

## Oblig 2 – anvendt AI

I denne oppgaven hadde jeg en del utfordringer både med sykdommer og hva jeg valgte av prosjekt. Først prøvde jeg å lage en historie/fortelling, men da jeg fikk en ukes utsettelse ved influensa, så valgte jeg å bytte prosjekt til noe som jeg likte mye mer, og noe jeg faktisk har en lidenskap for.

I denne oppgaven har jeg valgt å generere en type treningsplan-generator. En applikasjon hvor du legger inn enkel info i et skjema om dine mål og forutsetninger, og vil få en treningsplan generert spesifikt for deg. «**Idrettsutøvere trenger skreddersydde treningsprogrammer tilpasset mål, utstyr og eventuelle skader, men mangler tilgjengelig ekspertise» er problemstillingen** som chat gpt genererer ved idedrøftingen jeg hadde ved å finne nytt prosjekt. Det må understreses at ved idemyldringen og dannelsen av det nye prosjektet så brukte chat gpt sitt minne om tidligere samtaler med meg, den vet at jeg er fysisk aktiv og trener i hverdagen, både styrke og fotball 😊

For å lage dette produktet vil jeg bruke OpenAIs Chat GPT 4o som språkmodell, Github Copilot for kodegenerering, og jeg skulle opprinnelig ha sora til å generere video-illustrasjoner til øvelsene, men dette viste seg å være problematisk. Til slutt endte jeg med å bruke bildegeneratoren DALL-E for å lage eventuelle treningsillustrasjoner!

Selve grunn-utviklingen av appen tok kun et par timer (det vil si sette opp struktur, generere komponentene og koden), det var feilmeldinger ved integrering av backend i applikasjonen som stoppet opp prosjektet, dette er fordi jeg skulle integrere OpenAIs gpt 4 API for å kunne generere disse treningsplanene. Ved utviklingen så innså jeg at jeg har brukt opp min gratis kvote på alt av openai api, som betydde at jeg måtte endre API til noe som er gratis eller open source, da jeg ikke ville betale for å lage applikasjonen. Her kom de ekte utfordringene, jeg brukte altfor mange timer på å prøve ulike open source modeller via [hugging face](#). Her prøvde jeg meg på ulike modeller som gpt 2, flan-t5-large, falcon-7b-instruct og deepseek-R1 som er gratis open source modeller som er både veldig åpne, og spisset målrettet mot å følge instrukser.

Problemet med alle utenom deepseek var ekstreme mengder hallusinasjoner og feil ved formatering (fikk masse random informasjon om leger, institusjoner, årstall og helt

tilfeldige hendelser ...), flere av modellene ville for eksempel ikke generere svarene sine i ren JSON format, som var nødvendig for å få svar. Deepseek derimot krevde omtrent 612GB lokalt for å implementeres, så det var litt i overkant.

Grunnet sykdommen, all skole jeg måtte ta igjen + jobb, så satt jeg i de siste dagene og nettene og fortsatte å fikle med dette, jeg bestemte meg for å åpne API kontoen min i Open AI (tok den laveste hard limiten som mulig), og applikasjonen funket med en gang:

