WEEK 5 FUN LIFE

Image Uploads with Vercel Blob

Learning Objectives

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

- **Provision a Vercel Blob store by prompt** and have the AI wire it into an existing Next.js app end-to-end, including server-only upload tokens.
- **Describe a file-upload UX in English** — drag-and-drop, preview, size cap, content-type filter — and get a working implementation back.
- **Verify a binary round-trip** (pick a file → upload → persist URL in Neon → render from Vercel Blob CDN) as a first-class engineering skill.

Core Topics

- What blob storage is, at the level a beginner needs: URLs that point to files, not to HTML.
- Why you never let the browser upload directly to your server (hint: memory cost + attack surface) — and why signed upload tokens solve both.
- The one-week-ahead mental model: by Week 7 your guestbook entries will trigger Slack notifications. Keeping the schema clean today pays off then.

Tools / Stack

Tool	Role this week
Vercel Blob	File storage; CDN-backed URLs; free tier is generous.
`@vercel/blob` SDK	What your server action uses to issue upload tokens.
Drizzle ORM	Adds a new `image_url` column to existing `guestbook_messages`.
Cursor	Writes everything; you describe + verify.

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** ← NEW

Session Plan

Time	Activity
0 – 15 min	**Recap & Check-in.** Who had a friend / classmate sign up and post a guestbook message this week? 30 seconds each: share a favourite.
15 – 40 min	**Concept Teaching.** Server-generated upload tokens. Why the browser talks directly to Vercel Blob (not via your server). Content-type filtering as a security layer. How a URL-in-the-DB beats a blob-in-the-DB.
40 – 75 min	**Live Demo.** Instructor extends her guestbook to accept a meme. Watch a 10 MB image become a 40 KB thumbnail + 800 px full-size upload in under 5 seconds.
75 – 105 min	**Hands-On Lab.** Students extend their own guestbooks. The "post a meme" moment is the most fun class of the term so far — expect a lot of laughter.
105 – 120 min	**Q&A + Wrap.** Anyone whose upload silently fails gets debugged live — usually a content-type mismatch.

Hands-On Lab

Task. By the end of class a signed-in visitor can attach an image to their guestbook message. Photos show up inline above the text, clicking opens full size, and your Vercel Blob dashboard shows each upload.

PROMPT **Step 1 · Say to Cursor:**

> *Please enable **Vercel Blob** on my `my-portfolio` project via the Vercel CLI. When the store is created:*
>
> 1. *Add the resulting `BLOB_READ_WRITE_TOKEN` to my `.env.local` and to Vercel (production + preview) via the CLI.*
> 2. *Install `@vercel/blob` in my Next.js project.*
> 3. *Add a new column `image_url` (text, nullable) to my existing `guestbook_messages` table. Generate the Drizzle migration, show it to me, then push it to Neon.*
> 4. *Confirm the new column is on the table and the token is set in all three environments.*

VERIFY **Step 2 · Verify:**

- `.env.local` contains `BLOB_READ_WRITE_TOKEN=...`.
- `vercel env ls` shows the token for both production and preview.
- Drizzle migration SQL includes `ADD COLUMN image_url`.
- A Cursor-run `SELECT column_name FROM information_schema.columns WHERE table_name = 'guestbook_messages'` (or equivalent) lists `image_url`.

PROMPT **Step 3 · Say to Cursor:**

> *Add a Next.js route handler at `/api/upload` that uses `@vercel/blob`'s `handleUpload` to issue **client upload tokens**. Constraints:*
>
> - *Only signed-in users can request a token. Reject with 401 if no session.*
> - *Only accept content types `image/png`, `image/jpeg`, `image/webp`, `image/gif`.*
> - *Maximum file size: 4 MB.*
> - *Generated filenames should be prefixed with the user's ID so I can clean up per user later.*
>
> *Do not accept uploads from the browser to the Next.js server directly — the browser uploads straight to Vercel Blob using the token your route issues.*

VERIFY **Step 4 · Verify:**

- Cursor explains (in chat) the two-step flow: browser asks server for token, browser uploads to Blob, browser sends the resulting URL back to a second server action.
- `curl` (run by Cursor) against `/api/upload` without a session returns 401.

PROMPT **Step 5 · Say to Cursor:**

> *Extend the guestbook form I built in Week 4 to support an optional image:*
>
> - *Add a small paperclip / image-icon button next to the **Post** button.*
> - *Clicking opens the file picker. After pick, show a **preview thumbnail** immediately (client-side, before upload). The user can remove the image and pick a different one before posting.*
> - *On **Post**, if an image is attached: first request a token from `/api/upload`, upload directly to Vercel Blob, get the returned public URL, then submit the message server action with both `body` and `imageUrl`.*
> - *During upload, disable the Post button and show a small spinner next to the thumbnail.*
> - *If the upload fails, keep the message text in the textarea (don't lose it) and show an error toast: **"Image upload failed — try again or post without an image."***
>
> *In the server action, persist `imageUrl` into the new column. In the messages list, render the image above the body text with a max height of 240 px, rounded corners matching the site's theme, and a subtle shadow. Clicking opens it full size in a lightbox overlay with Escape to close.*

VERIFY **Step 6 · Verify:**

- Localhost: attach a PNG, see the thumbnail, post — it appears in the list with the image.
- Refresh — still there.
- In another browser (incognito), you can see the image without signing in (the URL is public).
- Check your Vercel Blob dashboard — the file is there, prefixed with your user ID.

