KelasIT.com

TRIK & TIPS
AI CODING

Panduan Praktis Ngoding Lebih Cepat, Rapi, dan Aman Bersama AI

Edisi 2026 • eBook Panduan Developer

Hifzon, M.Kom

Daftar Isi

BAB 1

Kenapa Perlu Belajar AI Coding?

Dunia development sudah bergeser. AI coding assistant seperti Claude, GitHub Copilot, dan sejenisnya bukan lagi "mainan", tapi sudah jadi bagian dari workflow harian banyak developer profesional. Bukan berarti AI menggantikan programmer — tapi programmer yang tahu cara memanfaatkan AI akan jauh lebih cepat dibanding yang tidak.

Buku kecil ini merangkum trik-trik praktis yang bisa langsung dipakai: mulai dari memilih tool, menyusun prompt yang tepat, debugging, refactoring, testing, sampai menjaga kode tetap aman saat dibantu AI.

Manfaat Nyata Memakai AI saat Coding

- Menulis boilerplate dan kode repetitif dalam hitungan detik.
- Menjelaskan error message yang membingungkan dengan bahasa manusia.
- Membantu memahami codebase asing dengan cepat.
- Menjadi "pair programmer" untuk brainstorming solusi.
- Mempercepat penulisan unit test dan dokumentasi.

TIPS: Ingat
AI adalah alat bantu, bukan pengganti logika berpikir. Selalu pahami kode yang dihasilkan sebelum dipakai di production.

BAB 2

Memilih Tool AI Coding yang Tepat

Ada banyak tool AI coding, dan masing-masing punya kekuatan berbeda. Memilih tool yang tepat untuk kebutuhanmu akan sangat memengaruhi produktivitas.

Kategori Tool AI Coding

- **Chat-based assistant** (Claude, ChatGPT) — cocok untuk diskusi arsitektur, debugging kompleks, dan menjelaskan konsep.
- **In-editor autocomplete** (GitHub Copilot, Codeium) — cocok untuk menulis kode baris demi baris lebih cepat.
- **Agentic coding tool** (Claude Code, Cursor Agent) — cocok untuk delegasi tugas besar: refactor multi-file, migrasi, atau membangun fitur dari awal.
- **Code review bot** (terintegrasi CI/CD) — cocok untuk menjaga kualitas kode sebelum merge.

Tips Memilih

- Kalau kerja sering ganti-ganti file besar, pilih tool agentic yang bisa membaca dan mengedit banyak file sekaligus.
- Kalau butuh kecepatan menulis kode baris per baris, pilih autocomplete di dalam editor.
- Kalau butuh sparring soal desain sistem, pakai chat assistant dengan konteks panjang.
- Perhatikan apakah tool tersebut mendukung bahasa dan framework yang kamu pakai sehari-hari.

TIPS: Kombinasi adalah Kunci
Banyak developer profesional memakai lebih dari satu tool sekaligus: autocomplete untuk kecepatan harian, plus agentic tool untuk tugas besar seperti refactor.

BAB 3

Seni Prompting untuk Ngoding

Kualitas output AI sangat bergantung pada kualitas instruksi yang kamu berikan. Ini beberapa prinsip prompting yang terbukti efektif untuk konteks coding.

1. Berikan Konteks yang Jelas

Jangan cuma bilang "perbaiki kode ini". Jelaskan bahasa, framework, tujuan fungsi, dan batasan yang ada.

```
Contoh prompt yang kurang baik:
"perbaiki fungsi ini"

Contoh prompt yang lebih baik:
"Fungsi Python ini harusnya menghitung total
harga dengan diskon, tapi hasilnya salah
kalau diskon lebih dari 50%. Gunakan
Python 3.11, tanpa library tambahan."
```

2. Minta Langkah Berpikir (Step-by-Step)

Untuk masalah yang kompleks, minta AI menjelaskan alur berpikirnya dulu sebelum menulis kode. Ini membantu menemukan kesalahan asumsi lebih awal.

3. Berikan Contoh (Few-Shot)

Kalau kamu punya gaya penulisan kode tertentu (naming convention, struktur folder, pola error handling), berikan satu contoh supaya AI mengikuti gaya yang sama.

4. Batasi Ruang Lingkup

Untuk tugas besar, pecah jadi beberapa prompt kecil: satu untuk struktur data, satu untuk logika utama, satu untuk test. Ini membuat hasil lebih mudah diverifikasi.

