# Chương 6: Sử dụng AI agent skill

chương này hướng dẫn cách hoạt động và sử dụng AI để thực hiện các tác vụ về tạo mới api

# Nội dung chi tiết

## 1. Cài đặt ban đầu — Clone thư mục .agent

Trước khi sử dụng skill, bạn cần clone repository cấu hình về thư mục gốc của project bằng lệnh sau:

git clone -b master http://192.168.0.95/minhnd/chat-gpt-agent.git .agent

Lệnh trên sẽ tải branch master và đặt toàn bộ nội dung vào thư mục .agent ngay trong thư mục hiện tại của bạn.

Lưu ý: Chạy lệnh này tại thư mục gốc (root) của project, không phải bên trong thư mục con nào khác.

## 2. Hướng dẫn cài đặt Google Antigravity Truy cập link sau https://antigravity.google/download

Sau đó cài đặt như bình thường.

## 3. Mục tiêu của Skill

Skill này được thiết kế để:

• Chuẩn hóa cấu trúc dự án Spring Boot.

• Đảm bảo tính nhất quán (Consistency) giữa các module.

• Tự động hóa việc ra quyết định về kiến trúc (Layered Architecture).

• Tuân thủ các quy tắc bảo mật và lưu vết (Audit log).

# 4. Cách kích hoạt (Activation)

Skill này hoạt động hoàn toàn TỰ ĐỘNG khi project có thư mục cấu hình .agent.

Cơ chế: Khi bạn gửi yêu cầu (ví dụ: "Tạo API User"), hệ thống nhận diện .agent, kích hoạt @backend-specialist và tự động áp dụng các quy chuẩn trong skill api-processing-flow.

Lưu ý: Bạn không cần copy nội dung skill vào chatbot, chỉ cần đảm bảo thư mục .agent nằm trong project.

# 5. Các Quy tắc "Vàng" (Mandatory Rules)

Cần tuân thủ 5 quy tắc bắt buộc sau khi viết code:

1. Kiến trúc 3 lớp: Luôn tuân theo luồng Controller -> Service -> Repository.

2. Không Logic ở Controller: Controller chỉ nhận request, check validation cơ bản và gọi Service.

3. ApiResponse Standard: Mọi API phải trả về object ApiResponse<T>. Không trả về trực tiếp DTO hay Entity.

4. DTO vs Entity: Tuyệt đối không trả về Entity cho Client. Luôn dùng DTO cho Request và Response.

5. In-memory Storage: Dùng ConcurrentHashMap hoặc List trong Repository nếu chưa có Database bên ngoài.

5b. Cấu trúc Module chuẩn

Mỗi module mới khi tạo ra phải có cấu trúc thư mục như sau:

modules/<module-name>/

+-- controller/ (Chua @RestController)

+-- service/ (Chua Business Logic & Interface)

+-- repository/ (Chua truy van du lieu)

+-- entity/ (Chua mapping du lieu)

+-- dto/ (Chua Request/Response models)

# 6. Quy trình phát triển 5 bước (API Workflow)

Để tạo một tính năng mới, hãy đi theo trình tự:

6. B1: Tạo DTO: Định nghĩa Request và Response model.

7. B2: Tạo Entity: Định nghĩa cấu trúc dữ liệu mapping.

8. B3: Tạo Repository: Định nghĩa các phương thức lưu trữ dữ liệu.

9. B4: Tạo Service: Viết Business Logic (Xử lý dữ liệu, chuyển đổi Entity <-> DTO).

10. B5: Tạo Controller: Khai báo Endpoint và gọi Service.

# 7. Coding Convention (Quy tắc đặt tên)

Bảng quy ước đặt tên các thành phần trong dự án:

Thành phần Quy tắc đặt tên Ví dụ

Controller <n>Controller UserController

Service <n>Service UserService

Repository <n>Repository UserRepository

Entity <n>Entity UserEntity

DTO Request <n>CreateRequest UserCreateRequest

DTO Response <n>Response UserResponse

# 8. Ví dụ một phương thức Service chuẩn

Dưới đây là cấu trúc chuẩn của một phương thức trong Service layer:

```
public UserResponse create(UserCreateRequest request) {

// 1. Validate (Kiem tra du lieu)
if (repo.existsByUsername(request.getUsername())) {
throw new BusinessException(ErrorCode.USER_EXISTS);
}

// 2. Log (Luu vet hanh dong)
log.info("Creating user: {}", request.getUsername());

// 3. Business logic (Xu ly)
UserEntity entity = mapper.toEntity(request);
repo.save(entity);

// 4. Return (Tra ve ket qua)
return mapper.toResponse(entity);
}
```

# 10. Cơ chế hoạt động

Thư mục .agent (Antigravity Kit) đóng vai trò là "bộ não" và hệ thống điều hành của trợ lý AI trong dự án. Nó chứa các định nghĩa về vai trò, kỹ năng, quy trình làm việc và các kịch bản tự động hóa đảm bảo chất lượng code.

