바이브 코딩**(Vibe Coding)** 프로젝트를 위한 지능형 문서 관리 및 구조화 전략

소프트웨어 개발이 AI 주도형 '바이브 코딩'으로 전환되면서, 마크다운(Markdown) 기반의 문서는 단순한 기록을 넘어 시스템의 동작을 제어하고 AI의 기억을 유지하는 **'실행 가능한 사양(Executable Specification)'**으로 진화하고 있습니다.[1] 특히 타입스크립트 기반 프로젝트에서는 강력한 타입 시스템과 구조화된 문서를 연계하여 AI의 환각을 최소화하고 생산성을 극대화할 수 있습니다.[4]

1. 지능형 프로젝트 디렉토리 및 문서 구조 **(The Golden Repository)**

AI가 최소한의 토큰으로 전체 맥락을 파악하고 작업을 수행하도록 돕는 'AI 친화적' 구조를 제안합니다. 프로젝트 루트의 AGENTS.md를 기점으로 계층화된 문서 체계를 구축하는 것이 핵심입니다.[6]

```
/root
├── AGENTS.md (또는 AGENT.md) : 프로젝트의 헌법. 기술 스택, 규칙, 금지사항 정의 6
├── CLAUDE.md : Claude Code 전용 지침 및 메모리 뱅크 링크
├──.docs/ (또는.ai/) : AI 전용 지식 저장소
│  ├── PRD.md : 제품 요구사항 및 비전 10
│  ├── SPEC.md : 기술적 사양 및 아키텍처 결정 4
│  ├── MEMORY.md : 장기 의사결정 기록 (Memory Bank)
│  ├── CONTEXT.md : 현재 작업 상태 및 세션 정보
│  └── lessons_learned.md : 재발 방지를 위한 실패/성공 패턴 기록 7
├──.docs/design-system/ (WDS 특화)
│  ├── tokens.md : 디자인 토큰 정의 및 명명 규칙
│  ├── figma-mcp-rules.md : 피그마 레이어-코드 매핑 규칙
│  └── design-system-rules.md : 컴포넌트 사용 가이드라인
├── plans/
│  └── plan_<feature_name>.md : 기능별 단계적 구현 계획서 7
├── tickets/
│  └── ticket-.md : 개별 작업 단위의 수락 기준(AC) 및 로그 7
└── src/ (타입스크립트 소스 코드)
```

2. 바이브 코딩 문서별 베스트 프랙티스 포맷

2.1. PRD 및 **SPEC.md:** 의도의 명확화

AI는 모호한 명령보다 구조화된 사양에서 성능이 비약적으로 향상됩니다.[4]

- **PRD:** 프로젝트 비전, 사용자 페인 포인트, 핵심 기능 리스트, 기술 스택, 성공 지표를

포함합니다.[10]

- **SPEC.md:** 구현 전 단계에서 AI가 '인터뷰'를 통해 작성하게 하며, 데이터 모델(Zod/TypeScript Interface), API 컨트랙트, 엣지 케이스를 명시합니다.[4]

2.2. 계획 및 작업 관리: 수직적 슬라이스 (Vertical Slice)

대규모 작업을 한 번에 지시하지 않고, AI가 한 번에 완결할 수 있는 작은 단위로 쪼개어 관리합니다.[15]

- **Plan:** "인증 시스템 개발" 대신 "로그인 폼 컴포넌트", "이메일 유효성 검사 로직", "JWT 엔드포인트 생성" 등으로 세분화합니다.
- **Ticket:** 각 티켓 문서에는 수락 기준(Acceptance Criteria)을 체크리스트 형태로 작성하여 AI가 스스로 검증(Self-Verification)하게 합니다.[7]

2.3. 회고 및 전역 인사이트: lessons_learned.md

세션 간의 문맥 소실을 막고 동일한 실수를 방지하는 '프로젝트 기억력' 시스템입니다.[19]

- **lessons_learned.md:** "d_proj=32 옵션은 정보 손실을 유발하므로 64 이상 사용"과 같은 구체적인 실패 패턴과 해결책을 3~5개 불렛으로 기록합니다.
- **SKILL.md:** 반복되는 작업 패턴(예: 특정 라이브러리 설정 방식)을 모듈화하여 저장하고, AI가 필요 시 /advise 명령으로 호출하게 합니다.[6]

3. 디자인 시스템(WDS) 구축을 위한 핵심 문서화 항목

Phase 2 자동화를 대비하여 Phase 1에서 정돈해야 할 핵심 요소들입니다.