5. Minta Format Output Spesifik

- Minta AI mengembalikan hanya kode (tanpa penjelasan) kalau kamu mau langsung copy-paste.
- Minta AI membuat tabel perbandingan kalau kamu sedang mengevaluasi beberapa pendekatan.
- Minta AI membuat daftar edge case yang perlu ditangani sebelum menulis kode.

TIPS: Trik Praktis
Simpan template prompt yang sering kamu pakai (misalnya untuk code review atau menulis test) supaya konsisten setiap kali dipakai.

BAB 4

Trik Debugging Bareng AI

Debugging adalah salah satu area di mana AI paling terasa manfaatnya, terutama untuk error yang membingungkan atau bug yang sulit direproduksi.

Langkah Efektif Debugging dengan AI

- Salin pesan error **lengkap**, termasuk stack trace — jangan dipotong.
- Sertakan kode yang relevan, bukan seluruh file kalau tidak perlu.
- Jelaskan apa yang **seharusnya** terjadi vs apa yang **benar-benar** terjadi.
- Sebutkan perubahan terakhir yang kamu buat sebelum bug muncul.
- Kalau bug sulit direproduksi, minta AI membantu menyusun skenario reproduksi minimal (minimal reproducible example).

Contoh Alur Prompt Debugging

```
Bahasa: TypeScript, Framework: Next.js 14

Error:
TypeError: Cannot read properties of
undefined (reading 'map')
  at ProductList (ProductList.tsx:22)

Kode terkait: [tempel fungsi ProductList]

Yang saya harapkan: daftar produk tampil.
Yang terjadi: halaman crash saat data
belum selesai di-fetch.
```

Manfaatkan AI untuk Root Cause Analysis

Selain memperbaiki gejala, minta AI menjelaskan kenapa bug itu terjadi. Ini membantu kamu belajar pola kesalahan yang sama supaya tidak terulang di bagian kode lain.

TIPS: Hati-hati
Jangan langsung menerima patch dari AI tanpa membaca. Kadang AI "menutupi" gejala (misalnya menambah null check) tanpa menyelesaikan akar masalahnya.

BAB 5

Refactoring & Code Review dengan AI

Refactoring

AI sangat membantu untuk refactoring skala kecil-menengah: mengganti pola callback jadi async/await, memecah fungsi besar jadi lebih kecil, atau menyesuaikan kode dengan best practice terbaru.

- Minta AI menjelaskan risiko sebelum melakukan refactor besar (misalnya perubahan pada shared utility).
- Refactor bertahap, lalu jalankan test setelah setiap perubahan kecil.
- Minta AI mempertahankan behavior yang sama persis, dan menyebutkan bagian yang berpotensi berubah.

Code Review

AI bisa jadi "reviewer pertama" sebelum kode masuk ke pull request, membantu menangkap masalah umum lebih awal.

- Minta AI mengecek edge case yang belum ditangani.
- Minta AI mengecek potensi masalah performa (misalnya query N+1, loop tidak efisien).
- Minta AI mengecek konsistensi penamaan dan gaya kode dengan sisa codebase.
- Gunakan AI untuk menulis ringkasan perubahan (changelog) pada pull request.

TIPS: Trik Tim
Beberapa tim menambahkan AI code review sebagai langkah otomatis di CI, tapi review manusia tetap jadi keputusan akhir sebelum merge.

BAB 6

Otomatisasi Testing dengan AI

Menulis test sering dianggap membosankan, padahal krusial. AI bisa mempercepat proses ini secara signifikan.

Yang Bisa Didelegasikan ke AI

- Membuat unit test dari fungsi yang sudah ada, termasuk edge case.
- Membuat data dummy (mock/fixture) yang realistis.
- Menulis test untuk skenario error/exception yang jarang terpikirkan.
- Mengonversi test dari satu framework ke framework lain (misalnya Jest ke Vitest).

Tips Supaya Test Tetap Berkualitas

- Selalu minta AI menjelaskan apa yang diuji oleh setiap test case.
- Cek apakah test benar-benar gagal saat kode sengaja dirusak (supaya tahu test-nya valid, bukan cuma "selalu hijau").
- Jangan biarkan AI membuat assertion yang terlalu longgar hanya supaya test lolos.

TIPS: Trik Cepat
Minta AI membuat daftar edge case dulu dalam bentuk poin-poin sebelum menulis kode test. Ini lebih mudah direview daripada langsung membaca kode test yang panjang.

