MAXIMIZING

Claude Code

for Software Engineering

A 2-hour workshop on getting more from AI in your daily SDLC

19 May 2026 · 2-hour workshop

Today

1. Reality check
2. Context engineering
3. Legacy code + MCP
4. Daily workflow
5. Workflow → Skill
6. agent-dev-team
7. End-to-end automation

Why this workshop matters — right now

30,000+	**21,000+**	**12,000+**
Oracle layoffs	**Meta**	**Google**
announced 2025	2022–2024	ongoing

Tech layoffs are no longer cyclical — they're structural.

AI fluency separates engineers who keep their seat from those who become the saving.

The only question that matters:

Are you the leverage,

or the cost?

Engineers who use AI to multiply output stay. The rest become the saving.

Today's agenda — 2 hours

10 min	Reality check	Why now, why us
15 min	Context engineering	Prompt vs context, Lab 1
25 min	Legacy code + MCP	codebase-memory MCP, Labs 2+3
10 min	Daily SDLC workflow	Analyst → Dev → QA
25 min	Workflow → Skill	Lab 4 + Lab 5
15 min	agent-dev-team showcase	Demo
15 min	End-to-end automation	Demo
5 min	Q&A + Close	Takeaways

SECTION 2

Context Engineering

The prompt is only the tip of the iceberg.

Prompt vs Context

PROMPT

The instruction.

`"Write a function to validate email."`

What you want done.

Without context → generic, often wrong.

CONTEXT

The world around it.

Project, conventions, examples, constraints, expected output shape.

Everything the model needs to do it right.

With context → specific, often correct.

Prompt A — bare

```
Write me a function to validate email.
```

What Claude doesn't know:

- Language? Framework?
- Naming style? Module system?
- Throw on error, or return object?
- Edge cases that matter?
- Where to put it?

Prompt B — context-rich

```
Project: BookShelf (Node.js + Express). CommonJS, not ES modules.
Existing validators live in utils/helpers.js as named exports.
Style: addDays, format_date, calc_fine — follow that pattern.
Error handling: return { ok: false, error: "..." }. Do NOT throw.
Accept: standard RFC 5322-ish. Reject: empty, no @, no dot, spaces, > 254 chars.
Called from auth.js during register and login.
No new dependencies — pure JS only.

Task:
Add validate_email(email) to utils/helpers.js. Update exports.
Show me the diff only. Include 5 inline test calls at the bottom.
```

Where context lives

Per-prompt context
What you type each time

CLAUDE.md / repo memory
Project conventions, stack, decisions

Skills
Reusable workflows w/ steps

MCP servers
Live data + tools (codebase, Jira, etc.)

Each layer compounds. The bottom is permanent. The top is throwaway.

LAB 1

Bare prompt vs context-rich prompt

Steps

1. Run Prompt A. Save output.
2. Run Prompt B. Save output.
3. Compare style, correctness, fit.

Expected outcome

- Visible quality gap.
- Prompt B fits the existing codebase.
- You feel the ROI of 2 mins of context.
- Question to ponder: where should that context live permanently?

Time: 10 min

S E C T I O N 3

Understanding legacy code with AI

Onboarding to a codebase in minutes, not weeks.

The onboarding tax

Without AI

- Read README that's 3 years stale
- Grep your way through the repo
- Ask the one person who's been here
- Wait a week for an answer
- Re-ask when you forget
- Repeat for each new joiner

With codebase-memory MCP

- Index the repo once
- Ask Claude in natural language
- Get answers grounded in source
- Onboard in an afternoon
- Same memory serves the next joiner
- README becomes a living doc

What is MCP?

Model Context Protocol

An open standard for connecting AI assistants to the systems where your data lives.

Claude — the reasoning → **MCP server** — the bridge → **Your data** — browser, files, database

MCPs are how Claude reaches outside its own brain.

codebase-memory MCP

An MCP that indexes a codebase into a queryable knowledge graph.

What it tracks

- Functions & their callers
- Modules & their dependencies
- Routes, endpoints, schemas
- Configuration entry points
- Cross-file references

Questions it answers

- "How does auth work here?"
- "Where is the loan flow handled?"
- "What touches the books table?"
- "Show me callers of calc_fine."
- "Where do we validate input?"

github.com / deusdata / codebase-memory-mcp

LAB 2

Install codebase-memory MCP

```
# 1. Clone the MCP
git clone https://github.com/deusdata/codebase-memory-mcp.git
cd codebase-memory-mcp && npm install

# 2. Register with Claude Code
claude mcp add codebase-memory --command "node /path/to/codebase-memory-mcp/server.js"

# 3. Verify
claude   →   /mcp     (you should see codebase-memory in the list)
```

Time: 5 min · Note: pre-installed via setup email is recommended.

