WEEK 1 JOB

AI Developer Toolkit Setup

Learning Objectives

By the end of this session, students should be able to:

- **Install and sign into the three AI coding environments we will use all term** — Cursor (primary IDE), Claude Code (terminal-native AI), and Gemini CLI (free-tier scripting AI) — and know when to reach for each one.
- **Install a Skill and an MCP server into Claude Code / Cursor** and describe, in one sentence, what each extension lets the AI do on the student's behalf.
- **Recognise and apply the Delegation Rules** — know which operations to hand off to AI through a CLI or MCP server, and which (OAuth, DNS, provider-side credentials) the student must do with their own hands.

Core Topics

- Why "natural language is the source code" — course doctrine and how it changes the way students think about programming.
- The three AI environments: Cursor, Claude Code, Gemini CLI — when to use each.
- What MCP servers and Skills are, and how they turn an AI chat into a tool-using agent.
- Voice-first prompting with Wispr Flow (optional but strongly recommended).
- GitHub as the source-of-truth layer that every later week's deploy flow depends on.

Tools / Stack

Tool	Purpose this term
Cursor	Primary IDE; built-in AI chat; all weekly work lives here
Claude Code	Terminal-native AI with MCP + Skills; preferred for W8 PDF flow
Gemini CLI	Free-tier AI for throwaway scripts and quick explorations
Wispr Flow	Voice-to-text; dictate long prompts without typing
GitHub	Source of truth; auto-triggers every Vercel deploy
Vercel CLI	AI invokes it to deploy, read logs, manage env vars
Neon CLI + Neon MCP	AI invokes them to provision databases and auth (from W4)
Typst skill	AI invokes it to generate pixel-perfect PDFs (used in W8)

Your personal site — cumulative features through this week:

```
Wk 1 ·
```

Dev toolkit installed — Cursor, Claude Code, Gemini CLI, MCPs, Typst skill ← NEW

Session Plan

Time	Activity
0 – 15 min	**Recap & Check-in.** First class — no recap. 10-min icebreaker: everyone says what they think "programming" means before and after watching a Cursor demo.
15 – 40 min	**Concept Teaching.** The Delegation Rules. Why we let AI drive CLIs. What MCP servers are. How voice + AI changes keyboard-heavy work. Live comparison: Cursor vs Claude Code vs Gemini CLI on the same task.
40 – 75 min	**Live Demo.** Instructor opens Cursor from scratch, installs the Vercel MCP, and tells Cursor to "deploy an empty Next.js site to my Vercel". Class watches the AI call the Vercel CLI, approve, deploy, and return a live URL.
75 – 105 min	**Hands-On Lab.** Each student repeats the demo on their own laptop and ends with a live `*.vercel.app` URL they can share.
105 – 120 min	**Q&A + Wrap.** Troubleshooting as a group. Everyone leaves with working tools.

Hands-On Lab

Task. Get from "no dev tools installed" to "Cursor just deployed an empty Next.js site to Vercel under my account, and I watched it happen". This is the only lecture with heavier-than-usual manual setup — from Week 2 onwards, almost every step is a single spoken prompt.

> NOTE · Pacing note for students
>
> If you fall behind during steps 1–4 (installations), **don't stress**. The goal is to finish Step 7 (your first AI-driven Vercel deploy) before we wrap. Installers can finish while we continue — AI can start working even while Claude Code is still downloading in the background.

MANUAL **Step 1 · Manual (human only):**

Install **Cursor** from cursor.com. Open it, sign in with GitHub, and accept the terms. When Cursor asks "Would you like to import VS Code settings?" choose **Skip** — we want a clean slate.

Why manual: Cursor's installer + GitHub OAuth require clicks and consent screens that no CLI can perform for you.

MANUAL **Step 2 · Manual (human only):**

Install **Claude Code** by following the official quickstart. On Windows, run the installer and accept defaults. When it prompts for login, sign in with your Anthropic account.

Why manual: Anthropic's login flow opens a browser for consent — human required.

MANUAL **Step 3 · Manual (human only):**

Install **Gemini CLI** by following Google's readme. Run `npm install -g @google/gemini-cli` in any terminal (Cursor's built-in terminal works), then `gemini login` and sign in with your Google account.

Why manual: Installing Node + signing into Google both require consent screens.