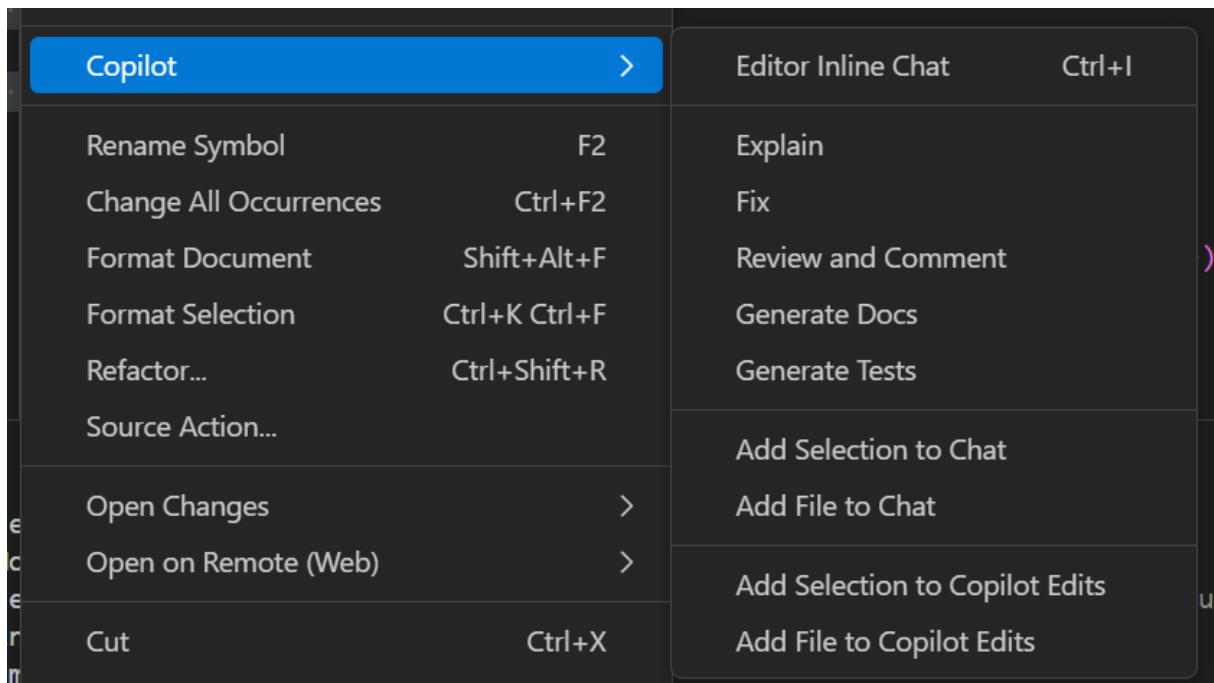


Prosjektet er laget med Vite og React, i frontend og Hono i backend. Dette er kjente teknologier for meg fra faget webapplikasjoner, også veldig standard i arbeidslivet som utvikler. Jeg brukte Chat GPT 4o til å generere enkle kommentarer som man legger i toppen av hver av kode-komponentene, jeg instruerte 4o til å gi klare men også detaljerte nok beskrivelser av komponentet og hvordan det skal reagere med de andre komponentene:

```
ai-treningsprogram > src > pages > HomePage.tsx > HomePage
1 // This is the landing page of the application where the InputForm component is displayed.
2 // It should guide the user to fill out the form and submit it.
3 // After submission, the page should route the user to the WorkoutPage with the generated workout plan.
4 import React from 'react';
5 import { useNavigate } from 'react-router-dom';
6 import InputForm from '../components/InputForm';
7 import { FormData } from '../components/InputForm';
8
9 const HomePage: React.FC = () => [
10   const navigate = useNavigate();
11
12   const handleFormSubmit = async (formData: FormData) => {
13     try {
14       const response = await fetch(`.${import.meta.env.VITE_BACKEND_URL}/api/generate-program` , {
```

, og deretter github copilot til å generere koden via denne promten: «**Use and follow the**

**instructions in the comments to generate code fitting to the rest of the components and the codebase»,** slikt genererte jeg alle komponentenes grunnlag, men en problemstilling som var vanskelig i github copilot kontra gpt 4o var å kontekstualisere problemer for å debugge. Den skjønte sjeldent kontekst selv om man legger til filer den skal bruke, hele kodebasen, og promt. Den visste for eksempel ikke hvilke teknologier vi brukte, og hadde ingen minne om kontekst fra tidligere debugging eller samtaler over tid. Dette viste seg å væreeldig tidskonsumerende for lite utbytte, så 80% av debugging ble håndtert med gpt 4o.