BAB 7

Keamanan Kode saat Pakai AI

Kecepatan tidak boleh mengorbankan keamanan. Berikut beberapa hal penting yang wajib diperhatikan saat coding dibantu AI.

Prinsip Dasar

- Jangan pernah menempelkan kredensial asli (API key, password, connection string) ke dalam prompt.
- Selalu review kode yang berhubungan dengan autentikasi, otorisasi, dan validasi input — jangan asal terima.
- Waspada terhadap dependency/library yang direkomendasikan AI; pastikan library tersebut benar-benar ada dan terpercaya sebelum diinstall.
- Jalankan static analysis atau security scanner tambahan untuk kode hasil AI, terutama sebelum production.

Waspadai "Halusinasi" Package

AI kadang menyarankan nama package yang terdengar masuk akal tapi sebenarnya tidak ada, atau merupakan package berbeda dari yang dimaksud. Ini bisa dimanfaatkan pihak jahat untuk menyusupkan package berbahaya. Selalu verifikasi nama dan sumber package sebelum instalasi.

TIPS: Wajib Diingat
Perlakukan setiap kode yang berhubungan dengan keamanan (login, pembayaran, akses data sensitif) dengan level review paling ketat, apa pun sumbernya — termasuk dari AI.

BAB 8

Workflow Efisien: AI + Git + CI/CD

Integrasi ke Alur Kerja Harian

- Gunakan AI untuk menulis pesan commit yang jelas berdasarkan diff perubahan.
- Minta AI membuat deskripsi pull request otomatis, lengkap dengan ringkasan perubahan dan cara testing.
- Manfaatkan agentic coding tool untuk tugas berulang seperti update dependency atau migrasi versi framework.
- Gunakan AI untuk menulis atau memperbaiki file konfigurasi CI/CD (misalnya GitHub Actions).

Contoh Alur Kerja Ringkas

```
1. Tulis kebutuhan fitur singkat
2. Minta AI membuat rencana/langkah kerja
3. Review rencana, revisi jika perlu
4. Delegasikan implementasi per bagian
5. Jalankan test setelah setiap bagian
6. Minta AI membuat pesan commit & deskripsi PR
7. Review manual sebelum merge
```

TIPS: Trik Produktivitas

Jangan minta AI mengerjakan seluruh fitur besar dalam satu prompt panjang. Pecah jadi tahapan supaya lebih mudah diverifikasi dan di-rollback kalau ada yang salah.

BAB 9

Kesalahan Umum yang Wajib Dihindari

- **Copy-paste tanpa membaca.** Kode yang terlihat benar belum tentu benar secara logika atau keamanan.
- **Terlalu bergantung pada AI untuk hal fundamental.** Memahami dasar algoritma dan struktur data tetap penting.
- **Prompt yang terlalu umum.** Semakin jelas konteks yang diberikan, semakin relevan hasilnya.
- **Tidak menjalankan test setelah perubahan dari AI.** Selalu verifikasi bahwa perubahan tidak merusak fitur lain.
- **Menyertakan data sensitif dalam prompt.** Selalu anonimkan atau ganti dengan data dummy.
- **Mengabaikan konteks proyek.** AI tidak otomatis tahu konvensi tim kamu kecuali diberi tahu atau ditunjukkan contoh.

TIPS: Kesimpulan Singkat

AI mempercepat proses, tapi tanggung jawab kualitas kode tetap ada di tangan developer.

BAB 10

Penutup & Sumber Belajar Lanjutan

AI coding assistant akan terus berkembang, dan cara terbaik untuk tetap relevan adalah terus bereksperimen: coba tool baru, uji berbagai gaya prompting, dan evaluasi mana yang paling cocok dengan gaya kerjamu.

Ringkasan Trik Utama

- Pilih tool sesuai jenis pekerjaan (chat, autocomplete, atau agentic).
- Berikan konteks yang jelas dan spesifik saat prompting.
- Gunakan AI untuk debugging, refactor, dan testing — tapi tetap verifikasi hasilnya.
- Jaga keamanan: jangan share kredensial, selalu review kode sensitif.
- Integrasikan AI ke workflow Git/CI/CD secara bertahap.

> *"AI tidak menggantikan developer yang baik — ia menggantikan developer yang tidak mau belajar memakainya dengan benar."*

Selamat mencoba dan selamat ngoding lebih cepat!