Mastering the Claude Code session

This series will teach you the session management framework that Anthropic's Claude Code team uses internally.

You used Claude Code . It worked perfectly for the first 20 minutes. Then things changed. Claude started forgetting what you told it to do. It repeated methods you had rejected. It conjured up a function that didn't exist in your codebase.

It's a phenomenon of "losing context"—and that's the biggest frustration for Claude Code users. Not because the model is bad, but because most people never learn how to manage their sessions.

This series will teach you the session management framework that Anthropic's Claude Code team uses internally. Based on the official guide by Thariq Shihipar (over 3400 likes on X) and refined with community-proven techniques, you'll learn 5 steps to keep your context clean, sessions efficient, and Claude sharp from the first prompt to the last.

What you will learn

1. Explain the phenomenon of context degradation and why Claude Code's performance declines during long sessions.
2. Apply the 5-step decision framework (Continue, Rewind, Clear, Compact, Subagent) after each iteration.
3. Use /rewind instead of correction to keep the context clean and focused.
4. Compare /compact with /clear and choose the appropriate strategy for each situation.
5. Design a subagent workflow to isolate context for research, experimentation, and documentation.
6. Develop a personal session management strategy using CLAUDE.md rules, logging, and proactive collapsing.

After this course, you will be able to

1. Running Claude Code sessions for 2 hours and remaining sharp from start to finish - no more instances of AI confusion at the 45-minute mark.
2. Use rewind, compact, and subagent intuitively – choose the right step in under 5 seconds.
3. Session architecture design for complex projects - parallel sessions, context isolation, structured handovers
4. Configure the CLAUDE.md rules so that Claude automatically manages its own context without manual intervention.
5. Add "Claude Code mastery" to your development toolkit—a skill that is clearly demonstrated in code review speed and product deployment speed.

What you will build

Session Decision Handbook

A personalized reference card maps your common work scenarios to the optimal session operation—Continue, Rewind, Compact, Clear, or Subagent.

CLAUDE.md Session Configuration

A CLAUDE.md configuration ready for a production environment with context rules, compression policies, and session management automation tailored to your actual codebase.

Mastering the Claude Code session

Demonstrate your ability to manage Claude Code sessions, prevent contextual errors, and design multi-session workflows.

Prerequisites

1. Basic experience with Claude Code (you've used it for at least a few sessions)
2. Familiar with terminal commands (cd, ls, basic navigation)

Suitable candidates

1. Developers use Claude Code but find sessions become chaotic or unreliable after more than 30 minutes.
2. Experienced users want to understand why Claude's quality deteriorates and how to prevent it.
3. The team leader is standardizing Claude Code's workflow across engineering teams.
4. Anyone who encounters a 'contextual error' and wants an official solution from the Anthropic manual.

Why is your programming session failing (and how to fix it)?

Understanding contextual errors – why Claude Code has gotten worse over time – and previewing a 5-step framework to fix this problem.

The moment Claude Code stopped supporting it.

You spent 40 minutes in a programming session. Claude built the validation module perfectly. You asked him to write tests for that module. And he. wrote tests for a completely different function. A function that doesn't exist in your codebase.

You correct the error. It apologizes and tries again. It fails again. You correct it more forcefully. It starts repeating its previous responses exactly. Something has changed—and you can sense it, even if you can't name

it. That's called: contextual error.

In this lesson, you will:

1. Understanding what "contextual degeneration" actually is (and isn't).
2. See why the 1 million token window doesn't solve the problem.
3. Preview the 5-step framework you'll master in this course.

This course teaches you session management techniques used by Anthropic's Claude Code team. It's not theory – it's practical steps you can apply in your next work session.

Explain your context window.

Claude Code's context window is 1 million tokens. That's all the model can "see" at the moment when generating the next response:

1. System prompt and CLAUDE.md file
2. The entire conversation so far.
3. Every tool call that Claude made and its output.
4. All the files that Claude has read

Imagine it like a table. A million tokens is a huge table – enough room for hundreds of files, thousands of messages. But even a huge table can become cluttered.

The problem is that when your desk is covered in papers, it becomes harder to find the document you need. The model doesn't crash. It doesn't report errors. It just becomes. less sharp. Less accurate. More prone to missing details or making things up.

? **Quick check** : Which four things take up space in the Claude Code context window?

Answer : System prompt/CLAUDE.md, conversation history, output of tool call commands, and file contents.

Contextual degradation is real – and it's not a bug.

Studies on advanced models show that performance begins to degrade long before you reach the context limit. Some benchmarks show accuracy dropping at just 25% of the window. The practical threshold that most professional users have agreed upon: beyond 50-60% of the context, quality begins to decline.