Det skal derimot understrekkes at github copilots funksjoner fungerte veldig fint for å generere den initiale koden, og komme med enkle fikser i et enkelt komponent via «fix», men for et større og mer kompleks system så var 4o overlegen.

Jeg satt i noe som føltes ut som en evighet med å prøve open source modeller i hugging face, så når jeg integrerte GPT4 slikt i koden:

```

// ✅ API Endpoint
app.post('/api/generate-program', async (c) => {
  console.log('✅ Received request to /api/generate-program');
  const { userInput } = await c.req.json();

  try {
    const chatResponse = await openai.chat.completions.create({
      model: 'gpt-4',
      messages: [
        {
          role: 'user',
          content: `

You are a professional fitness coach.

Your task is to generate a safe, family-friendly weekly workout plan for general fitness.
 Avoid anything related to injury, pain, violence, nudity, or unsafe exercises.

User input:
${userInput}

Respond ONLY with valid JSON structured like this:
[

  {
    "day": "Day 1 - Upper Body",
    "exercises": [
      { "name": "Push-up", "description": "Standard push-up exercise.", "sets": 3, "reps": 12 }
    ]
  }
]

No explanation. JSON only.
      `,
      max_tokens: 1000,
      temperature: 0.7,
    });
  }
}

```

Så fikk jeg endelig resultatet jeg ønsket:

## Your AI-Generated Workout Plan

### Day 1 - Upper Body & Flexibility

#### Dumbbell Bench Press

Lying on a flat bench, hold the dumbbells at your chest level, then push them up till your arms are straight.

Sets: 3, Reps: 10

#### Resistance Band Pull Apart

Hold the band with both hands at chest level. Pull your hands apart, stretching the band and squeezing your shoulder blades together.

Sets: 3, Reps: 15

#### Standing Hamstring Stretch

Stand up straight, then bend forward at the hips, keeping your legs straight. Hold for 10-30 seconds.

Sets: 2, Reps: 1

### Day 2 - Lower Body & Flexibility

#### Dumbbell Deadlift

Treningsplan generert av AI! Dette var et stort øyeblink i prosjekt utviklingen. Jeg hadde sittet lenge med problemet om API-integrering.

Etter dette prøvde jeg å implementere sora, dette viste seg å ikke være mulig teknisk sett enda ved integrering av openAI sin API, så jeg valgte å heller generere bilder via DALL-E, et stort problem her var user-policyen til DALL-E, ord som body og generelle øvelses-relaterte ord ble blokket som «upassende», og ville ikke genereres. Jeg fikk det til å funke til slutt, og promten brukt for dette i kode var som følgende: «**Clean digital illustration of gym equipment used for "\${exercise.name}". Minimalist, no people, no unsafe behavior, no body parts.** »

✓ Received request to /api/generate-program ✓ Received request to /api/generate-program ⚠ DALL-E failed for "Glute Bridge", skipping image. ⚠ DALL-E failed for "Bodyweight Squat", skipping image. ⚠ DALL-E failed for "Bird Dog", skipping image. ⚠ DALL-E failed for "Dumbbell Deadlift", skipping image. ⚠ DALL-E failed for "Resistance Band Seated Row", skipping image. ⚠ DALL-E failed for "Dynamic Lunge", skipping image. ⚠ DALL-E failed for "Plank", skipping image.

Over er direkte kopiert fra backend terminalen, altså hvor spørringene kjøres, som du ser så satt jeg inn tester for at den skulle «skippe» å lage bildet om det brøt policy. Mange

feilet, men jeg fikk noen bilder og illustrasjoner, selv om noen var bedre enn andre!

### Dumbbell Bench Press

Lie on a flat bench holding a pair of dumbbells. Lower the weights to the sides of your chest, then press them back up to the start.

Sets: 3, Reps: 10



### Tricep Extensions

Hold a dumbbell with both hands and raise it above your head, then lower it behind your head.

