

START HERE

Building projects with AI.

From early discussion to a tested, maintainable result.

~/GUIDES/006

Discuss the idea. Write the brief. Build in stages. Verify everything.



00 / INTRODUCTION

Think before you build.

AI can help you create software, websites and useful tools. Planning and checking still determine whether the result works.

An impressive prototype can appear quickly, but real projects also need requirements, reliable data, security, testing and maintenance. The strongest AI-assisted workflow is not one huge prompt. It is a sequence of conversations, scoped tasks, practical checks and recorded decisions.

THE CORE IDEA

Discuss and refine the idea first. Turn the agreed plan into a clear brief. Let the agent build bounded pieces that you can test.

IN THIS GUIDE

- 01 Planning conversations**
Work through uncertainty with AI.
- 02 Project briefs**
Make decisions and scope explicit.
- 03 Small milestones**
Build useful pieces in order.
- 04 Testing**
Prove what works, not just what looks good.
- 05 Recovery**
Debug, checkpoint and hand off.
- 06 Final checklist**
A reusable workflow for future work.

01 / DISCUSSION

Talk the idea through.

A good project often begins with several rounds of conversation, not a perfect first instruction.

01

Describe the problem

Explain who the project is for and why the existing solution is not good enough.

02

Explore alternatives

Ask about implementation options, trade-offs, costs, accessibility and practical limitations.

03

Make the decisions

Choose essential features, what can wait and what you explicitly do not want.

04

Challenge assumptions

Ask AI to identify missing requirements, risky dependencies and decisions that need human input.

05

Produce the agreed brief

Request a self-contained specification that a fresh coding-agent session can follow.

VERY IMPORTANT

Back-and-forth is normal. Clarifying a complicated idea can take longer than typing the final build prompt.

02 / REQUIREMENTS

What a useful brief needs.

A concise project specification prevents the agent from silently inventing requirements.

GOAL

Who and why

Describe the audience, core problem and what a successful first version will accomplish.

SCOPE

What is included

List the required features and explicit exclusions. Separate “must have” from later ideas.

CONSTRAINTS

Technical boundaries

Name existing frameworks, file locations, performance targets, privacy needs and deployment restrictions.

ACCEPTANCE

Definition of done

Specify observable outcomes: test results, working forms, mobile layouts, no broken links and documented limitations.

A GOOD HANDOFF

Include the decisions made during discussion, not the entire rambling chat transcript.

03 / GIVE DIRECTION

Specific beats enormous.

A big task becomes manageable when the agent knows where to work, what to build and how to verify it.

VAGUE INSTRUCTION

“Build me a website.”

No audience, design reference, existing code, essential pages, acceptance checks or security boundaries. The assistant has to guess nearly everything, so revision is inevitable.

ACTIONABLE INSTRUCTION

“Build the first milestone.”

“In the local project folder, create a responsive three-page service site using the supplied branding. Keep all data fictional. Add a working contact-form demo, run checks and report what is unfinished.”

KEEP TALKING

If the brief is still unclear, return to planning. Do not begin implementation just because you have written a long prompt.

04 / MILESTONES

Build one piece at a time.

A staged build gives you opportunities to catch mistakes while they are small and easy to reverse.

01

Create a safe baseline

Set up version control, dependencies, a working local build and a backup of existing files.

02

Build the basic structure

Implement navigation, page layout and a minimal working example without extras.

03

Finish one vertical slice

Take one feature from interface through application logic to testable behaviour.

04

Run focused checks

Test the changed feature and its immediate dependencies before adding more.

05

Commit and summarise

Save verified work, record limitations and select the next small milestone.

PRACTICAL RHYTHM

Plan implement test review commit. Repeat instead of piling unfinished features together.

05 / ONE FEATURE

Build a comment form.

Imagine a blog needs a basic account-based commenting feature. Treat this as a sequence, not a single unchecked code dump.

Before writing code, decide who can comment, what moderation means and how personal information is handled.

BEHAVIOUR

Describe the user journey

"Signed-in visitors can submit a comment. New comments remain pending until approved by a moderator. Visitors cannot impersonate staff."