This isn't just Claude's problem. It's how attention-driven models work. The more tokens compete for attention, the less focus each individual token receives. Old, irrelevant context—a debugging session from an hour ago, file read operations you no longer need—doesn't just sit there passively. It actively distracts the model.

Thariq Shihipar, a member of Anthropic's Claude Code team, stated frankly in his recent article: Contextual degradation is "observed as model performance decreases as context increases because attention is spread across more tokens, and old, irrelevant content begins to distract from the current task."

5-Step Framework (Preview)

Most people treat Claude Code like a chat window – they just keep typing. But after each response from Claude, you actually have five options:

1. **Continue** — send another message (everything remains in context)
2. **Rewind** — return to the previous point, ignoring everything that happened afterward.
3. **/clear** — starts a completely new session
4. **/compact** — to allow Claude to summarize the session, replacing the history with a summary.
5. **Subagent** — assigning tasks to a separate agent with its own clean context.

Each action has a different impact on your context. The skill lies in knowing which action to choose – and that's what the next 7 lessons will teach you.

? **Quick Check** : If you've been debugging for 30 minutes and want to move on to a completely new task, which operation gives you the cleanest start?

Answer : /clear - it clears the context so you start over.

Why wouldn't 1 million tokens solve this problem?

When Claude Code scaled up to 1 million contexts, many people argued that session management was no longer important. More space = less cleanup. Right?

Wrong. A larger workbench means more room for clutter. The model still has to process every token in the window in each turn. And research shows that even with 1 million tokens, performance still drops at the same percentage – it just takes longer to reach that level.

The 1 million token window is a safety margin, not a license to dump everything into it. The best Claude Code users treat it as working memory that needs constant management – ??not an endless storage.

Key points to remember

1. Contextual degradation is real: The quality of Claude Code gradually decreases as your gaming session lasts, even with 1 million tokens.
2. The degradation begins when using about 50-60% of the context – long before the window is full.
3. You have 5 moves after each turn, it's not just a matter of "keep typing".
4. Session management is the number one skill that distinguishes frustrated users from skilled users.
5. This course teaches the working framework that the Anthropic team uses.

Try this now: Self-check context state

Open Claude Code (or any Claude session in the terminal) and paste this prompt into the current session. It tells you if your current context is clean or experiencing problems – without having to restart.

```
Hãy ?óng vai trò là ng??i ki?m tra tr?ng thái phiên Claude Code c?
a tôi. Tôi s? d?ng ?i?u khi?n và ?? b?n ki?m tra phiên hi?n t?
i. Hãy cho tôi bi?t li?u chúng ta có nên ti?p t?c, tua l?i, nén, xóa hay t?
o m?t subagent hay không. Trong thông ?i?p này, hãy phân tích: 1. ?i?m ?
ánh giá ch?t l??ng ng? c?nh (1–5) - 5 = rõ ràng, m?i thông ?i?p ??u ???
```