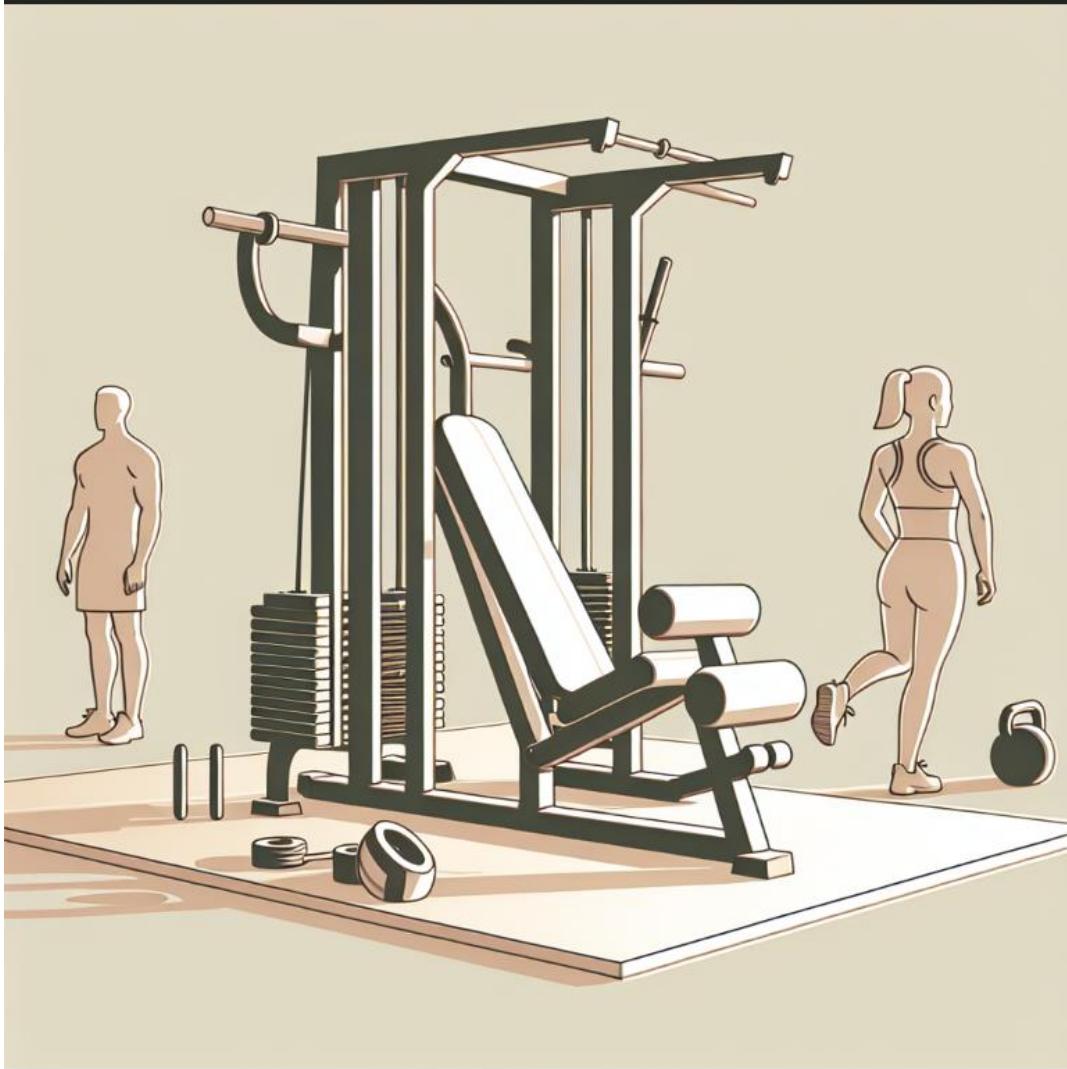
Sets: 3, Reps: 12



### **Standing Hamstring Stretch**

Stand and bend forward from your waist with arms hanging down.

