K.R. MANGALAM UNIVERSITY

THE COMPLETE WORLD OF EDUCATION

Mini Project ENSI-152

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SCHOOL OF ENGINEERING AND TECHNOLOGY

TEAM LEADER

- MOHAMMED ZAID - 2401730232

TEAM CO-MEMBERS

1. AADITYA SAINI - 2401730024
2. LOKESH - 2401730235
3. VIKRAM – 2401730170

PROJECT INTRODUCTION AND METHODOLOGY

Project Overview

- It is Next.js web dev application that help users to write a job cover letter, Interview preparation and Build Resume. a Next.js project that helps users create resumes, prepare for interviews, and write job cover letters using the Gemini AI API. It addresses real-world job search challenges by making the process easier and more efficient. A growth tool button further enhances career development.

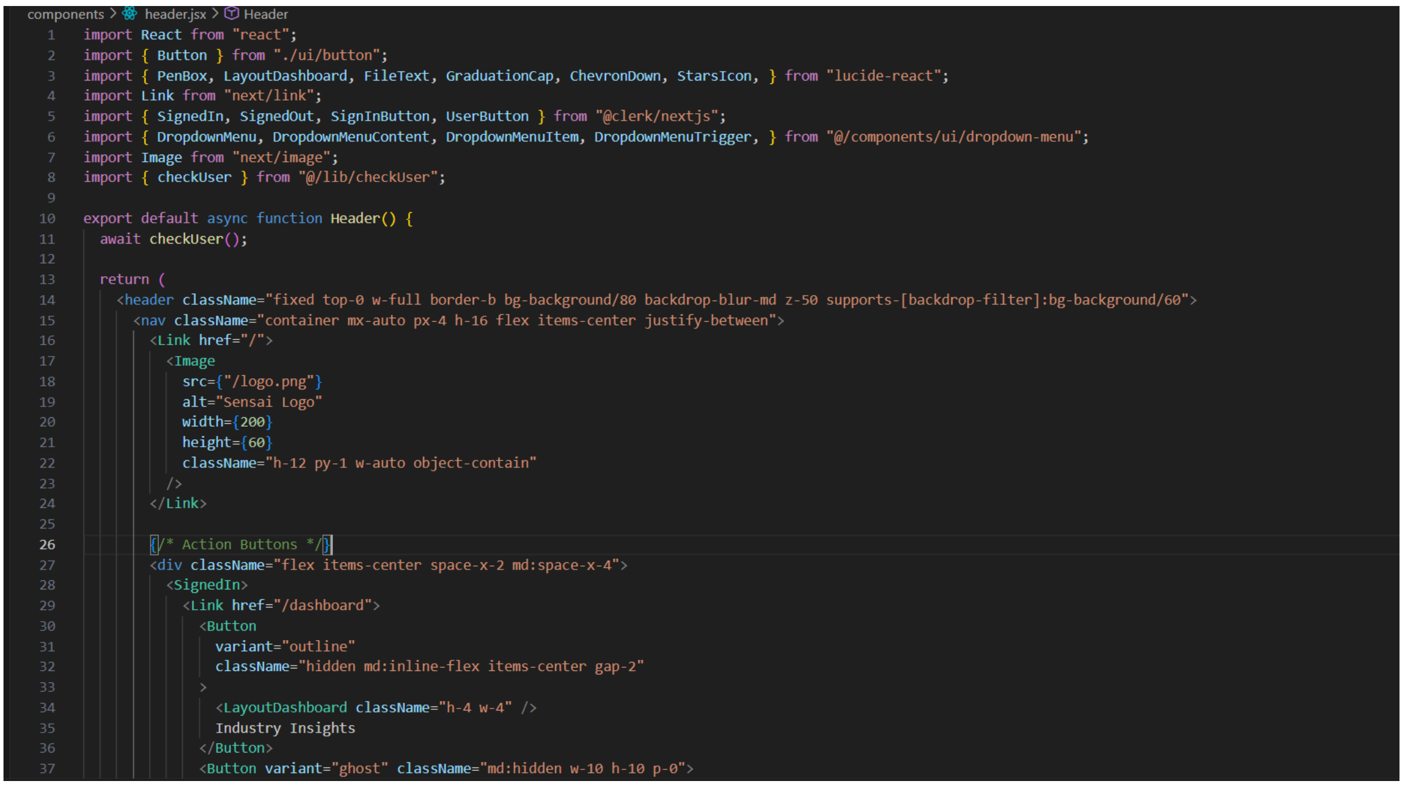
Working Of Project

1. Automatic Resume Building – Enable users to generate professional resumes quickly Using Ai
2. Enhance Interview Preparation – Provide AI-driven insights and practice questions to improve interview skills.
3. Simply Cover Letter Writing – Help users craft personalized and impactful job cover letters effortlessly.
4. Provide Weekly Industry Insights – Keep users updated with the latest job market trends and hiring demands.
5. Integrate Growth Tools - Offer a dedicated growth tool button to enhance career development and job-readiness.

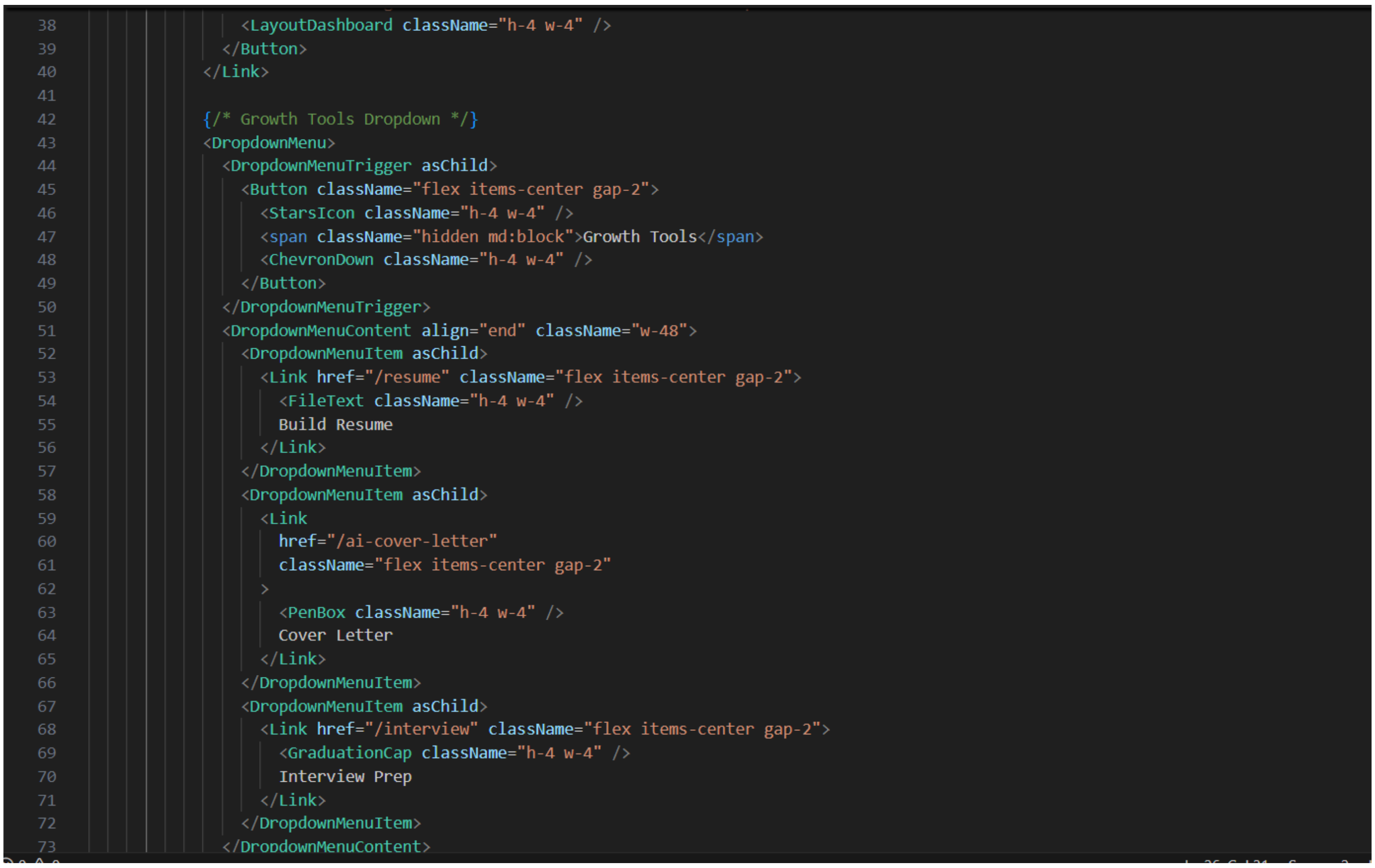
Methodology

1. Frontend Development – The user interview is built using Next.js with a responsive Header/Navigation and a well-structured Landing-page, styled using Tailwind CSS for a modern and clean design User Authentication – clerk is used for secure and seamless user authentication, ensuring a smooth sign-up and login experience.
2. Backend and Database – The project Utilizes Inngest or Neon for database management, ensuring efficient data storage and retrieval.
3. AI Integration – The Google Gemini API powers resume building, interview preparation, and job cover letter generation, providing AI-driven assistance to users.
4. Industry Insights and Growth Tools – Weekly Updates on job trends and market insights help users stay ahead, while a dedicated growth tool button enhances career development.