c truy?n ??t - 3 = có m?t s? thông tin th?a nh?ng v?n s? d?ng ???
c - 1 = gây hi?u nh?m — các quy?t ??nh c? mâu thu?n v?i các quy?t ??nh hi?
n t?i 2. 5 thành ph?n h?u ích nh?t trong phiên hi?n t?i (???ng d?n file ?
ã truy c?p, quy?t ??nh ?ã ??a ra, m?u mã ?ã h?i t?, ràng bu?c ?ã ???c ch?
p nh?n) 3. 5 thành ph?n không c?n thi?t nh?t (khám phá b? b? d?, ch?nh s?
a b? hoàn tác, ??u ra công c? dài không còn liên quan, l?c ??) 4. Mâu thu?
n - Li?t kê b?t k? n?i nào chúng ta nói m?t ?i?u tr??c ?ó và ?i?u ng??c l?
i sau ?ó - ?ánh d?u n?i tôi th?c s? mu?n gi? l?i 5. ?? xu?t b??c ti?
p theo kèm lý do: - Ti?p t?c: phiên làm vi?c ?n ??nh, hãy ti?p t?c - Quay l
?i: nêu rõ l??t quay l?i c? th? - /compact: nh?ng gì m?t thao tác nén t?t s
? b?o toàn; nh?ng gì nó s? b? qua - /clear: ch? khi nhi?m v? còn l?
i khác bi?t v? c? b?n so v?i nh?ng gì ?ã ???c th?c hi?n - Subagent: ch?
 dành cho m?t cu?c ?i?u tra ??c l?p ho?c s? d?ng công c? n?ng n? có th?
 làm r?i lo?n ng? c?nh chính 6. N?u b?n ?? xu?t /compact, hãy so?n th?
o ghi chú bàn giao mà tôi s? l?u vào .claude/notes/ TR??C KHI nén, ?? nh?
ng ?i?u c?n thi?t v?n còn ngay c? khi quá trình nén làm m?t chi ti?t: - Nhi
?m v? tôi ?ang làm (m?t câu) - Các file ?ã ???c s? d?ng + lý do - Các quy?
t ??nh quan tr?ng ?ã ???c ch?t (kèm lý do) - Các câu h?i ch?a ???c gi?i quy
?t - B??c ti?p theo duy nh?t 7. N?u b?n ?? xu?t m?t subagent, hãy so?n th?
o prompt subagent chính xác — ??c l?p, v?i các ???ng d?n file và ràng bu?
c c? th? mà subagent c?n ?? nó không l?y ng? c?nh t? phiên chính QUY T?C B?
N V?NG: - Không bao gi? ?? xu?t "c? ti?p t?c" n?u ?i?m s?c kh?e ng? c?
nh là 2 ho?c th?p h?n - Không bao gi? ?? xu?t /compact n?u phiên ch?
a các quy?t ??nh ch?a ???c cam k?t có t?i — nên l?u ghi chú bàn giao tr??
c - Không bao gi? ?? xu?t /clear trong khi Có nh?ng công vi?c ch?a ???
c cam k?t mà b? nh? c?a mô hình là b?n ghi duy nh?t — hãy b?o tôi cam k?
t và l?u ghi chú tr??c - N?u phiên làm vi?c có liên quan ??
n các thao tác phá h?
y (rm -rf, DROP TABLE, git reset --hard, force push) và k?t qu? c?
a chúng ch?a ???c xác nh?n, hãy g?n c? ?i?u ?ó TR??C KHI ?? xu?t b?t k?
 hành ??ng d?n d?p nào - N?u phiên làm vi?c v??t qua ranh gi?i tin c?
y (ví d?: dán mã NDA c?a khách hàng, tác ??ng ??n môi tr??ng s?n xu?t, s? d
?ng thông tin ??ng nh?p), hãy ?? xu?t l?u b?ng ch?ng và ghi chú tr??c khi d
?n d?p - N?u tôi ?ang ? trên kho l?u tr? nhóm, hãy nh?c tôi r?ng vi?c d?n d
?p phiên làm vi?c là R?I RO C?A TÔI — ??ng nghi?p c?a tôi không th? th?y nh
?ng gì trong c?a s? ng? c?nh - Không bao gi? ?? xu?t s? d?
ng subagent cho công vi?c ch? yêu c?u ki?n ??th?c trong ng? c?
nh cha — subagent s? ?o t??ng v? nó

What you will see : Contextual health scores from 1–5, top artifacts + burdens, conflict lists, clear next-step suggestions, and (if compressing or creating a subagent) accurate transfer notes or prompt subagent prompts – so that cleaning up the session becomes a purposeful action rather than a surprise.

1. Question 1:

   After Claude has finished replying, how many session management options do you have?

   1. A. Two - continue the session or restart completely.
   2. B. One - just keep typing whatever comes to mind next.
   3. C. Year - continue, rewind, delete, collapse, create subagent

4. D. Three - continue, delete the conversation, or collapse the entire conversation.

EXPLAIN:

You have five different operations: Continue, Rewind, /clear, /compact, or create a subagent (delegate to a separate context). Most people only use Continue.

2. Question 2:

   Claude Code has a context window of 1 million tokens. What does this mean in practice?

   1. A. You should always fill the entire window for maximum performance.
   2. B. Everything Claude sees at once - instructions, chat, files, tools
   3. C. That is the number of words Claude can produce in a single response.
   4. D. That means Claude can remember all your previous sessions permanently.

   EXPLAIN:

   The context window contains everything Claude processes for each response—your system prompt, conversation history, file contents, and tool output. A larger window means more space, but filling it completely will cause a loss of context.

3. Question 3:

   What is the phenomenon of "contextual errors" in Claude Code?

   1. A. Performance decreases when the context window is full of tokens.
   2. B. A setting that automatically deletes old conversations from memory.
   3. C. The process of compressing conversation history to release tokens.
   4. D. An error that causes Claude Code to crash after long programming sessions.

   EXPLAIN:

   Contextual error is the phenomenon where a model's performance decreases as the context increases—attention is scattered across more tokens, and older content distracts from the current task. This isn't an error; it's a fundamental property of how attention-driven models work.

Submit your work

Training results

You have completed **0** questions.

-- / --

Review the lesson