Sets: 3, Reps: 15



### **Lunge with Spinal Twist**

Perform a lunge and twist your body towards the side of your front leg.

Sets: 3, Reps: 10

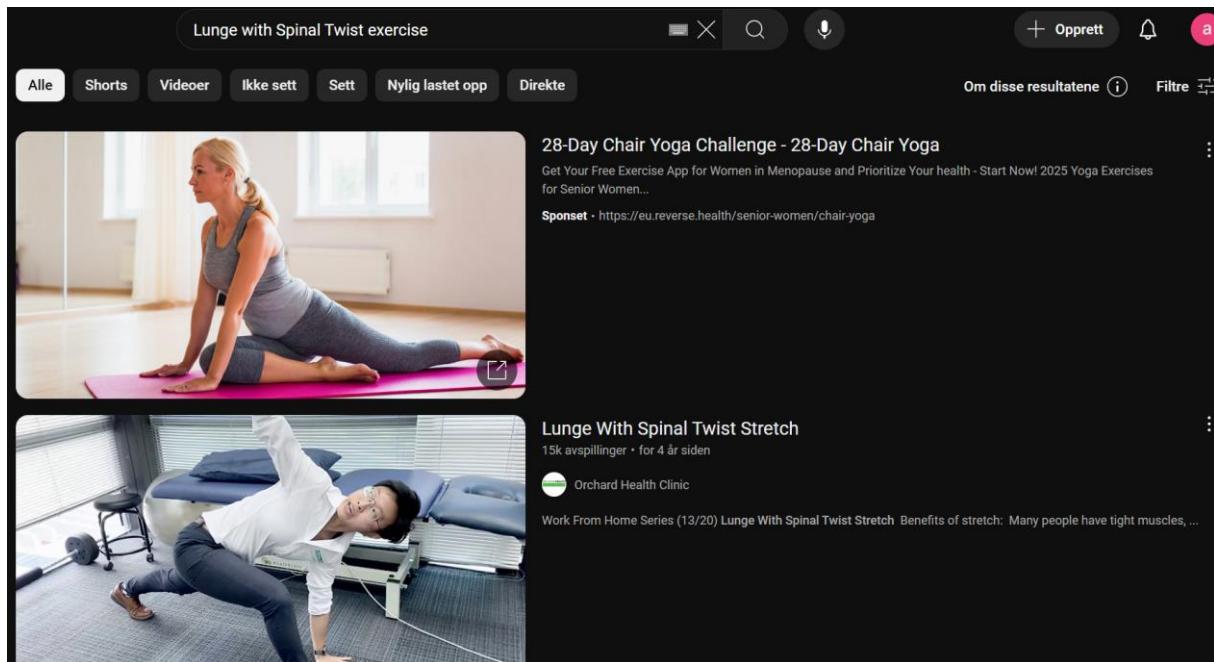
[Watch "Lunge with Spinal Twist" on YouTube](#)

Enda en «fallback»/backup plan jeg lagde for de øvelsene som ikke ville genereres var at jeg kodet en url som ville søke på øvelsen som illustrasjon i youtube, for tilgjengelighet og universell utforming, sentrale prinsipper ved utvikling:

```
// ✅ DALL-E with YouTube fallback
for (const day of plan) {
  for (const exercise of day.exercises) {
    const safePrompt = `Clean digital illustration of gym equipment used for "${exercise.name}". Minimalist, no people, no unsafe content.`

    try {
      const image = await openai.images.generate([
        model: 'dall-e-3',
        prompt: safePrompt,
        n: 1,
        size: '1024x1024'
      ]);
      exercise.imageUrl = image.data[0]?.url;
    } catch {
      console.warn(`⚠️ DALL-E failed for "${exercise.name}", adding YouTube fallback.`);
      exercise.imageUrl = null;
    }
    // ✅ Automatisk YouTube søker fallback
    exercise.videoUrl = `https://www.youtube.com/results?search_query=${encodeURIComponent(exercise.name + ' exercise')}`;
  }
}
```

