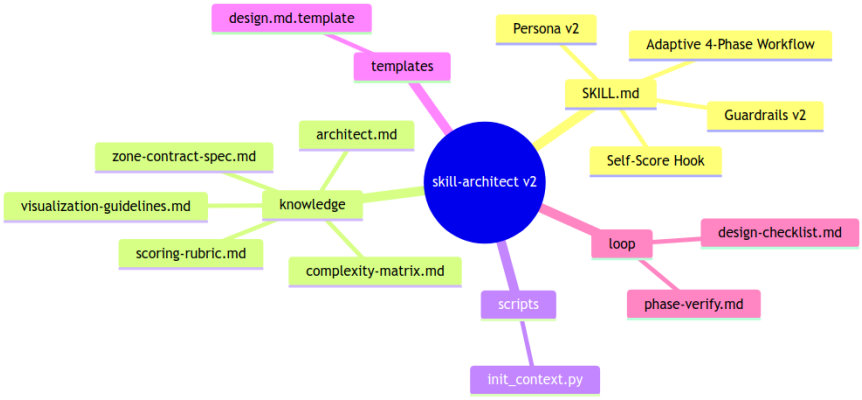
Skill Architect đóng vai trò như một ”kiến trúc sư trưởng”, nhận yêu cầu từ người dùng và tạo ra bản thiết kế chi tiết (`design.md`) cho skill mới. Vị trí của nó trong pipeline được minh họa trong Hình 2.3.



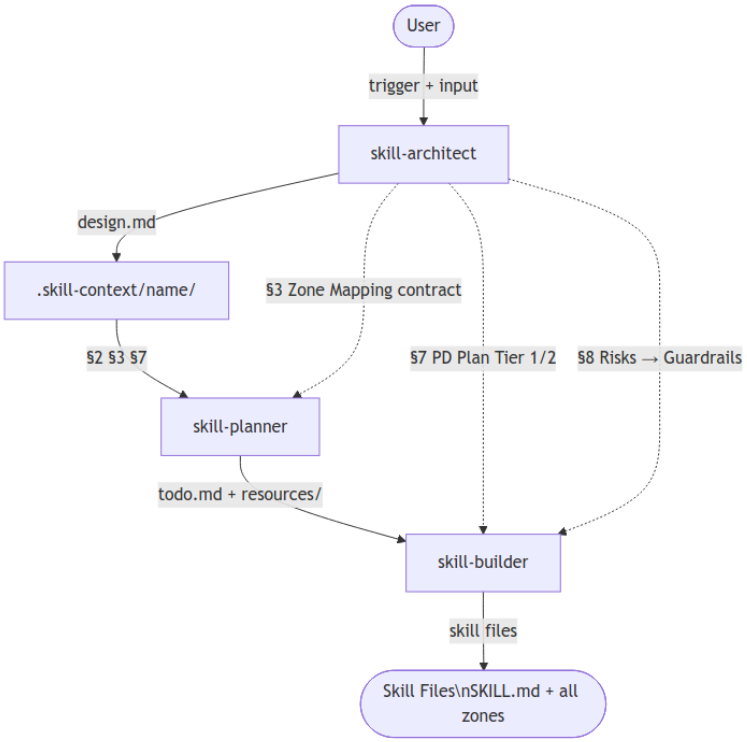
Hình 2.3: Pipeline Skill Suite: Architect → Planner → Builder

Cấu trúc thư mục và Skill Suite Pipeline

Cấu trúc thư mục của Skill Architect v2 bao gồm SKILL.md, 5 knowledge files, scripts, templates và loop. Hình 2.4 minh họa cấu trúc này. Vị trí của Skill Architect trong Skill Suite Pipeline (Architect → Planner → Builder) được thể hiện trong Hình 2.5.



Hình 2.4: Cấu trúc thư mục của Skill Architect v2



Hình 2.5: Pipeline Skill Suite: Skill Architect → Planner → Builder

Kiến trúc 7 Zones

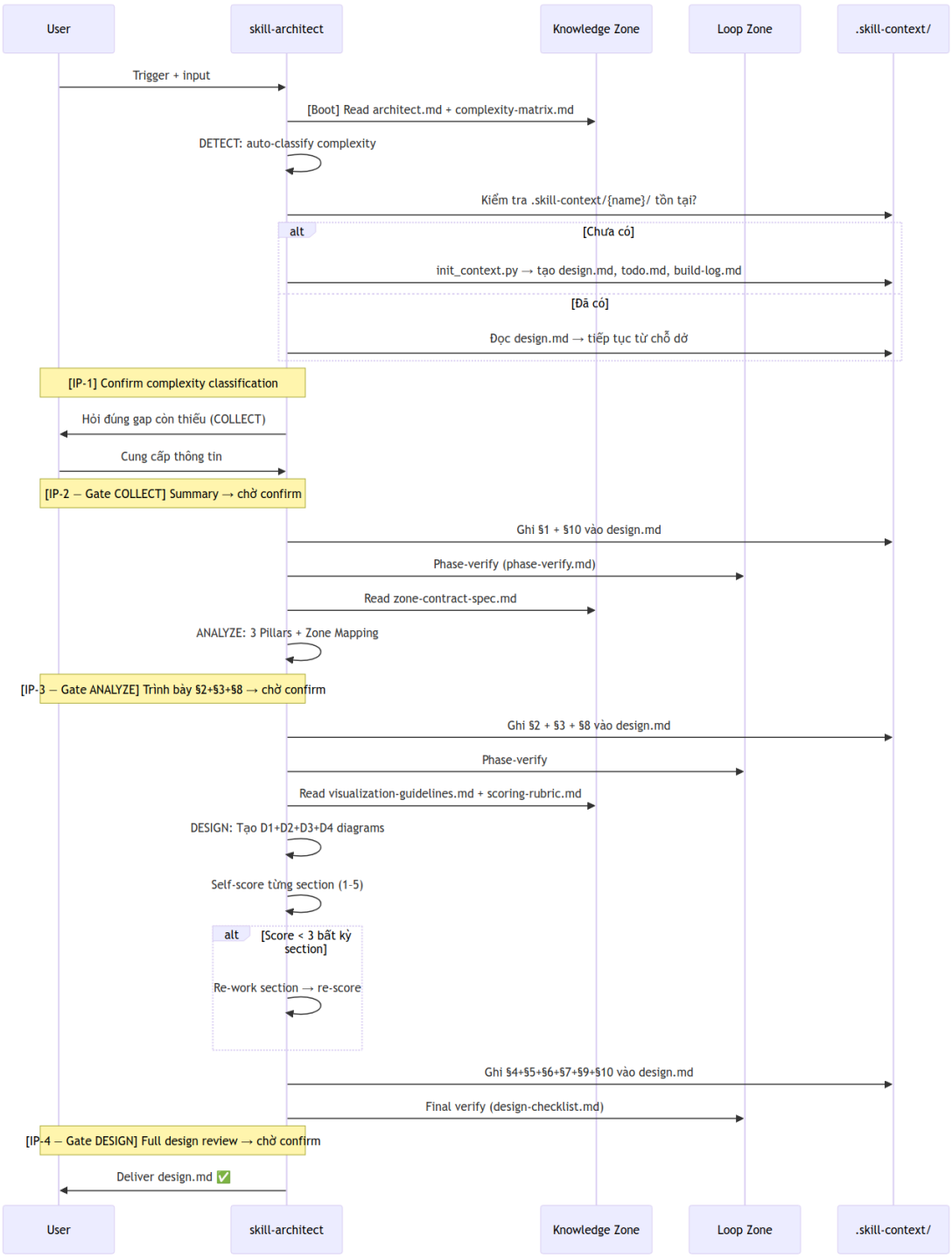
Mọi Agent Skill được tổ chức theo cấu trúc 7 Zones chuẩn, mỗi zone phục vụ một mục đích riêng biệt:

Bảng 2.1: 7 Zones trong cấu trúc Agent Skill

Zone	Nội dung	Bắt buộc?
Core	`SKILL.md` — Persona, workflow, guardrails	Có
Knowledge	Các file `knowledge/*.md` — Standards, best practices, domain knowledge	Có
Scripts	Automation scripts (Python/Bash) — Validation, linting, data processing	Tùy chọn
Templates	Output templates (Mermaid, YAML, code stubs)	Tùy chọn
Data	Static config, registries, sample data	Không bắt buộc
Loop	Quality control checklists, verification scripts	Có
Assets	Images, diagrams, media files	Không bắt buộc

Quy trình Adaptive Workflow và luồng Runtime

Luồng runtime của Skill Architect đi qua 4 interaction points bắt buộc (IP-1: xác nhận hiểu yêu cầu; IP-2: duyệt Zone Mapping; IP-3: duyệt design.md nháp; IP-4: xác nhận giao hàng). AI không được phép bỏ qua bất kỳ gate nào. Hình 2.6 mô tả chi tiết các tương tác giữa User, Skill Architect, Knowledge Zone và thư mục output.



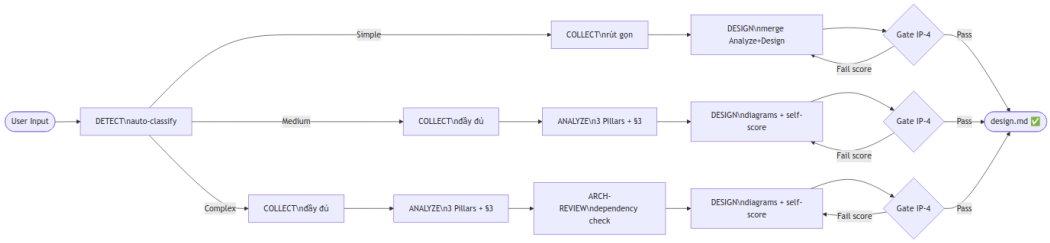
Hình 2.6: Luồng thực thi Runtime (Sequence Diagram) của Skill Architect

Skill Architect v2 giới thiệu cơ chế Adaptive Workflow — tự động phân loại độ phức tạp của yêu cầu và chọn workflow tối ưu:

- **Simple**: COLLECT (rút gọn) → DESIGN (merge Analyze+Design)
- **Medium**: COLLECT → ANALYZE → DESIGN

- **Complex**: COLLECT → ANALYZE → ARCH-REVIEW → DESIGN

Complexity detection dựa trên: số Zones cần dùng, số file cần tạo, mức độ phụ thuộc với skills khác, và độ mới của domain. Hình 2.7 thể hiện 3 nhánh này với Gate IP-4 bắt buộc trước khi bàn giao.



Hình 2.7: Adaptive Workflow của Skill Architect: Simple/Medium/Complex paths

Cơ chế Self-Scoring

Để đảm bảo chất lượng thiết kế, Skill Architect tích hợp cơ chế Self-Scoring — AI tự đánh giá từng section của `design.md` theo thang điểm 1-5. Các tiêu chí đánh giá bao gồm: độ rõ ràng (Clarity), tính đầy đủ (Completeness), tính khả thi (Feasibility), và tuân thủ chuẩn (Standards Compliance).

Quy tắc: Nếu bất kỳ section nào có điểm dưới 3/5, AI phải tự động re-work section đó trước khi chuyển giao cho Planner.

Guardrails chống Hallucination

Skill Architect áp dụng 5 guardrails cứng:

1. **Zone Mapping Contract**: §3 trong `design.md` PHẢI có tên file cụ thể cho mọi Zone, tuân thủ regex `[a-z][a-z0-9_]+\.[a-z]+`
2. **Confidence Threshold**: Nếu confidence < 70% về bất kỳ thông tin nào → BLOCK → Hỏi user → Không tiếp tục
3. **Interaction Gates**: 4 gates bắt buộc (IP-1 đến IP-4) — AI không được bỏ qua

4. **Progressive Disclosure Tiering**: Phân biệt rõ Tier 1 (mandatory) vs Tier 2 (conditional loading)

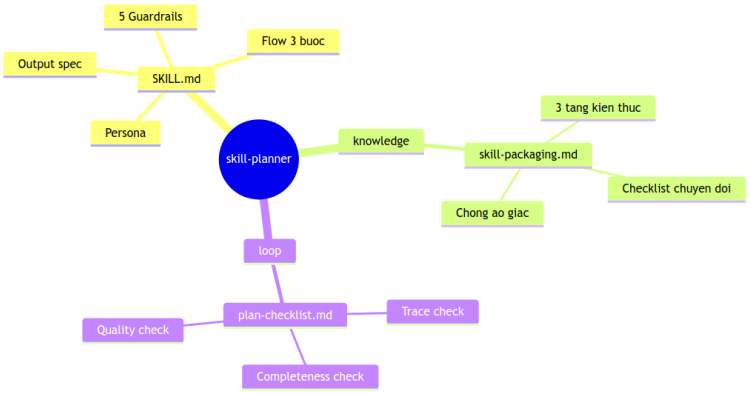
5. **Validation Before Delivery**: Regex validation cho §3, checklist verification từ `loop/design-checklist.md`

2.2.2 Skill Planner — Lập kế hoạch triển khai

Skill Planner là skill #2 trong bộ Master Skill Suite, nhận đầu vào là `design.md` từ Architect và tạo ra `todo.md` — kế hoạch triển khai chi tiết. Planner không tự mình thêm yêu cầu, chỉ phân rã thiết kế thành các task có truy xuất nguồn gốc rõ ràng.

Cấu trúc thư mục

Skill Planner có cấu trúc tối giản — chỉ cần Core (SKILL.md), 1 knowledge file (`skill-packaging.md`) và 1 loop file (`plan-checklist.md`). Hình 2.8 minh họa cấu trúc này.



Hình 2.8: Cấu trúc thư mục tối giản của Skill Planner

Mô hình 3 tầng kiến thức

Planner phân tích mọi Zone trong thiết kế theo 3 tầng:

1. **Tầng 1 — Domain Knowledge**: Kiến thức miền — hiểu bản chất thứ cần làm (UML standards, MongoDB patterns, Business logic)

2. **Tầng 2 — Technical Knowledge**: Kỹ thuật triển khai — công cụ, cú pháp (Mermaid syntax, Python scripting, Regex)

3. **Tầng 3 — Packaging Knowledge**: Đóng gói — map human skill vào 7 Zones (SKILL.md structure, template formatting)

Với mỗi tầng, Planner liệt kê: kiến thức nào cần chuẩn bị, công cụ nào cần, task nào cần thực hiện. Mọi task PHẢI trace về một section cụ thể trong `design.md`.

Quy trình Read → Analyze → Write

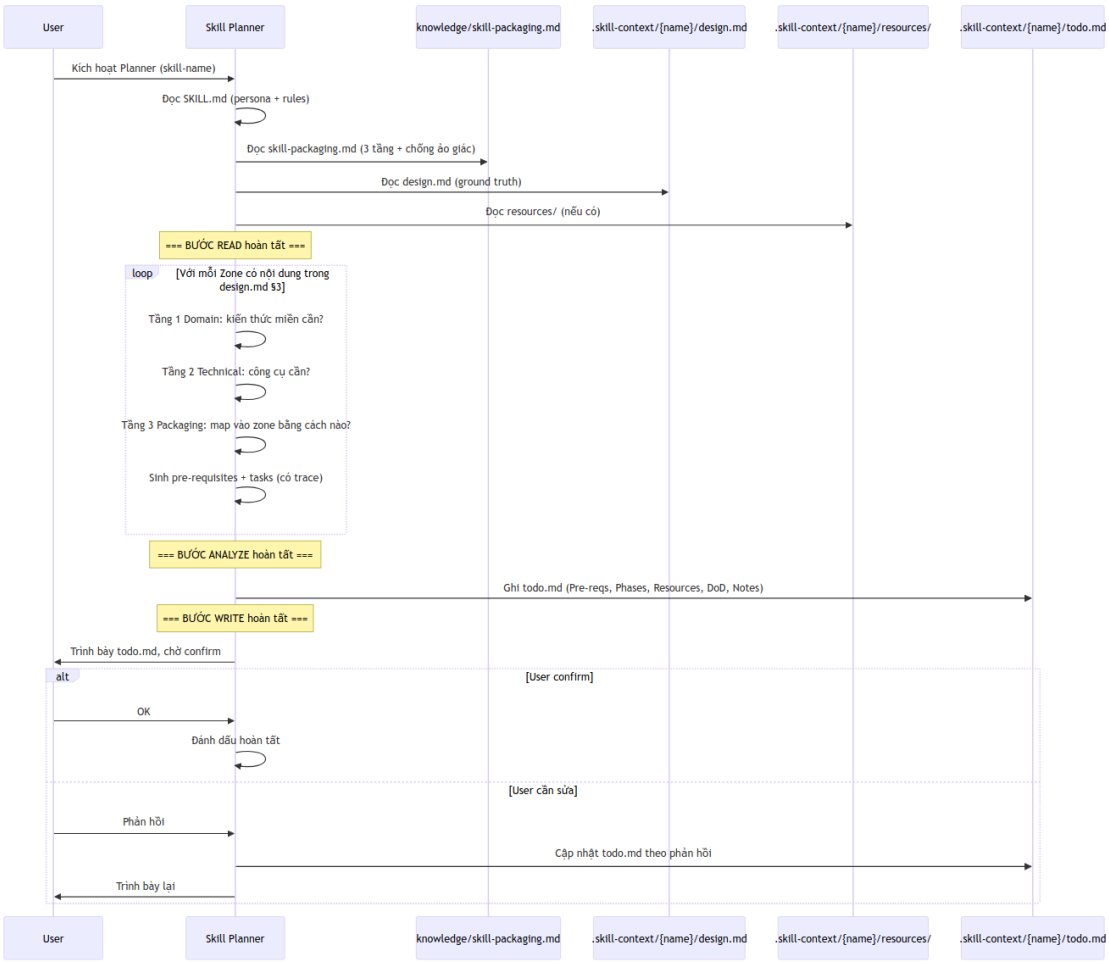
Planner hoạt động theo flow 3 bước nội bộ, 1 interaction point cuối:

1. **READ**: Đọc `design.md`, `resources/`, context từ user

2. **ANALYZE**: Với mỗi Zone, phân tích 3 tầng → sinh pre-requisites + tasks (có trace)

3. **WRITE**: Ghi `todo.md` với format: Pre-reqs, Phases, Resources, Definition of Done, Notes

Interaction point duy nhất: Sau khi ghi `todo.md`, trình bày cho user xem và chờ confirm. Hình 2.9 thể hiện toàn bộ luồng tương tác này giữa User, Planner, knowledge files, design.md và todo.md.



