WEEK 8 JOB LIFE

Typst PDF Automation for Job Applications

Learning Objectives

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

- **Invoke the Typst skill through Claude Code** to turn structured profile data in Neon into a pixel-perfect, downloadable PDF — without writing a single line of Typst by hand.
- **Unify one data source (Neon `profile`) across two outputs** — the portfolio site and the CV / cover letter PDFs — proving that **data first, documents later** is a senior engineering habit.
- **Ship one-click "Download CV" and "Generate cover letter for [job description]" buttons** on your live portfolio that produce custom PDFs on demand.

Core Topics

- Typst, in one paragraph: think "Markdown with superpowers that compiles to PDF". Same tool this lecture handout is compiled with.
- Skills in Claude Code — the Typst skill is shipped by Anthropic and knows how to generate common documents from structured input.
- Data-driven documents: why storing your CV fields in Neon (not Google Docs) makes the CV AI-editable forever.
- Cover letters are the perfect LLM task: structured input (your profile + a job description) + structured output (a formatted PDF).
- The meta moment: students realise this lecture and their CV are both made with the same tool.

Tools / Stack

Tool	Role this week
Typst CLI	Compiles `.typ` files to PDFs — fast and deterministic.
Typst skill (Claude Code)	AI writes the `.typ` template and drives the compile.
Neon Postgres + Drizzle	Stores your `profile` row: one source of truth.
Claude API (optional)	Generates tailored cover-letter prose from a JD.
Vercel serverless function	Runs Typst compilation on demand and streams PDF.

Your personal site — cumulative features through this week:

Wk 1 · Dev toolkit installed — Cursor, Claude Code, Gemini CLI, MCPs, Typst skill

Wk 2 · Personal website deployed to Vercel

Wk 3 · AI chat widget (Gemini 2.5 Flash) emulating you

Wk 4 · Neon Postgres + Neon Auth + Drizzle guestbook

Wk 5 · Vercel Blob image uploads in guestbook

Wk 6 · MDX blog system with RSS

Wk 7 · Real-time Slack notifications on contact form

Wk 8 · **Typst-powered CV & cover-letter PDFs from Neon profile** ← NEW

Session Plan

Time	Activity
0 – 15 min	**Recap & Check-in.** Who got a real contact form submission this week from someone outside the class? (There's usually at least one.)
15 – 40 min	**Concept Teaching.** Why Typst is better than Word / Pages / Docs for generated documents. Why your CV should live in Postgres. How the Typst skill works — you describe, it writes `.typ`, CLI renders, you verify. What makes a good system prompt for cover-letter generation.
40 – 75 min	**Live Demo.** Instructor generates her own CV from her Neon profile row in real time via Claude Code. Then feeds a fake job description and watches a tailored cover letter appear in 15 seconds.
75 – 105 min	**Hands-On Lab.** Students do the same with their own data. Bonus: the tailored-cover-letter button stays on their site forever, ready for every real job application.
105 – 120 min	**Q&A + Wrap.** Every student shows their generated CV. Friendly critique round.

Hands-On Lab

Task. By the end of class your portfolio site has two new buttons: **Download CV** (produces a polished one-page CV from your Neon profile data) and **Generate cover letter** (paste a job description, get a custom PDF). Both compile via the Typst skill.

PROMPT **Step 1 · Say to Cursor:**

> *Let's add a `profile` table to Neon that will be the source of truth for my CV data. Design the Drizzle schema for this. A **single** profile row per user — meaning I'll be the only populated row, but future multi-user versions should support more. Include:*
>
> - `user_id` *(FK → users.id, unique) — one-to-one with a user*
> - `headline` *(text, 1 line, e.g., "CS undergrad building AI agents")*
> - `summary` *(text, 3–5 sentences)*
> - `email` , `phone` *(nullable),* `location_city` , `location_country` , `website` , `github` , `linkedin` *(all text, nullable)*
> - `education` *(JSON: array of* `{school, degree, start, end, notes}` *)*
> - `experience` *(JSON: array of* `{org, role, start, end, highlights: []}` *)*
> - `projects` *(JSON: array of* `{name, url, one_liner, stack: []}` *)*
> - `skills` *(JSON: array of strings)*
> - `updated_at` *(timestamp auto-updates)*
>
> *Generate + push the migration. Also add an admin-only `/edit-profile` page on my site where I can edit each field with a textarea or JSON editor. Only my signed-in user can access this page.*

VERIFY **Step 2 · Verify:**

- Migration applied; `profile` table exists.
- Visiting `/edit-profile` on localhost while signed in as you shows the editor.
- Visiting while signed out — or as a different user — redirects to home.
- Save a minimal profile (headline + summary + one education + one experience). Confirm the row is in Neon.

