

Les 18 — Lesopdracht

Kies één modaliteit, bouw een mini-demo

Vak: AI-Assisted Development

Duur: 30 min in-class

Duur: 30 min in-class

Doel

Pak één van de modaliteiten uit deze les en bouw daar een minimale werkende demo van. Voice, vision, image gen, of local LLM — jouw keuze.

Opties

Optie 1 — Voice transcriptie

Bouw een mic-button + transcribe. Werkende voice → text in een Next.js page.

```

// app/voice/page.tsx
"use client";
import { useState, useRef } from "react";

export default function VoicePage() {
  const [text, setText] = useState("");
  const [recording, setRecording] = useState(false);
  const recorder = useRef<MediaRecorder | null>(null);
  const chunks = useRef<Blob[]>([]);

  async function start() {
    const stream = await navigator.mediaDevices.getUserMedia({ audio: true });
    recorder.current = new MediaRecorder(stream);
    recorder.current.ondataavailable = (e) => chunks.current.push(e.data);
    recorder.current.onstop = async () => {
      const blob = new Blob(chunks.current, { type: "audio/webm" });
      chunks.current = [];
      const fd = new FormData();
      fd.append("audio", blob);
      const res = await fetch("/api/transcribe", { method: "POST", body: fd });
      const { text } = await res.json();
      setText(text);
    };
    recorder.current.start();
    setRecording(true);
  }

  function stop() {
    recorder.current?.stop();
    setRecording(false);
  }

  return (
    <main className="p-8">
      <button onClick={recording ? stop : start} className="bg-blue-600 text-white px-6"
        {recording ? "Stop" : "Start opnemen"}
      </button>
      <p className="mt-6">{text}</p>
    </main>
  );
}

```

app/api/transcribe/route.ts:

```
import { experimental_transcribe as transcribe } from "ai";
import { openai } from "@ai-sdk/openai";

export async function POST(req: Request) {
  const fd = await req.formData();
  const audio = fd.get("audio") as Blob;
  const result = await transcribe({
    model: openai.transcription("whisper-1"),
    audio: new Uint8Array(await audio.arrayBuffer()),
  });
  return Response.json({ text: result.text });
}
```

Optie 2 — Vision foto-analyse

Bouw een upload-form + analyse. User upload foto → AI beschrijft.

```
// app/vision/page.tsx
"use client";
import { useState } from "react";

export default function VisionPage() {
  const [desc, setDesc] = useState("");
  const [loading, setLoading] = useState(false);

  async function analyze(e: React.ChangeEvent<HTMLInputElement>) {
    const file = e.target.files?.[0];
    if (!file) return;
    setLoading(true);
    const fd = new FormData();
    fd.append("image", file);
    const res = await fetch("/api/analyze", { method: "POST", body: fd });
    const { description } = await res.json();
    setDesc(description);
    setLoading(false);
  }

  return (
    <main className="p-8">
      <input type="file" accept="image/*" onChange={analyze} />
      {loading && <p>Bezig...</p>}
      <p className="mt-6">{desc}</p>
    </main>
  );
}
```

app/api/analyze/route.ts:

```
import { generateText } from "ai";
import { openai } from "@ai-sdk/openai";

export async function POST(req: Request) {
  const fd = await req.formData();
  const image = fd.get("image") as File;
  const buffer = await image.arrayBuffer();

  const result = await generateText({
    model: openai("gpt-4o"),
    messages: [{
      role: "user",
      content: [
        { type: "text", text: "Beschrijf wat er op deze foto staat in 3 zinnen." },
        { type: "image", image: new Uint8Array(buffer) },
      ],
    }],
  });

  return Response.json({ description: result.text });
}
```

Optie 3 — Image generation

Bouw een prompt-input → generate → preview.

```
// app/generate/page.tsx
"use client";
import { useState } from "react";

export default function GeneratePage() {
  const [prompt, setPrompt] = useState("");
  const [imageUrl, setImageUrl] = useState("");
  const [loading, setLoading] = useState(false);

  async function generate() {
    setLoading(true);
    const res = await fetch("/api/generate", {
      method: "POST",
      body: JSON.stringify({ prompt }),
    });
    const { image } = await res.json();
    setImageUrl(`data:image/png;base64,${image}`);
    setLoading(false);
  }

  return (
    <main className="p-8 max-w-2xl mx-auto">
      <textarea value={prompt} onChange={(e) => setPrompt(e.target.value)}
        placeholder="Beschrijf je afbeelding..." rows={3}
        className="w-full border p-2 rounded" />
      <button onClick={generate} disabled={loading}
        className="mt-3 bg-blue-600 text-white px-4 py-2 rounded">
        {loading ? "..." : "Generate"}
      </button>
      {imageUrl && <img src={imageUrl} alt="" className="mt-6 rounded" />}
    </main>
  );
}
```

app/api/generate/route.ts:

```
import { experimental_generateImage as generateImage } from "ai";
import { fal } from "@ai-sdk/fal";

export async function POST(req: Request) {
  const { prompt } = await req.json();
  const result = await generateImage({
    model: fal.image("fal-ai/flux/schnell"),
    prompt,
    size: "1024x1024",
  });
  return Response.json({ image: result.image.base64 });
}
```

pnpm add @ai-sdk/fal — gratis tier op fal.ai.

Optie 4 — Local LLM met Ollama

Run llama lokaal, chat ermee via je app.

```
brew install ollama
ollama pull llama4:8b
ollama serve
```

```
pnpm add ollama-ai-provider
```

```
// app/local/page.tsx + /api/local-chat
import { createOllama } from "ollama-ai-provider";
import { streamText } from "ai";

const ollama = createOllama();

export async function POST(req: Request) {
  const { messages } = await req.json();
  const result = streamText({
    model: ollama("llama4:8b"),
    messages,
  });
  return result.toUIMessageStreamResponse();
}
```

Frontend kan dezelfde useChat gebruiken als in Les 12.

Eisen

Kies één optie. Aan het eind:

- ☐ Demo werkt lokaal
- ☐ Eén feature compleet (transcribe, analyze, generate, of chat)
- ☐ Geen errors in console
- ☐ Optioneel: gedeeld in Teams met screenshot

Tijdsindeling (30 min)

Stap	Tijd
1 — Project setup	5 min
2 — Backend route	10 min
3 — Frontend page	10 min
4 — Test + debug	5 min

Veelvoorkomende problemen

Symptoom	Oplossing
MediaRecorder undefined	Browser support — Chrome werkt, Safari iets anders
Fal API key	Sign up + create key — gratis
Ollama "connection refused"	ollama serve runnen + check :11434
Vision error	Image te groot? Resize to <4MB
Transcribe error op localhost	Gebruik buffer (Uint8Array) niet Blob direct

Klaar?

Dat was de laatste lesopdracht van de leerlijn. Voor je eindopdracht — wat je nu kunt:

- Multi-modal features toevoegen (voice/vision)
- Lokale modellen voor privacy
- Image generation voor branding
- Edge AI voor performance

Veel succes!