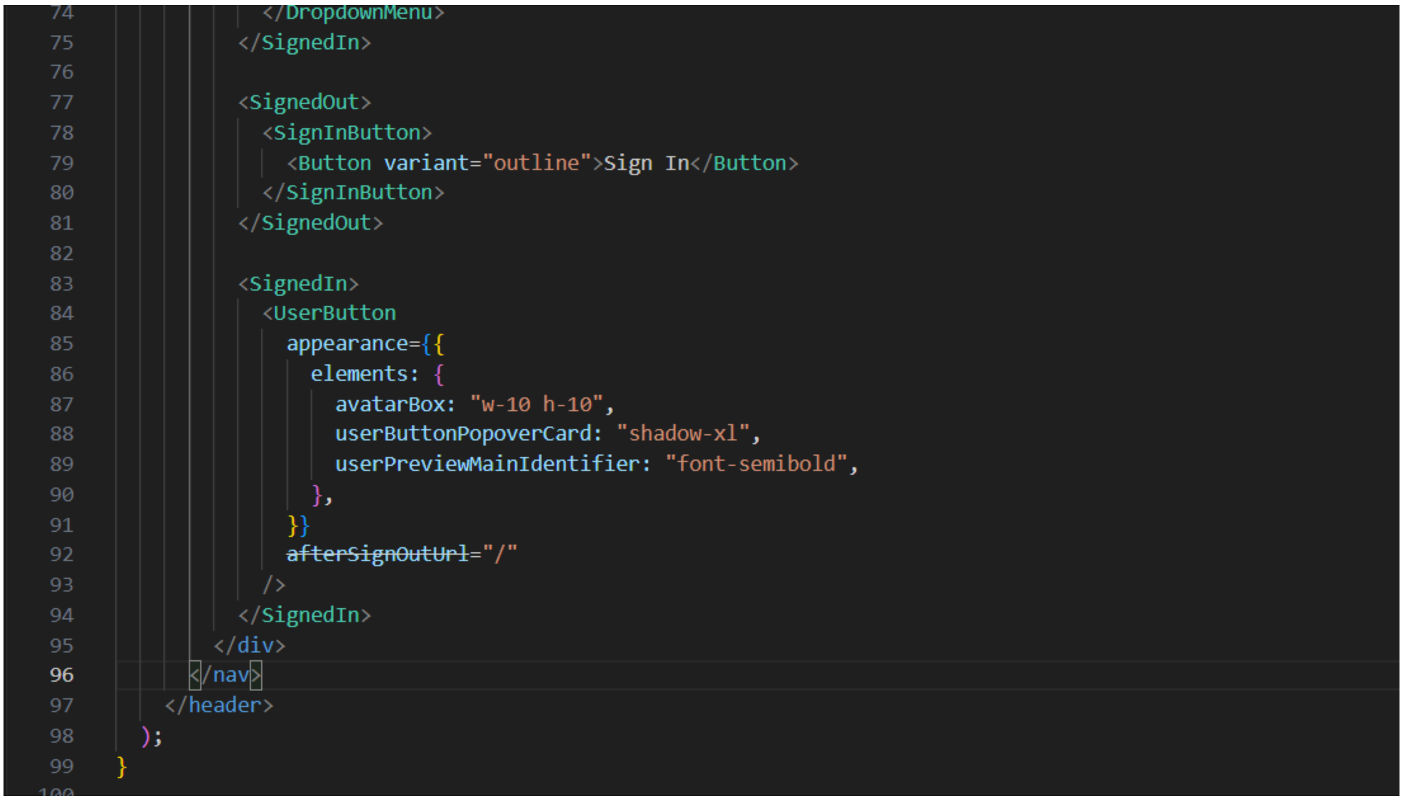
WEBSITE NAVIGATION



```jsx
import React from "react";
import { Button } from "./ui/button";
import { PenBox, LayoutDashboard, FileText, GraduationCap, ChevronDown, StarsIcon, } from "lucide-react";
import Link from "next/link";
import { SignedIn, SignedOut, SignInButton, UserButton } from "@clerk/nextjs";
import { DropdownMenu, DropdownMenuContent, DropdownMenuItem, DropdownMenuTrigger, } from "@/components/ui/dropdown-menu";
import Image from "next/image";
import { checkUser } from "@/lib/checkUser";

export default async function Header() {
  await checkUser();

  return (
    <header className="fixed top-0 w-full border-b bg-background/80 backdrop-blur-md z-50 supports-[backdrop-filter]:bg-background/60">
      <nav className="container mx-auto px-4 h-16 flex items-center justify-between">
        <Link href="/">
          <Image
            src={"/logo.png"}
            alt="Sensai Logo"
            width={200}
            height={60}
            className="h-12 py-1 w-auto object-contain"
          />
        </Link>

        {/* Action Buttons */}
        <div className="flex items-center space-x-2 md:space-x-4">
          <SignedIn>
            <Link href="/dashboard">
              <Button
                variant="outline"
                className="hidden md:inline-flex items-center gap-2"
              >
                <LayoutDashboard className="h-4 w-4" />
                Industry Insights
              </Button>
              <Button variant="ghost" className="md:hidden w-10 h-10 p-0">
```

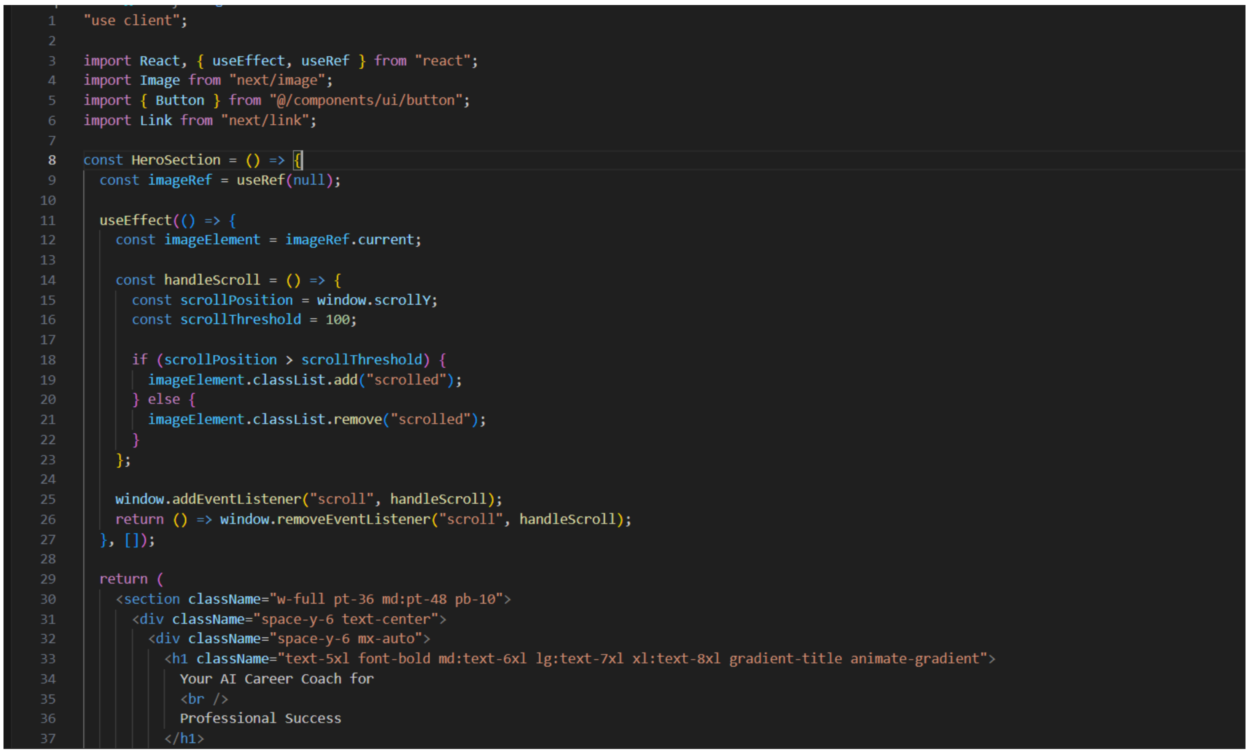


```jsx
                <LayoutDashboard className="h-4 w-4" />
              </Button>
            </Link>

            {/* Growth Tools Dropdown */}
            <DropdownMenu>
              <DropdownMenuTrigger asChild>
                <Button className="flex items-center gap-2">
                  <StarsIcon className="h-4 w-4" />
                  <span className="hidden md:block">Growth Tools</span>
                  <ChevronDown className="h-4 w-4" />
                </Button>
              </DropdownMenuTrigger>
              <DropdownMenuContent align="end" className="w-48">
                <DropdownMenuItem asChild>
                  <Link href="/resume" className="flex items-center gap-2">
                    <FileText className="h-4 w-4" />
                    Build Resume
                  </Link>
                </DropdownMenuItem>
                <DropdownMenuItem asChild>
                  <Link
                    href="/ai-cover-letter"
                    className="flex items-center gap-2"
                  >
                    <PenBox className="h-4 w-4" />
                    Cover Letter
                  </Link>
                </DropdownMenuItem>
                <DropdownMenuItem asChild>
                  <Link href="/interview" className="flex items-center gap-2">
                    <GraduationCap className="h-4 w-4" />
                    Interview Prep
                  </Link>
                </DropdownMenuItem>
              </DropdownMenuContent>
```