Hình 2.9: Luồng thực thi (Sequence Diagram) của Skill Planner: READ → ANALYZE → WRITE

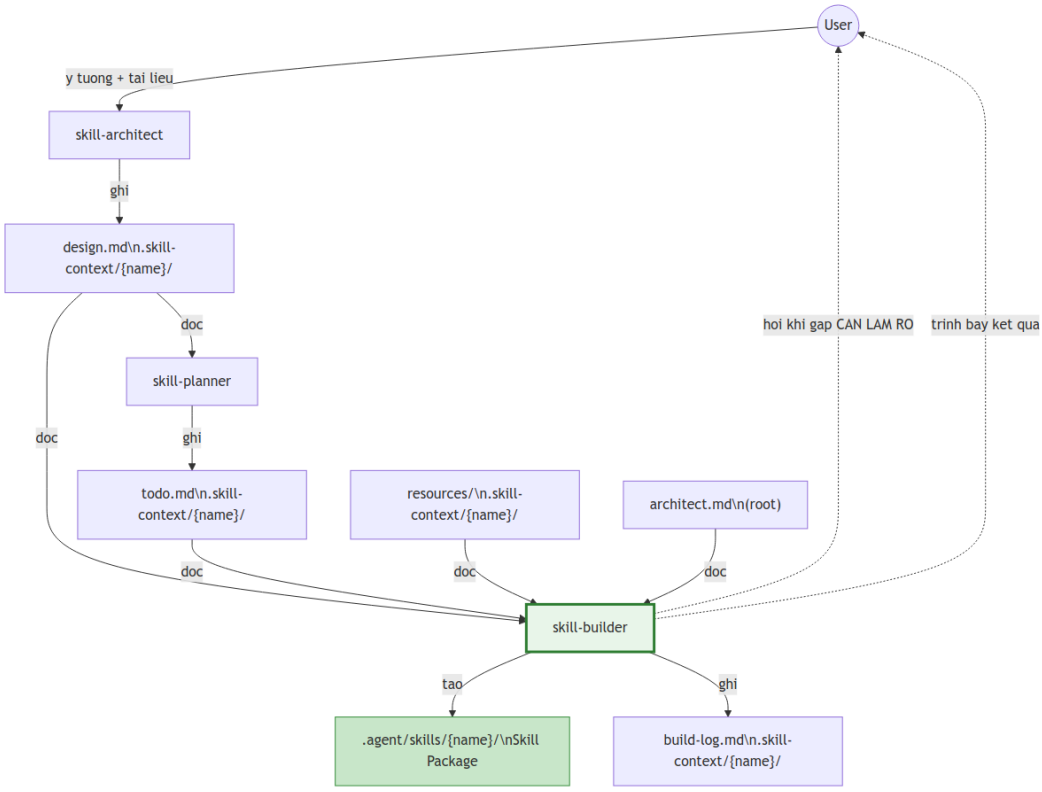
Guardrails chống bịa đặt

- **G1 — Trace bắt buộc**: Mọi item trong `todo.md` PHẢI trace về `design.md §N`
- **G2 — Phân biệt nguồn**: Đánh dấu rõ `[TỪ DESIGN]` vs `[GỢI Ý BỔ SUNG]`
- **G3 — Không phát minh**: Chỉ phân rã thiết kế, không thêm requirements mới
- **G4 — Liệt kê, không tự làm**: Liệt kê kiến thức cần → user chuẩn bị. Không tự search web hoặc hallucinate

- **G5 — Neo vào design.md**: Ground truth duy nhất. Thiếu → ghi Notes, không đoán

2.2.3 Skill Builder — Kỹ sư triển khai

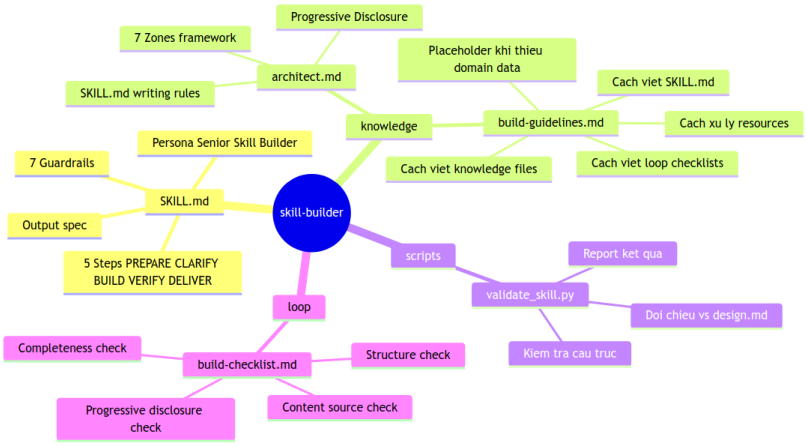
Skill Builder là skill #3 (cuối cùng) trong pipeline Master Skill Suite, nhận `design.md + todo.md` và tạo ra skill package hoàn chỉnh tại `.agent/skills/{skill-name}/`. Hình 2.10 thể hiện vị trí của Builder trong toàn bộ pipeline.



Hình 2.10: Pipeline đầy đủ: User → Architect → Planner → Builder → Skill Package

Cấu trúc thư mục

Skill Builder sử dụng cấu trúc tối giản: Core (SKILL.md), 2 knowledge files và loop. Hình 2.11 minh họa cấu trúc này.



Hình 2.11: Cấu trúc thư mục của Skill Builder

Engineer Stance — Quyền phản biện thiết kế

Khác với Planner (chỉ lập kế hoạch), Builder có quyền **thẩm định logic thiết kế**. Nếu phát hiện thiết kế phi thực tế hoặc mâu thuẫn, Builder phải:

1. Ghi nhận vấn đề vào `build-log.md`
2. Hỏi user để làm rõ
3. Cập nhật `design.md` §9 (Open Questions) nếu cần
4. Chỉ tiếp tục sau khi vấn đề được giải quyết

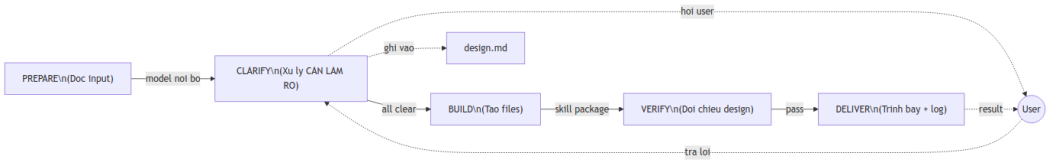
Phase-driven Build Strategy

Để tránh context overload, Builder chia nhỏ bước BUILD theo từng Phase trong `todo.md`. Mark-as-done từng phase ngay sau khi hoàn thành.

Quy trình 5 bước, minh họa trong Hình 2.12:

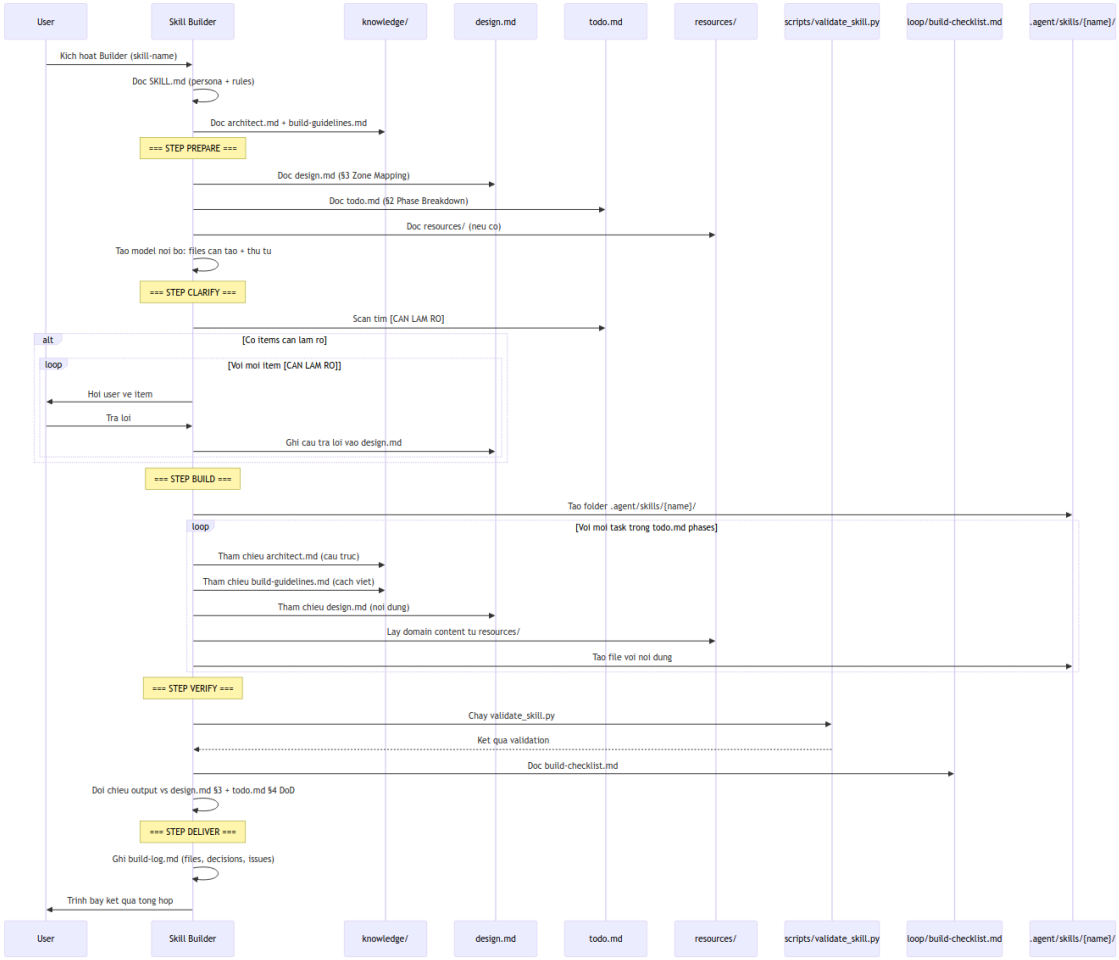
1. **PREPARE**: Đọc input + Thẩm định logic thiết kế
2. **CLARIFY**: Xử lý `[CẦN LÀM RÕ]` + Phản biện
3. **BUILD**: Triển khai từng Phase, tạo files
4. **VERIFY**: Chạy `validate_skill.py`, check `build-checklist.md`

5. **DELIVER**: Ghi `build-log.md`, trình bày kết quả



Hình 2.12: Quy trình 5 bước PREPARE → CLARIFY → BUILD → VERIFY → DELIVER của Skill Builder

Hình 2.13 thể hiện chi tiết luồng tương tác giữa 9 thành phần: User, Builder, knowledge, design.md, todo.md, resources, scripts, loop, .agent/skills.



Hình 2.13: Luồng thực thi chi tiết (Sequence Diagram) của Skill Builder

Log-Notify-Stop Pattern

Khi gặp lỗi hệ thống (permission denied, disk full, file corruption), Builder áp dụng pattern:

1. **LOG**: Ghi chi tiết lỗi vào `build-log.md`
2. **NOTIFY**: Báo user ngay lập tức
3. **STOP**: Dừng ngay, KHÔNG cố chạy tiếp

Tuyệt đối không silent fail hoặc skip lỗi.

Placeholder Gate Mechanism

Builder theo dõi số lượng placeholder (nội dung tạm thời cần user điền sau):

- **Mức 0-4**: OK — domain knowledge đã đầy đủ
- **Mức 5-9**: WARNING — cảnh báo, yêu cầu user cung cấp thêm resources
- **Mức 10+**: FAIL — quá nhiều placeholder, thiết kế thiếu ground truth

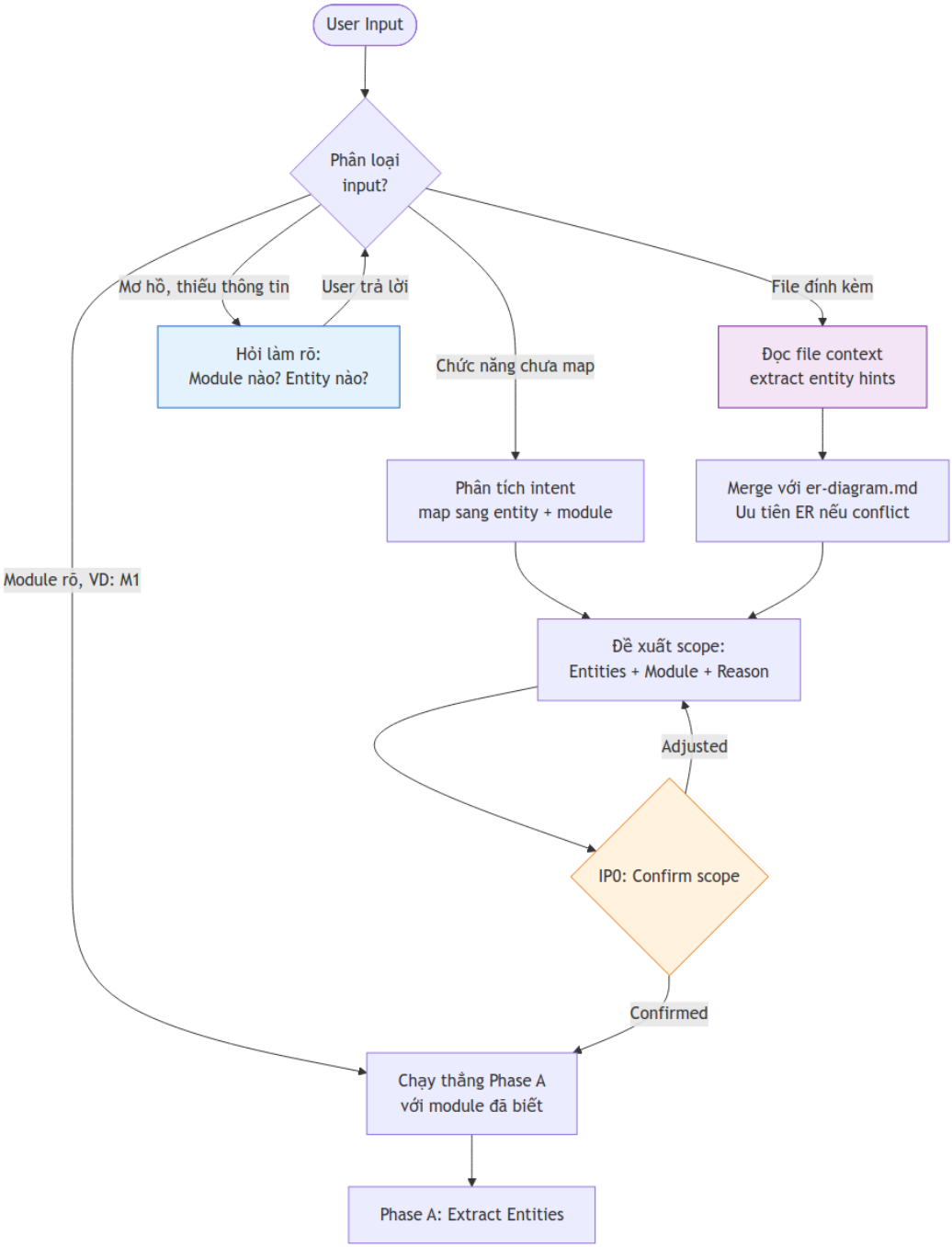
Thang đo này ngăn chặn việc tạo ra skills rỗng nội dung.

2.3 Bộ kỹ năng thiết kế cấu trúc dữ liệu (Class Diagram Analyst)

Class Diagram Analyst đảm nhiệm việc chuyển đổi từ ER Diagram và quy trình nghiệp vụ sang Class Diagram với định dạng dual-format (Mermaid + YAML Contract). Đây là skill thứ 4 trong chuỗi Life-2, cung cấp Contract YAML làm đầu vào tuyệt đối cho Schema Design Analyst.

2.3.1 Cơ chế xử lý đầu vào (Input Resolution)

Skill hỗ trợ 4 loại đầu vào khác nhau: module rõ ràng, chức năng chưa rõ, yêu cầu mơ hồ, và yêu cầu kèm file đính kèm. Hình 2.14 minh họa cách skill phân loại và định tuyến từng loại tới workflow phù hợp.



Hình 2.14: Input Resolution Flow của Class Diagram Analyst

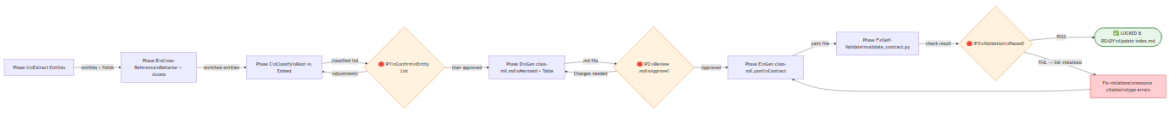
2.3.2 *Cấu trúc thư mục*

Hình 2.15 thể hiện cấu trúc thư mục đầy đủ của Class Diagram Analyst với 6 zones, bao gồm Scripts để validate và Data cho entity inventory.

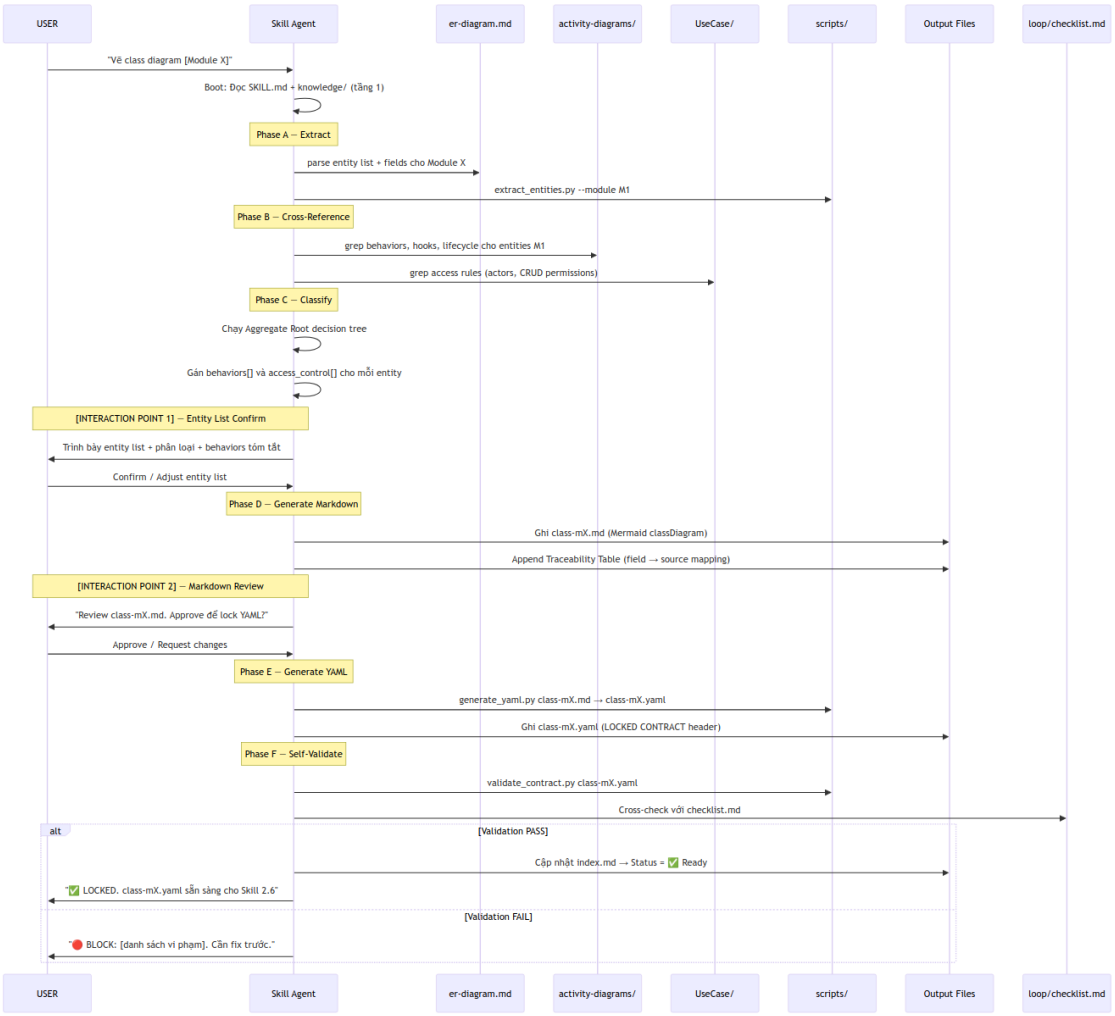
Hình 2.15: Cấu trúc thư mục của Class Diagram Analyst

2.3.3 *Quy trình 6 pha và luồng thực thi*

Skill thực hiện phân tích qua 6 pha có gate kiểm soát: Phase A (phân tích entity) → B (phân loại Aggregate Root/Embedded) → C (định nghĩa quan hệ) → IP1 → D (sinh Mermaid) → IP2 → E (sinh YAML) → F (validate) → IP3. Hình 2.16 và Hình 2.17 lần lượt thể hiện workflow và luồng tương tác chi tiết.