IMPLEMENT

Build a local slice

Add the form, server-side validation, storage and a moderation state. Use a local database and synthetic test accounts.

VERIFY

Test the awkward cases

Try signed-out submission, very long text, duplicate posts, unsafe HTML, authorisation failures and a moderator approving a pending comment.

Public registration, email verification, rate limits and live hosting should be separate tasks with their own tests.

A green screen is not proof.

A web application needs more than an AI-generated statement saying “everything works”.

FUNCTION

Run real tests

Build the project, run unit and integration tests, and verify user journeys against acceptance criteria.

SECURITY

Check access and inputs

Review permissions, input validation, secrets, dependency risk and unsafe data handling. Use appropriate specialist review.

ACCESSIBILITY

Test different users

Check keyboard controls, labels, contrast, focus states, error messages and readable content.

RESPONSIVE

Inspect actual screens

Test on narrow phones and wider screens. Catch overflow, broken navigation and awkward touch targets.

DON'T OVERCLAIM

Passing tests increases confidence but does not prove that an application is free of defects or secure in production.

07 / CONTROL

Make the agent manageable.

You can delegate plenty of development work while still controlling the scope and accepting responsibility for the result.

ACCESS**Limit the workspace**

Use a local repository or staging site. Keep production passwords and irreversible actions outside normal agent access.

BUDGET**Set usage boundaries**

Give a stop threshold and reserve allowance for verification, documentation and final commits.

STATUS**Ask for honest reports**

Require exact files changed, commands run, tests passed or failed and unresolved blockers.

HISTORY**Use branches and commits**

A clean Git history is a practical recovery tool. Commit stable milestones and keep a written handoff file.

HUMAN ROLE

Read the diffs, open the site yourself and decide what is acceptable before deployment.

When the agent gets stuck.

Repeated failures often mean the problem needs to be narrowed rather than solved by another enormous prompt.

01

Record the real symptom

Copy the error, failed test name or reproduction steps instead of relying on a vague description.

02

Find the smallest failing case

Reduce the problem to one file, route or feature and separate environmental faults from code faults.

03

Inspect assumptions

Confirm dependency versions, paths, permissions, inputs and previous fixes.

04

Try one targeted change

Make the smallest defensible adjustment and rerun the relevant check.

05

Checkpoint or roll back

If verification fails, report it honestly and return to a working revision when necessary.

AVOID ENDLESS LOOPS

Set retry limits. A clear report of a blocker is better than hours of speculative patching.

09 / QUICK REFERENCE

Your repeatable workflow.

Use these stages for your next website, application or automation project.

01

Discuss the idea

Clarify who it helps, how it works and what could go wrong.

02

Agree the essentials

Write must-haves, exclusions and acceptance criteria.

03

Prepare a fresh-session brief

Include paths, decisions, tools, limits and definitions of done.

04

Set a safe baseline

Use source control, backups, synthetic data and local development.

05

Build a small milestone

Finish one usable piece before widening the scope.

06

Test and review

Check commands, functionality, usability and security assumptions.

07

Commit and hand over

Record results, outstanding work and exactly where to resume.

PROMPT TO REUSE

"Turn our agreed decisions into a concise implementation brief for a fresh agent. Include constraints, milestones, tests and stop conditions."

10 / KEEP BUILDING

Ship work you can trust.

The advantage of AI-assisted development is faster iteration, not the removal of engineering responsibility.

Good projects grow from conversations and explicit decisions into manageable tasks. Ask AI to help you plan, implement and troubleshoot, but insist on tests, visible evidence and careful review. When work spans sessions, a good brief and handoff keep everyone aligned.

REMEMBER THESE THREE THINGS

01 Planning saves rework

Discussion and a clean specification beat rushing into an enormous build.

02 Verification is part of building

Test each feature, inspect real screens and challenge unsupported claims.

03 A good handoff makes work durable

Commits and clear notes let the next session continue without guessing.

FURTHER READING / REFERENCE

[Anthropic / Long-running agents](#)



[GitHub / Review AI-generated code](#)



[OpenAI / Building agents](#)



Explore more at nathm.net