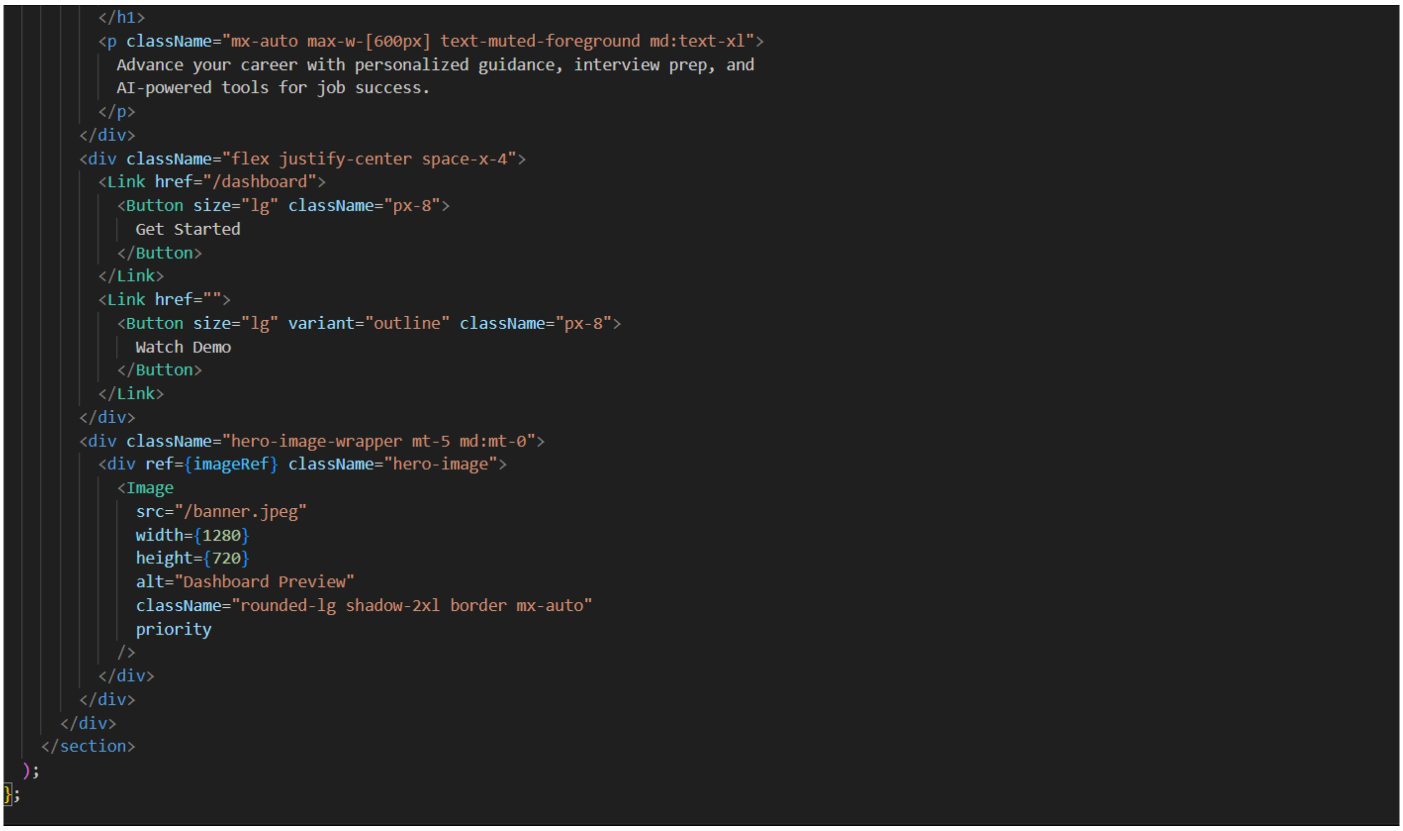


```
            </DropdownMenu>
          </SignedIn>

          <SignedOut>
            <SignInButton>
              <Button variant="outline">Sign In</Button>
            </SignInButton>
          </SignedOut>

          <SignedIn>
            <UserButton
              appearance={{
                elements: {
                  avatarBox: "w-10 h-10",
                  userButtonPopoverCard: "shadow-xl",
                  userPreviewMainIdentifier: "font-semibold",
                },
              }}
              ~~afterSignOutUrl~~="/"
            />
          </SignedIn>
        </div>
      </nav>
    </header>
  );
}
```

WEBSITE LANDING PAGE/FRONTEND PART

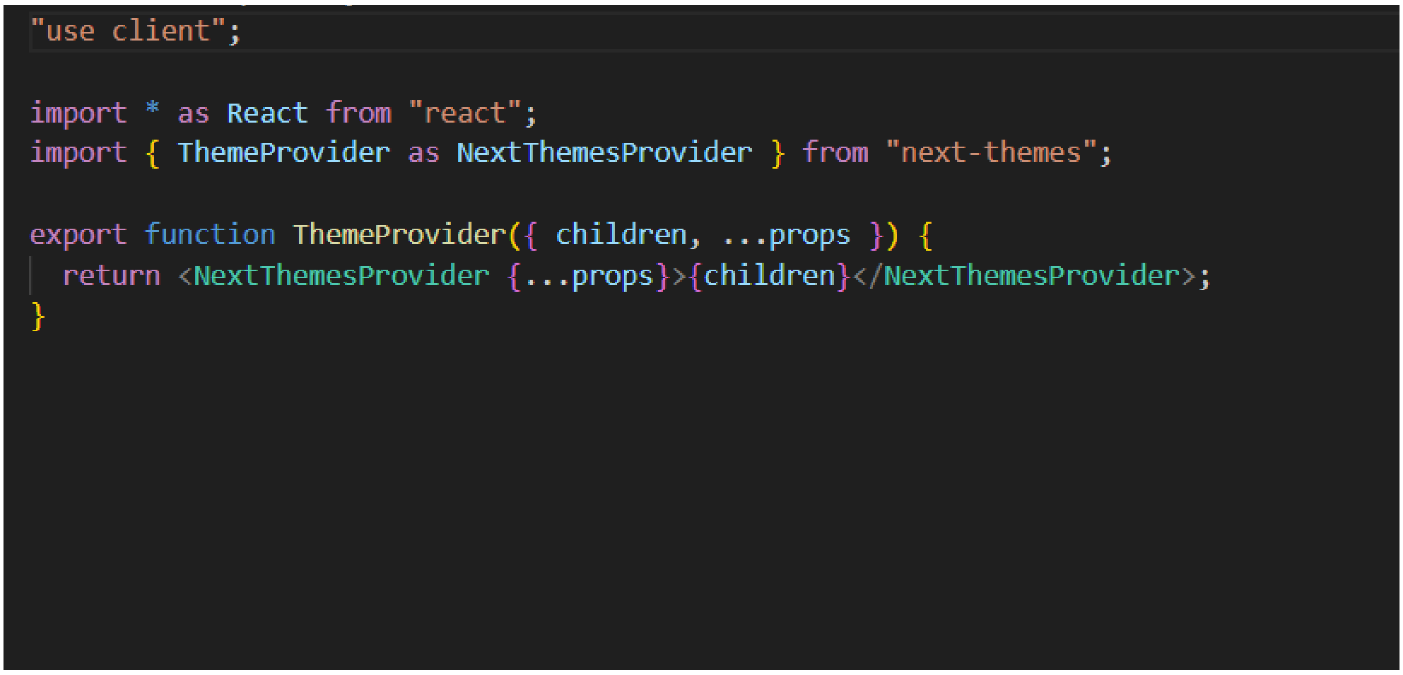


```
"use client";

import React, { useEffect, useRef } from "react";
import Image from "next/image";
import { Button } from "@/components/ui/button";
import Link from "next/link";

const HeroSection = () => {
  const imageRef = useRef(null);

  useEffect(() => {
    const imageElement = imageRef.current;

    const handleScroll = () => {
      const scrollPosition = window.scrollY;
      const scrollThreshold = 100;

      if (scrollPosition > scrollThreshold) {
        imageElement.classList.add("scrolled");
      } else {
        imageElement.classList.remove("scrolled");
      }
    };

    window.addEventListener("scroll", handleScroll);
    return () => window.removeEventListener("scroll", handleScroll);
  }, []);

  return (
    <section className="w-full pt-36 md:pt-48 pb-10">
      <div className="space-y-6 text-center">
        <div className="space-y-6 mx-auto">
          <h1 className="text-5xl font-bold md:text-6xl lg:text-7xl xl:text-8xl gradient-title animate-gradient">
            Your AI Career Coach for
            <br />
            Professional Success
          </h1>
```



```
          </h1>
          <p className="mx-auto max-w-[600px] text-muted-foreground md:text-xl">
            Advance your career with personalized guidance, interview prep, and
            AI-powered tools for job success.
          </p>
        </div>
        <div className="flex justify-center space-x-4">
          <Link href="/dashboard">
            <Button size="lg" className="px-8">
              Get Started
            </Button>
          </Link>
          <Link href="">
            <Button size="lg" variant="outline" className="px-8">
              Watch Demo
            </Button>
          </Link>
        </div>
        <div className="hero-image-wrapper mt-5 md:mt-0">
          <div ref={imageRef} className="hero-image">
            <Image
              src="/banner.jpeg"
              width={1280}
              height={720}
              alt="Dashboard Preview"
              className="rounded-lg shadow-2xl border mx-auto"
              priority
            />
          </div>
        </div>
      </div>
    </section>
  );
};
```