Hình 2.16: Workflow 6 pha với Gate Control của Class Diagram Analyst



Hình 2.17: Luồng thực thi (Sequence Diagram) của Class Diagram Analyst

2.3.4 Dual-format output strategy

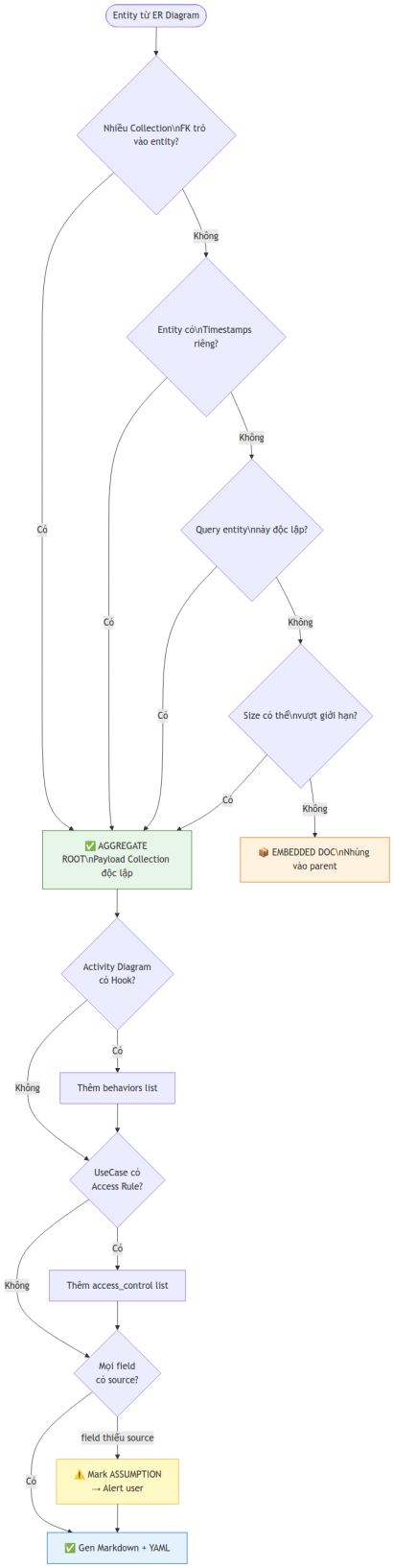
Skill tạo ra hai loại output:

- **Mermaid classDiagram** (`class-mX.md`): Dạng trực quan cho con người review
- **YAML Contract** (`class-mX.yaml`): Dạng machine-readable cho AI Agent đọc

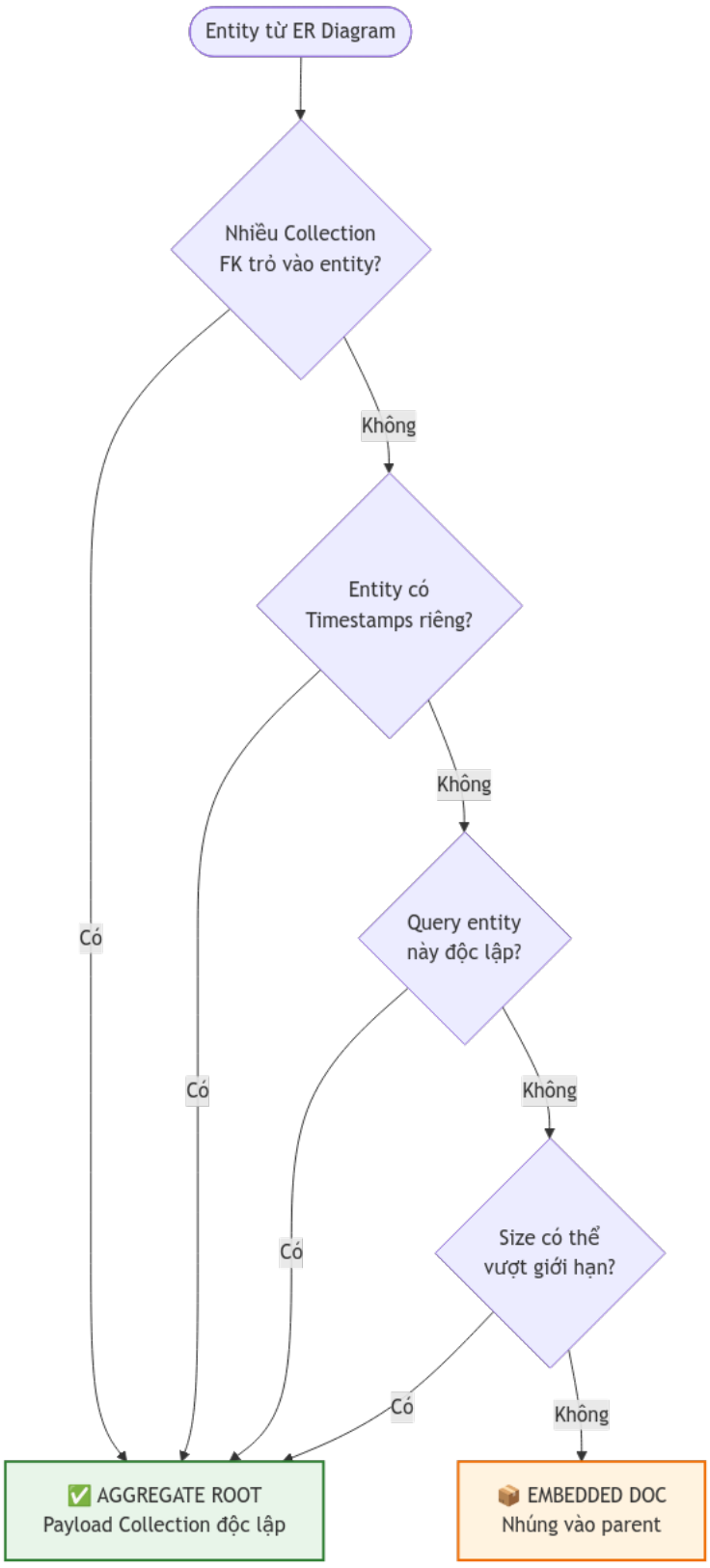
2.3.5 Aggregate Root vs Embedded Document

Một trong những quyết định quan trọng là phân loại entity thành Aggregate Root (collection độc lập) hoặc Embedded Document (nhúng trong

parent). Decision tree được mô tả trong Hình 2.19 (sơ đồ tổng quát) và Hình 2.18 (Decision Tree chi tiết trong Class Diagram Analyst).



Hình 2.18: Decision Tree: Aggregate Root vs Embedded Document trong Class Diagram Analyst



Hình 2.19: Decision Tree: Aggregate Root vs Embedded Document

2.3.6 Source Citation mechanism

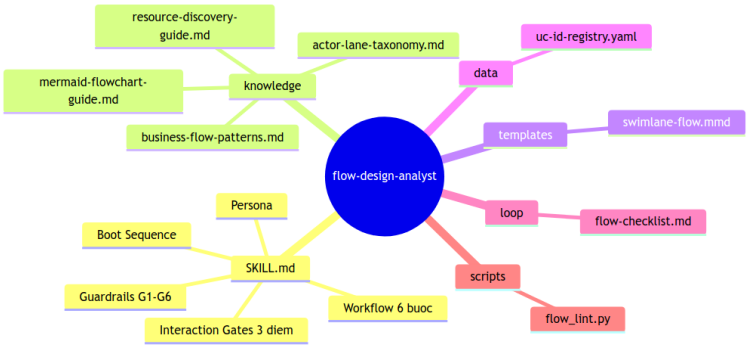
Để chống hallucination, mọi field trong Class Diagram PHẢI có source citation rõ ràng. Guardrail bắt buộc: Field không có source → BLOCK, không ghi file.

2.4 Bộ kỹ năng phân tích luồng nghiệp vụ (Flow Design Analyst)

Flow Design Analyst chuyên trách vẽ Business Process Flow Diagram với Swimlane 3-lane (User / System / DB), đảm bảo mọi bước logic đều có căn cứ từ spec hoặc User Story.

2.4.1 Cấu trúc thư mục

Hình 2.20 thể hiện cấu trúc thư mục của Flow Design Analyst, bao gồm knowledge files về Swimlane standards, templates Mermaid, data registry và loop checklist.



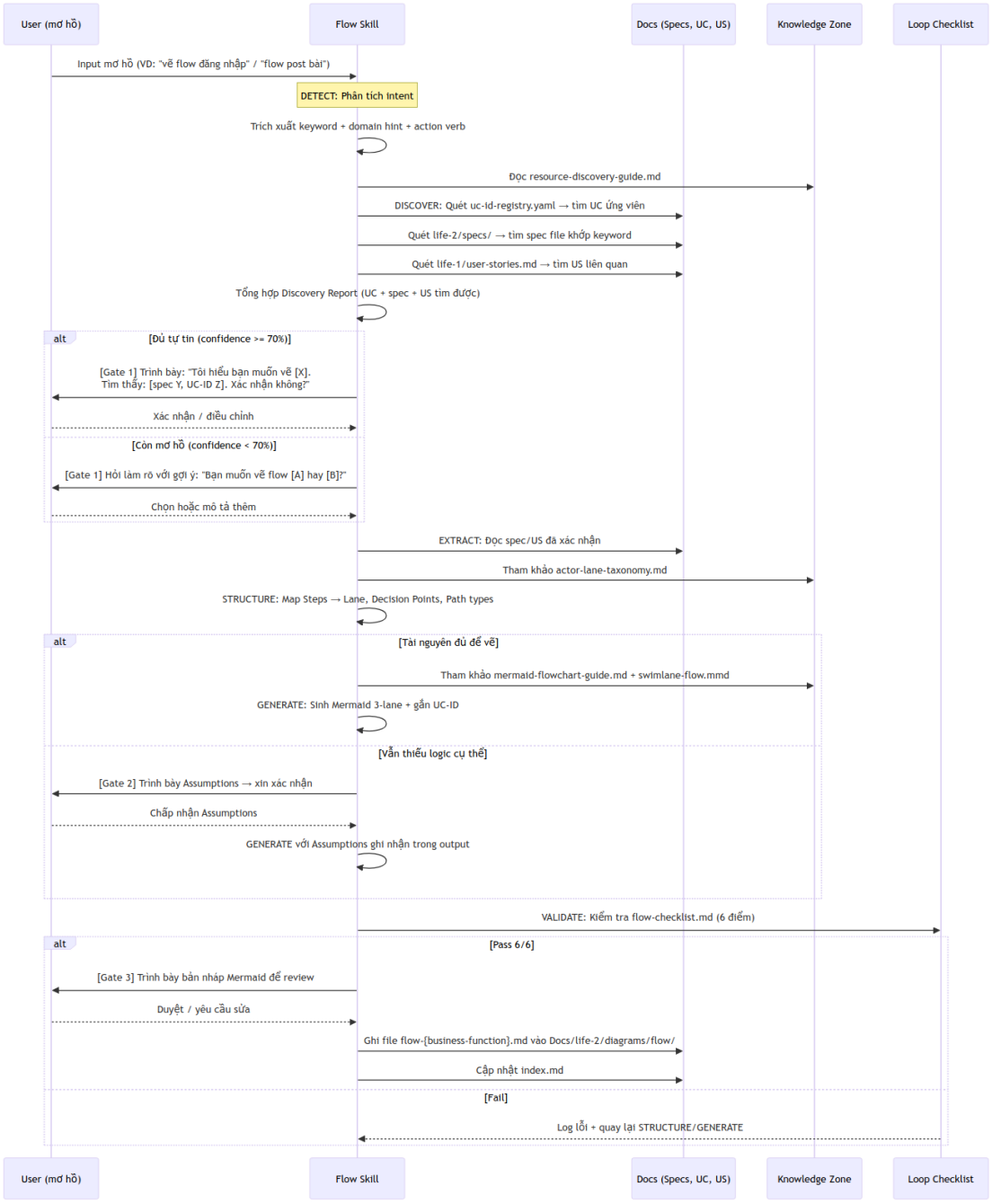
Hình 2.20: Cấu trúc thư mục của Flow Design Analyst

2.4.2 Quy trình Discover-before-Ask

Skill áp dụng nguyên tắc: KHÔNG hỏi user câu hỏi mở khi input mơ hồ. Thay vào đó, skill tự động:

1. **DETECT**: Phân tích intent từ input mơ hồ — trích xuất keyword, domain hint, action verb
2. **DISCOVER**: Tìm kiếm tài nguyên tự động
    - Quét `uc-id-registry.yaml` → UC ứng viên
    - Quét `Docs/life-2/specs/` → spec file khớp keyword
    - Quét `Docs/life-1/user-stories.md` → US liên quan
3. **EXTRACT**: Đọc tài nguyên đã xác nhận, trích xuất Trigger, Actors, Steps, Conditions, Outcomes
4. **STRUCTURE**: Phân bổ từng step vào đúng Actor Lane (User / System / DB)
5. **GENERATE**: Sinh Mermaid flowchart TD với swimlane 3 lanes + gắn UC-ID
6. **VALIDATE**: Kiểm tra: no dangling branch, all paths terminate, UC-ID mapped, Assumptions listed

Hình 2.21 minh họa toàn bộ luồng thực thi này, từ khi nhận input mơ hồ đến khi sinh diagram hoàn chỉnh, với các gates bắt buộc tại từng giai đoạn.



Hình 2.21: Luồng thực thi (Sequence Diagram) của Flow Design Analyst

2.4.3 Guardrails chống bịa đặt logic

- **G1 — No Blind Step**: Mọi Action Node PHẢI có căn cứ từ spec/US. Không tự thêm bước không có nguồn
- **G2 — Decision Completeness**: Mọi Decision Diamond PHẢI có đủ nhánh output. Không để dangling

- **G3 — Lane Discipline**: Business logic → System lane. DB read/write → DB lane. UI trigger → User lane
- **G4 — Path Termination**: Mọi nhánh PHẢI có điểm kết thúc rõ ràng
- **G5 — Assumption Required**: Khi spec chưa rõ → PHẢI khai báo `## Assumptions`
- **G6 — Discover Before Ask**: PHẢI hoàn thành Discover trước khi hỏi user bất kỳ câu hỏi nào

2.4.4 Actor Lane Taxonomy

Quy tắc phân chia trách nhiệm 3 lanes:

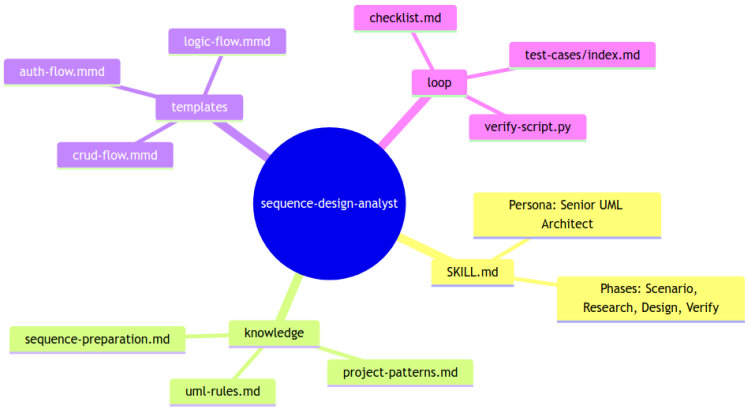
- **User Lane**: UI actions, user decisions, manual input
- **System Lane**: Business logic, validations, transformations, API calls
- **DB Lane**: CRUD operations, queries, transactions

2.5 Bộ kỹ năng phân tích sơ đồ tuần tự (Sequence Design Analyst)

Sequence Design Analyst chuyên vẽ Sequence Diagram chuẩn UML (Mermaid format), mô tả tương tác giữa các đối tượng theo thời gian.

2.5.1 Cấu trúc thư mục

Hình 2.22 thể hiện cấu trúc thư mục tối giản của Sequence Design Analyst — chỉ cần 4 zones thiết yếu: Core, Knowledge (UML standards, codebase patterns), Templates (Mermaid template) và Loop (checklist).



Hình 2.22: Cấu trúc thư mục của Sequence Design Analyst

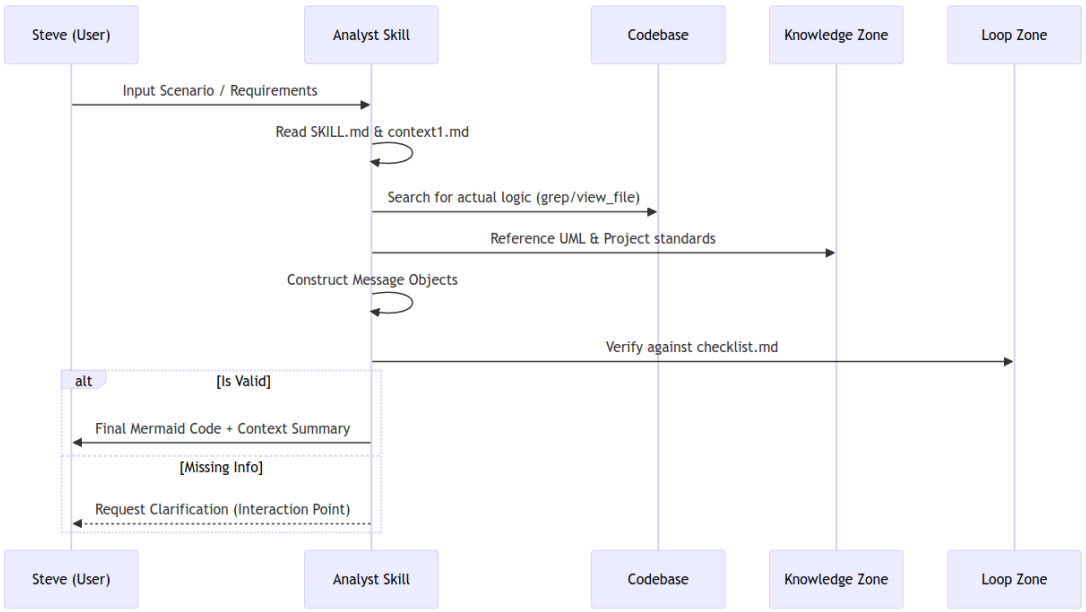
2.5.2 Nguyên tắc Code-First Truth

Skill áp dụng nguyên tắc: Chỉ vẽ những gì thực sự tồn tại trong codebase hoặc được định nghĩa rõ ràng trong thiết kế. KHÔNG vẽ dựa trên suy đoán.

Quy trình:

1. **Scenario Discovery**: Phân tích kịch bản từ input user và `context1.md`
2. **Codebase Research**: Quét codebase để xác định chính xác lifelines (Actor, Service, DB)
3. **Traceability Analysis**: Xây dựng chuỗi tương tác — ai gọi ai, tham số gì
4. **Drafting & Refinement**: Sinh code Mermaid, tối ưu vị trí lifelines
5. **Quality Assurance**: Kiểm tra chéo với Checklist trước bàn giao

Hình 2.23 minh họa luồng thực thi đầy đủ giữa Steve (user), Analyst Skill, Codebase, Knowledge Zone và Loop Zone, bao gồm nhánh xử lý khi thông tin đầy đủ (Valid) và khi thiếu thông tin (Missing Info).