LAB 3

Index the BookShelf legacy app

1. Point the MCP at 04_Sample_Legacy_App/
2. Run /codebase-memory index
3. Ask: "How does borrowing a book work?"
4. Then disable MCP, ask the same. Compare.

Time: 10 min

SECTION 4

Your daily SDLC workflow

What you actually do, every day.

The roles, the loop

Software Analyst

Specs, AC, edge cases

→

Backend Dev

API, data, business logic

→

Frontend Dev

UI, state, integration

→

QA

Test, regression, sign-off

↻ *Constant back-and-forth*

The hidden cost: context switching.

Every handoff loses information. Every loop adds rework. Senior engineers spend more time re-explaining than building.

From workflow to Skill

What you do → What Claude does

Manual workflow

- Lives in your head
- Re-explained each time
- Lost when you leave
- Inconsistent across team
- Not auditable

Codified as a Skill

- Lives in a file
- Invoked with one word
- Stays when you leave
- Same flow for whole team
- Versioned in git

Anatomy of a SKILL.md

```
---
name: bug-fixer
description: |
  Use when fixing a bug ticket / feedback. Investigates code, proposes fix,
  writes test, opens PR.
triggers: ["fix bug", "BUG-", "resolve ticket"]
---

# Steps
1. Read the ticket + acceptance criteria.
2. Use codebase-memory to find related code.
3. Propose a minimal patch. Wait for approval.
4. Apply patch + add a regression test.
5. Run tests. Iterate until green.
6. Open a PR with a summary of what changed and why.
```

LAB 4

Convert YOUR workflow into a Skill

1 Pick one task you do weekly

2 Write down the steps you take (5–10)

3 Fill the SKILL.md template (in handbook)

4 Drop it in .claude/skills/your-skill/

Time: 15 min

LAB 5

Use your Skill on a real bug

Goal

Fix one ticket from BookShelf's BUG_TICKETS.md using your new skill.

Suggested tickets

- `BUG-002  Loan never decrements available_copies  (easy first win)`
- `BUG-003  Pagination off-by-one  (multi-file investigation)`
- `BUG-001  SQL injection in search  (security-flavoured)`

Time: 10 min · Stretch: chain two skills (analyst → backend)

SECTION 6

Showcase

`agent-dev-team`

What happens when you take your own workflow seriously.

agent-dev-team — what it is

A skill I built from my own workflow.

It runs Analyst → Backend → Frontend → QA on one ticket / feedback — in sequence.

Analyst	Backend	Frontend	QA

Source: `uscloud3.dxn2u.net:3000/cipta/agent-dev-team`

Real ROI

BEFORE

2–3 hrs

per ticket / feedback

Context-switching between roles, re-reading specs, copy-paste fatigue.

AFTER

3–5 min

per ticket / feedback

One invocation, all four roles, human reviews the final output.

LIVE DEMO

agent-dev-team

Watch the skill run end-to-end on a fake ticket.

SECTION 7

Skill + automation at the door of your inbox

From manual invocation to autonomous trigger.

The automation loop

Inbox

Customer feedback email

→

Cowork watch

Polls inbox, triggers

→

Claude Code

Runs agent-dev-team

→

Reply + commit

Auto reply, real PR

What this unlocks

- Feedback never sits in an inbox unread.
- Tickets move while you sleep.
- Human stays in the loop for review — not grunt work.

Guardrails — non-negotiable

1	**Human-in-the-loop review**	Auto-commit to a branch, never main. PR requires human merge.
2	**Audit trail**	Every action logged. Every reply traceable to a source email.
3	**Cost caps**	Daily budget per agent. Hard stop on runaway loops.
4	**Sandboxed write access**	Agents work in dedicated repos / branches / containers.
5	**Kill switch**	One command to disable all automation.

L I V E   D E M O

Email → Agent → Reply

I'll send an email from my phone. Watch what happens.

What we covered

1	**Reality check**	AI is reshaping who keeps a seat at the table.
2	**Context engineering**	Prompts are the tip. Context is the iceberg.
3	**codebase-memory MCP**	Onboard a legacy app in minutes.
4	**Daily SDLC workflow**	Map your roles, find the friction.
5	**Workflow → Skill**	Encode it once. Reuse forever.
6	**agent-dev-team**	A real skill, doing real work.
7	**End-to-end automation**	Agents that meet the world where it is.

Where to start tomorrow

Today	Install Claude Code on your work machine.
Day 1	Index ONE codebase with codebase-memory MCP.
Day 2	Write CLAUDE.md for that repo. Commit it.
Week 1	Convert ONE recurring task into a skill.
Week 2	Share the skill with one teammate.
Month 1	Hook one automation trigger into your inbox or chat.

Thank you.

Questions?

Share Your Feedback

Scan to fill feedback form