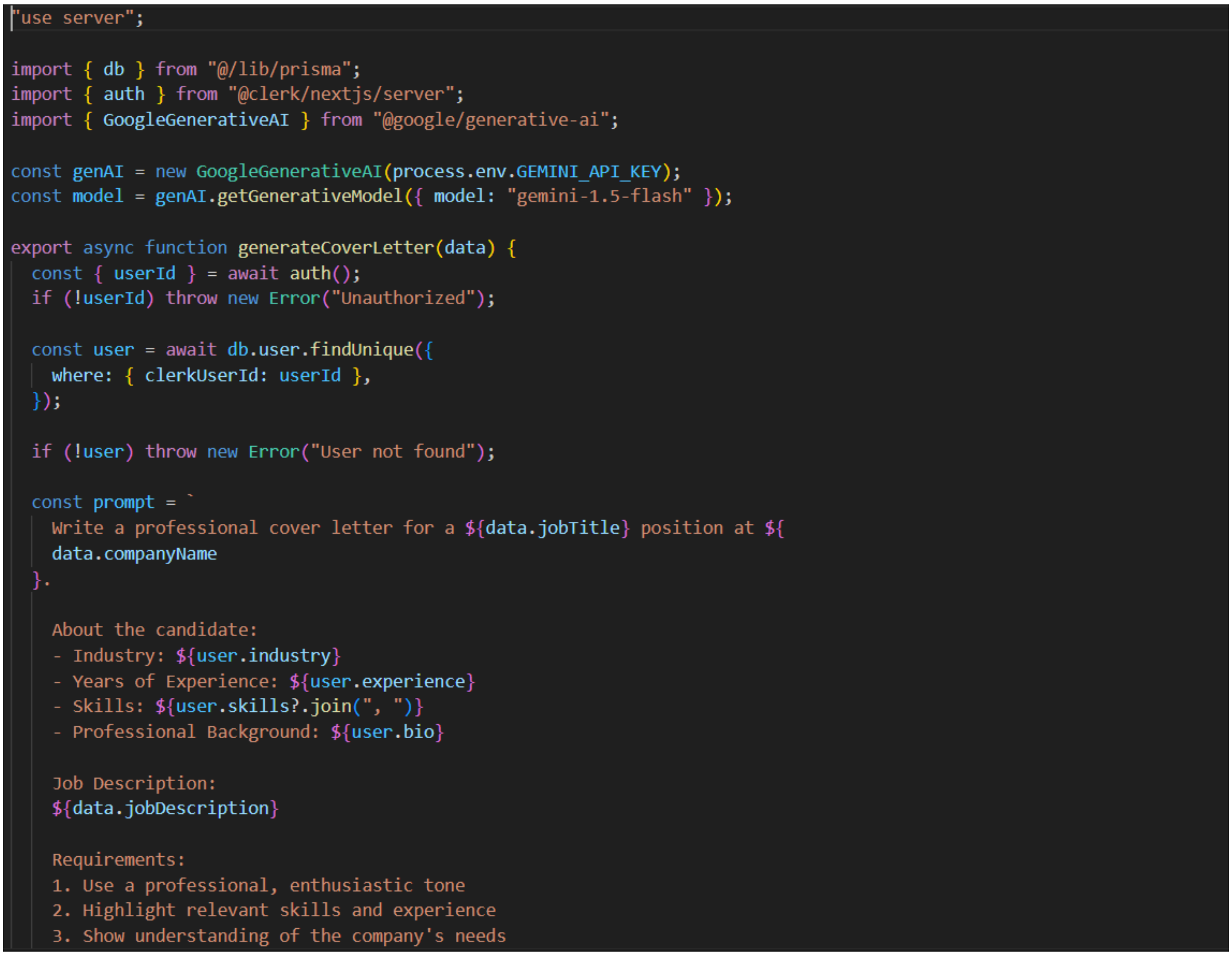
WEBSITE THEME PROVIDER / DARK OR LIGHT THEME



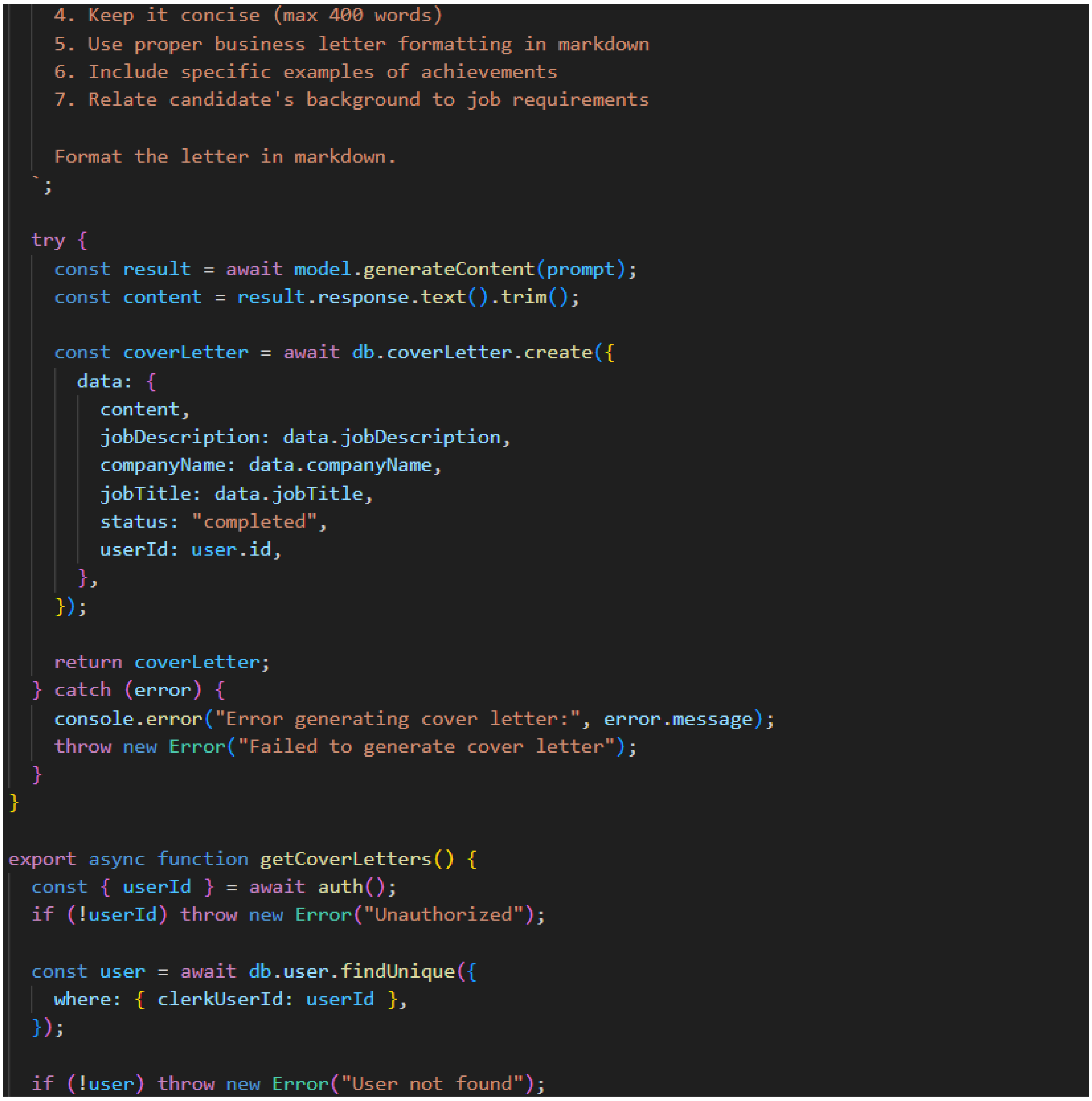
```
"use client";

import * as React from "react";
import { ThemeProvider as NextThemesProvider } from "next-themes";

export function ThemeProvider({ children, ...props }) {
  return <NextThemesProvider {...props}>{children}</NextThemesProvider>;
}
```