PROMPT **Step 3 · Say to Claude Code:**

> *Please confirm the Typst skill is installed. List the kinds of prompts it handles. If it's not installed, install it from the official Anthropic skills registry, then confirm Typst CLI (`typst`) is on my PATH by running `typst --version`.*

VERIFY **Step 4 · Verify:**

- Claude Code confirms the Typst skill is installed.
- `typst --version` prints a version number.

PROMPT **Step 5 · Say to Claude Code:**

> *I want to generate a one-page CV from my Neon profile row. Please:*
>
> 1. *Read my `profile` row from the database (use the `DATABASE_URL` from my `.env.local`).*
> 2. *Use the Typst skill to write a clean single-column CV template at `cv/template.typ` with: name + headline in the header, summary paragraph, then sections for Education, Experience, Projects, Skills. Use a tasteful serif body + sans headings. Match the accent colour I chose in Week 2.*
> 3. *Write a small Node script at `cv/build.ts` that reads the DB row, fills the Typst template, and shells out to `typst compile cv/template.typ cv/my-cv.pdf`.*
> 4. *Run the script once. Open the resulting PDF and paste me a screenshot / description of how it looks. If anything feels off, iterate — but do the iteration through the skill, never by hand-editing the `.typ`.*

VERIFY **Step 6 · Verify:**

- `cv/my-cv.pdf` exists, is one page, and contains your real profile data.
- Headings, spacing, typography all look professional.
- The accent colour matches your site.

PROMPT **Step 7 · Say to Cursor:**

Now wire this into the site:

1. *Add a serverless function at `/api/cv` that regenerates my CV PDF on demand — same flow as the Node script, but running inside a Vercel function. Stream the PDF response back with `Content-Type: application/pdf` and `Content-Disposition: attachment; filename="ChanMeng-CV.pdf"` (substitute my real name from the profile row).*
2. *Add a **Download CV** button on the homepage hero next to my existing social links. Click triggers the download.*
3. *Rate-limit the route to 20 requests per hour per IP — CV generation is cheap but let's not waste compute. Use the same rate-limit pattern we set up for the contact form in Week 7.*
4. *Make sure `typst` is available in the Vercel function runtime. If it's not (it's not by default), use the approach the Typst skill recommends — likely shipping a pre-built binary or using a fly.io / Railway side-service.*

VERIFY **Step 8 · Verify:**

- Localhost: click **Download CV** — a PDF downloads with your real data.
- Production after deploy: same button, same result.
- Click it 21 times in quick succession: 21st returns 429.

PROMPT **Step 9 · Say to Claude Code:**

*Now the cover letter generator. I want a form on my site where I paste a job description, optionally a company name, and click **Generate cover letter**. It produces a custom one-page PDF, written in my voice (using `content/persona.md` as the voice guide), referencing **actual** items from my profile's experience and projects arrays.*

Here's how to build it:

1. *Write a Typst template at `cv/cover-letter.typ` — a one-page letter layout: sender block (me), recipient block (company name + "Dear Hiring Manager" if unknown), body paragraphs, sign-off. Same typography language as the CV.*
2. *Write a server action that: a. Takes the JD + company name + (optional) specific role. b. Calls the Claude API with a system prompt: **"You write a crisp one-page cover letter in the voice of [persona.md]. Use only facts from the profile JSON provided — don't fabricate experience. Structure: opening hook, why this role, two specific project / experience references relevant to the JD, sign-off."** Return structured JSON: `{ opening, body1, body2, body3, signoff }`. c. Fills the Typst template with this JSON. d. Compiles with `typst compile` and streams the PDF back.*
3. *Add a **Generate cover letter** section on `/edit-profile` (signed-in only): two inputs (company, JD), one textarea, a **Generate** button. After success, the PDF downloads.*
4. *Log every generation to a new `cover_letters` table (company, role, created_at, body length). I want to be able to look back at what I applied to.*

Set `ANTHROPIC_API_KEY` in both `.env.local` and Vercel prod.

MANUAL **Step 10 · Manual (human only):**

Open console.anthropic.com/settings/keys, sign in, click **Create Key**, name it `technest-cover-letter`, copy, and paste back to Claude Code / Cursor.

Why manual: Anthropic's console requires human login + terms acceptance before key generation — standard for every major LLM provider.

VERIFY **Step 11 · Verify:**

- Paste a real job description (e.g., one from LinkedIn you might actually apply to).
- Click **Generate cover letter**. Within ~20 seconds, a 1-page PDF downloads.
- The letter references real items from your profile (not invented).
- The voice matches your `persona.md`.
- A row appears in `cover_letters` table.

RECOVER **Step 12 · If stuck, say to AI:**

The cover letter generates but the typography is cramped — the body text bleeds to a second page. Can you work with the Typst skill to tweak the template so everything stays on one page, even for longer JDs? Prefer condensing paragraphs over shrinking fonts below 10 pt.

