

Building with Claude Code

From Simple Tasks to Research-Scale Projects

MGMT 408 – UCLA Anderson

UCLA Anderson School of Management

February 12, 2026

Outline

- 1 What is Claude Code?
- 2 Case Study: Finance Practice Tool
- 3 The CLAUDE.md File
- 4 Applications for Research
- 5 Getting Started

What is Claude Code?

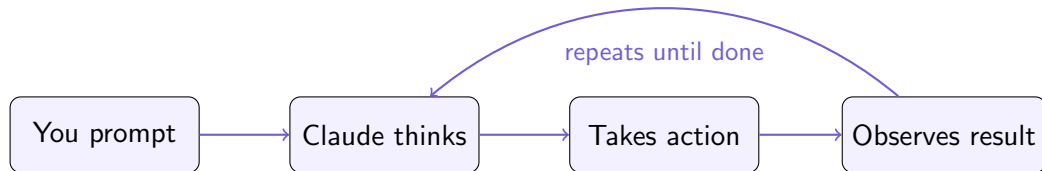
Claude Code is Anthropic's agentic coding tool that:

- Runs in your terminal alongside your codebase
- Reads, writes, and edits files directly
- Executes shell commands (git, npm, python, etc.)
- Browses documentation and searches the web
- Maintains context across an entire session
- **Asks clarifying questions** when needed

Unlike ChatGPT: Claude Code *operates* on your project, not just *discusses* it.

```
$ claude
> What would you like
to work on?
I can read your files,
run commands, and help
you build.
```

How It Works: The Agentic Loop



Actions include:

- Read – examine files in your project
- Write/Edit – create or modify code
- Bash – run terminal commands
- WebFetch – retrieve documentation
- AskUserQuestion – request clarification from you

The Task: Build a Practice Tool

Starting point: PDF lecture slides for MGMT 408 (Foundations of Finance)

Goal: Create an interactive web-based practice tool with:

- 40 practice problems across 4 topics (TVM, Capital Budgeting, Stocks, Bonds)
- Adaptive hints (3 levels) and full solutions
- Built-in calculator
- Progress tracking with localStorage
- 2 practice midterms (40 multiple-choice questions)
- LaTeX formula rendering

Result: 8 HTML files, 3,200 lines of code

Built entirely through natural language conversation with Claude Code.

Prompt #1: Initial Request

User:

"I uploaded my finance lecture slides. Can you create a web-based practice tool with problems similar to what's covered in the slides? Include a calculator and hint system."

What Claude Code did:

- 1 Read the uploaded PDF slides
- 2 Identified key topics (TVM, NPV, stock valuation, bonds)
- 3 Generated 40 problems with solutions
- 4 Built responsive HTML/CSS/JS interface
- 5 Added scientific calculator component
- 6 Implemented 3-level hint system

No manual coding required – Claude wrote, tested, and iterated autonomously.

Interactive Clarification

When Claude needed decisions, it asked:

What format would you like for the slides?

- Google Slides – *I'll provide content you can copy*
- PowerPoint – *I'll create a .pptx file*
- **PDF/Beamer** – *Code-based slides in LaTeX*
- Markdown/reveal.js – *Browser-based slides*

What's the target audience?

- Technical (CS/Engineering)
- Business/MBA students

Prompt #2: Code Review

User:

“Can you explain what this does so far?”

Claude's response: A structured analysis of the codebase:

Structure: 8 HTML files (6,765 lines)

File	Purpose	-----	-----	
index.html	Landing page / navigation		week1-tvm.html	Time Value of Money (10)
week2-capbudget.html	Capital Budgeting (10)		week3-stocks.html	Stock Valuation (10)
week4-bonds.html	Bond Valuation (10)		midterm1.html	Practice Exam 1 (20 MCQ)
midterm2.html	Practice Exam 2 (20 MCQ)			

Prompt #3: Improvement Suggestions

User:

“Do you have suggestions to improve this?”