MAIN WEBSITE CONTENT

1. COVER LETTER

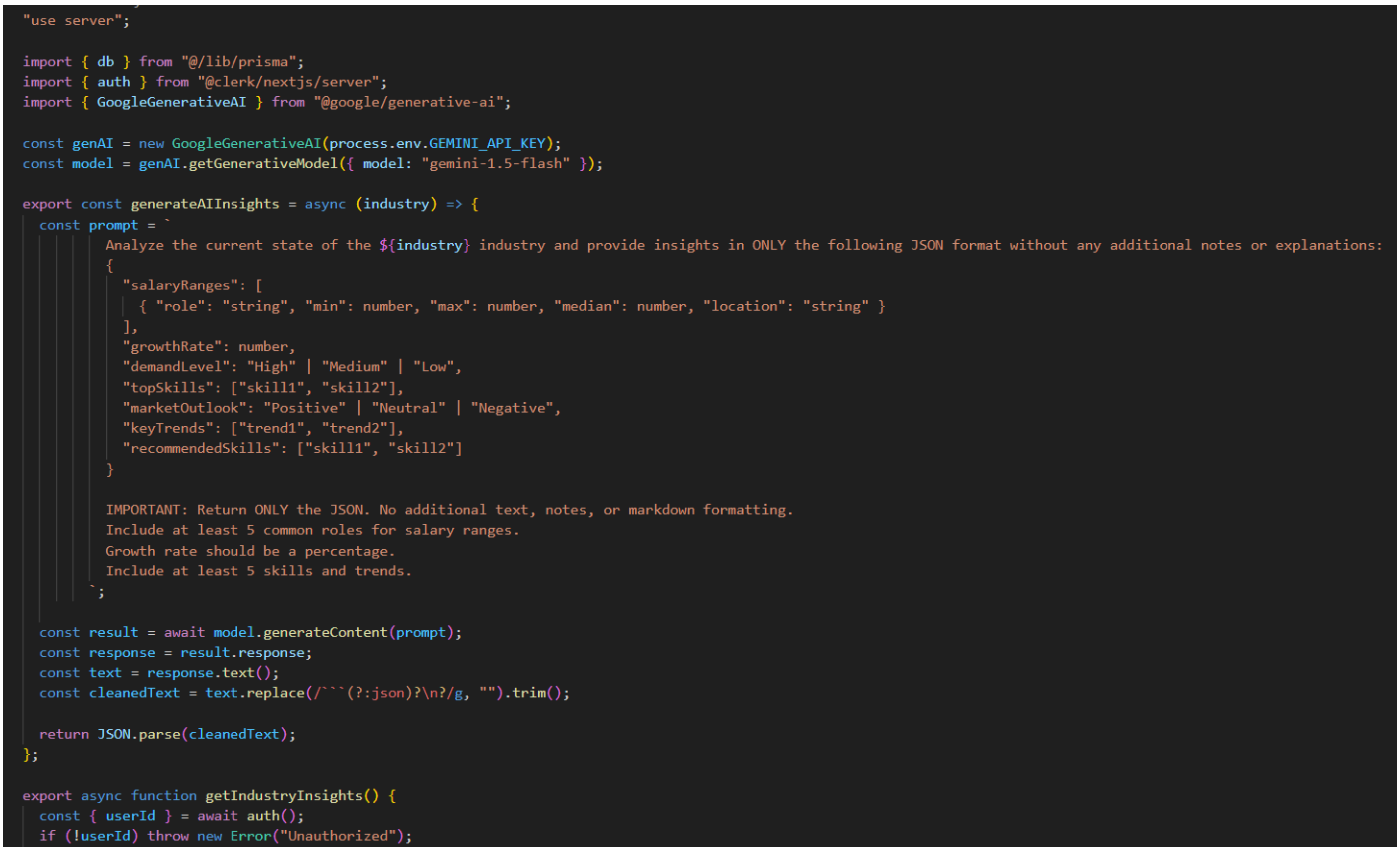


```
"use server";

import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { GoogleGenerativeAI } from "@google/generative-ai";

const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });

export async function generateCoverLetter(data) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  const prompt = `
    Write a professional cover letter for a ${data.jobTitle} position at ${
    data.companyName
  }.

    About the candidate:
    - Industry: ${user.industry}
    - Years of Experience: ${user.experience}
    - Skills: ${user.skills?.join(", ")}
    - Professional Background: ${user.bio}

    Job Description:
    ${data.jobDescription}

    Requirements:
    1. Use a professional, enthusiastic tone
    2. Highlight relevant skills and experience
    3. Show understanding of the company's needs
```

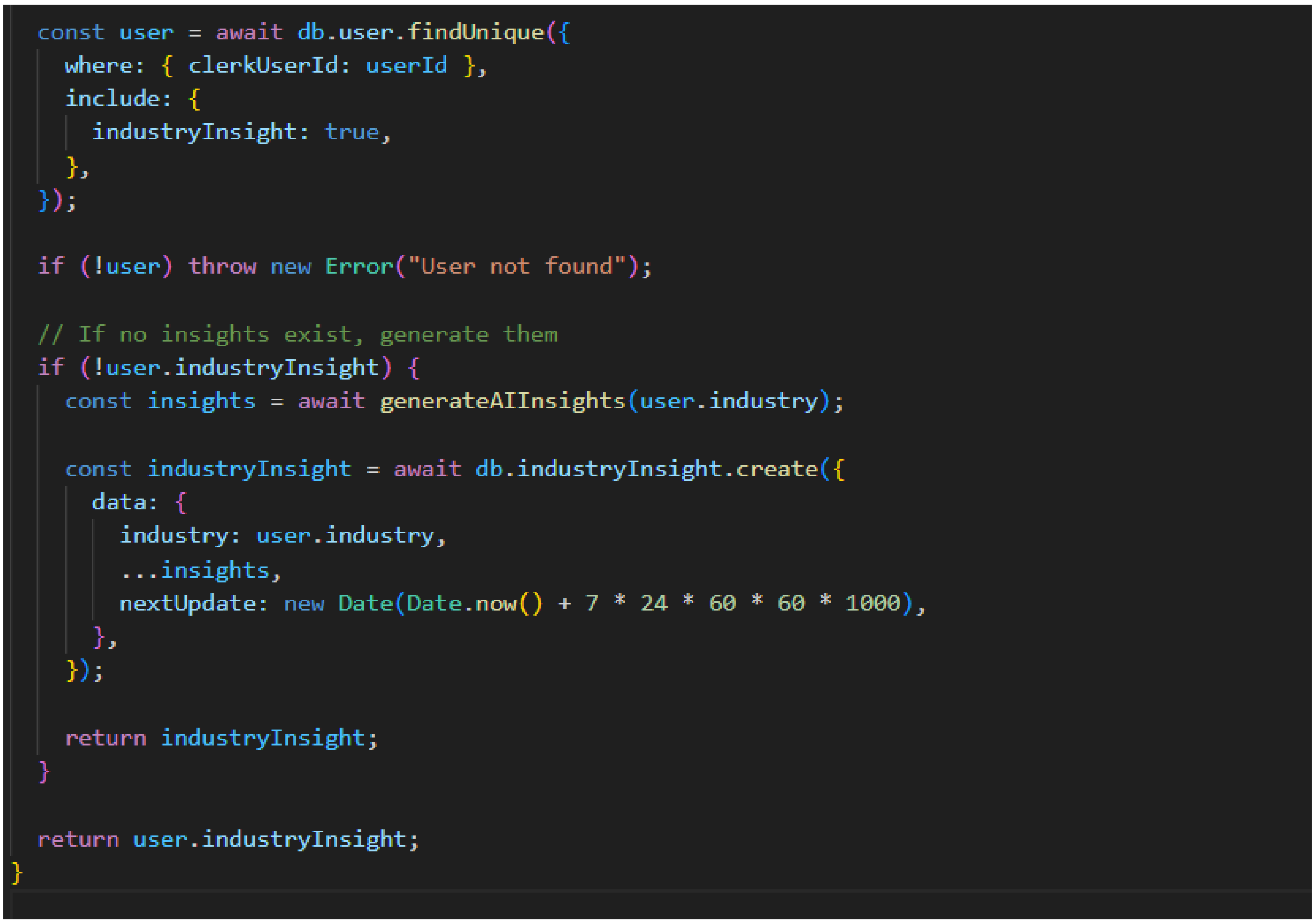


```
    4. Keep it concise (max 400 words)
    5. Use proper business letter formatting in markdown
    6. Include specific examples of achievements
    7. Relate candidate's background to job requirements

    Format the letter in markdown.
  `;

  try {
    const result = await model.generateContent(prompt);
    const content = result.response.text().trim();

    const coverLetter = await db.coverLetter.create({
      data: {
        content,
        jobDescription: data.jobDescription,
        companyName: data.companyName,
        jobTitle: data.jobTitle,
        status: "completed",
        userId: user.id,
      },
    });

    return coverLetter;
  } catch (error) {
    console.error("Error generating cover letter:", error.message);
    throw new Error("Failed to generate cover letter");
  }
}

export async function getCoverLetters() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");
```