항목	문서 내용 및 목적	AI 활용 전략
디자인 토큰 (Tokens)	색상, 타이포그래피, 간격, 그림자 값 정의	JSON/CSS 형식으로 관리하여 코드로 즉시 변환[18]
컴포넌트 명세 (Spec)	Anatomy, Props 인터페이스, Variants(Default, Hover, Active)	JSDoc 및 Zod 스키마와 연계하여 타입 안전성 확보[22]
레이어 컨벤션	피그마 오토 레이아웃 규칙 및 레이어 네이밍 (BEM 등)	figma-mcp-rules.md에 정의하여 1:1 코드 변환 유도
접근성 및 정책	WCAG 기준, 상태별 디자인	AI가 코드 생성 시 접근성

	정책, 사용 가이드	속성(Aria-label 등)을 자동 포함

특히 피그마의 **오토 레이아웃(Auto Layout)**은 AI가 레이아웃 구조를 Flexbox/Grid로 해석하는 결정적인 단서가 되므로 반드시 적용되어야 합니다.

4. 바이브 코딩 워크플로우 및 자동화 전략

4.1. 플랜 모드(Plan Mode) 활용

코드를 직접 수정하기 전에 Claude Code의 plan mode를 사용하여 아키텍처와 구현 경로를 먼저 승인받습니다.[17]

- 명령어: claude --permission-mode plan
- 이 단계에서 AI는 코드베이스를 분석하고 plan_*.md를 업데이트하며, 인간 개발자는 이 '설계도'를 검토하고 보정합니다.[17]

4.2. MCP(Model Context Protocol) 및 툴 연동

- **Figma MCP:** 피그마의 레이어 속성, 스크린샷, 노드 정보를 직접 쿼리하여 디자인 의도를 코드로 실시간 변환합니다.
- **Zod Interface Automation:** AI가 JSON 스키마나 API 응답 샘플(Fixtures)을 읽어 타입스크립트 인터페이스와 런타임 유효성 검사기(Zod)를 자동으로 생성하게 합니다.
- **Context Manager MCP:** 프로젝트의 전역 규칙과 현재 상태(CONTEXT.md)를 실시간으로 추적하여 AI의 기억력을 보조합니다.

4.3. 인사이트 자동화: /retrospective

세션 종료 시 /retrospective 명령을 실행하면, AI가 대화 내역에서 핵심 지식을 추출하여 SKILL.md나 lessons_learned.md를 자동으로 업데이트하고 PR을 생성하게 할 수 있습니다.

5. 결론 및 실천 가이드

바이브 코딩은 **"문서가 곧 코드"**라는 전제하에 움직입니다. Phase 1에서는 다음 세 가지에 집중하십시오.

1. **Constitutionalize:** 프로젝트 루트에 AGENTS.md를 작성하여 AI의 행동 지침을 확립하십시오.[6]
2. **Modularize Knowledge:** 지식을 중앙 README에 몰아넣지 말고, 도메인별 하위 폴더의 .md 파일로 분산하여 AI의 주의력을 최적화하십시오.[21]
3. **Capture Failures:** 성공보다 실패를 더 기록하십시오. 기록된 실패는 AI가 다음 세션에서 밟지 않을 훌륭한 이정표가 됩니다.

이러한 문서 체계가 잡히면 Phase 2에서 피그마 데이터를 기반으로 디자인 시스템 코드를 자동 생성할 때, AI는 이미 정의된 tokens.md와 figma-mcp-rules.md를 참고하여 인간 개발자의

의도와 일치하는 정밀한 코드를 생산하게 될 것입니다.