Og det funket på første forsøk:

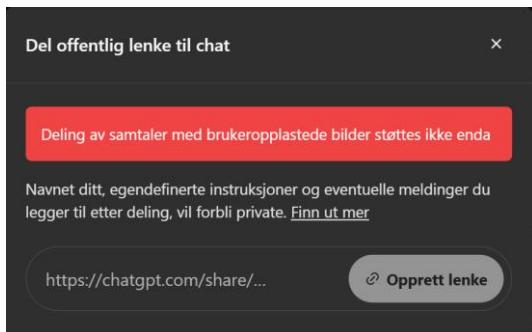


# Konklusjon

Prosjektet jeg valgte å ta for meg er i praksis en fullstendig applikasjon med moderne og noe avansert frontend, som snakker med systemet i backend (serveren) som snakker med open AI sitt API, genererer basert på prompt i JSON format og skjemaet man sender inn til serveren. Jeg vet ikke hva terskelen var for denne oppgaven, men om jeg hadde gjort det igjen hadde jeg nok ikke valgt noe som skulle være like teknisk utfordrende igjen, AI produktene jeg valgte var perfekte for jobben, men jeg måtte jo ende opp med å betale for Open AI sitt API for at applikasjonen skulle funke, både for oppgavens og min mentale helses del! Det var mange timer, men tusenvis mer av linjer med kode som gikk inn i dette prosjektet. Spesielt underveis da jeg ble ganske avhengig av gpt 4o til å hjelpe med den komplekse debuggingen, jeg opplevde stadig noen hallusinasjoner bli mer og mer frekvent jo lenger jeg snakket med den, og det er som forventet. Jeg jobbet imot dette effektivt ved å kontekstualisere, prompte på en måte som forklarer hvordan ting er, om den gjør noe feil, og sendte oversikt over hvordan de forskjellige komponentenes kode så ut.

Jeg er veldig fornøyd med resultatet, det å se faktiske treningsplaner bli generert basert på skjemaet og hva man fyller inn der føltes kjempebra og belønnende ut. Jeg hadde nok gjort litt mer undersøkelse på hvordan det er med selve AI-teknologiene og kompleksiteten det vil ta å implementere i egne prosjekter. Basert på forutsetningen om at jeg ble syk, endret retning og hadde mye å ta igjen i hverdagen, så føler jeg at jeg har besvart oppgaven til min beste evne, og levert et produkt jeg er storfornøyd med 😊

Jeg får dessverre ikke delt lenke til chatteren med GPT 4o, da samtaler med brukeropplastede bilder ikke støttes enda som funksjon. Dette var f.eks. brukt til å vise hvordan ting så ut i nettleseren, konsollen, feilmeldinger.. og når ting funket!



Jeg føler jeg har gitt Github Copilot mye tyn, så her er en teknisk oppsummering for den å gjøre opp for seg !

## How It Works

### 1. User Interaction:

- The user fills out the form on the [HomePage](#) and submits it.
- The frontend sends the input to the backend API.

### 2. Workout Plan Generation:

- The backend uses GPT-4 to generate a JSON-structured workout plan based on the user's input.
- Each exercise in the plan is enhanced with an image from DALL·E or a YouTube link.

### 3. Display:

- The frontend retrieves the workout plan and displays it on the [WorkoutPage](#) using the [WorkoutPlan](#) component.
- 

## Key Technologies

- **Frontend:**

- React, React Router, TypeScript

- **Backend:**

- Hono, OpenAI API (GPT-4, DALL·E), TypeScript

- **Build Tools:**

- Vite, pnpm

- **Environment Management:**

- dotenv
- 

## Summary

This project combines a React-based frontend with a Hono-powered backend to deliver a personalized workout plan generator. It leverages OpenAI's GPT-4 for text generation and DALL·E for image generation, providing a seamless and interactive user experience.

Veldig bra oppsummert 😊