# 11. Cấu trúc thư mục .agent

Cấu trúc chuẩn của thư mục .agent:

```
.agent/
```

+-- rules/

| +-- GEMINI.md # Quy tắc tối cao(P0), định nghĩa hành vi cơ bản

+-- agents/ # Danh sách 20 Agent(Personas)

+-- skills/ # Các kỹ năng mo-dun hóa (Domain knowledge)

+-- workflows/ # Quy trình làm việc (/plan, /create, /debug...)

+-- scripts/ # Các kịch bản kiểm tra và tự động hóa master

+-- ARCHITECTURE.md # Bản đồ tổng thể hế thống Agent

+-- tasks/ # Lưu trữ tài liệu và các tác vụ({task-slug}.md)

# 12. Luồng xử lý — Từ yêu cầu đến code

Mỗi yêu cầu gõ vào chạy qua đúng 6 cổng theo thứ tự sau. Không cổng nào bị bỏ qua:

[USER INPUT]

|

v

[1] Request Classifier <- rules/GEMINI.md doc vao dau tien

| QUESTION --> trả lời thẳng,kết thúc.

| SIMPLE CODE --> chuyển sang bước 2.

| COMPLEX --> tạo tasks/{task-slug}.md rồi mới sang bước 2.

v

[2] Agent Routing <- đọc agents/ chọn đúng agent

| ví dụ: viết API --> @backend-specialist

| viết UI --> @frontend-specialist

| kiểm tra sec --> @security-auditor

v

[3] Modular Skill Loading <- chỉ đọc skills/ cần thiết, không đọc hết

| ví dụ: @backend + API --> skills/api-patterns + nodejs-best-practices

| mọi folder Skill gồm: SKILL.md (hướng dẫn) + scripts/ (công cụ)

v

[4] Rule Priority Check <- áp dụng theo thứ tự P0 > P1 > P2

| P0: rules/GEMINI.md (Toàn cục -cao nhất, không thể override)

| P1: agents/{agent}.md (quy tắc riêng của Specialist)

| P2: skills/{skill}.md (kỹ thuật cụ thể cửa Skill được nạp)

v

[5] Socratic Gate <- chỉ áp dụng với COMPLEX CODE

| AI hỏi >= 3 câu trước khi viết bất kỳ dòng code nào:

| - Mục tiêu chính là gì?

| - Edge cases nào cần xử lý?

| - Ràng buộc kiến trúc/công nghệ?

v

[6] Implementation & Validation <- viết code + tự động kiểm tra

| Viết code: Clean Code + TDD

| scripts/checklist.py --> Bảo mật, Lint, Schema, UX (nhanh)

| scripts/verify_all.py --> E2E, Performance, SEO (truoc deploy)

v

[OUTPUT] Code đạt chuẩn, đã quan kiểm tra, sẵn sàng review.

# 13. Các lệnh Slash Commands phổ biến

Hệ thống hỗ trợ các quy trình chuẩn hóa thông qua lệnh /. Gõ trực tiếp trong chat để kích hoạt:

/plan — Lập kế hoạch chi tiết, chia nhỏ tác vụ (không viết code).

/create — Bắt đầu tạo một tính năng hoặc ứng dụng mới.

/debug — Kích hoạt chế độ kiểm tra lỗi hệ thống 4 giai đoạn.

/orchestrate — Phối hợp nhiều chuyên gia để giải quyết vấn đề đa chiều.

/status — Kiểm tra tiến độ và trạng thái hiện tại của dự án.

Ghi chú: Mọi thay đổi đối với cấu trúc hệ thống Agent nên được thực hiện thông qua việc cập nhật các file trong .agent để đảm bảo AI luôn cập nhật được “tư duy” mới nhất.