Autonomous Development Agent System

Claude Code + RAG + Agent Teams 기반 자율 개발 에이전트

Portfolio Project

Problem & Motivation

왜 이 시스템이 필요한가

반복적 수작업

빌드 실패 → 수동 디버깅 → 수정 → 재빌드 반복

일관성 없는 코드

개발자마다 다른 패턴, 네이밍, 구조 사용

불충분한 테스트

Edge case 누락, 경계값 테스트 부재

문서화 지연

코드 완성 후 문서는 항상 후순위

사람 + AI 협업 구조

기획 · 설계 · 디자인은 사람과 함께, 구현 · 검증은 에이전트가 자율 수행

사람이 주도

에이전트와 대화하며 결정

- 기획 · 요구사항 정의
- UI/UX 디자인 방향 결정
- 아키텍처 · 기술 스택 논의
- 스펙 최종 확정 (spec.md)

에이전트가 자율 수행

스펙 확정 후 100% 완성까지 무한 반복

- 프로젝트 구성 · 코드 구현
- 테스트 작성 · 실행 · 수정
- 대량 QC (1만건 · 10만건)
- 문서 자동 생성 (README, API)

Solution Overview

4가지 핵심 기술 축

Claude API

Orchestrator 두뇌
계획 수립, 품질 판단
이슈 분류

Agent SDK

실행 계층 손발
코드 작성, 테스트
빌드 수행

RAG (Skills)

코드 품질 보장
일관된 디자인 패턴
강제

Agent Teams

6개 전문 에이전트
역할 분담
병렬 협업

System Architecture

Human → Orchestrator → Agent Teams → RAG/Code 계층 구조

Human — 기획 · 설계 · 디자인 논의 후 스펙 확정, 크리티컬 이슈 응답

Orchestrator (Claude API)

Planner · Issue Classifier · Token Manager · State Manager

Agent Teams (Agent SDK)

Architect · Coder · Tester · Reviewer · QC · Documenter

RAG Skills (6개 SKILL.md)

Codebase + Tests + Docs

Agent Teams

6개 전문 에이전트 역할 분담

Architect

Opus

설계, 구조 결정
API 설계, 모듈 분리

Coder

Sonnet

코드 구현
Skills 참조, 패턴 준수

Tester

Sonnet

유닛/통합 테스트
100% 통과 목표

Reviewer

Sonnet

코드 리뷰
보안, 성능, 패턴 검증

QC

Sonnet

대량 QC 테스트
모듈 1만건, E2E 10만건

Documenter

Sonnet

전체 문서 생성
README, API, 아키텍처

RAG — Skills 기반 코드 품질 보장

Claude Code가 자동 발견 · 참조하는 코딩 규칙 시스템

Design Patterns

Layered Architecture

Repository · DTO · DI

Code Standards

네이밍 규칙 · 함수 규칙

import 순서 · 주석 규칙

Testing Strategy

AAA 패턴 · 테스트 네이밍

필수 케이스 정의

Error Handling

커스텀 에러 계층

재시도 패턴 · 레이어별 처리

Project Architecture

기술 스택 · 모듈 의존성

새 파일 체크리스트

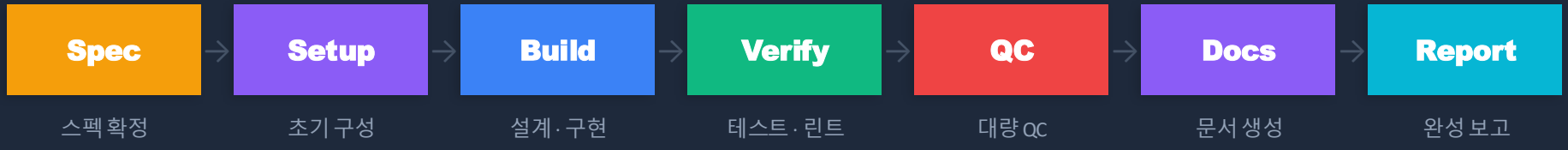
RAG Search

기존 코드에서 유사 패턴

검색하여 일관성 유지

Autonomous Loop

스펙 확정 후 무중단 자율 개발 루프



자율 해결

빌드/테스트 실패 → 사람에게 넘기지 않고 스스로 수정 반복

크리티컬만 질문

스펙 모호, 외부 연동 정보, 스펙 간 모순, 보안 결정만 사람에게

토큰 자동 복구

한도 도달 → 지수 백오프 대기 → 리셋 후 이어서 진행

상태 저장

state.json에 진행 상황 저장 → 재시작 시 이어서 진행

QC Agent — 대량 테스트 자동화

인풋/아웃풋 세트 대량 생성 → 실행 → 실패 시 자동 수정

Module QC

10,000건

모듈 하나 완성될 때마다 실행

Normal 30% · Boundary 20%

Invalid 20% · Stress 15% · Random 15%

E2E QC

100,000건

기능 하나 완성될 때마다 실행

Normal 25% · Sequence 15%

Concurrent 10% · Stress 10% · Random 10%

실패 시 자동 수정 루프

실패 패턴 클러스터링 → root cause 추정 → fix_requests 생성 → Coder에 수정 지시 → QC 재실행 → 100% 될 때까지 무한 반복

API 분업 구조

Claude API (두뇌) + Agent SDK (손발) 역할 분리

Claude API

"두뇌" — Orchestrator의 판단 엔진

- 다음 작업 계획 수립
- 이슈 critical/non-critical 분류
- 코드 품질 완성도 판단
- 작업 우선순위 결정

Agent SDK

"손발" — 실제 코드 수정/실행 계층

- 파일 읽기/쓰기/수정
- 테스트 실행, 빌드 수행
- 린트/타입체크 실행
- Git 커밋/브랜치 관리

Token 관리 & Self-Healing

토큰 한도 · 에러 발생 시에도 중단 없는 연속 실행

Token Manager

1. API 호출 시 토큰 사용량 추적
2. Rate Limit 감지 (429 응답)
3. 지수 백오프 대기 (60s → 120s → 240s → 300s)
4. API 가용성 테스트 (health check)
5. 성공 시 작업 이어서 진행

Self-Healing Loop

1. 예상치 못한 에러 → _self_heal() 호출
2. 에러 분석 → 자동 수정 시도
3. state.json에서 마지막 성공 지점 복원
4. run.sh: 프로세스 종료 시 자동 재시작
5. 최대 500 iteration · 50회 재시작

Tech Stack & Structure

프로젝트 구성 요소

Tech Stack

AI/LLM	Claude API · Agent SDK · Agent Teams
RAG	Skills (SKILL.md) · ChromaDB · MCP Server
언어/런타임	Python 3.12+ · asyncio · Pydantic
테스트	pytest · ruff · mypy · QC Generator
인프라	tmux · bash scripts · JSON state

Project Files

32+ 파일 구성

2	CLAUDE.md + Settings
6	Skills (SKILL.md)
6	Agent Definitions (.md)
4	Orchestrator (Python)
4	Agents / Utils (Python)
10+	Scripts / Config / etc

Thank You

Q & A

Autonomous Development Agent System · 구현 중