PROMPT **Step 7 · Say to Cursor:**

> *Let's harden the upload a little. Add three things:*
>
> 1. *Server-side check that the file reaching the token step is actually one of the allowed content types (inspect the magic bytes, not just the `Content-Type` header).*
> 2. *Client-side guard: if the picked file is over 4 MB, compress it in the browser (using `browser-image-compression` or similar) before upload. Show a toast: **"Your 8 MB photo was compressed to 1.2 MB. Tap to post."***
> 3. *A tiny **delete** button (×) on each of my **own** messages. Clicking prompts confirm, then removes the row and the blob. Other people's messages don't get a delete button.*

VERIFY **Step 8 · Verify:**

- Try to upload a 20 MB photo — client compresses it, toast appears, post succeeds.
- Rename a `.exe` to `.png` and try to upload — server rejects it.
- Delete one of your own messages — it disappears; check the Blob dashboard to confirm the file is gone too.
- You can't see a delete button on someone else's message.

PROMPT **Step 9 · Say to Cursor:**

> *Commit all changes in reasonable chunks, push to GitHub, and let the Vercel deploy complete. When it's live, sign in on the production URL, post a meme with a clever caption, and confirm every part of the flow works in production.*

VERIFY **Step 10 · Verify:**

- Production guestbook now supports image uploads.
- Your Vercel Blob dashboard shows production uploads.
- Neon dashboard shows rows with populated `image_url` columns.

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

> *The production upload is returning `403 Forbidden` but works locally. Please check whether `BLOB_READ_WRITE_TOKEN` is actually set in Vercel's production environment (not just preview), and fix it if it's missing. Redeploy and confirm.*

> **TIP · Keep your browser console open during upload**
>
> For the whole upload flow, open DevTools → Network. Watch the two distinct requests: first `POST /api/upload` (tiny, just metadata) then `POST https://*.blob.vercel-storage.com/...` (the actual file bytes). Understanding this shape matters — you'll see it in every modern upload flow for the rest of your career.

Weekly Assignment

Build / Implement.

- Guestbook on your live site now accepts image uploads.

- At least five real images uploaded (from your friends / classmates).
- Delete works on your own messages only.

Requirements.

- Direct-to-Blob upload (no image bytes traversing your Next.js server).
- Content-type + size limits enforced.
- A screenshot of your Vercel Blob dashboard showing prefixed filenames.
- A 30-second screen recording of the upload flow.

Submission. Live URL + screenshot + recording in Slack before Week 6.

Resources

Docs

- Vercel Blob — client uploads
- `@vercel/blob` — `handleUpload` and `put`
- `browser-image-compression` — npm

Videos

- Instructor demo: "Direct uploads in 8 minutes"

Repos

- `vercel/examples/blob` — reference patterns

Real-World Application

File uploads are in every product you'll build. Getting the **direct-to-bucket** pattern into your fingers this early — instead of the naive "ship bytes through my server" pattern — saves you from 90% of outages you'd otherwise cause in production. This is exactly how companies like Figma, Notion, and Linear architect their uploads.

> **CAREER**
>
> Add this line to your resume: **"Implemented direct-to-CDN uploads with signed tokens, enforced server-side MIME validation, and client-side compression — handling 20 MB inputs at 1.2 MB payloads."** This sentence is interview gold.

Challenges & Tips

- **"The upload works, but the image is rotated 90° on mobile."** iPhones embed EXIF orientation. Ask Cursor **"Strip EXIF or respect orientation during compression."**
- **"`fetch` to /api/upload returns 413."** Next.js body parser limit. Ask Cursor **"Raise the route's body size limit, but only for the upload route — the rest of the site stays at defaults."**
- **"Images take forever to load on the guestbook."** You're rendering the original size. Ask **"Add Next.js Image component with appropriate sizes + placeholder blur."**
- **"The delete button works but the Blob file stays."** Two-step delete was missed. Say **"Delete the Blob file** before **removing the row; if the Blob delete fails, don't remove the row."**
- **"Blob free tier exhausted."** You uploaded too many test files. Ask Cursor **"Clean up all Blob files older than 24 hours whose DB rows are gone."**

> **STUCK? RESCUE**

> If a file disappears silently — no errors, just never shows up — add temporary logging: **"Add console.log calls at every stage of the upload flow so I can see exactly which step drops the file."** Remove the logs once the bug's found.

Instructor Notes (Internal)

- **Have one MP4 and one .exe ready to test content-type guards.** If all students only upload clean PNGs, the security lesson falls flat.
- **The delete step is where students learn about referential integrity.** Some will get a foreign-key error when deleting a user. Good — it teaches the lesson. Don't pre-empt with cascade rules.
- **Photo compression on the client is the "wow" step.** Make everyone upload a 20 MB phone photo and watch it compress to 1 MB before their eyes. Screenshot the before/after — it's LinkedIn material.
- **Gently warn students about Blob costs in real usage.** Free tier ends fast if the site goes semi-viral. The skill of estimating storage cost is one they can practise now.
- **If the class runs ahead**, extend with: add a tiny thumbnail column (`image_thumb_url`) generated server-side at 200 px wide for faster list rendering. This is a 10-minute prompt.

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.