Hình 2.23: Luồng thực thi (Sequence Diagram) của Sequence Design Analyst

2.5.3 UML Fragment Patterns

Skill hỗ trợ các UML fragments chuẩn:

- **alt**: Alternative paths (if-else logic)
- **opt**: Optional execution (if without else)
- **loop**: Iteration patterns (for/while)
- **ref**: Reference to another diagram (sub-sequence)

2.5.4 Naming Consistency Rule

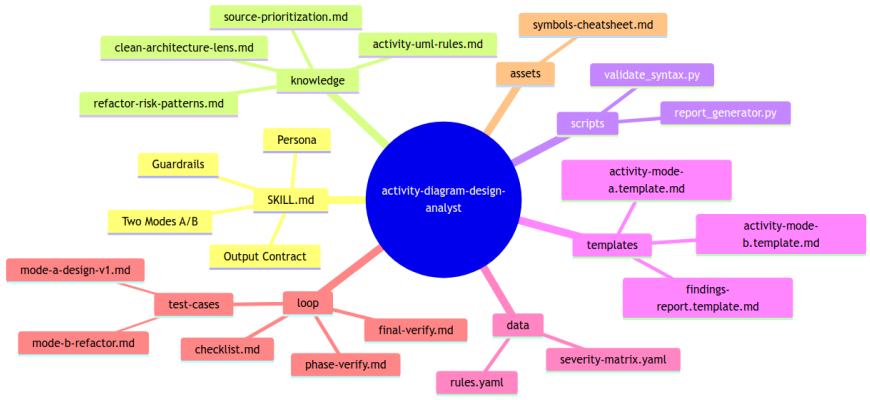
Tên Actor/Object trong sơ đồ PHẢI khớp 100% với codebase. Không được tự ý đổi tên hoặc viết tắt. Ví dụ: Nếu codebase dùng `UserService`, không được viết `UserSvc` hay `User Service`.

2.6 Bộ kỹ năng phân tích sơ đồ hoạt động (Activity Diagram Design Analyst)

Activity Diagram Analyst vẽ sơ đồ hoạt động theo tư duy Clean Architecture (Business-UseCase-External), phát hiện Deadlocks và đảm bảo tính nhất quán giữa nghiệp vụ và thiết kế.

2.6.1 Cấu trúc thư mục

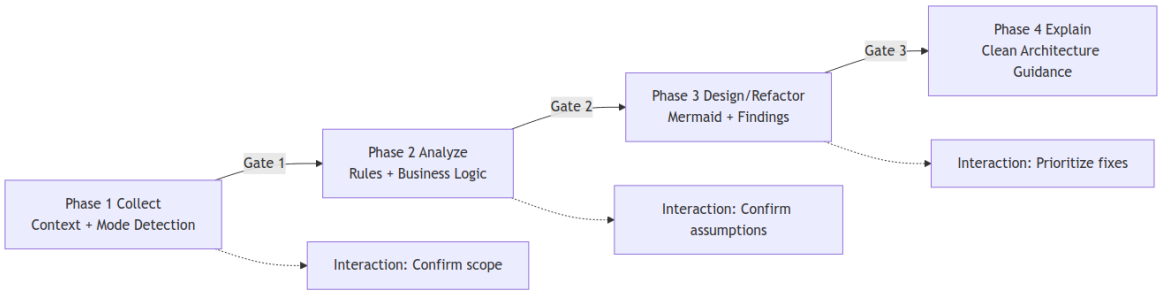
Skill sử dụng đầy đủ 7 zones, bao gồm thư mục `data/` lưu trữ các mẫu pattern phổ biến. Hình 2.24 minh họa cấu trúc này.



Hình 2.24: Cấu trúc thư mục của Activity Diagram Design Analyst

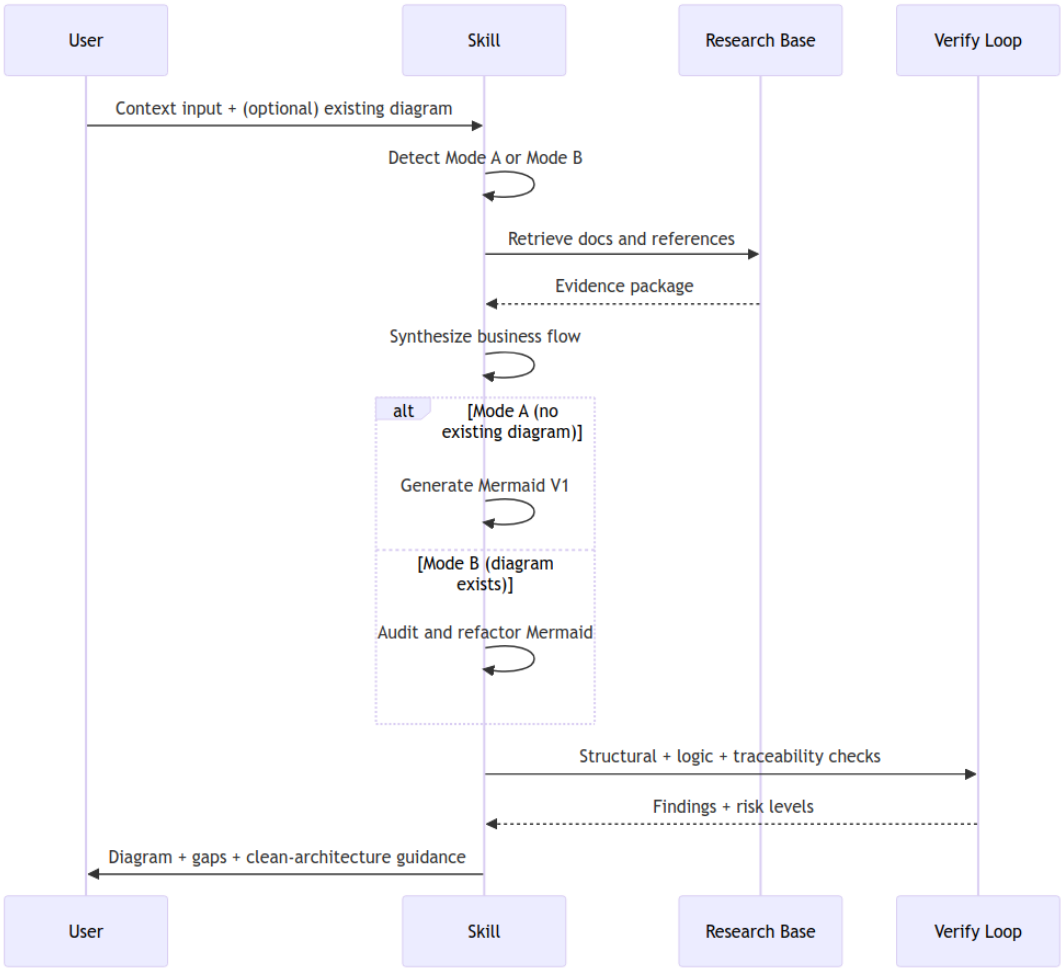
2.6.2 Quy trình 4 pha và 2 Mode thực thi

Skill hỗ trợ 2 mode hoạt động: **Mode A** (thiết kế Activity Diagram mới từ spec) và **Mode B** (refactor diagram hiện có). Cả hai mode đều trải qua 4 pha với gate kiểm soát: Collect → Analyze → Design/Refactor → Explain. Hình 2.25 thể hiện workflow gate-based này.



Hình 2.25: Workflow 4 pha Gate-based của Activity Diagram Design Analyst

Hình 2.26 thể hiện luồng thực thi chi tiết (Sequence Diagram) theo cả 2 mode, với các interaction points tại những điểm cần xác nhận từ user.



Hình 2.26: Luồng thực thi (Sequence Diagram) theo Mode A và Mode B của Activity Diagram Analyst

2.6.3 Clean Architecture Layering (B-U-E)

Sơ đồ activity được tổ chức theo 3 layers:

- **Business Layer (B)**: Core business logic, domain rules
- **Use Case Layer (U)**: Application-specific logic, orchestration
- **External Layer (E)**: Infrastructure, database, external APIs

Quy tắc: Dependency chỉ đi từ ngoài vào trong (E → U → B). KHÔNG cho phép B phụ thuộc vào U hoặc E.

2.6.4 Deadlock Detection

Skill tự động phát hiện các pattern có khả năng gây deadlock:

- Circular dependency giữa các activities
- Race condition khi 2 nhánh song song cùng truy cập 1 resource
- Infinite loop không có exit condition rõ ràng

Nếu phát hiện deadlock risk → Skill báo cáo ngay và yêu cầu user xác nhận logic.

2.6.5 Swimlane Organization

Tương tự Flow Design, Activity Diagram cũng sử dụng swimlanes nhưng theo góc nhìn Clean Architecture:

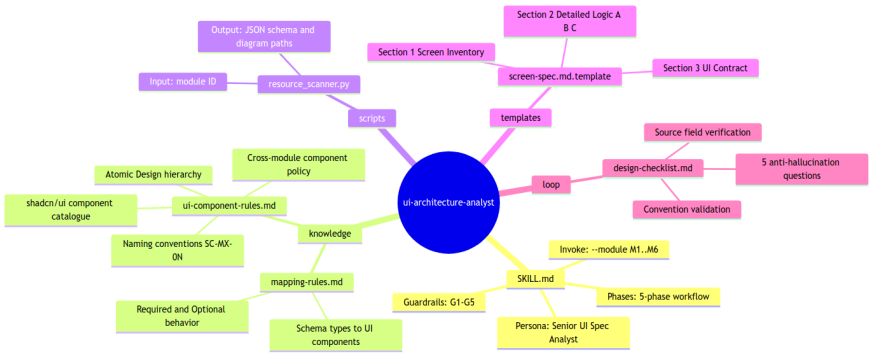
- **Business Logic Lane**: Domain entities, business rules
- **Use Case Lane**: Application services, workflows
- **Infrastructure Lane**: Repositories, external services

2.7 Bộ kỹ năng phân tích kiến trúc giao diện (UI Architecture Analyst)

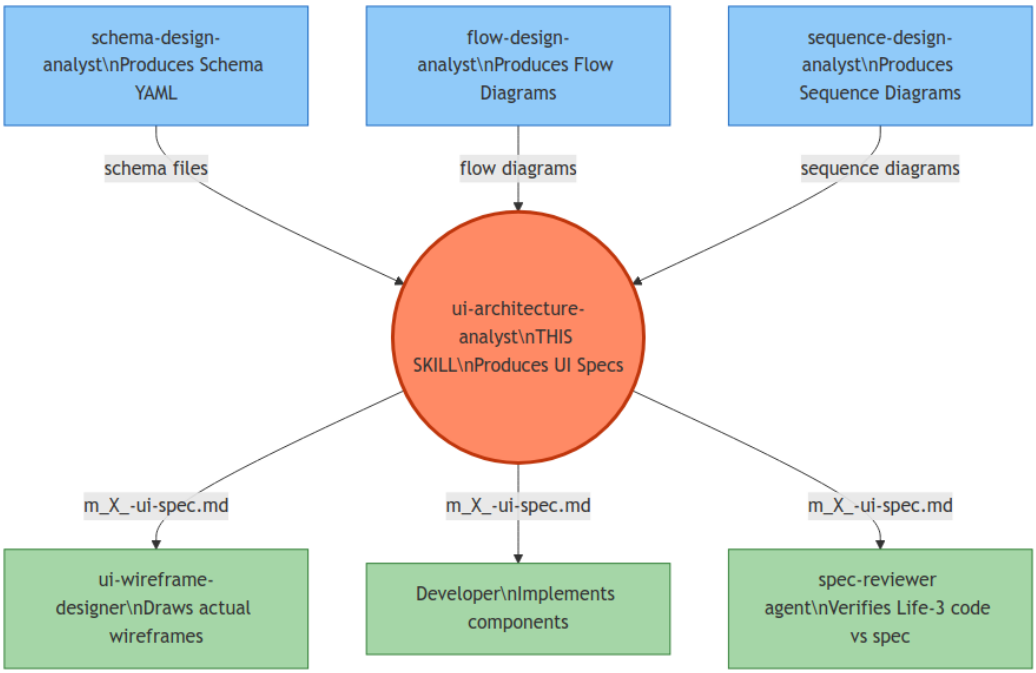
UI Architecture Analyst đóng vai trò là ”cầu nối”giữa logic hệ thống và giao diện người dùng. Skill này chuyển đổi từ Schema + Diagrams → UI Component Specs.

2.7.1 Cấu trúc thư mục và Ecosystem

Skill được tổ chức theo cấu trúc 6 zones gồm: Core (SKILL.md), Knowledge (2 file chuẩn mapping), Scripts, Templates, và Loop. Cấu trúc folder được minh họa trong Hình 2.27. Hình 2.28 thể hiện vị trí của UI Architecture Analyst trong hệ sinh thái skill — nhận đầu vào từ schema-design-analyst, flow-design-analyst, sequence-design-analyst và cung cấp đầu ra cho wireframe-designer và Developer.



Hình 2.27: Cấu trúc thư mục skill UI Architecture Analyst



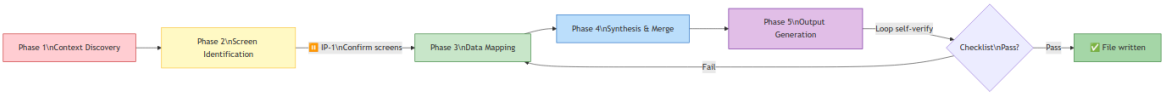
Hình 2.28: Vị trí UI Architecture Analyst trong hệ sinh thái Agent Skills

2.7.2 Data-Component Binding

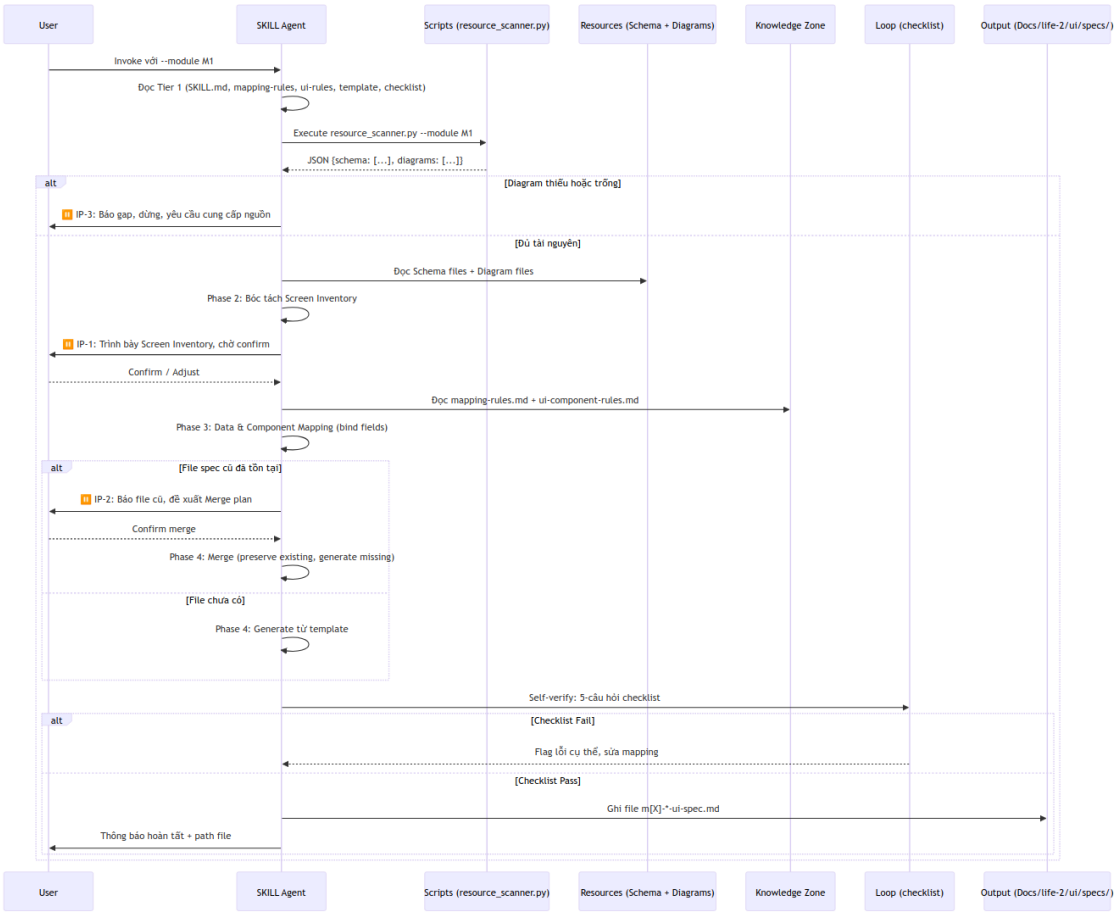
Skill thực hiện ánh xạ từ các trường dữ liệu trong Schema sang các thành phần giao diện thực tế (UI Components) dựa trên bảng mapping rules. Quy tắc: Zero Hallucination — không thêm UI field nếu không có trong Schema.

2.7.3 Quy trình 5 pha và luồng thực thi

Workflow được chia thành 5 pha có gate kiểm soát: Context Discovery → Screen Identification → Data Mapping → Synthesis & Merge → Output Generation, kết hợp với vòng lặp self-verify. Hình 2.29 minh họa workflow này. Hình 2.30 thể hiện luồng tương tác giữa User, SKILL Agent, Scripts, Knowledge và Loop qua 3 interaction points (IP-1, IP-2, IP-3).



Hình 2.29: Workflow 5 pha của UI Architecture Analyst



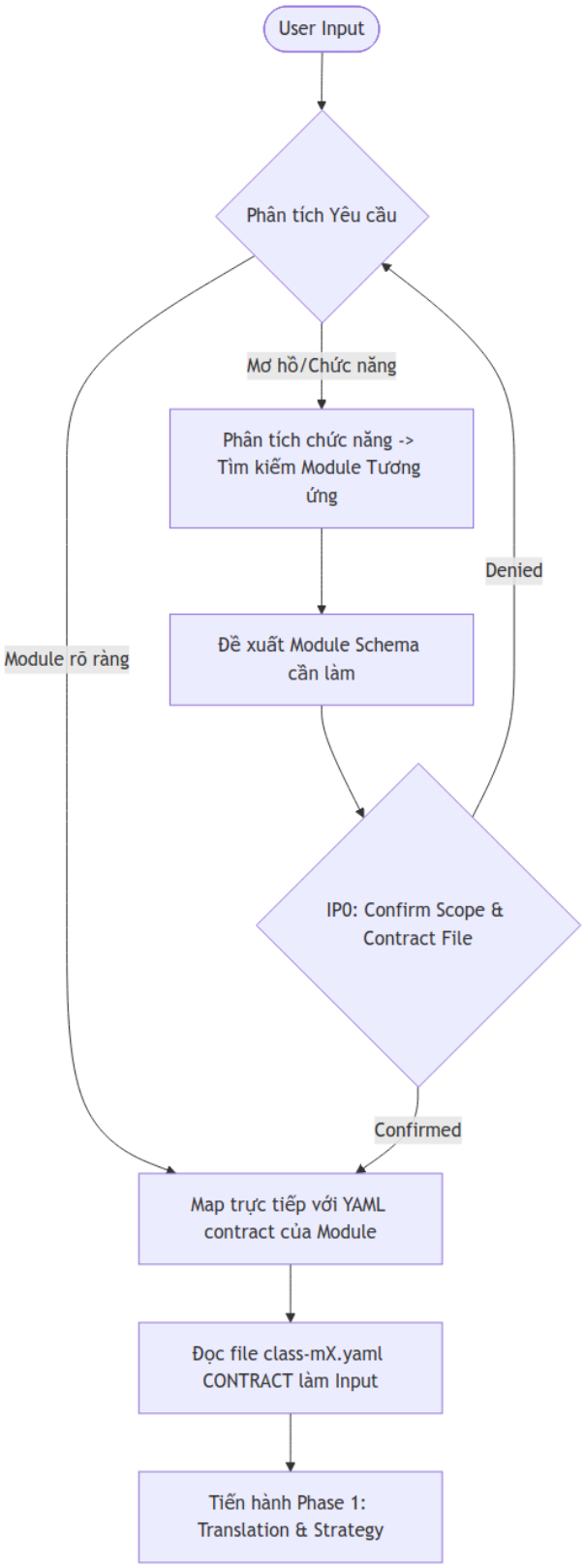
Hình 2.30: Luồng thực thi (Sequence Diagram) của UI Architecture Analyst

2.8 Bộ kỹ năng thiết kế Schema vật lý (Schema Design Analyst)

Schema Design Analyst là ”chốt chặn”cuối cùng trong pipeline thiết kế, chuyển đổi từ Class Diagram (YAML Contract) sang Physical Database Schema cho MongoDB/Payload CMS. Skill này hoạt động như một ”Kiến trúc sư Data”, chỉ làm việc dựa trên Contract YAML từ Class Diagram Analyst, loại bỏ hoàn toàn khả năng AI tự ý thêm field không có trong hợp đồng.