참고 자료

1. What are the Best Vibe Coding Practices? - Emergent, 1월 15, 2026에 액세스, https://emergent.sh/learn/vibe-coding-best-practices
2. How to Build Entire AI Systems Using Only Text Files | by Hex Shift, 1월 15, 2026에 액세스, https://hexshift.medium.com/markdown-driven-development-how-to-build-entire-ai-systems-using-only-text-files-b8b3c8d054b8
3. How We Use Claude Code Skills to Run 1,000+ ML Experiments a ..., 1월 15, 2026에 액세스, https://huggingface.co/blog/sionic-ai/claude-code-skills-training
4. Figma Design System — Best Practices for AI and Developers, 1월 15, 2026에 액세스, https://medium.com/@minhtruonghoang28/figma-design-system-best-practices-for-ai-and-developers-70099a8a06a7
5. Generate TypeScript Interfaces with AI: A Practical Guide, 1월 15, 2026에 액세스, https://www.augmentcode.com/guides/generate-typescript-interfaces-with-ai-a-practical-guide
6. agentmd/agent.md: This repository defines AGENT.md, a ... - GitHub, 1월 15, 2026에 액세스, https://github.com/agentmd/agent.md
7. Improve your AI code output with AGENTS.md (+ my best tips), 1월 15, 2026에 액세스, https://www.builder.io/blog/agents-md
8. What tools and MCPs are you using with Claude Code? Let's share ..., 1월 15, 2026에 액세스, https://www.reddit.com/r/ClaudeAI/comments/1lx4277/what_tools_and_mcps_are_you_using_with_claude/
9. Dear LLM, here's how my design system works - UX Collective, 1월 15, 2026에 액세스, https://uxdesign.cc/dear-llm-heres-how-my-design-system-works-b59fb9a342b7
10. Resources / Best Practices for Using PRDs with Cursor - ChatPRD, 1월 15, 2026에 액세스, https://www.chatprd.ai/resources/PRD-for-Cursor
11. Team & Enterprise Scale | AI Native Development Guide, 1월 15, 2026에 액세스, https://danielmeppiel.github.io/awesome-ai-native/docs/team-adoption/
12. Concepts | AI Native Development Guide, 1월 15, 2026에 액세스, https://danielmeppiel.github.io/awesome-ai-native/docs/concepts/
13. AI-Ready Design Systems: Preparing Your Design System for ..., 1월 15, 2026에 액세스, https://www.supernova.io/blog/ai-ready-design-systems-preparing-your-design-system-for-machine-powered-product-development
14. How I use AI agents to automate shipping complex features - Mark ..., 1월 15, 2026에 액세스, https://markptorres.com/ai_workflows/2025-11-17-linear-ticket-execution-orchestrator

15. How to write a good spec for AI agents - AddyOsmani.com, 1월 15, 2026에 액세스, https://addyosmani.com/blog/good-spec/
16. Master the Blueprint: LLM Prompts for Perfect Product Requirements ..., 1월 15, 2026에 액세스, https://reeganalward.com/master-the-blueprint-llm-prompts-for-perfect-product-requirements-documents-prd-192b23835462
17. The Ultimate Vibe Coding Guide : r/ClaudeAI - Reddit, 1월 15, 2026에 액세스, https://www.reddit.com/r/ClaudeAI/comments/1kivv0w/the_ultimate_vibe_coding_guide/
18. automata/aicodeguide: AI Code Guide is a roadmap to start ... - GitHub, 1월 15, 2026에 액세스, https://github.com/automata/aicodeguide
19. How to build reliable AI workflows with agentic primitives and ..., 1월 15, 2026에 액세스, https://github.blog/ai-and-ml/github-copilot/how-to-build-reliable-ai-workflows-with-agentic-primitives-and-context-engineering/?utm_source=blog-release-oct-2025&utm_campaign=agentic-copilot-cli-launch-2025
20. AI Agent Template MCP Server - LobeHub, 1월 15, 2026에 액세스, https://lobehub.com/mcp/bswa006-mcp-context-manager
21. Claude Code: Part 2 - CLAUDE.md Configuration Files - Luiz Tanure, 1월 15, 2026에 액세스, https://www.letanure.dev/blog/2025-07-31--claude-code-part-2-claude-md-configuration
22. Learning TypeScript by Building a Markdown Editor | by Raj Kundalia, 1월 15, 2026에 액세스, https://medium.com/@rajkundalia/learning-typescript-by-building-a-markdown-editor-52399ff9e910
23. spec-kit/spec-driven.md at main - GitHub, 1월 15, 2026에 액세스, https://github.com/github/spec-kit/blob/main/spec-driven.md