2. DASHBOARD

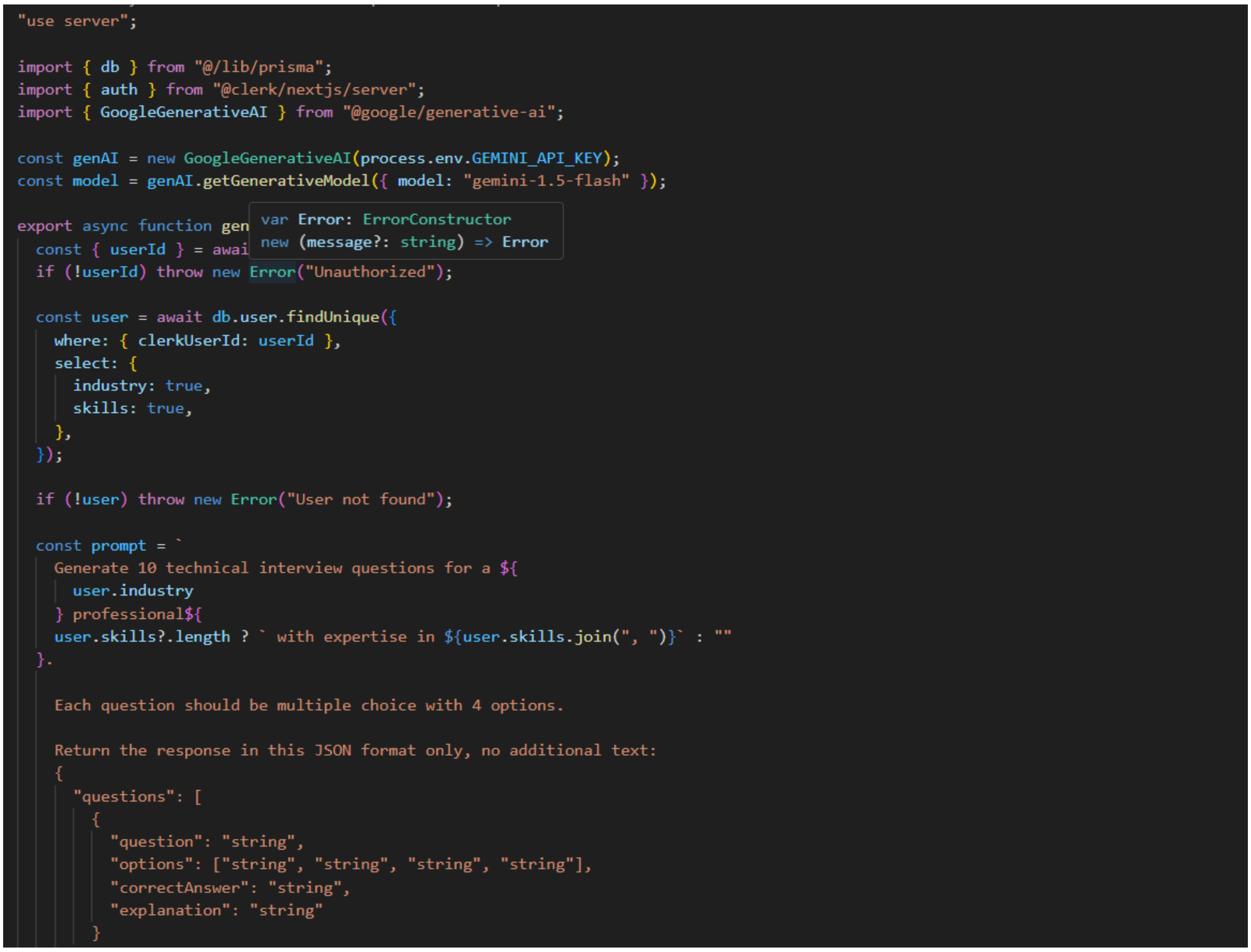


```
"use server";

import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { GoogleGenerativeAI } from "@google/generative-ai";

const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });

export const generateAIInsights = async (industry) => {
  const prompt = `
          Analyze the current state of the ${industry} industry and provide insights in ONLY the following JSON format without any additional notes or explanations:
          {
            "salaryRanges": [
              { "role": "string", "min": number, "max": number, "median": number, "location": "string" }
            ],
            "growthRate": number,
            "demandLevel": "High" | "Medium" | "Low",
            "topSkills": ["skill1", "skill2"],
            "marketOutlook": "Positive" | "Neutral" | "Negative",
            "keyTrends": ["trend1", "trend2"],
            "recommendedSkills": ["skill1", "skill2"]
          }

          IMPORTANT: Return ONLY the JSON. No additional text, notes, or markdown formatting.
          Include at least 5 common roles for salary ranges.
          Growth rate should be a percentage.
          Include at least 5 skills and trends.
        `;

  const result = await model.generateContent(prompt);
  const response = result.response;
  const text = response.text();
  const cleanedText = text.replace(/```(?:json)?\n?/g, "").trim();

  return JSON.parse(cleanedText);
};

export async function getIndustryInsights() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");
```

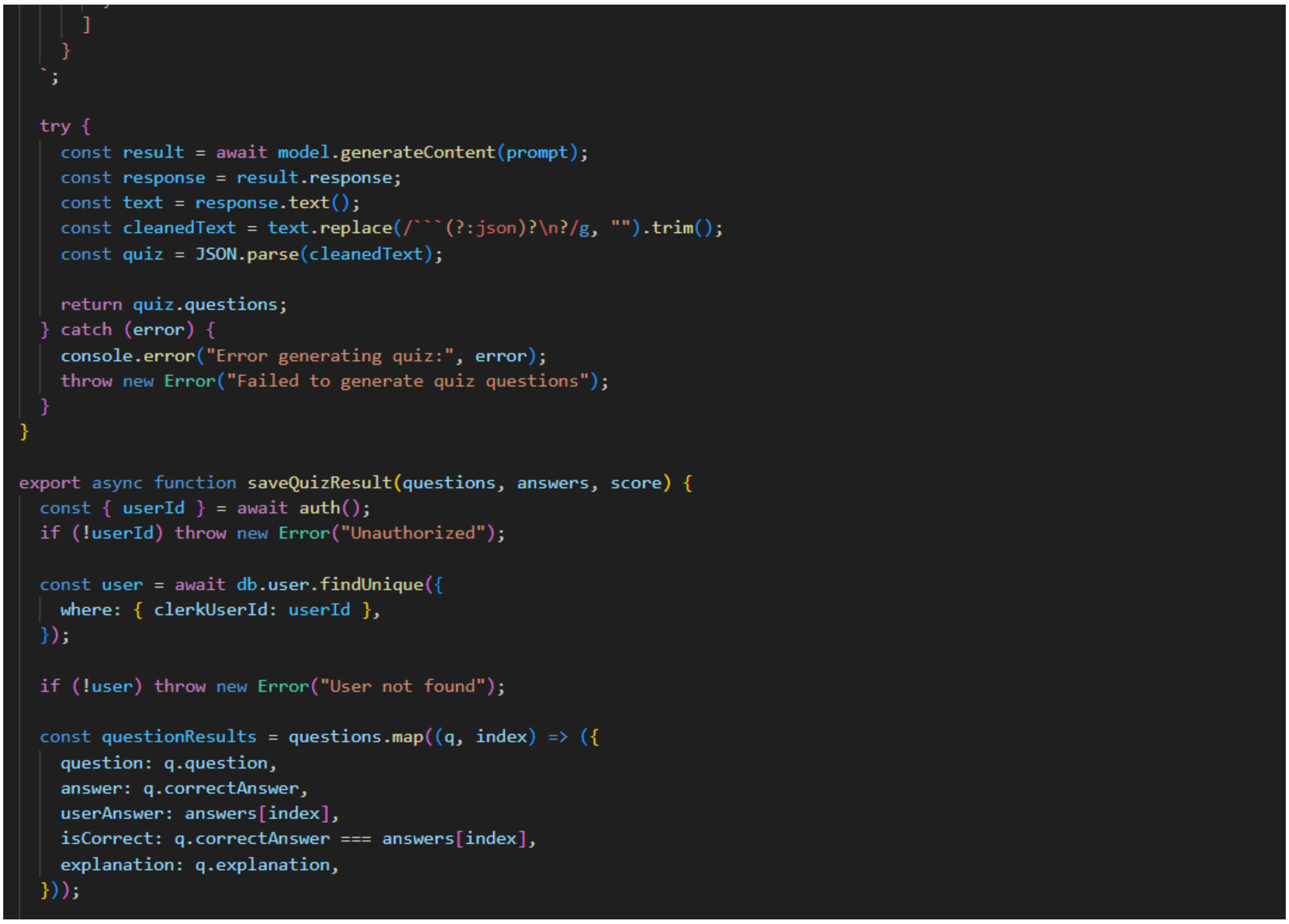


```
  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
    include: {
      industryInsight: true,
    },
  });

  if (!user) throw new Error("User not found");

  // If no insights exist, generate them
  if (!user.industryInsight) {
    const insights = await generateAIInsights(user.industry);

    const industryInsight = await db.industryInsight.create({
      data: {
        industry: user.industry,
        ...insights,
        nextUpdate: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000),
      },
    });

    return industryInsight;
  }

  return user.industryInsight;
}
```