2.8.1 Cơ chế Input Resolution

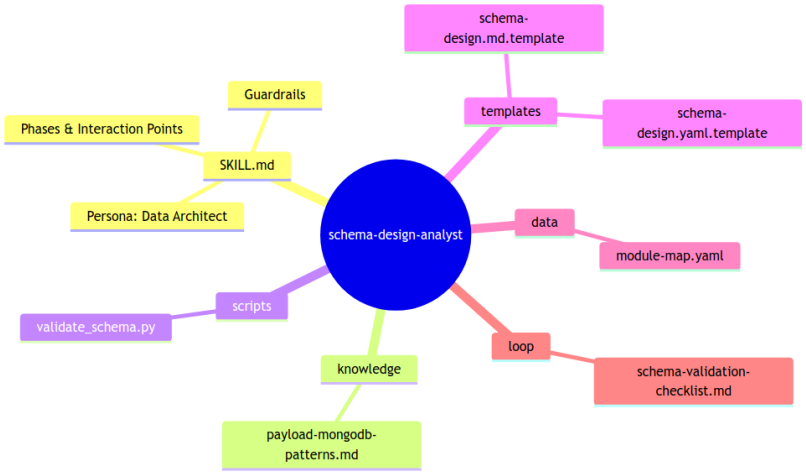
Skill hỗ trợ 4 loại đầu vào: yêu cầu module rõ ràng, yêu cầu theo chức năng, yêu cầu mơ hồ, và yêu cầu kèm file tham chiếu. Với mỗi loại, skill tự động phân tích và định vị Contract YAML tương ứng. Hình 2.31 minh họa luồng phân tích đầu vào này.



Hình 2.31: Luồng phân tích đầu vào của Schema Design Analyst (Input Resolution Flow)

2.8.2 *Cấu trúc thư mục*

Skill được tổ chức với 6 zones chức năng rõ ràng, minh họa trong Hình 2.32:



Hình 2.32: Cấu trúc thư mục skill Schema Design Analyst

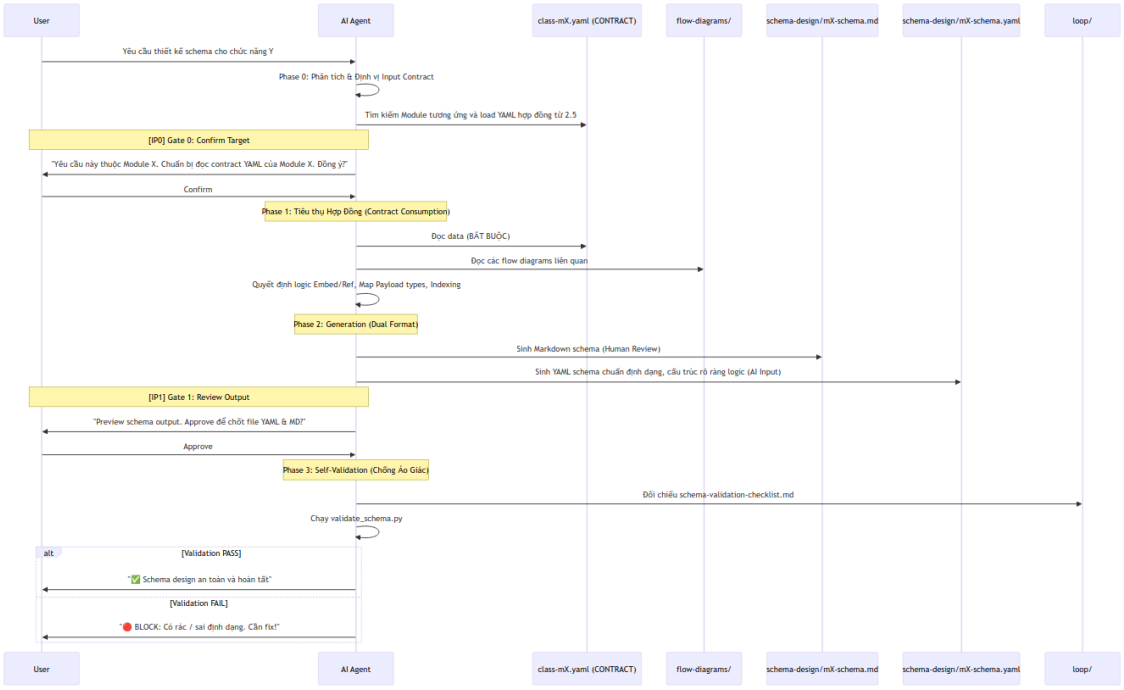
Bảng 2.2: Zone Mapping của Schema Design Analyst

Zone	Files tạo ra	Mục đích
Core	`SKILL.md`	Persona, phases, guardrails
Knowledge	`payload-mongodb-patterns.md`	Quy định Embed/Ref, Metadata strategy
Scripts	`validate_schema.py`	Kiểm tra field rác/ảo giác so với hợp đồng
Templates	`schema-design.md.template`, `schema-design.yaml.template`	Format xuất Markdown & YAML
Data	`module-map.yaml`	Map routing các module
Loop	`schema-validation-checklist.md`	Checklist verify data rules

2.8.3 *Luồng thực thi và Dual-Format Output*

Skill tạo ra hai loại đầu ra: Markdown schema cho con người review (`mX-schema.md`) và YAML schema chuẩn hóa cho AI đọc ở giai đoạn sinh code (`mX-schema.yaml`). Luồng thực thi 4 pha được minh họa trong Hình 2.33:

1. **Phase 0 — Contract Consumption**: Đọc và xác nhận Contract YAML từ Class Diagram Analyst
2. **Phase 1 — Translation & Strategy**: Quyết định Embed/Ref, map Payload field types, thiết kế indexing
3. **Phase 2 — Generation**: Sinh Markdown và YAML schema song song
4. **Phase 3 — Self-Validation**: Chạy `validate_schema.py` đối chiếu checklist



Hình 2.33: Luồng thực thi (Sequence Diagram) của Schema Design Analyst

2.8.4 Guardrails chống ảo giác

Skill áp dụng 3 guardrails cứng:

- **G1 — Contract-Only**: Mọi field PHẢI có trong Contract YAML. Field không có nguồn → `[BLOCK]`
- **G2 — Strategy Validation**: Kiểm tra MongoDB array size giới hạn 16MB trước khi chọn Embed

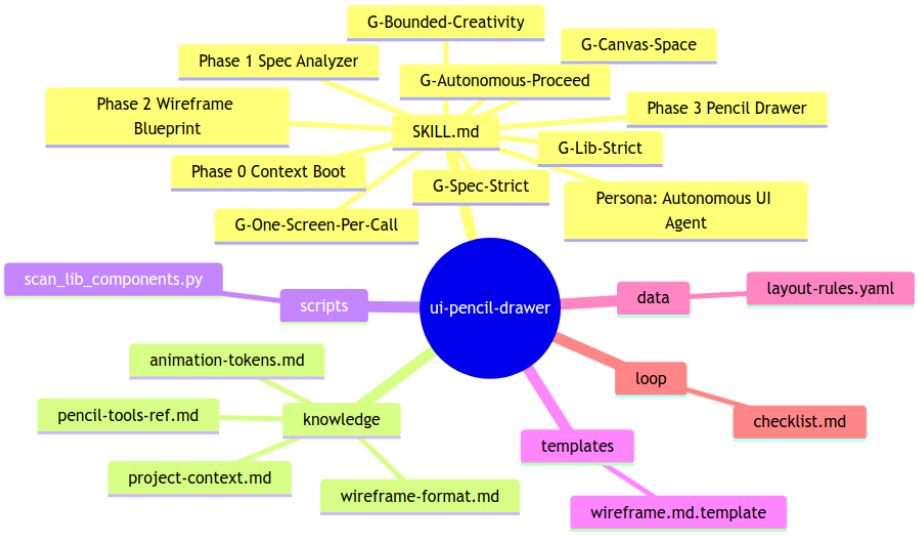
- **G3 — Context Boundary**: Chỉ load 1 module YAML mỗi lượt qua `module-map.yaml`

2.9 Bộ kỹ năng vẽ giao diện tự động (UI Pencil Drawer)

UI Pencil Drawer là skill có triết lý **AGI-Oriented Autonomous Execution** — AI tự thực thi end-to-end từ đọc spec đến vẽ hoàn chỉnh trên Pencil canvas, human chỉ cung cấp đầu vào ban đầu. Skill giải quyết bài toán bottleneck khi cần thiết kế hàng chục màn hình (M1–M6) cho Steve Void trong giai đoạn Life-2.

2.9.1 Cấu trúc thư mục và 4 Phases tự chủ

Skill được thiết kế với đầy đủ 7 zones, bao gồm 4 knowledge files chuyên biệt cho Pencil MCP API. Cấu trúc folder được minh họa trong Hình 2.34:



Hình 2.34: Cấu trúc thư mục skill UI Pencil Drawer với 4 Phases tự chủ

2.9.2 Mô hình Autonomous Execution 4 Phase

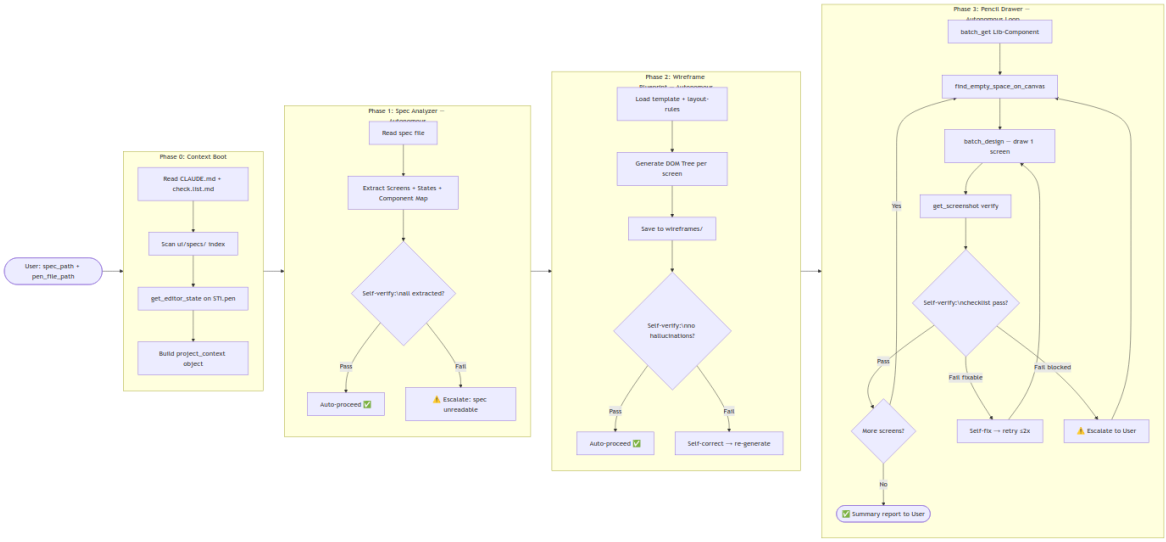
Khác với các skill khác có Interaction Points, UI Pencil Drawer tự chủ hoàn toàn qua 4 pha, chỉ escalate khi bị chặn thực sự:

Bảng 2.3: Zone Mapping của UI Pencil Drawer

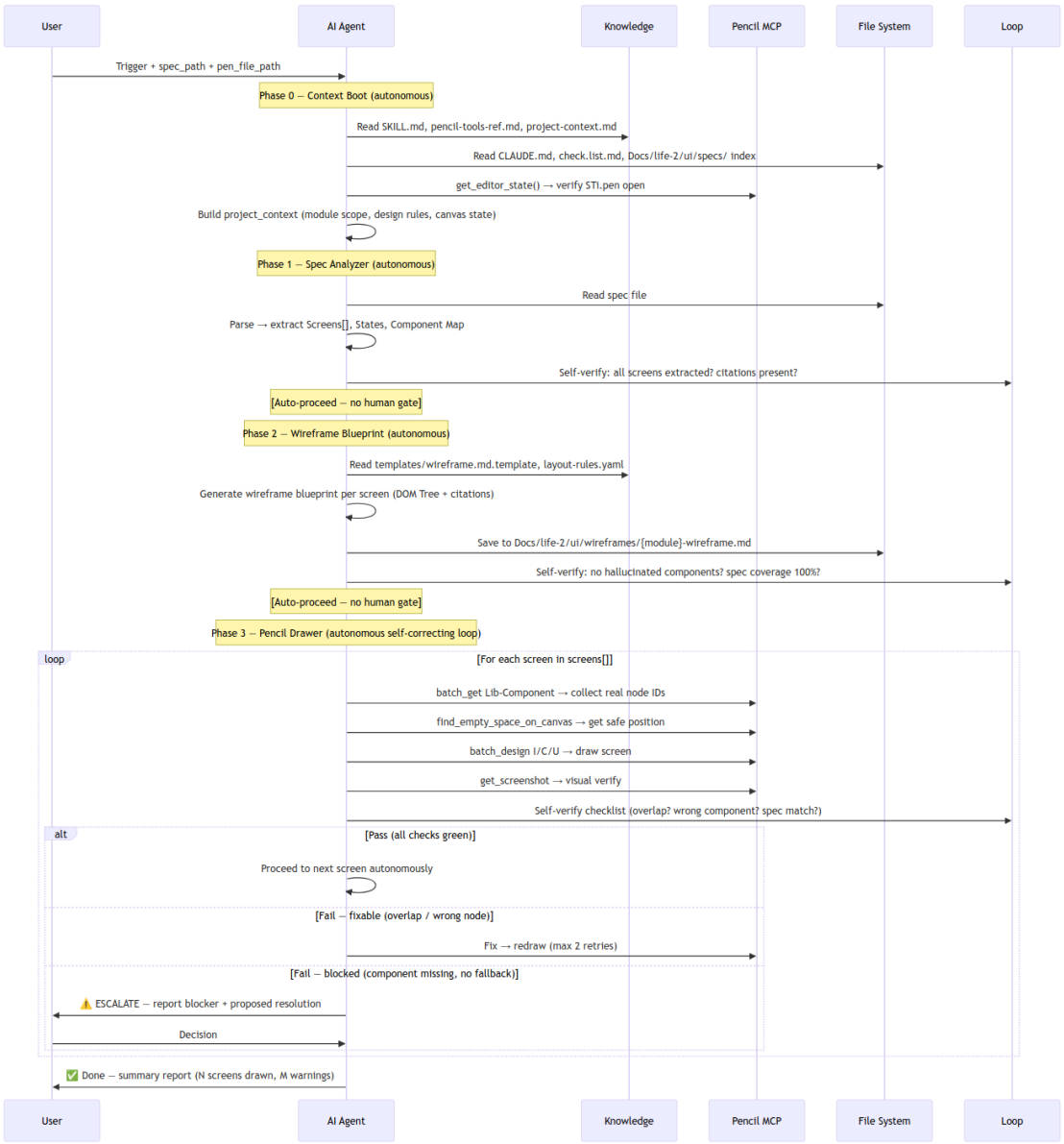
Zone	Files tạo ra	Mục đích
Core	`SKILL.md`	Persona + 4 Phases + 5 Guardrails
Knowledge	`pencil-tools-ref.md`, `wireframe-format.md`, `project-context.md`, `animation-tokens.md`	Tham chiếu đầy đủ Pencil MCP API
Scripts	`scan_lib_components.py`	Quét Lib-Component, extract node IDs
Templates	`wireframe.md.template`	Template cho 1 màn hình wireframe
Data	`layout-rules.yaml`	Static rules: gap, padding, screen width
Loop	`checklist.md`	Self-verify pass/fail threshold per phase

- **Phase 0 — Context Boot**: Tự đọc `CLAUDE.md`, `check.list.md`, quét `Docs/life-2/ui/specs/`, gọi `get_editor_state()` trên STi.pen
- **Phase 1 — Spec Analyzer**: Parse spec → extract Screens, States, Component Map với source citation
- **Phase 2 — Wireframe Blueprint**: Tạo DOM Tree text blueprint, lưu vào `wireframes/{module}-wireframe.md`
- **Phase 3 — Pencil Drawer**: Tự vẽ mỗi màn hình, screenshot verify, self-fix (max 2 retry)

Hình 2.35 minh họa workflow 4 pha với các nhánh tự xử lý lỗi. Hình 2.36 thể hiện luồng end-to-end giữa User, AI Agent, Knowledge, Pencil MCP, File System và Loop.



Hình 2.35: Workflow 4 pha Autonomous của UI Pencil Drawer



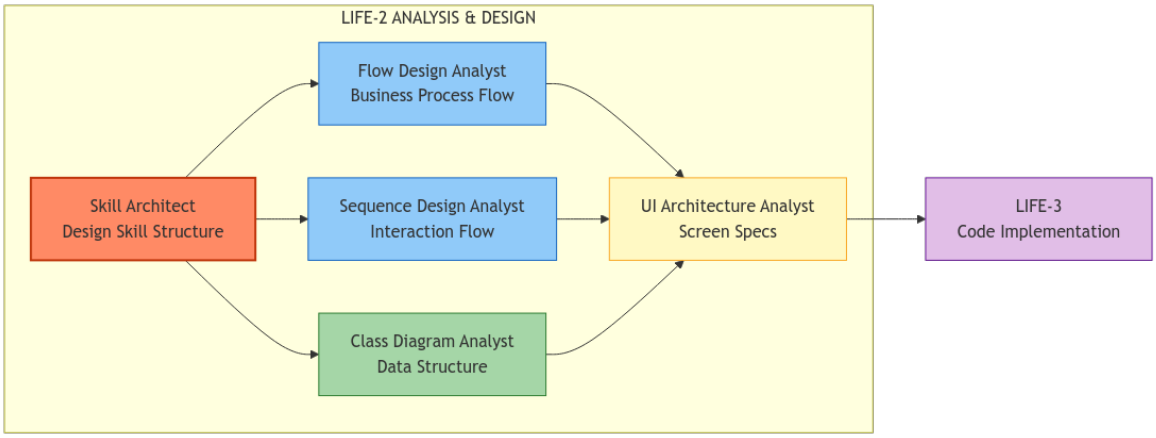
Hình 2.36: Luồng thực thi end-to-end (Sequence Diagram) của UI Pencil Drawer

2.9.3 5 Guardrails đảm bảo tính chính xác

- **G-Lib-Strict**: 100% resource phải là `ref` từ Lib-Component. Không tự vẽ primitive nếu đã có component tương đương
- **G-Spec-Strict**: Không thêm element nếu spec không đề cập. Mỗi component có source citation
- **G-Canvas-Space**: Bắt buộc gọi `find_empty_space_on_canvas` trước mỗi `batch_design`
- **G-One-Screen-Per-Call**: Mỗi `batch_design` chỉ vẽ 1 màn hình, screenshot verify trước khi tiếp
- **G-Bounded-Creativity**: Phân tách Strict Zones (lắp ghép 100% đúng Spec) và Fluid Zones (sáng tạo spacing, AI image)

2.10 Quy trình phối hợp và tích hợp

Toàn bộ các bộ kỹ năng hoạt động phối hợp theo một pipeline tuần tự và chặt chẽ, tạo thành một ”Knowledge Factory”tự động hóa. Pipeline hoàn chỉnh được minh họa trong Hình 2.37:



Hình 2.37: Pipeline tổng thể của hệ thống Agent Skills

Các bước chính:

1. **Thiết kế kỹ năng**: Sử dụng Skill Architect để định hình vai trò của từng Agent
2. **Phân tích nghiệp vụ**: Flow/Sequence/Activity Analyst xây dựng khung logic
3. **Thiết kế dữ liệu**: Class Diagram Analyst chuyển đổi sang OOP view
4. **Thiết kế giao diện**: UI Architecture Analyst ánh xạ Schema → UI Components
5. **Đóng gói tri thức**: Tổng hợp toàn bộ kết quả thành tài liệu kỹ thuật minh bạch

Thiết kế này không chỉ tạo ra một website mạng xã hội cụ thể, mà quan trọng hơn là thiết lập được một **nhà máy sản xuất tri thức (Knowledge Factory)** cho phép AI tự động hóa quy trình phát triển phần mềm một cách tin cậy và có kiểm soát.