Claude identified 14 specific issues, prioritized by impact:

- 1 **Code duplication** – 3,400 lines of CSS/JS repeated across 4 files
- 2 **localStorage broken** – progress tracking never saved
- 3 **KaTeX not rendering** – `renderMath()` was a no-op
- 4 Calculator uses `eval()` – minor security concern
- 5 No question randomization
- 6 Missing accessibility features
- 7 *...and 8 more*

Claude doesn't just build – it reviews, critiques, and suggests improvements.

Prompt #4: Implement Fixes

User:

“sure” (in response to “Want me to implement any of these?”)

Claude autonomously:

- Created `practice.css` (shared styles)
- Created `practice.js` (shared logic + `localStorage` + KaTeX fix)
- Refactored all 4 weekly pages to use shared files
- Updated LaTeX formulas to proper syntax
- Committed changes with descriptive message
- Pushed to GitHub

```
Result: 6 files changed 882 insertions(+), 3,385 deletions(-)
Net: 2,500 lines removed (deduplication)
```

Prompt #5: Verification

User:

“Can you visit the website and make sure it is all working as intended?”

Claude ran a comprehensive test suite:

- Started local HTTP server
- Verified all 10 files return HTTP 200
- Checked 16 required DOM elements in each page
- Validated JavaScript syntax
- Verified all 40 problems have required fields
- Tested cross-page navigation links

```
=== week1-tvm.html === topicKey:  tvm Problem count:  10 All fields OK  
All pages validated successfully.
```

Project Memory: CLAUDE.md

Claude Code reads a CLAUDE.md file in your repo root to understand project context:

```
# Finance Practice Tool
```

```
## Project Structure
```

- index.html: Main landing page
- week{1-4}-*.html: Practice problem pages
- midterm{1,2}.html: Exam simulations
- practice.css/js: Shared code

```
## Tech Stack
```

- Pure HTML/CSS/JS (no framework)
- KaTeX for LaTeX rendering
- localStorage for progress tracking

```
## Development Notes
```

- Problems are embedded as JS arrays
- Each problem needs: id, topic, difficulty,

Why CLAUDE.md Matters

Without CLAUDE.md:

- Claude must re-explore the codebase each session
- May make inconsistent architectural decisions
- Loses track of project conventions

With CLAUDE.md:

- Instant context on project structure
- Consistent coding style across sessions
- Remembers key decisions (“no frameworks”, “problems need 3 hint levels”)
- Can include TODOs, known issues, deployment notes

Think of it as onboarding documentation for your AI collaborator.

Scaling to Research Projects

The same workflow applies to larger, research-scale tasks:

Data Analysis

- Parse and clean datasets
- Run statistical analyses
- Generate publication-ready figures
- Iterate on methodology

Experiment Infrastructure

- Build survey tools (like this one)
- Create participant interfaces
- Log and process results

Literature & Writing

- Search and summarize papers
- Draft methodology sections
- Format citations
- Convert between formats

Reproducibility

- Generate Docker configs
- Write setup documentation
- Create replication packages

Key Advantages for Researchers

① Reduces context-switching

Stay in natural language; Claude handles the implementation details

② Handles tedious tasks

Data cleaning, formatting, boilerplate code – delegate confidently

③ Catches errors proactively

Reviews its own work, runs tests, validates outputs

④ Documents as it goes

Commit messages, code comments, CLAUDE.md updates

⑤ Learns your project

Maintains context across long sessions; CLAUDE.md persists across sessions

Getting Started

Installation:

```
npm install -g @anthropic-ai/claude-code
```

Usage:

```
cd your-project claude
```

Tips for effective use:

- Start with a clear, specific request
- Let Claude ask clarifying questions (don't over-specify upfront)
- Review outputs and provide feedback
- Create a CLAUDE.md for ongoing projects
- Use Git – Claude commits its work

Questions?

Finance Practice Tool: `github.com/[your-repo]`

Claude Code: `anthropic.com/claude-code`