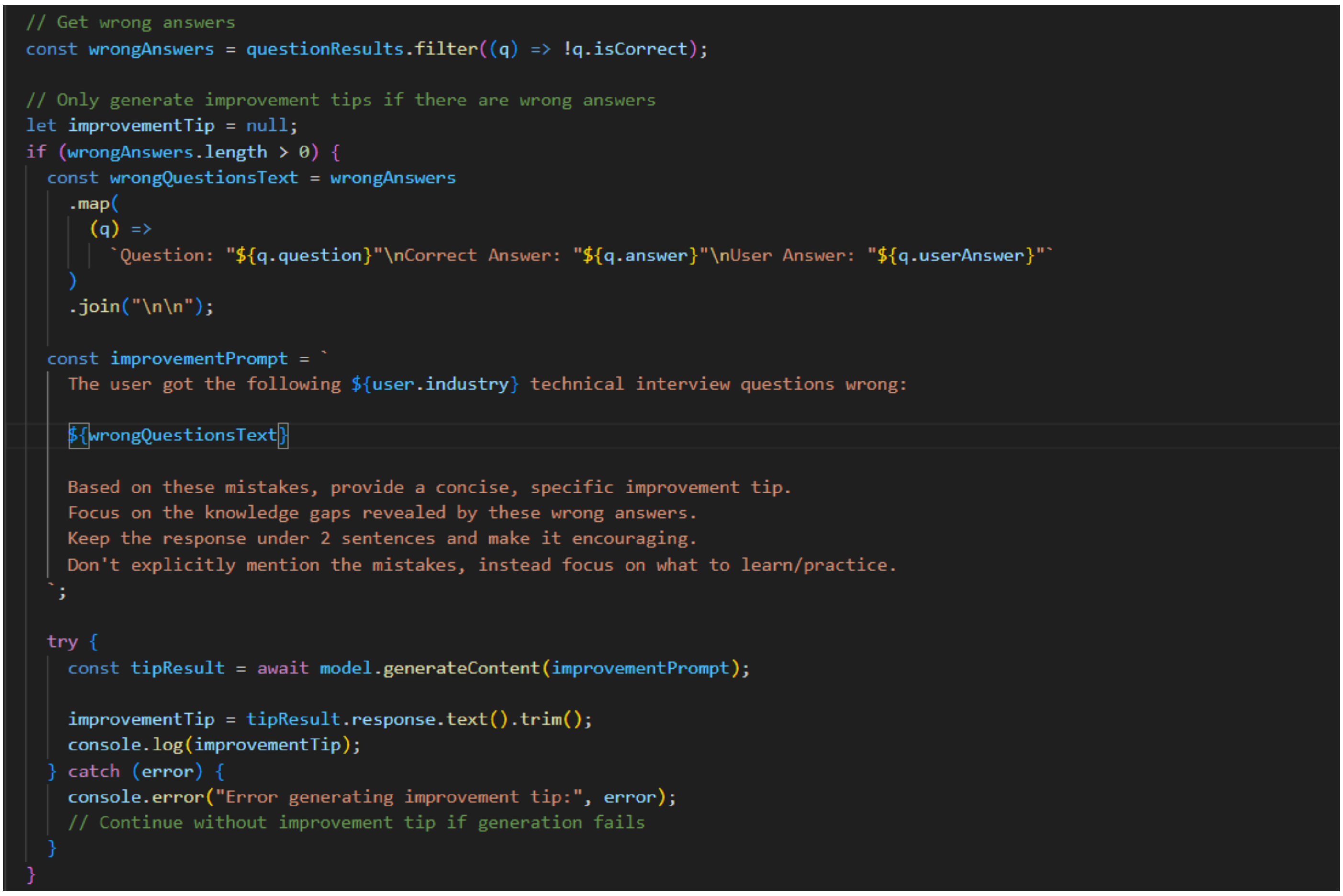
3. INTERVIEW



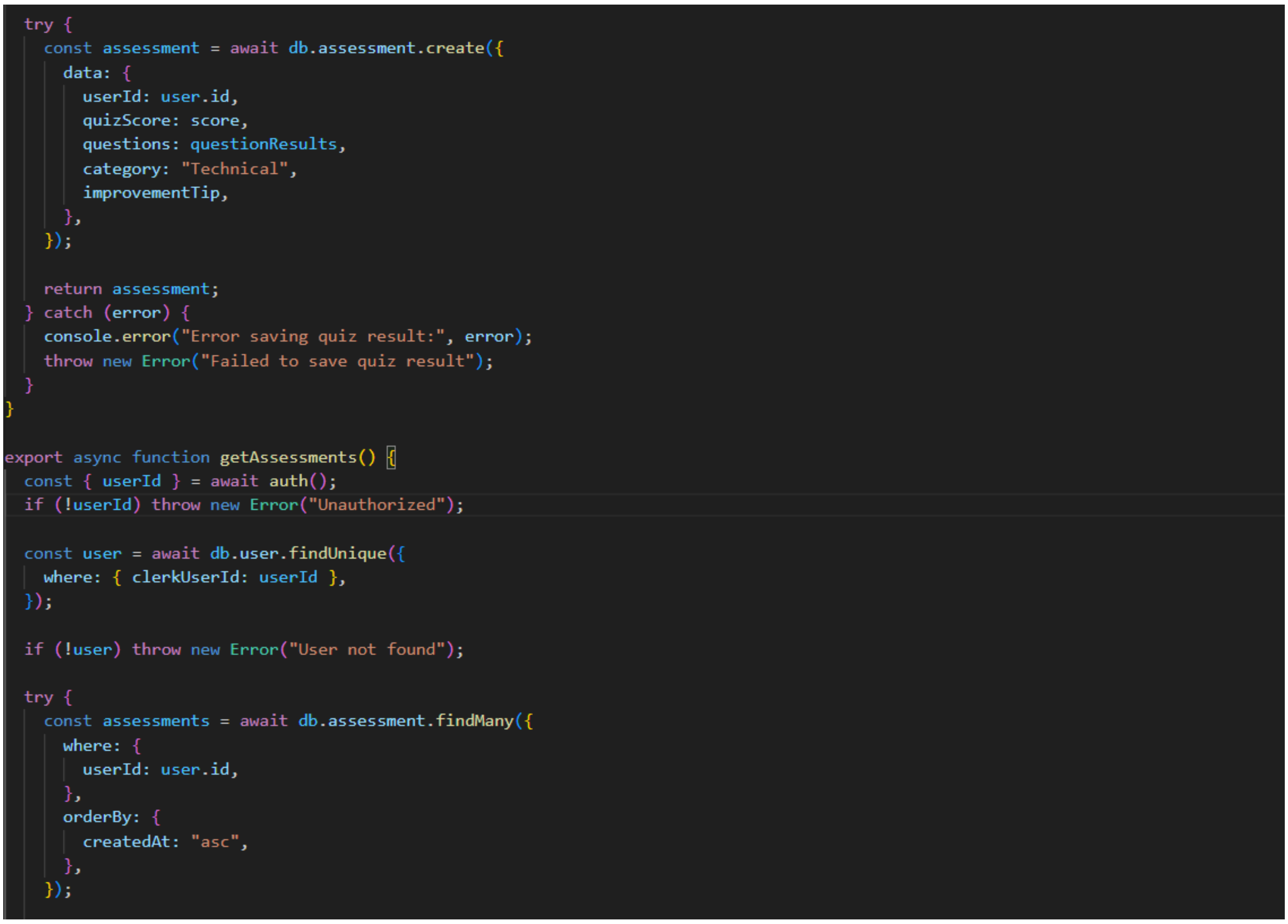
```
"use server";

import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { GoogleGenerativeAI } from "@google/generative-ai";

const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });

export async function gen var Error: ErrorConstructor
  const { userId } = awai new (message?: string) => Error
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
    select: {
      industry: true,
      skills: true,
    },
  });

  if (!user) throw new Error("User not found");

  const prompt = `
    Generate 10 technical interview questions for a ${
      user.industry
    } professional${
    user.skills?.length ? ` with expertise in ${user.skills.join(", ")}` : ""
  }.

    Each question should be multiple choice with 4 options.

    Return the response in this JSON format only, no additional text:
    {
      "questions": [
        {
          "question": "string",
          "options": ["string", "string", "string", "string"],
          "correctAnswer": "string",
          "explanation": "string"
        }
```



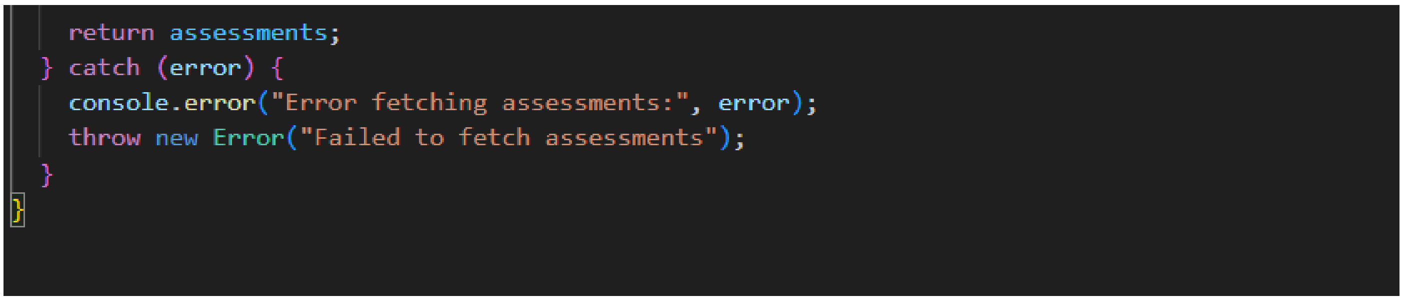
```
      ]
    }
  `;

  try {
    const result = await model.generateContent(prompt);
    const response = result.response;
    const text = response.text();
    const cleanedText = text.replace(/```(?:json)?\n?/g, "").trim();
    const quiz = JSON.parse(cleanedText);

    return quiz.questions;
  } catch (error) {
    console.error("Error generating quiz:", error);
    throw new Error("Failed to generate quiz questions");
  }
}

export async function saveQuizResult(questions, answers, score) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  const questionResults = questions.map((q, index) => ({
    question: q.question,
    answer: q.correctAnswer,
    userAnswer: answers[index],
    isCorrect: q.correctAnswer === answers[index],
    explanation: q.explanation,
  }));
```



```
  // Get wrong answers
  const wrongAnswers = questionResults.filter((q) => !q.isCorrect);

  // Only generate improvement tips if there are wrong answers
  let improvementTip = null;
  if (wrongAnswers.length > 0) {
    const wrongQuestionsText = wrongAnswers
      .map(
        (q) =>
          `Question: "${q.question}"\nCorrect Answer: "${q.answer}"\nUser Answer: "${q.userAnswer}"`
      )
      .join("\n\n");

    const improvementPrompt = `
      The user got the following ${user.industry} technical interview questions wrong:

      ${wrongQuestionsText}

      Based on these mistakes, provide a concise, specific improvement tip.
      Focus on the knowledge gaps revealed by these wrong answers.
      Keep the response under 2 sentences and make it encouraging.
      Don't explicitly mention the mistakes, instead focus on what to learn/practice.
    `;

    try {
      const tipResult = await model.generateContent(improvementPrompt);

      improvementTip = tipResult.response.text().trim();
      console.log(improvementTip);
    } catch (error) {
      console.error("Error generating improvement tip:", error);
      // Continue without improvement tip if generation fails
    }
  }
```