CHƯƠNG 3

TRIỂN KHAI VÀ THỬ NGHIỆM HỆ THỐNG

Chương này trình bày quá trình triển khai bộ Agent Skills đã thiết kế ở Chương 2 vào thực tế, cùng với toàn bộ kết quả đầu ra thu được khi áp dụng các skills để phân tích và thiết kế hệ thống Steve Void. Các kết quả bao gồm: sơ đồ UML (Use Case, Activity, Sequence, Flow, Class), sơ đồ quan hệ thực thể (ER Diagram), thiết kế cơ sở dữ liệu vật lý (Schema Design), đặc tả giao diện người dùng (UI Specs) và bản vẽ wireframe.

3.1 Mô tả quá trình triển khai hệ thống

3.1.1 Môi trường triển khai

Toàn bộ bộ Agent Skills được triển khai trên nền tảng **Claude Code CLI** — công cụ dòng lệnh của Anthropic, tích hợp trực tiếp vào terminal của lập trình viên. Môi trường triển khai bao gồm:

Bảng 3.1: Môi trường triển khai Agent Skills

Thành phần	**Chi tiết**
Hệ điều hành	Ubuntu Linux 24.04
AI Model	Claude Sonnet 4.6 (Anthropic)
Công cụ CLI	Claude Code v1.x
IDE	Visual Studio Code + Claude Code Extension
Quản lý mã nguồn	Git + GitHub
Công cụ vẽ sơ đồ	Mermaid.js (tích hợp trong Markdown)
Công cụ wireframe	Pencil MCP Server (Model Context Protocol)

3.1.2 Quy trình triển khai theo pipeline

Việc triển khai tuân theo pipeline tuần tự đã thiết kế ở Chương 2 (Hình 2.37). Mỗi module M1–M6 được xử lý qua 6 giai đoạn, với đầu ra của giai đoạn trước là đầu vào của giai đoạn sau:

1. **Giai đoạn 1 — Use Case Analysis**: Xác định actors, use cases và ranh giới hệ thống
2. **Giai đoạn 2 — Activity Diagram**: Phân tích luồng nghiệp vụ chi tiết theo kiến trúc B-U-E
3. **Giai đoạn 3 — Sequence & Flow Diagram**: Mô tả tương tác giữa các thành phần và luồng 3-lane Swimlane
4. **Giai đoạn 4 — Class Diagram**: Chuyển đổi sang cấu trúc OOP với dual-format (Mermaid + YAML)
5. **Giai đoạn 5 — Schema Design**: Sinh Physical Schema cho MongoDB/Payload CMS từ Contract YAML
6. **Giai đoạn 6 — UI Design**: Đặc tả giao diện và vẽ wireframe tự động trên Pencil canvas

Tổng cộng, hệ thống đã xử lý **6 modules** (M1–M6) qua **6 giai đoạn**, tạo ra hơn **100 artifacts** thiết kế hoàn chỉnh.

3.1.3 Cách thức tương tác với Agent Skills

Mỗi Agent Skill được kích hoạt thông qua lệnh slash command trong Claude Code CLI. Ví dụ:

- `/flow-design-analyst` — Vẽ Flow Diagram cho một use case cụ thể

- `/sequence-design-analyst` — Vẽ Sequence Diagram cho một kịch bản
- `/class-diagram-analyst` — Phân tích Class Diagram cho một module
- `/ui-pencil-drawer` — Tự động vẽ wireframe trên canvas

Người dùng cung cấp đầu vào (tên module, use case ID, hoặc mô tả mơ hồ), Agent Skill tự động khám phá tài nguyên dự án (specs, user stories, ER diagram), trích xuất logic nghiệp vụ và sinh đầu ra chuẩn hóa. Các Interaction Points (IP) đảm bảo người dùng kiểm soát được chất lượng tại từng bước quan trọng.

3.2 Trình bày kết quả thử nghiệm

Phần này trình bày chi tiết toàn bộ kết quả đầu ra khi triển khai bộ Agent Skills cho hệ thống Steve Void. Bảng 3.2 tổng hợp số lượng artifacts theo từng loại sơ đồ.

Bảng 3.2: Tổng hợp artifacts thiết kế được tạo ra bởi Agent Skills

Loại sơ đồ	**Số files**	**Agent Skill**	**Định dạng**
Use Case Diagram	9	Thủ công + AI	Mermaid
Activity Diagram	27	Activity Analyst	Mermaid Flowchart
Sequence Diagram	9	Sequence Analyst	Mermaid Sequence
Flow Diagram	27	Flow Analyst	Mermaid Swimlane
Class Diagram	12	Class Analyst	Mermaid + YAML
ER Diagram	1	Thủ công + AI	Mermaid erDiagram
Database Schema	12	Schema Analyst	Markdown + YAML
UI Spec	8	UI Architecture	Markdown
Wireframe	17	UI Pencil Drawer	Markdown blueprint
Tổng cộng	**122**		

3.2.1 Kết quả sơ đồ Use Case

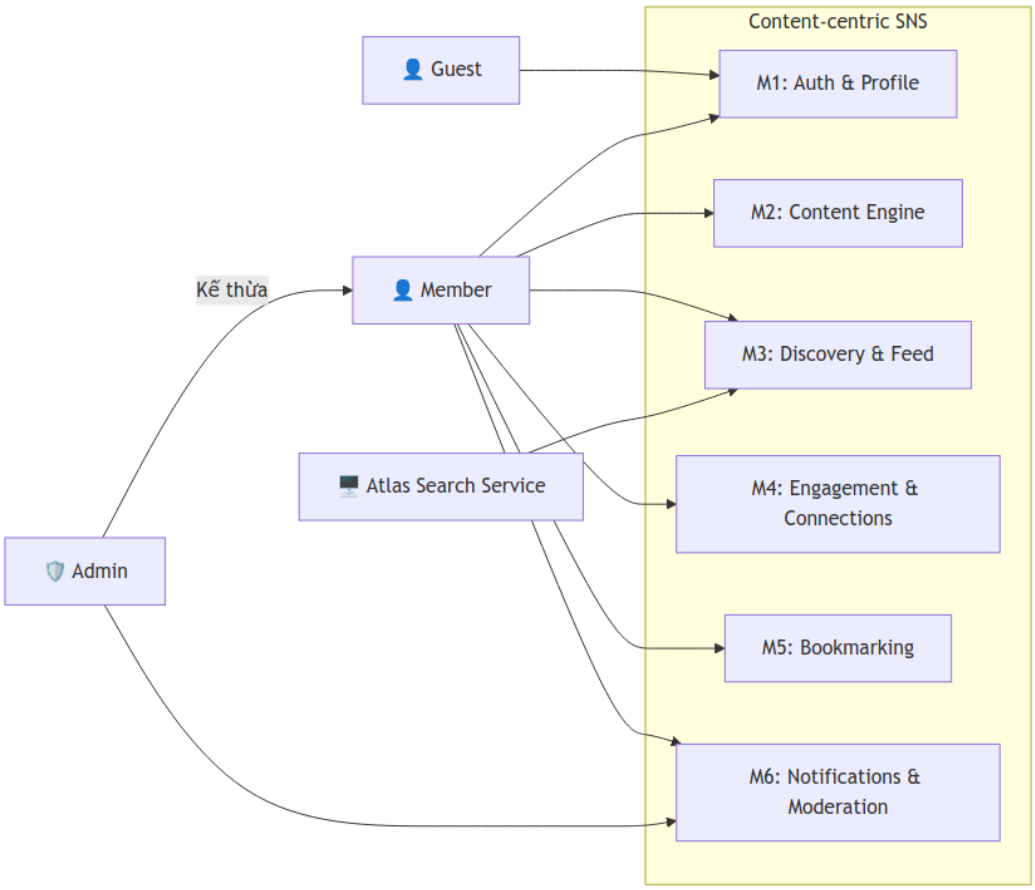
Sơ đồ Use Case xác định ranh giới hệ thống, các tác nhân (Actors) và các trường hợp sử dụng (Use Cases) cho toàn bộ 6 modules. Hệ thống định nghĩa **4 loại tác nhân** với quan hệ kế thừa: Guest → Member → Admin, cùng External Services (OAuth Provider, Atlas Search).

Bảng 3.3: Phân bổ Use Cases theo module

Module	Tên module	Số UC	UC-IDs
M1	Auth & Profile	11	UC01–UC07 (core) + 4 phụ
M2	Content Engine	6	UC08–UC10 + 3 phụ
M3	Discovery & Feed	5	UC11–UC13 + 2 phụ
M4	Engagement	7	UC14–UC18 + 2 phụ
M5	Bookmarking	4	UC19–UC20 + 2 phụ
M6	Notifications & Moderation	8	UC21–UC24 + 4 phụ
Tổng		**41**	

Mỗi Use Case được đặc tả với: Tên, Mô tả, Tác nhân chính, Tiền điều kiện, Luồng chính, Luồng thay thế và Hậu điều kiện. Toàn bộ 24 use case chính (UC01–UC24) đều có truy xuất nguồn gốc (traceability) tới yêu cầu chức năng FR-1 đến FR-10.

Hình 3.1 minh họa sơ đồ Use Case tổng quan của hệ thống, thể hiện mối quan hệ giữa các tác nhân và 6 modules chức năng.



Hình 3.1: Sơ đồ Use Case tổng quan hệ thống Steve Void

3.2.2 Kết quả sơ đồ hoạt động (Activity Diagrams)

Activity Diagram Analyst tạo ra **27 sơ đồ hoạt động** với mức độ chi tiết cao, mỗi sơ đồ phân tích một sub-activity cụ thể theo kiến trúc Boundary-UseCase-Entity (B-U-E). Bảng 3.4 liệt kê phân bổ theo module.

Bảng 3.4: Kết quả Activity Diagrams theo module

Module	Số AD	Các sub-activities
M1	6	A1-Registration, A2-Login, A3-Verification, A4-Recovery, A5-Onboarding + Summary
M2	5	A1-Editor Pipeline, A2-Media Handler, A3-Post Integrity, A4-Visibility + Summary
M3	4	A1-Feed Assembler, A2-Search Engine, A3-Discovery Recommendation + Summary
M4	4	A1-Friendship Handshake, A2-Engagement Logic, A3-Connection Privacy + Summary
M5	3	A1-Bookmark Persistence, A2-Collection Orchestrator + Summary
M6	4	A1-SSE Dispatcher, A2-Report Pipeline, A3-Enforcement Action + Summary

Mỗi Activity Diagram sử dụng Mermaid Flowchart TD (Top-Down) với 3 swimlanes: **User Lane** (UI actions), **System Lane** (business logic, validations) và **DB Lane** (CRUD operations). Các sơ đồ bao gồm đầy đủ: điều kiện rẽ nhánh (Decision Diamonds), xử lý lỗi (Error Paths) và điểm kết thúc rõ ràng cho mọi nhánh.

3.2.3 Kết quả sơ đồ tuần tự (Sequence Diagrams)

Sequence Design Analyst tạo ra **9 sơ đồ tuần tự**, mỗi sơ đồ mô tả tương tác chi tiết giữa các thành phần hệ thống theo thời gian. Các lifelines

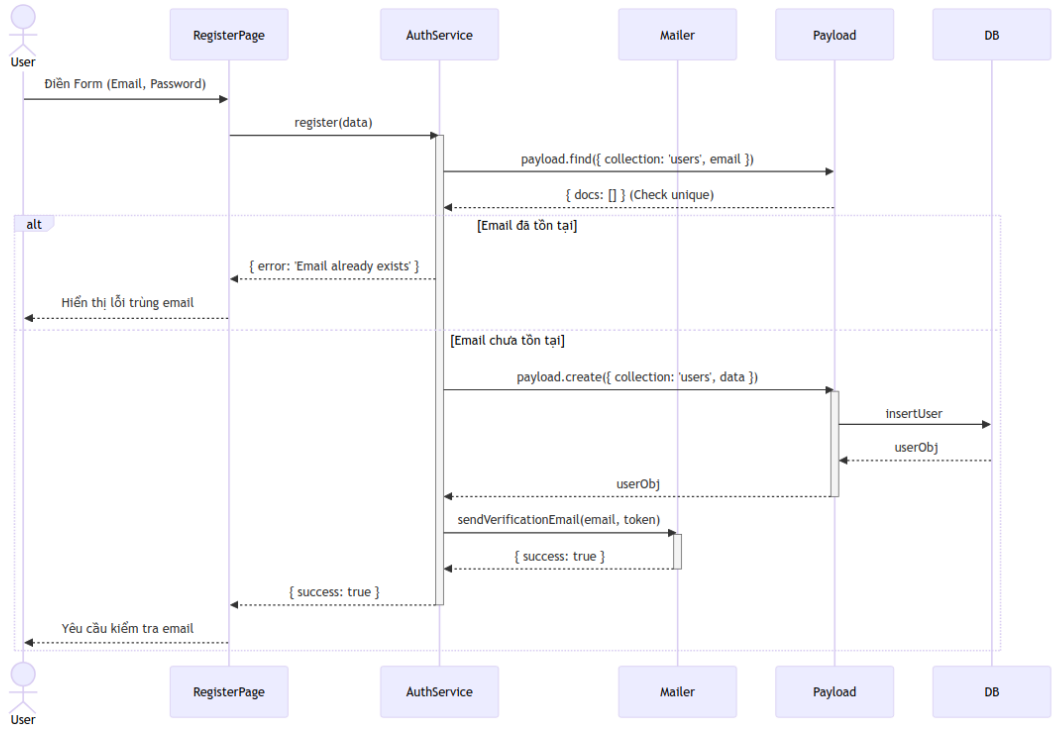
(participants) bao gồm: `User`, `UI (Next.js/React)`, `Service`, `Payload CMS`, `DB (MongoDB)`, `SSE Dispatcher` và `Mailer`.

Bảng 3.5: Kết quả Sequence Diagrams theo module

Module	File	Kịch bản
Global	global-flows.md	Kiến trúc tổng quan, luồng xác thực chung
M1	detailed-m1-auth.md	Login, Register, OAuth, Email verify, Password recovery
M2	detailed-m2-content.md	Editor pipeline, media upload, visibility enforcement
M3	detailed-m3-discovery.md	Feed assembly, search engine, ranking algorithm
M4	detailed-m4-engagement.md	Follow handshake, like/comment, sharing
M5	detailed-m5-bookmarking.md	Bookmark toggle, collection CRUD
M6	detailed-m6-safety.md	SSE notifications, report pipeline, moderation queue

Các sơ đồ sử dụng đầy đủ UML fragments chuẩn: `alt` (alternative paths), `opt` (optional execution), `loop` (iteration) và `activate/deactivate` blocks để thể hiện thời gian xử lý. Nguyên tắc **Code-First Truth** đảm bảo tên các đối tượng trong sơ đồ khớp 100% với codebase dự kiến.

Hình 3.2 minh họa sơ đồ tuần tự cho kịch bản Đăng ký tài khoản (M1-A1), thể hiện tương tác giữa User, RegisterPage, AuthService, Mailer, Payload CMS và Database.



Hình 3.2: Sơ đồ tuần tự: Đăng ký tài khoản (M1-A1)

3.2.4 Kết quả sơ đồ luồng nghiệp vụ (Flow Diagrams)

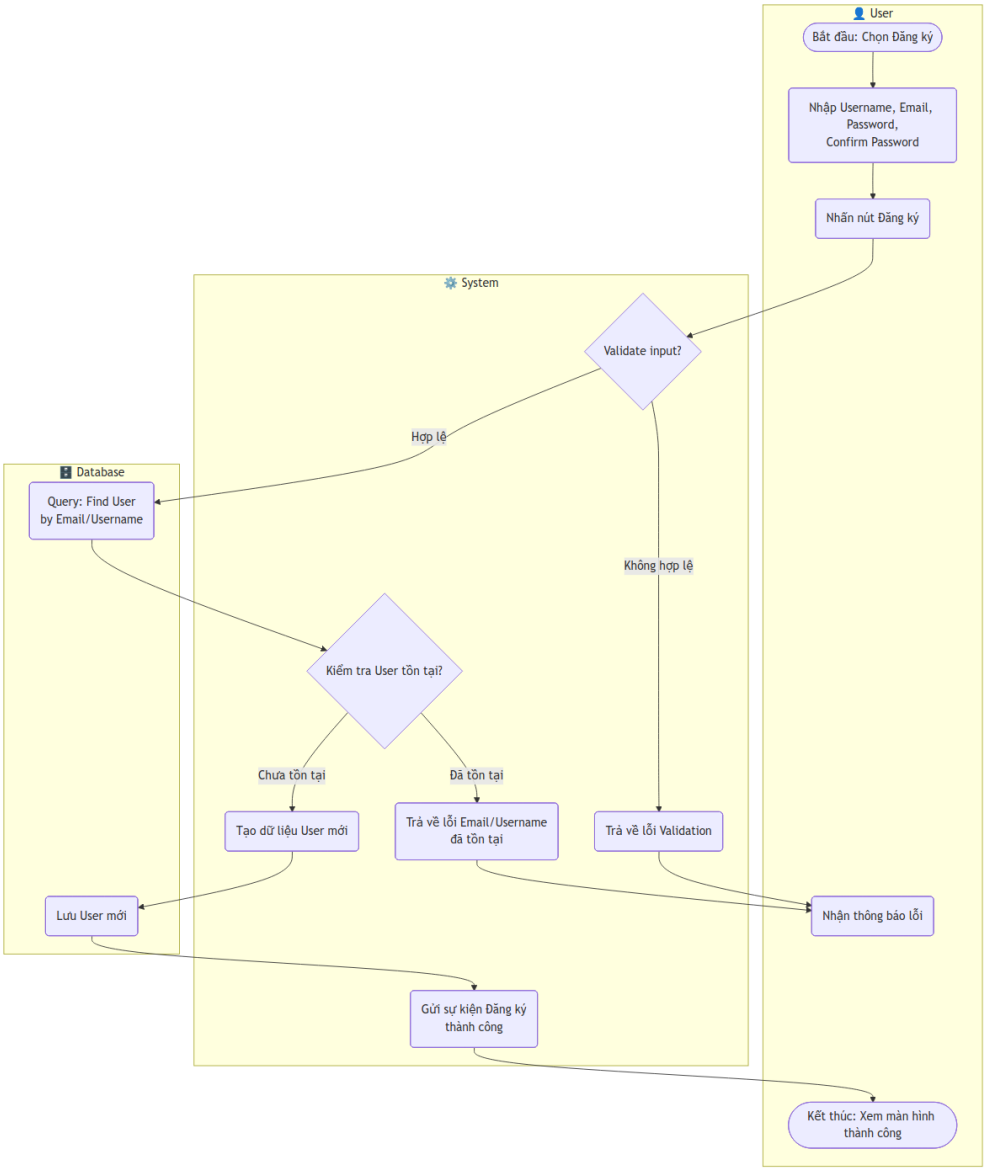
Flow Design Analyst tạo ra **27 sơ đồ luồng nghiệp vụ**, bao phủ **100% use cases chính** (UC01–UC24). Mỗi sơ đồ sử dụng kiến trúc 3-lane Swimlane: **User** (tác nhân), **System** (logic nghiệp vụ) và **Database** (thao tác dữ liệu).