TIP · Install Wispr Flow now if you're game

Wispr Flow (wisprflow.ai) lets you dictate prompts instead of typing. Everything from Step 5 onwards is a spoken sentence — Wispr Flow makes the rest of this term significantly faster.

PROMPT **Step 4 · Say to Cursor:**

I just installed you. Can you introduce yourself, tell me what tools you have access to (for example, can you run shell commands in a terminal, edit files on disk, and open URLs?), and create a brand-new empty folder called `technest-week-1` in my home directory? After creating it, open that folder as the Cursor workspace.

VERIFY **Step 5 · Verify:**

- Cursor reports its capabilities in its own words — take a moment to read them.
- A folder `technest-week-1` exists in your home directory.
- The Cursor window is now inside that folder (the file-tree pane shows it).

PROMPT **Step 6 · Say to Cursor:**

*Please install the **Vercel MCP server** so you can call Vercel's API on my behalf. Add it to my Cursor configuration. Once installed, tell me which tools it gives you — I want to see the list. If the MCP server needs me to paste a Vercel token, pause and tell me exactly where to get the token before continuing.*

MANUAL **Step 7 · Manual (human only):**

When Cursor pauses and asks for a Vercel token, open vercel.com/account/tokens in your browser, sign in, click **Create Token**, name it `cursor-mcp`, copy the token, and paste it back into Cursor's chat.

Why manual: Vercel never exposes account-creation tokens through any API — only the web dashboard.

VERIFY **Step 8 · Verify:**

- Cursor prints the list of Vercel tools it now has (create deployment, read logs, manage env vars, etc.).
- No error messages.

PROMPT **Step 9 · Say to Cursor:**

*Now prove to me that the delegation model works. Create a brand-new empty Next.js project inside this folder (call it `hello-technest`), initialise a git repo, push it to a **new** public GitHub repo under my account, and deploy it to Vercel using the Vercel MCP. When the deployment is live, open the production URL in my browser. Commit along the way with sensible messages. I'll just watch.*

VERIFY **Step 10 · Verify:**

- Your browser opens a live `hello-technest-*.vercel.app` URL showing the default Next.js starter page.
- A new repo appears on your GitHub account.
- Your local folder has a commit history Cursor wrote for you.

RECOVER **Step 11 · If stuck, say to AI:**

> *The deployment failed with a build error. Can you read the Vercel build logs through the MCP, find the root cause, fix it, and redeploy? I want the production URL to load without errors before we move on.*

PROMPT **Step 12 · Say to Claude Code:**

> *Open a terminal, `cd` into `hello-technest`, and confirm you can see the project. Then install the **Typst skill** from the Anthropic skills registry. When installed, tell me what prompts the skill is now able to handle.*

VERIFY **Step 13 · Verify:**

- Claude Code confirms the Typst skill is installed.
- It lists a handful of prompts like "generate a PDF", "compile a typst file". We'll use this in Week 8.

> **CAREER · Remember this table — you'll apply it every week**
>
> **Let the AI drive** provisioning, env-var plumbing, git, installs, edits, migrations, deploys, and PDF generation — all through CLIs, APIs, MCP servers, and Skills. **Do it yourself** only for OAuth consent screens, provider-side API-key creation, DNS, and the rare `.env.local` line where no CLI exists to set it. If you're ever unsure, **ask the AI which category a task falls into** — this is a perfectly good prompt: "Is this something you can do via a CLI, or do I need to do it myself?"

Weekly Assignment

Build / Implement.

- One live `*.vercel.app` URL deployed entirely through AI-invoked CLIs.
- A screenshot of your Cursor conversation showing the prompt that kicked off the deploy.
- A second screenshot showing the Vercel MCP tool-call output.

Requirements.

- The repo must exist on your GitHub account.
- The deploy must have been triggered by AI, not by you running `vercel deploy` by hand. (This is the whole point of the term.)
- You must have both Cursor and Claude Code installed, and the Typst skill loaded in Claude Code.

Submission. Post the three artifacts (live URL + two screenshots) in the course Slack channel before the start of Week 2.

