VIBE CODING

CHEATSHEET

AI-directed development: define intent, constrain scope, verify hard, ship clean.

Director mindset
Intent • Constraints • Proof

🎯 THE VIBE LOOP

1. **Frame outcome**
   Define user result and acceptance checks
2. **Scope change**
   Identify files, boundaries, non-goals
3. **Generate**
   Single pass, minimal output
4. **Review diff**
   Align with intent, reject drift
5. **Verify**
   Tests, UX, edge cases, then commit

📋 PROMPT DNA

Goal	Outcome + user impact
Context	Key files, naming, constraints
Non-goals	What must not change
Checks	Tests, UI steps, edge cases

💻 PROMPT TEMPLATE

```
Goal: <what the user should be able to do> Context: <files, constraints, stack> Non-goals: <what to avoid, scope limits> Checks: <tests, UI verification steps>
```

Ask for a plan first when risk is high.

✅ QUALITY GATES

Diff review	Verify files, scope, and intent
Run it	Local test or manual UX pass
Edge checks	Hit a non-happy path
Stop if	Failing tests, unclear ownership, hidden scope

⚖️ WHEN TO VIBE VS CODE

Great Fits	Switch to Classic
MVPs and demos	Security-heavy flows
UI/content iteration	Perf-critical logic
Internal tools	Ambiguous scope
Docs scaffolding	No tests or owners
Data cleanup scripts	Regulated systems
Refactors with tests	Large architecture

Higher blast radius = more design before generation.

⚡ CORE PRINCIPLES

Director mindset	You define outcomes and approve diffs
Small scopes	Short prompts, tight diffs, faster review
Verify by default	Tests + behavior checks beat vibes
State discipline	Commit often, log prompts, keep rollback

🛠️ THE VIBE STACK

Explore	Editor + browser agent
Build	Repo-aware CLI agent
Verify	Tests, lint, runtime checks
Anchor	Real files, real constraints

🧭 PROMPT PATTERNS

Plan first	Outline approach before code
One feature	Keep scope tight per prompt
Diff-first	Ask for patches, not rewrites
Sample data	Realistic inputs = accurate output
Force checks	Require explicit success criteria

⚠️ DRIFT SIGNALS

- ✗ Spec expands mid-prompt or adds new features
- ✗ Renames or rewires files you did not touch
- ✗ Introduces new dependencies without approval
- ✗ Deletes tests or bypasses verification

🚫 ANTI-PATTERNS

- ✗ Letting the model redefine goals
- ✗ Stacking changes before verification
- ✗ Accepting diffs without a local run
- ✗ Shipping without ownership or rollback

👥 TEAM & SAFETY

Roles

Owner sets scope • Builder runs loop • Reviewer verifies risk

Guardrails

No secrets in prompts • Pin dependencies • Audit changes • Keep rollback

📝 RELEASE CHECKLIST

Use every iteration

- ☐ Scope locked, acceptance defined
- ☐ Diff reviewed for drift
- ☐ Tests run or UX verified
- ☐ Edge case probed
- ☐ Prompt log updated
- ☐ Rollback plan confirmed