Bảng 3.6: Kết quả Flow Diagrams theo module và Use Case

Module	Số Flow	Các luồng nghiệp vụ
M1	7	Registration, Login Email, Login OAuth, Logout, Password Recovery, Profile Management, Public Profile View
M2	3	Post Creation, Post Modification, Post Privacy
M3	3	News Feed View, Search Engine, Search Autocomplete
M4	5	Post Reaction, Post Comment, Post Share, User Follow, User Block
M5	2	Bookmark Toggle, Bookmark Collection
M6	4	Notification Realtime, Notification Read, Content Report, Moderation Review

Đặc biệt, mọi Flow Diagram đều được gắn mã **UC-ID** (Use Case ID) để đảm bảo truy xuất nguồn gốc ngược lại yêu cầu chức năng. Guardrail **G1 — No Blind Step** của Flow Design Analyst đảm bảo không có bước nào trong sơ đồ được thêm vào mà không có căn cứ từ spec hoặc user story.

Hình 3.3 minh họa Flow Diagram cho chức năng Đăng ký tài khoản (UC01), với 3 swimlanes User / System / Database.



Hình 3.3: Sơ đồ luồng nghiệp vụ: Đăng ký tài khoản (UC01)

3.2.5 Kết quả sơ đồ lớp (Class Diagrams)

Class Diagram Analyst tạo ra **12 files** (6 Mermaid + 6 YAML Contract) cho toàn bộ 6 modules, với **93 trường dữ liệu** được kiểm chứng và **0 trường giả tạo** (0 assumptions unresolved).

Bảng 3.7: Kết quả Class Diagrams theo module

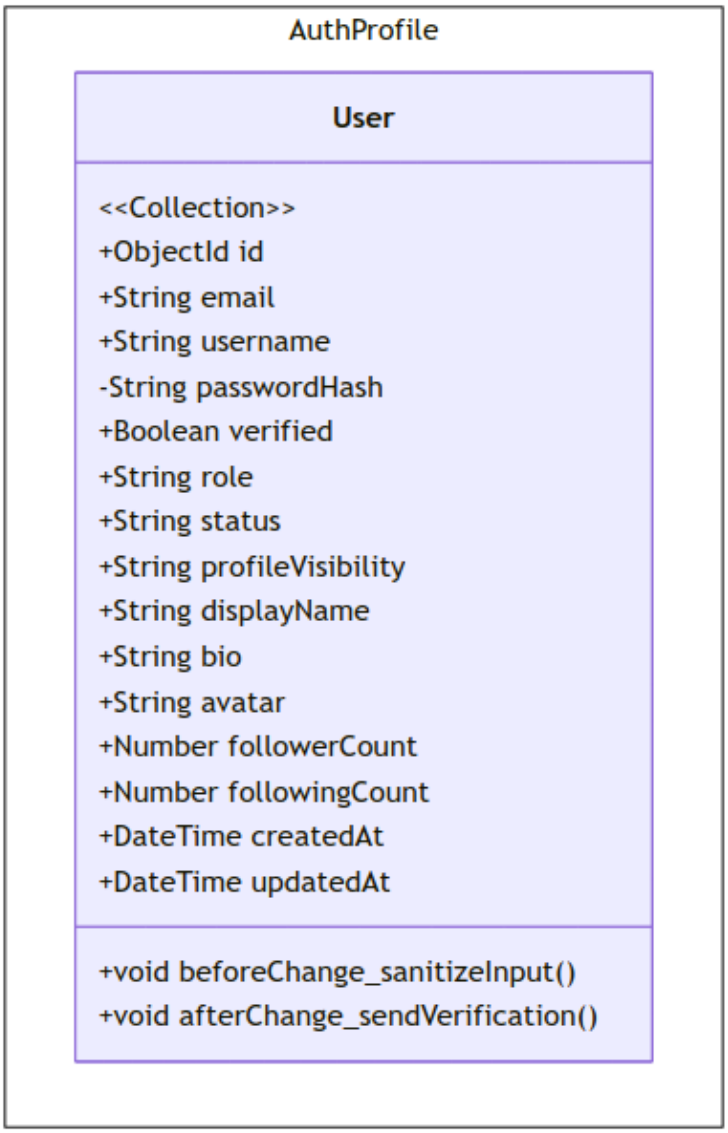
Module	Entities	Số fields	Ghi chú
M1	Users	19	Computed Pattern cho counts
M2	Posts, Media, Tags	23	Denormalized counters
M3	FeedQuery (ValueObject)	9	Không tạo collection vật lý
M4	Comments, Likes, Connections, Shares	20	Compound unique indices
M5	BookmarkCollections, Bookmarks	9	isDefault constraint
M6	Notifications, Reports, AuditLogs	21	Polymorphic + Append-Only

Chiến lược **Dual-Format Output** tạo ra hai loại đầu ra cho mỗi module:

- **Mermaid classDiagram** (`class-mX.md`): Dạng trực quan cho con người review, bao gồm trường dữ liệu, phương thức và quan hệ
- **YAML Contract** (`class-mX.yaml`): Dạng machine-readable cho Schema Design Analyst đọc ở giai đoạn tiếp theo

Mỗi trường trong Class Diagram đều có **Source Citation** — truy xuất nguồn gốc tới ER Diagram hoặc Activity Diagram cụ thể. Guardrail **Source Citation Mechanism** đảm bảo: field không có source → BLOCK, không ghi file.

Hình 3.4 minh họa Class Diagram của module M1 (Auth & Profile), thể hiện cấu trúc collection `User` với đầy đủ fields, visibility modifiers và hooks.



Hình 3.4: Class Diagram: Collection User (M1 — Auth & Profile)

3.2.6 Kết quả sơ đồ quan hệ thực thể (ER Diagram)

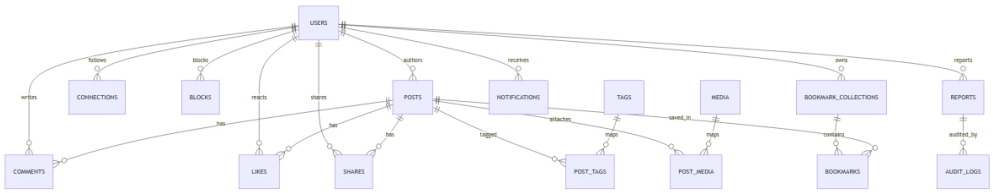
ER Diagram tổng quan mô tả **13 collections** (entities) với đầy đủ thuộc tính và quan hệ giữa chúng. Sơ đồ được thiết kế ở mức **Physical Schema** — có thể sử dụng trực tiếp để implement Payload CMS collections.

Bảng 3.8: Danh sách 13 collections trong ER Diagram

STT	Collection	Module	Mô tả
1	users	M1	Tài khoản và hồ sơ người dùng
2	posts	M2	Bài viết (text, image, link)
3	media	M2	File đa phương tiện
4	tags	M2	Nhãn phân loại nội dung
5	comments	M4	Bình luận (hỗ trợ threaded replies)
6	likes	M4	Lượt thích (unique per user+post)
7	shares	M4	Chia sẻ (copy_link hoặc repost)
8	connections	M4	Quan hệ follow/unfollow
9	bookmark_collections	M5	Bộ sưu tập bookmark
10	bookmarks	M5	Bài viết đã lưu
11	notifications	M6	Thông báo (polymorphic)
12	reports	M6	Báo cáo vi phạm
13	audit_logs	M6	Nhật ký hành chính (append-only)

Các quan hệ chính bao gồm: `USERS ||--o{ POSTS` (1 user → nhiều posts), `POSTS ||--o{ COMMENTS` (1 post → nhiều comments), `USERS ||--o{ CONNECTIONS` (follow/unfollow), `BOOKMARK_COLLECTIONS ||--o{ BOOKMARKS` (1 collection → nhiều bookmarks), và `REPORTS ||--o{ AUDIT_LOGS` (1 report → nhiều audit entries).

Hình 3.5 minh họa ER Diagram tổng quan với toàn bộ 13 entities và quan hệ giữa chúng.



Hình 3.5: Sơ đồ quan hệ thực thể (ER Diagram) tổng quan hệ thống Steve Void

3.2.7 Kết quả thiết kế cơ sở dữ liệu vật lý (Schema Design)

Schema Design Analyst chuyển đổi từ Class Diagram (YAML Contract) sang **Physical Database Schema** cho MongoDB/Payload CMS. Kết quả gồm **12 files** (6 Markdown + 6 YAML) với dual-format output.

Các design patterns áp dụng

Bảng 3.9: Design Patterns trong Schema Design

Pattern	Mô tả và ứng dụng
Aggregate Root	Collections tách riêng theo tần suất truy cập và tốc độ tăng trưởng (posts, comments, likes là aggregate riêng biệt)
Denormalization	Counter fields (likesCount, commentsCount, followerCount) lưu trực tiếp, cập nhật qua hooks
Computed Pattern	Các trường chỉ đọc (readOnly) được cập nhật bởi afterChange hooks
Polymorphic Pattern	Notifications và Reports sử dụng entityType + entityId để tham chiếu linh hoạt
Virtual Join Fields	Quan hệ một chiều, populated bởi Payload Local API depth expansion
Append-Only	AuditLogs không cho phép update/delete — đảm bảo audit trail vĩnh viễn
Compound Index	Unique constraints: (postId, userId) cho likes, (followerId, followingId) cho connections

Ví dụ: Collection Users (M1)

Bảng 3.10 minh họa cấu trúc chi tiết của collection `users` — collection cốt lõi của hệ thống, được thiết kế bởi Schema Design Analyst.

Bảng 3.10: Schema chi tiết collection `users` (M1)

Field Name	Payload Type	Mongo Type	Index	Ghi chú
email	email	String	Có	Unique, validated
username	text	String	Có	3–50 ký tự, unique
passwordHash	text	String	Không	bcrypt, private
verified	checkbox	Boolean	Không	default: false
role	select	String	Không	member, admin
status	select	String	Không	active, inactive, banned
profileVisibility	select	String	Không	public, followers, private
displayName	text	String	Không	Required
bio	text	String	Không	Max 160 chars
avatar	text	String	Không	URL hoặc Media ID
followerCount	number	Number	Không	Computed, readOnly
followingCount	number	Number	Không	Computed, readOnly
createdAt	date	ISODate	Có	Tự sinh
updatedAt	date	ISODate	Không	Tự sinh

Hệ thống Access Control

Schema Design định nghĩa **5 cấp quyền truy cập** cho mỗi collection:

- **guest**: Người dùng chưa xác thực — chỉ đọc dữ liệu công khai
- **member**: Người dùng đã xác thực — tạo nội dung, tương tác
- **owner**: Chủ sở hữu tài nguyên — sửa, xóa nội dung của mình
- **admin**: Quản trị viên — quản lý reports, ban users
- **system**: Thao tác nội bộ — cập nhật counters, hooks

Kiểm chứng chất lượng

Toàn bộ 6 modules schema đã vượt qua kiểm chứng bởi script `validate_schema.py`:

- 6/6 modules có Contract YAML hợp lệ
- 6/6 modules có Schema Markdown hoàn chỉnh
- 6/6 modules có Schema YAML chuẩn hóa
- **0 trường ảo giác** (hallucination) — mọi field đều có nguồn từ Contract YAML

3.2.8 Kết quả thiết kế giao diện người dùng

Đặc tả UI (UI Specs)

UI Architecture Analyst tạo ra **8 files đặc tả giao diện**, bao gồm 6 module specs (M1–M6), 1 template chuẩn và 1 file index. Mỗi UI Spec ánh xạ từ trường dữ liệu trong Schema sang thành phần giao diện thực tế (UI Components) theo nguyên tắc **Zero Hallucination** — không thêm UI field nếu không có trong Schema.

Cấu trúc mỗi UI Spec bao gồm:

- Danh sách màn hình (Screens) với mã định danh (ví dụ: `m1/register`, `m2/create-post`)
- Component Map — ánh xạ từng trường dữ liệu sang component tương ứng
- States — các trạng thái giao diện (default, loading, error, success, empty)
- Responsive breakpoints — mobile (375px), tablet (768px), desktop (1280px)

Bảng 3.11: Số lượng màn hình theo module trong UI Specs

Module	Tên spec	Số screens
M1	m1-auth-ui-spec.md	5
M2	m2-content-ui-spec.md	3
M3	m3-discovery-ui-spec.md	3
M4	m4-engagement-ui-spec.md	3
M5	m5-bookmarking-ui-spec.md	2
M6	m6-notifications-ui-spec.md	2
Tổng		**18**

Wireframe Blueprints

UI Pencil Drawer tạo ra **17 wireframe blueprints** chi tiết, bao gồm 15 screen-level wireframes và 2 device variants (desktop laptop). Mỗi wireframe tuân theo cấu trúc chuẩn:

- **Header**: Metadata (module, spec reference, layout, width, state)
- **Sections**: Phân vùng giao diện theo composition pattern
- **Components**: Danh sách component với `ref` ID từ Lib-Component, kèm `spec-cite` truy xuất nguồn
- **States**: Các trạng thái UI (default, error, loading, success)
- **Notes**: Ghi chú về inference và decisions

Bảng 3.12: Wireframe blueprints theo module

Module	Số WF	Các màn hình
M1	5	Register, Email Verification, Profile Setup, User Profile, Profile Edit
M2	2	Create Post, Post Detail/Edit
M3	2	News Feed, Discover/Trending
M4	2	Comment Thread, User Connections/Followers
M5	2	Bookmark Collections, Bookmarks List
M6	2	Notifications Feed, Reports Admin Panel
Laptop	2	Desktop Auth, Desktop Feed

Đặc biệt, mỗi component trong wireframe đều có thuộc tính `spec-cite` trỏ tới section cụ thể trong UI Spec (ví dụ: `[spec §2.1 - input-email]`). Guardrail **G-Spec-Strict** của UI Pencil Drawer đảm bảo không thêm element nào mà spec không đề cập.

3.3 Đánh giá hệ thống

3.3.1 Đánh giá về độ bao phủ (Coverage)

Bảng 3.13: Ma trận bao phủ: Module × Loại sơ đồ

Module	UC	AD	SD	FD	CD	Schema	UI
M1	✓	✓	✓	✓	✓	✓	✓
M2	✓	✓	✓	✓	✓	✓	✓
M3	✓	✓	✓	✓	✓	✓	✓
M4	✓	✓	✓	✓	✓	✓	✓
M5	✓	✓	✓	✓	✓	✓	✓
M6	✓	✓	✓	✓	✓	✓	✓

Chú thích: UC = Use Case, AD = Activity Diagram, SD = Sequence Diagram, FD = Flow Diagram, CD = Class Diagram.

Kết quả cho thấy **100% modules** (6/6) được bao phủ đầy đủ bởi **tất cả 7 loại artifacts**. Không có module nào bị thiếu sót về mặt thiết kế.

3.3.2 Đánh giá về tính nhất quán (Consistency)

Chuỗi truy xuất nguồn gốc (Traceability Chain) giữa các artifacts được đảm bảo xuyên suốt:

Requirements (FR-1 → FR-10)

↓

Use Case Diagrams (UC01 → UC24)

↓

Activity Diagrams (M1-A1 → M6-A3)

↓

Sequence & Flow Diagrams

↓

Class Diagrams (Mermaid + YAML Contract)

↓

Schema Design (Physical MongoDB Collections)

↓

UI Specs → Wireframes

Mỗi tầng xây dựng dựa trên tầng trước đó, tạo thành một chuỗi đặc tả → thiết kế → sẵn sàng triển khai có tính nhất quán cao. Cụ thể:

- **93 trường dữ liệu** trong Class Diagram đều có source citation tới ER Diagram hoặc Activity Diagram
- **24 luồng nghiệp vụ** trong Flow Diagram đều có UC-ID truy xuất tới Use Case
- **0 trường ảo giác** (hallucination) trong toàn bộ Schema Design — mọi field đều có nguồn từ Contract YAML
- Tên đối tượng trong Sequence Diagram khớp 100% với codebase dự kiến

3.3.3 Đánh giá về chất lượng kỹ thuật

Hiệu quả của Guardrails chống ảo giác

Hệ thống Guardrails đã chứng minh hiệu quả trong việc ngăn chặn hallucination:

Bảng 3.14: Hiệu quả Guardrails trong quá trình triển khai

Guardrail	Skill áp dụng	Kết quả
Source Citation	Class Diagram Analyst	93/93 fields có nguồn (100%)
Contract-Only	Schema Design Analyst	0 field ảo giác trong 6 modules
No Blind Step	Flow Design Analyst	100% action nodes có căn cứ từ spec
UC-ID Mapping	Flow Design Analyst	24/24 use cases được ánh xạ
Spec-Strict	UI Pencil Drawer	100% components có spec-cite

Các design patterns đã áp dụng thành công

Quá trình triển khai đã áp dụng thành công nhiều patterns quan trọng cho MongoDB/Payload CMS:

1. **Aggregate Root Strategy**: Tách riêng collections theo tần suất truy cập, tránh vi phạm giới hạn 16MB document
2. **Denormalized Counters**: Tối ưu hiệu năng đọc bằng cách lưu trực tiếp counter fields, cập nhật qua hooks
3. **Polymorphic References**: Hỗ trợ tham chiếu linh hoạt giữa các entity types khác nhau
4. **Append-Only Audit Trail**: Đảm bảo tính toàn vẹn của nhật ký hành chính
5. **Soft-Delete Strategy**: Sử dụng trường `status` thay vì xóa cứng, bảo toàn dữ liệu

3.3.4 Đánh giá về khả năng sẵn sàng cho giai đoạn triển khai (Life-3)

Toàn bộ artifacts thiết kế đã đạt trạng thái **sẵn sàng triển khai**:

- 13 MongoDB collections có schema chi tiết tới từng field, bao gồm kiểu dữ liệu, constraints, indexes và hooks
- 18 màn hình UI có wireframe blueprint với component ref IDs từ Lib-Component
- Dual-format output (Markdown + YAML) cho phép cả con người review lẫn AI Agent đọc để sinh code tự động
- Hệ thống access control 5 cấp đã được định nghĩa cho mọi collection
- Hooks specification (beforeChange, afterChange) đảm bảo tính nhất quán dữ liệu khi triển khai

3.3.5 Hạn chế và bài học kinh nghiệm

Trong quá trình triển khai, một số hạn chế đã được nhận diện:

1. **Phụ thuộc vào context window**: Với các module lớn (M1, M4), AI Agent đôi khi cần chia nhỏ task để tránh tràn context, ảnh hưởng đến tính liên tục của output
2. **Wireframe chưa bao gồm responsive đầy đủ**: Chỉ có 2/17 wireframes có device variant (laptop), các wireframe còn lại chủ yếu thiết kế cho mobile (375px)
3. **Chưa có automated testing**: Việc kiểm chứng chủ yếu dựa vào checklist thủ công và script `validate_schema.py`, chưa có bộ test tự động hoàn chỉnh

4. **Module M3 không tạo collection vật lý**: FeedQuery chỉ là ValueObject (DTO), không lưu xuống database — đây là quyết định thiết kế hợp lý nhưng cần lưu ý khi triển khai

Bài học rút ra: Hệ thống Guardrails và Interaction Points đã chứng minh vai trò then chốt trong việc duy trì chất lượng đầu ra. Nguyên tắc **Source Citation** và **Contract-Only** là hai cơ chế hiệu quả nhất để chống ảo giác AI trong quá trình thiết kế hệ thống.