Resources

Docs

- Cursor docs — getting started
- Claude Code — quickstart
- Vercel MCP server — readme
- Wispr Flow — setup

Videos

- The instructor's recorded "first AI deploy" demo
- Fireship — "Cursor in 100 seconds"

Repos

- `her-waka/tutorial/vibe-coding` — deeper dive for curious students
- `anthropics/skills` — the Skill registry

Real-World Application

Every modern AI-engineering job interview expects you to **demo a workflow**, not recite syntax. Hiring managers in 2026 want to see a candidate open Cursor, describe a feature, and ship it live in 10 minutes — because that's how senior engineers on their teams already work. Today's lab is exactly that interview artefact: a clip of you dictating a sentence and getting a live URL. Save the screen recording.

> CAREER
>
> LinkedIn bio upgrade for next week: **"Full-stack AI developer · I prompt, I ship · v.ercel.app/my-project"**.

Challenges & Tips

- **"Cursor is asking me to confirm each shell command."** Good — in a classroom setting keep manual approvals on. By Week 3 you can enable auto-approve for low-risk commands if you want faster iteration.
- **"The MCP server says 'authentication required'."** Your Vercel token is missing or expired. Re-paste a fresh token from `vercel.com/account/tokens`.
- **"Gemini CLI says I've hit a rate limit."** Use Cursor or Claude Code for the rest of the class. Gemini's free tier resets hourly.
- **"I can't tell whether I should do the step myself or ask the AI."** Ask: **"Can you do this through your tools, or do I need to do it?"** The AI will answer honestly.
- **"My prompts aren't specific enough — the AI keeps asking follow-ups."** Watch the instructor's prompts in this lecture carefully. Good prompts name the **tool** the AI should use, the **inputs** it has, and the **signal that the task is done**.

> STUCK? RESCUE · If you're completely stuck
>
> Paste your last prompt **and** the AI's last response into a fresh Claude Code window and say: **"This didn't work. What's the most likely next step to try, and why?"** Treating a stuck AI session like a debuggable object — that you can restart, inspect, and redirect — is the most important skill of this course.

Instructor Notes (Internal)

- **Expect 60–70% of the class to finish Step 10 inside the 2-hour window.** That's fine. Anyone still wrestling with installations after the demo should pair with a finished classmate for the deploy phase.

- **Pacing hazard: the Vercel token step.** Some students land on the wrong token page (team vs personal). If they paste the wrong token, the MCP call silently fails. Have them read the token's permissions aloud to catch it.
- **Don't skip the "introduce yourself" prompt in Step 4.** Students underestimate how much knowing what the AI can do changes what they ask it. Make them read its reply out loud.
- **Resist explaining what npm/node is.** If the AI says `pnpm install`, students don't need to know what pnpm is. They need to know that **Cursor knows**. Protect the delegation ritual — it's what separates this course from any other "learn to code" class.
- **If a student has a strong Python background**, they may push to use a Python stack instead. Gently redirect: the whole term is cumulative; W4 Neon + W5 Blob + W7 Slack all assume Next.js + Vercel. They can build Python side-projects later.

Student Feedback

Challenges Observed

Extra Information

MEET YOUR INSTRUCTOR

Chan Meng

Full-Stack Engineer · AI Agent Builder · Minimalist

> *"Subtraction for life, addition for thought."*

AI AGENT BUILDER

AGENTIC SYSTEMS · LLM INTEGRATION

FEMTECH INNOVATOR

MINIMALIST CODE · UN CSW 69 SPEAKER

Chan builds **production-grade AI agents** at the intersection of **agentic systems, LLM integration, and women's health**. Between China and New Zealand, she ships systems that go to work for real users at real companies — and she believes code, like life, becomes most powerful when you subtract what isn't essential. A **Master of Applied Computing with Distinction** from Lincoln University, NZ (Dean's List), she was featured at **UN Women CSW69** for her work on AI and gender equality.

Currently · Founding Engineer at Gavigo · Senior Full-Stack Engineer at She Sharp (NZ's leading women-in-STEM nonprofit, 2,200+ members) · CTO of FemTech Weekend · Open Source Contributor to CopilotKit (24.6k★). For the full project portfolio and code, visit github.com/ChanMeng666.

Stay in the loop

NEWSLETTER · RECOMMENDED

"What's Shipping in AI" — a weekly curated digest, every Monday. chanmeng.org/#newsletter

LINKEDIN
chanmeng666

GITHUB
ChanMeng666

PORTFOLIO
chanmeng.org

EMAIL
chanmeng.dev@gmail.com

Building tools that matter. Let's connect.