PROMPT **Step 13 · Say to Cursor:**

> *Commit everything in logical chunks (profile schema, CV generation, cover letter generation as three separate commit groups), push, and confirm the Vercel deploy succeeds. Test both the CV download and the cover letter generation on the live production URL. Final check: add a **CV / Cover Letter** link to the site header next to **Blog · Contact**.*

VERIFY **Step 14 · Verify:**

- Live site shows **Blog · Contact · CV / Cover Letter** in the header.
- Live **Download CV** produces your real PDF.
- Live **Generate cover letter** with a real JD produces a real, well-written, 1-page PDF in your voice.
- Your Neon dashboard shows the `profile` row and any `cover_letters` rows from your tests.

CAREER · You just built a real job-search tool.

Most students in this course will use this button **dozens of times** over the next year. Every time you apply to a role, paste the JD, click **Generate**, tweak one paragraph, submit. That's 10 minutes per application, not 60.

TIP · The meta moment

Notice that this lecture handout you're holding was also compiled with the Typst CLI — by your instructor, through the same Typst skill you just used. **The tool that teaches the tool is the tool.** Remember this symmetry when you build your own teaching material later.

Weekly Assignment

Build / Implement.

- Live **Download CV** button that produces a PDF from your Neon profile.
- Live **Generate cover letter** flow that produces a custom PDF from a pasted JD.
- At least one real cover letter you'd actually send, submitted as your deliverable.

Requirements.

- Your profile data lives in Neon, not in a hand-written `.typ` file.
- The Typst template is under version control.
- A screenshot of the generated CV.
- A screenshot of a generated cover letter (redact company name if sensitive).
- 30-second screen recording of the end-to-end cover-letter flow on your live site.

Submission. Live URL + all three artifacts in Slack before the Week 9 capstone kickoff.

Resources

Docs

- Typst — getting started + language reference

Videos

- Instructor demo: "Your CV as code in 20 minutes"

Repos

- `typst/typst` — compiler source

• Anthropic Skills — Typst skill
• Anthropic Claude API — getting started
• Vercel serverless functions — binary dependencies
• `anthropics/skills` — Typst skill examples
• `her-waka/tutorial/professional-pdf` — original tutorial

Real-World Application

Hiring will keep compressing timelines. The candidate who can send a **specific, thoughtful, typographically clean** cover letter within an hour of seeing a job post will out-compete the candidate who spends two days on it — every time. You now have that machine. Keep it. Refine it. Never pay for a CV builder SaaS again.

> **CAREER**
>
> On your live site today, the **CV / Cover Letter** link is gold. Recruiters who visit and click it get a one-page PDF of your real, current data — and if they paste their JD, they can see a cover letter from you written **about their role** in 20 seconds. That's an asymmetric hiring signal. Use it.

Challenges & Tips

- **"Typst isn't available in the Vercel function runtime."** Two options — (a) ship a pre-built binary in the function bundle (5 MB), (b) offload compile to a tiny Fly.io / Railway side-service. The Typst skill knows both patterns — ask it to pick the one that fits your Vercel plan.
- **"The cover letter hallucinated a company I didn't work for."** Strengthen the system prompt: **"Cite only items explicitly present in `experience` or `projects` arrays. If there's nothing relevant, say 'While I haven't worked at X directly, …' and pivot to the closest real item."**
- **"The PDF has a weird font."** Vercel's runtime may not have Inter / your chosen font. The Typst skill handles this by embedding fonts in the bundle — ask it: **"Ensure the CV uses an embedded font so it renders identically on my machine and on Vercel."**
- **"My cover letter has the wrong accent colour on Vercel."** You hardcoded a theme value in two places. Ask **"Factor the theme into one Typst variable and reference it from both CV and cover-letter templates."**
- **"Claude is too expensive after 10 test generations."** Ask Cursor to cache identical (JD, company) pairs for 24 hours — most of your testing is with the same inputs.

> **STUCK? RESCUE**
>
> If Typst compilation fails in production with a cryptic error, re-run the exact same input locally — 95% of the time Typst produces the same error locally and you can ask Claude Code (with the Typst skill) to fix it. If it only fails on Vercel, it's a runtime / font / binary issue and the skill knows how to diagnose.

Instructor Notes (Internal)

- **This class runs long for ~50% of the room.** CV design choices are a natural time sink. Put a 45-minute cap on CV polish; students must move to cover-letter work even if they're "not happy with the CV yet".
- **The "meta moment" callout lands emotionally.** Hold for it — let students look at the handout in their hands and realise **this was built by the same tool they just used**. It reframes the whole term.
- **Have a real JD ready.** Paste the same JD (e.g., an actual junior AI-engineer posting) into every student's generator so the comparison is apples-to-apples.
- **Showcase two generated cover letters at the end.** Cover the name, read the body aloud, have the class guess whose it is. It's a surprisingly moving exercise — students see their own voice preserved by the tool.
- **Next week: capstone kickoff.** Remind students that capstone teams form at the start of Week 9 — suggest they look around the classroom **today** for people whose builds they admire.

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.