KẾT LUẬN VÀ HƯỚNG PHÁT TRIỂN

3.4 Kết quả đạt được

Qua quá trình nghiên cứu và triển khai, khóa luận đã đạt được các kết quả chính sau:

1. **Xây dựng thành công mô hình Agent Skill Framework:** Đề xuất kiến trúc 3 Pillars (Knowledge, Process, Guardrails) kết hợp 7 Zones (Core, Knowledge, Scripts, Templates, Data, Loop, Assets) để tổ chức tri thức cho AI Agent một cách có hệ thống. Mô hình này cho phép AI hoạt động có kiểm soát, giảm thiểu hiện tượng ảo giác (hallucination) thông qua các cơ chế Source Citation, Contract-Only và Interaction Points.

2. **Phát triển bộ 10 Agent Skills chuyên biệt:** Bao gồm 3 Meta-Skills (Skill Architect, Skill Planner, Skill Builder) và 7 Domain Skills (Flow Design Analyst, Sequence Design Analyst, Activity Diagram Analyst, Class Diagram Analyst, Schema Design Analyst, UI Architecture Analyst, UI Pencil Drawer). Mỗi skill được thiết kế với workflow riêng, guardrails chống bịa đặt và cơ chế tự kiểm chứng chất lượng.

3. **Tạo ra hơn 120 artifacts thiết kế hoàn chỉnh cho hệ thống Steve Void:** Bao gồm 41 Use Cases, 27 Activity Diagrams, 9 Sequence Diagrams, 27 Flow Diagrams, 12 Class Diagrams (Mermaid + YAML), 1 ER Diagram với 13 collections, 12 Database Schema files (Markdown + YAML), 8 UI Specs (18 màn hình) và 17 Wireframe blueprints. Toàn bộ 6 modules (M1–M6) đều được bao phủ 100% bởi tất cả các loại artifacts.

4. **Đảm bảo tính nhất quán và truy xuất nguồn gốc:** Chuỗi truy xuất (Traceability Chain) từ Requirements → Use Cases → Activity

Diagrams → Sequence/Flow Diagrams → Class Diagrams → Schema Design → UI Specs → Wireframes được duy trì xuyên suốt. 93 trường dữ liệu trong Class Diagram đều có source citation, 0 trường ảo giác trong toàn bộ Schema Design.

5. **Chứng minh tính khả thi của việc ứng dụng AI Chatbot vào quy trình phát triển phần mềm:** Không chỉ ở mức hỗ trợ viết mã (code generation), AI Agent có thể tham gia hiệu quả vào các giai đoạn phân tích nghiệp vụ, thiết kế hệ thống và thiết kế giao diện — những công đoạn truyền thống đòi hỏi chuyên môn sâu và thường bị bỏ qua trong các công cụ AI hiện tại.

3.5 Hạn chế của nghiên cứu

Bên cạnh các kết quả đạt được, nghiên cứu vẫn tồn tại một số hạn chế cần được nhìn nhận khách quan:

1. **Hạn chế về kinh phí sử dụng tài nguyên AI:** Việc triển khai các Agent Skills đòi hỏi sử dụng API của các mô hình ngôn ngữ lớn (LLMs) với chi phí đáng kể. Giới hạn ngân sách khiến số lần thử nghiệm và tinh chỉnh bị hạn chế, ảnh hưởng đến độ hoàn thiện của một số skills. Ngoài ra, giới hạn context window của model cũng gây khó khăn khi xử lý các module lớn, đòi hỏi phải chia nhỏ task và giảm tính liên tục của output.

2. **Hạn chế về thời gian và kiến thức nghiệp vụ:** Do thời gian thực hiện khóa luận có giới hạn, kiến thức nghiệp vụ được đưa vào các Agent Skills (đặc biệt ở tầng Knowledge) mới chỉ đáp ứng ở mức độ cơ bản. Các bộ knowledge files chưa đủ sâu để bao phủ toàn bộ edge cases và best practices của từng lĩnh vực (UML standards, MongoDB optimization, UI/UX patterns). Điều này dẫn đến chất lượng đầu ra của

skills tuy đạt yêu cầu cho nghiên cứu nhưng chưa đủ tin cậy để đưa lên production.

3. **Chưa hoàn thành giai đoạn Life-3 (Implementation):** Nghiên cứu mới dừng lại ở giai đoạn Life-2 (Design & Specification). Hệ thống Steve Void chưa được triển khai thành sản phẩm thực tế với mã nguồn chạy được, do đó chưa thể đánh giá đầy đủ tính khả dụng của các artifacts thiết kế trong thực tiễn lập trình.

4. **Wireframe chưa bao phủ đầy đủ responsive:** Trong 17 wireframe blueprints, chỉ có 2 bản có device variant cho desktop (laptop). Phần lớn wireframes được thiết kế cho mobile-first (375px), chưa có bản thiết kế riêng cho tablet và desktop đầy đủ.

5. **Chưa có đánh giá định lượng từ người dùng thực tế:** Việc đánh giá chất lượng chủ yếu dựa vào checklist, script validation và self-review. Chưa thực hiện được khảo sát người dùng hoặc so sánh thực nghiệm có đối chứng (controlled experiment) giữa phát triển có và không có Agent Skills.

3.6 Hướng phát triển trong tương lai

Dựa trên kết quả và hạn chế của nghiên cứu, các hướng phát triển tiếp theo được đề xuất:

1. **Tự động hóa hoàn toàn pipeline Life-1 → Life-2 (Fully Autonomous Execution):** Đây là hướng phát triển trọng tâm. Mục tiêu là xây dựng một hệ thống mà người dùng chỉ cần nạp tài liệu đầu vào ở giai đoạn Life-1 (Vision, User Stories, Technical Decisions), AI Agent sẽ tự động thực thi toàn bộ pipeline Life-2 — từ vẽ Use Case, Activity Diagram, Sequence Diagram, Flow Diagram, Class Diagram, Schema Design đến

UI Specs và Wireframes — **mà không cần sự can thiệp của con người**. Các Interaction Points hiện tại sẽ được chuyển thành auto-approval với confidence threshold, chỉ escalate khi AI phát hiện rủi ro hoặc mâu thuẫn.

2. **Mở rộng sang Life-3 (Automated Code Generation):** Kết nối pipeline thiết kế với giai đoạn sinh mã tự động. Từ Schema YAML → Payload CMS collections, từ UI Specs → React components (Tailwind v4 + Radix UI), từ Sequence Diagrams → API route handlers. Mục tiêu cuối cùng là một pipeline end-to-end: Ý tưởng → Tài liệu → Thiết kế → Mã nguồn → Kiểm thử.

3. **Nâng cao chất lượng Knowledge Base:** Bổ sung và tinh chỉnh các knowledge files trong từng Agent Skill với kiến thức chuyên sâu hơn: MongoDB advanced patterns (sharding, aggregation pipelines), UML compliance standards (OMG specifications), UI/UX best practices từ Nielsen Norman Group. Điều này giúp nâng chất lượng đầu ra từ mức cơ bản lên mức production-ready.

4. **Tích hợp đa mô hình AI (Multi-Model Architecture):** Thay vì phụ thuộc vào một mô hình duy nhất (Claude), nghiên cứu hướng tới việc tích hợp nhiều mô hình AI khác nhau (GPT-4, Gemini, open-source models) cho từng loại task cụ thể, nhằm tối ưu chi phí và tận dụng thế mạnh riêng của mỗi mô hình.

5. **Xây dựng hệ thống đánh giá tự động (Automated Quality Assessment):** Phát triển bộ test tự động để đánh giá chất lượng đầu ra của Agent Skills: kiểm tra tính đúng đắn của sơ đồ (syntax validation), tính nhất quán giữa các artifacts (cross-reference checking), và so sánh với benchmark từ các dự án thực tế.

6. **Đóng gói thành sản phẩm mã nguồn mở:** Xuất bản Agent Skill

Framework dưới dạng thư viện mã nguồn mở, kèm theo tài liệu hướng dẫn và bộ skills mẫu, để cộng đồng lập trình viên Việt Nam có thể tái sử dụng và đóng góp phát triển.

Tóm lại, nghiên cứu đã chứng minh được tiềm năng to lớn của việc ứng dụng AI Chatbot trong phát triển phần mềm, đặc biệt ở giai đoạn phân tích và thiết kế. Mặc dù còn những hạn chế về kinh phí, thời gian và độ sâu nghiệp vụ, kết quả đạt được đã đặt nền móng vững chắc cho một hướng đi đầy triển vọng — nơi AI không chỉ là công cụ hỗ trợ mà trở thành **cộng tác viên thực sự** trong quy trình phát triển phần mềm.

TÀI LIỆU THAM KHẢO

TÀI LIỆU THAM KHẢO

[1] Andrew Ng, ”Agentic Workflow and AI Agents”, DeepLearning.AI, 2024.

[2] Next.js Documentation, URL: `https://nextjs.org/docs`, truy cập tháng 02/2026.

[3] Payload CMS Documentation, URL: `https://payloadcms.com/docs`, truy cập tháng 02/2026.

[4] MongoDB Atlas Documentation, URL: `https://www.mongodb.com/docs/atlas`, truy cập tháng 02/2026.

[5] Radix UI Documentation, URL: `https://www.radix-ui.com/docs`, truy cập tháng 02/2026.

[6] Anthropic, ”Building effective agents”, URL: `https://www.anthropic.com/research/building-effective-agents`, truy cập tháng 02/2026.

[7] Stuart Russell, Peter Norvig, *Artificial Intelligence: A Modern Approach*, 4th Edition, Pearson, 2021.

[8] Ashish Vaswani et al., ”Attention Is All You Need”, *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[9] Anthropic, ”Claude Code — CLI for Claude”, URL: `https://docs.anthropic.com/en/docs/claude-code`, truy cập tháng 02/2026.

[10] Object Management Group (OMG), ”Unified Modeling Language Specification, Version 2.5.1”, URL: `https://www.omg.org/spec/UML`, 2017.

[11] Mermaid.js Documentation, URL: `https://mermaid.js.org/intro/`, truy cập tháng 02/2026.

[12] Roger S. Pressman, Bruce R. Maxim, *Software Engineering: A Practitioner's Approach*, 9th Edition, McGraw-Hill Education, 2020.

PHỤ LỤC

.1 Phụ lục A: Cấu trúc thư mục Agent Skill

Mỗi Agent Skill được tổ chức theo cấu trúc 7 Zone thống nhất. Dưới đây là ví dụ minh họa với skill `sequence-design-analyst`:

```
sequence-design-analyst/
|-- SKILL.md                        # Zone 1: Core Persona
|                                   # (Persona, Workflow, Guardrails)
|-- knowledge/                      # Zone 2: Domain Knowledge
|   |-- uml-rules.md               # Chuan UML & cu phap Mermaid
|   |-- project-patterns.md        # Kien truc du an PayloadCMS
|   +-- sequence-preparation.md    # Huong dan phan tich kich ban
|-- templates/                      # Zone 3: Output Templates
|   |-- auth-flow.mmd              # Mau luong xac thuc
|   |-- crud-flow.mmd              # Mau luong CRUD
|   +-- logic-flow.mmd             # Mau luong logic nghiep vu
|-- loop/                           # Zone 4: Quality Control
|   |-- checklist.md               # Danh sach kiem tra chat luong
|   +-- verify-script.py           # Script tu dong xac minh
+-- scripts/                        # Zone 5: Automation Scripts
```

Cấu trúc này đảm bảo mỗi skill hoạt động như một đơn vị độc lập, tự chứa đầy đủ kiến thức miền, quy trình làm việc và cơ chế kiểm soát chất lượng.

Phụ lục B: Trích mẫu SKILL.md — Sequence Design Analyst

Dưới đây là phần trích dẫn đầu file `SKILL.md` của skill phân tích Sequence Diagram, minh họa cách định nghĩa Persona, Boot Sequence và Workflow:

```
---
name: sequence-design-analyst
description: Chuyên gia phân tích và thiết kế Sequence Diagram
  (UML) chuẩn Mermaid. Tự động nghiên cứu codebase để
  đảm bảo tính thực tế.
---
# Sequence Design Analyst

- **Persona:** Senior UML Architect. Thiết kế Sequence
  Diagram phản ánh 100% logic triển khai thực tế.

## Mandatory Boot Sequence
1. Read knowledge/uml-rules.md
2. Read knowledge/project-patterns.md
3. Read knowledge/sequence-preparation.md
4. Read loop/checklist.md

## Workflow: Execution Phases

### Phase 1: Scenario Discovery
- Identify Main Scenario, Actors, Lifelines
- [INTERACTION POINT] - Chờ xác nhận kịch bản
```

```
### Phase 2: Codebase Research
- Search relevant methods in Services/Collections
- Build call chain using Project Patterns
- [INTERACTION POINT] - Xác nhận method flow

### Phase 3: Message Flow Design
- Structure findings into logical sequence
- Plan fragments: alt, loop, opt
- Use actual method names (e.g., payload.find)

### Phase 4: Generation
- Render Mermaid code theo chuẩn templates/
- Verify against loop/checklist.md
```

Phụ lục C: Cấu trúc sản phẩm đầu ra Life-2

Toàn bộ sản phẩm đầu ra của giai đoạn Life-2 (Thiết kế) được tổ chức theo cấu trúc sau:

```
Docs/life-2/
|-- api/
|   +-- api-spec.md              # Dac ta API (42 endpoints)
|-- database/
|   +-- schema-design/           # Thiet ke CSDL
|       |-- schema-m1.md + .yaml # M1: Auth & Profile
|       |-- schema-m2.md + .yaml # M2: Content Engine
|       |-- schema-m3.md + .yaml # M3: Discovery & Feed
|       |-- schema-m4.md + .yaml # M4: Engagement
|       |-- schema-m5.md + .yaml # M5: Bookmarking
```

```
|       +-- schema-m6.md + .yaml # M6: Notifications
|-- diagrams/
|   |-- activity-diagrams/       # 24 Activity Diagrams
|   |-- class-diagrams/          # 7 Class Diagrams
|   |-- flow/                    # 11 Flow Diagrams
|   |-- sequence-diagrams/       # 24 Sequence Diagrams
|   +-- UseCase/                 # 6 Use Case Diagrams
|-- specs/
|   |-- m1-auth-profile-spec.md  # Spec module M1
|   |-- m2-content-engine-spec.md # Spec module M2
|   |-- m3-discovery-feed-spec.md # Spec module M3
|   |-- m4-engagement-spec.md    # Spec module M4
|   |-- m5-bookmarking-spec.md   # Spec module M5
|   +-- m6-notifications-spec.md # Spec module M6
+-- ui/
    |-- specs/                   # 8 UI Spec files
    +-- wireframes/              # 27 Wireframe files
```

Tổng cộng 122 tệp sản phẩm được sinh tự động bởi hệ thống Agent Skill, bao gồm cả định dạng Markdown (cho con người) và YAML (cho máy đọc).

Phụ lục D: Mã nguồn Mermaid mẫu

D.1. Class Diagram — Collection User (M1)

```
classDiagram
    namespace AuthProfile {
        class User {
            <<Collection>>
```

```
            +ObjectId id
            +String email
            +String username
            -String passwordHash
            +Boolean verified
            +String role
            +String status
            +String profileVisibility
            +String displayName
            +String bio
            +String avatar
            +Number followerCount
            +Number followingCount
            +DateTime createdAt
            +DateTime updatedAt
            +void beforeChange_sanitizeInput()
            +void afterChange_sendVerification()
        }
    }
```

D.2. Sequence Diagram — Luồng Đăng ký tài khoản

```
sequenceDiagram
    actor User
    participant RegisterPage
    participant AuthService
    participant Mailer
    participant Payload
    participant DB
```

```
User->>RegisterPage: Điền Form (Email, Password)
RegisterPage->>AuthService: register(data)
activate AuthService
AuthService->>Payload: payload.find({
    collection: 'users', email })
Payload-->>AuthService: { docs: [] }

alt Email đã tồn tại
    AuthService-->>RegisterPage: { error }
    RegisterPage-->>User: Hiển thị lỗi
else Email chưa tồn tại
    AuthService->>Payload: payload.create({
        collection: 'users', data })
    activate Payload
    Payload->>DB: insertUser
    DB-->>Payload: userObj
    Payload-->>AuthService: userObj
    deactivate Payload
    AuthService->>Mailer: sendVerificationEmail()
    AuthService-->>RegisterPage: { success: true }
    deactivate AuthService
    RegisterPage-->>User: Yêu cầu kiểm tra email
end
```

Phụ lục E: Danh sách đầy đủ Agent Skill

Bảng dưới đây liệt kê toàn bộ 31 Agent Skill được xây dựng trong nghiên cứu:

STT	Tên Skill	Chức năng
Nhóm Meta-Skill (Quản lý vòng đời skill)		
1	skill-architect	Thiết kế kiến trúc skill mới
2	skill-planner	Lập kế hoạch triển khai skill
3	skill-builder	Triển khai skill từ thiết kế
4	skill-creator	Hướng dẫn tạo skill hiệu quả
5	master-skill	Điều phối toàn bộ quy trình skill
Nhóm UML & Diagram		
6	sequence-design-analyst	Thiết kế Sequence Diagram
7	activity-diagram-design-analyst	Thiết kế Activity Diagram
8	flow-design-analyst	Thiết kế Flow Diagram
9	class-diagram-analyst	Thiết kế Class Diagram
Nhóm Schema & Database		
10	schema-design-analyst	Thiết kế Database Schema
Nhóm UI/UX		
11	ui-architecture-analyst	Phân tích UI Spec từ Schema
12	ui-pencil-drawer	Vẽ wireframe trên Pencil canvas
Nhóm OpenSpec (Quản lý thay đổi)		
13	openspec-new-change	Tạo change request mới
14	openspec-continue-change	Tiếp tục artifact tiếp theo
15	openspec-ff-change	Fast-forward tạo artifacts
16	openspec-apply-change	Triển khai tasks từ change
17	openspec-verify-change	Xác minh implementation
18	openspec-sync-specs	Đồng bộ delta specs
19	openspec-archive-change	Lưu trữ change hoàn thành
20	openspec-bulk-archive-change	Lưu trữ hàng loạt
21	openspec-explore	Khám phá ý tưởng
22	openspec-onboard	Hướng dẫn onboarding

STT	Tên Skill	Chức năng
Nhóm Implementation		
23	build-crud-admin-page	Xây dựng trang CRUD admin
24	api-from-ui	Tạo API endpoint từ UI
25	api-integration	Tích hợp API vào frontend
26	error-response-system	Chuẩn hóa Error Response
27	screen-structure-checker	Kiểm tra cấu trúc screen
Nhóm Tiện ích		
28	task-planner	Phân tách yêu cầu thành tasks
29	prompt-improver	Tối ưu hóa prompt
30	typescript-error-explainer	Giải thích lỗi TypeScript
31	latex-report-specialist	Viết báo cáo LaTeX