



AI-Assisted Modern Software Development

Structuring AI tools not as a coding replacement, but as a high-speed accelerator demanding elevated critical thinking and architectural discipline.

Engineering as an Accelerator



Beyond Code Generation

AI tools do not eliminate the core responsibilities of software engineering. Instead, they shift developer focus from manual syntax writing to high-level evaluation.

Engineers must double down on domain modelling, system boundaries, and continuous logic verification.

High-Speed Acceleration

When framed correctly, AI acts as a high-speed execution force multiplier that rapidly collapses prototype-to-production feedback loops.

Success requires stronger architectural discipline to prevent low-quality code from overwhelming the codebase.

Model Selection & Context



Low-Latency Models

Use smaller, rapid models for inline autocomplete and minor syntax fixes where instantaneous latency is required.



Frontier Reasoning

Deploy high-reasoning models (Claude 3.7, GPT-4o, Gemini 2.5) for architecture, refactoring, and agent loops.



Context Rot Control

Avoid full workspace indexing. Teach explicit context scoping using references like or .

`@file`

`#line`

Repository Rules & Instructions



Project-level instruction files serve as permanent system prompts, enforcing architecture, test mandates, and coding conventions across the team.

Platform / Tool	Rule File Location	Primary Responsibility
GitHub Copilot	.github/copilot-instructions.md	Enforces repository-wide coding style, testing frameworks, and linting standards.
Cursor IDE	.cursorrules or .cursor/rules/	Defines workspace agent rules, tech stack constraints, and framework patterns.
Claude / Anthropic CLI	CLAUDE.md	Specifies CLI build commands, project layout guidelines, and bash safety rules.

Execution Modes: Ask, Plan, Agent



Ask Mode

Read-only exploratory mode for searching codebases, explaining complex logic, and answering architectural questions without file mutation.



Plan Mode

Generates a step-by-step execution blueprint (Plan) before touching code, preventing unwanted runaway edits.



Agent Mode

Autonomous editing loop that edits files, runs terminal commands, inspects build errors, and self-corrects until tasks pass.

Vibe Coding vs Engineering



Vibe Coding

Intuition-led prompting coined by Andrej Karpathy where developers accept generated code without reviewing implementation details.

Trade-off: Enables rapid non-technical prototyping, but creates security vulnerabilities, fragile edge cases, and untraceable technical debt.

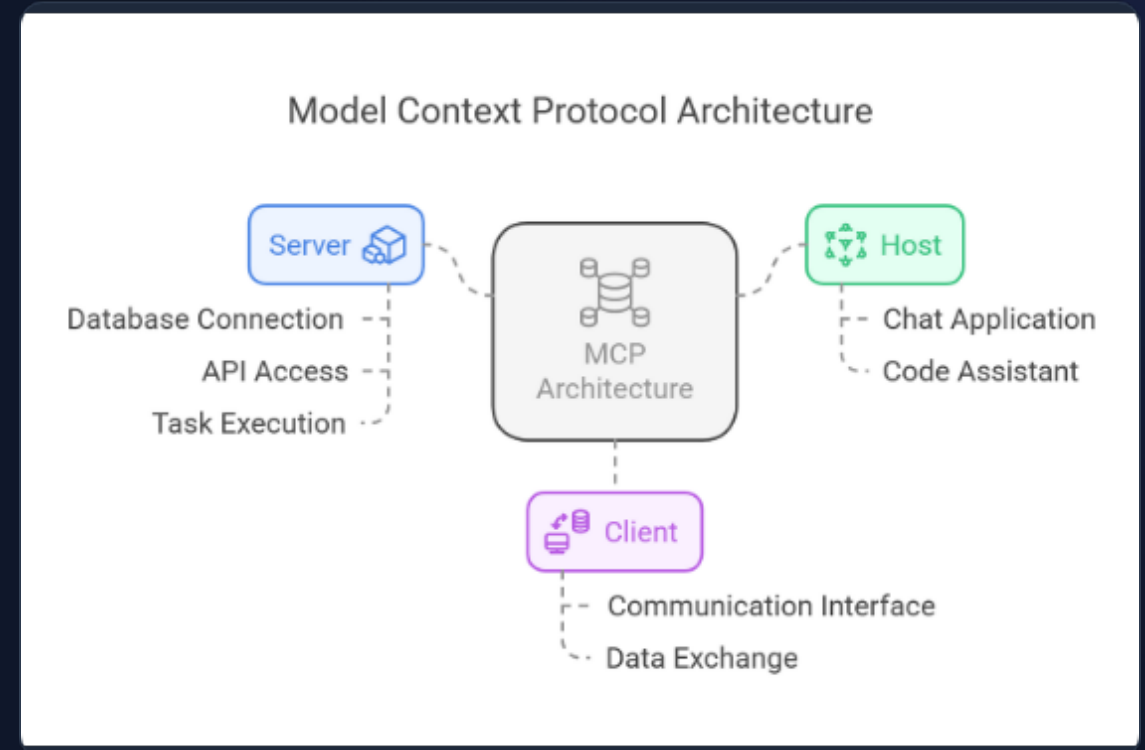
Software Engineering

A disciplined, verification-first strategy leveraging AI to write syntax while humans control system architecture.

Key Takeaway: Shifts developer responsibility from typing syntax to evaluating security, maintainability, and foundational logic.

Tool Calling & MCP Standard

- > Modern AI extends beyond static generation by executing tools, running builds, and querying live databases.
- > **Model Context Protocol (MCP)** is Anthropic's open standard acting as a universal adapter between host IDEs and external tools.
- > Eliminates fragmented N×M custom integrations by establishing a unified client-server communication standard.
- > Enables AI agents to query local storage, GitHub, enterprise tools, and databases seamlessly and securely.



Plan-Driven Dev & AI-TDD



☞ Plan-Driven Development

Decompose complex feature specs into atomic, isolated tasks within a plan file.

Instruct the AI to execute tasks one at a time, verify each task via automated testing, and commit changes incrementally.

🔧 AI-Assisted TDD

Prompt the AI to write failing unit tests based purely on product specifications first.

Switch execution modes to generate minimal implementation code that satisfies the test suite, preventing hallucinated tests.

Code Hygiene & Mindset



-  **Git Hygiene & Differential Auditing:** Treat as your primary inspection lens. Commit immediately after every successful AI task, and reset liberally when AI strays.
`git diff`
-  **Security & Secret Leaks:** Strictly audit context to prevent uploading files or A `.env` entials, and review terminal commands before granting approval.
-  **The Code Review Rule:** *Never accept or merge AI-generated code that you cannot explain line-by-line to someone else.*

 DISCUSSION & QUESTIONS

?

Thank You