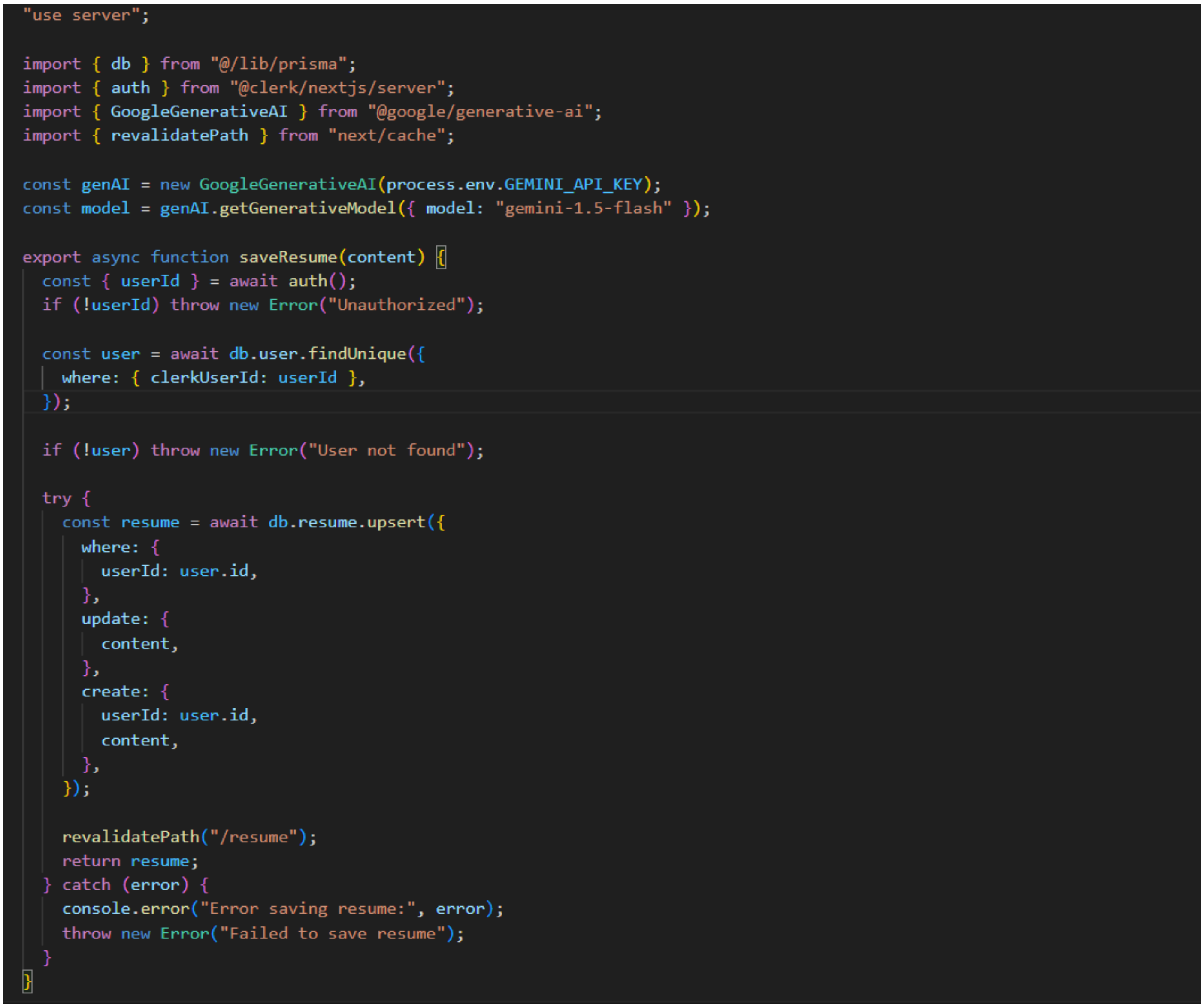


```
  try {
    const assessment = await db.assessment.create({
      data: {
        userId: user.id,
        quizScore: score,
        questions: questionResults,
        category: "Technical",
        improvementTip,
      },
    });

    return assessment;
  } catch (error) {
    console.error("Error saving quiz result:", error);
    throw new Error("Failed to save quiz result");
  }
}

export async function getAssessments() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  try {
    const assessments = await db.assessment.findMany({
      where: {
        userId: user.id,
      },
      orderBy: {
        createdAt: "asc",
      },
    });
```

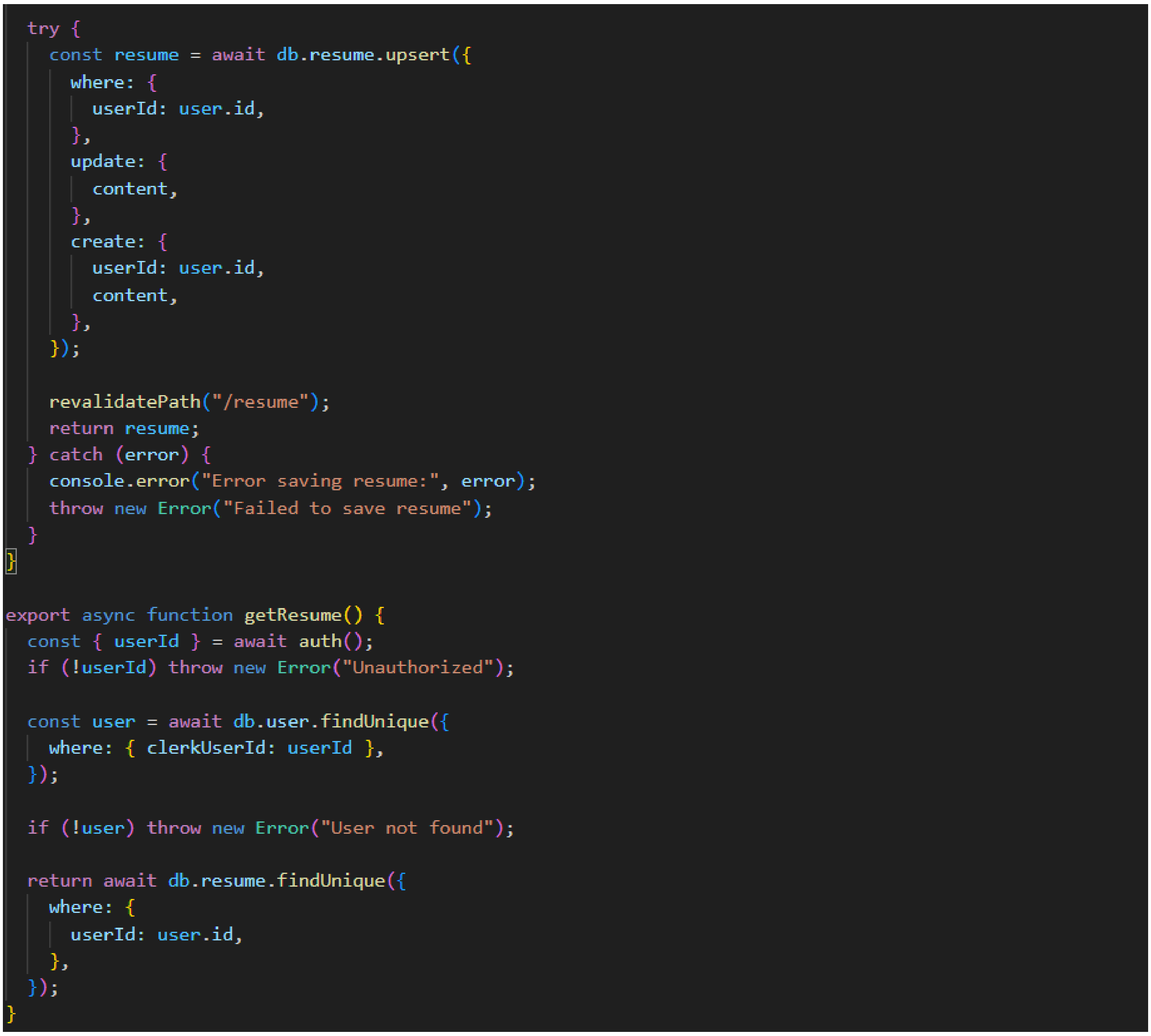


```
    return assessments;
  } catch (error) {
    console.error("Error fetching assessments:", error);
    throw new Error("Failed to fetch assessments");
  }
}
```

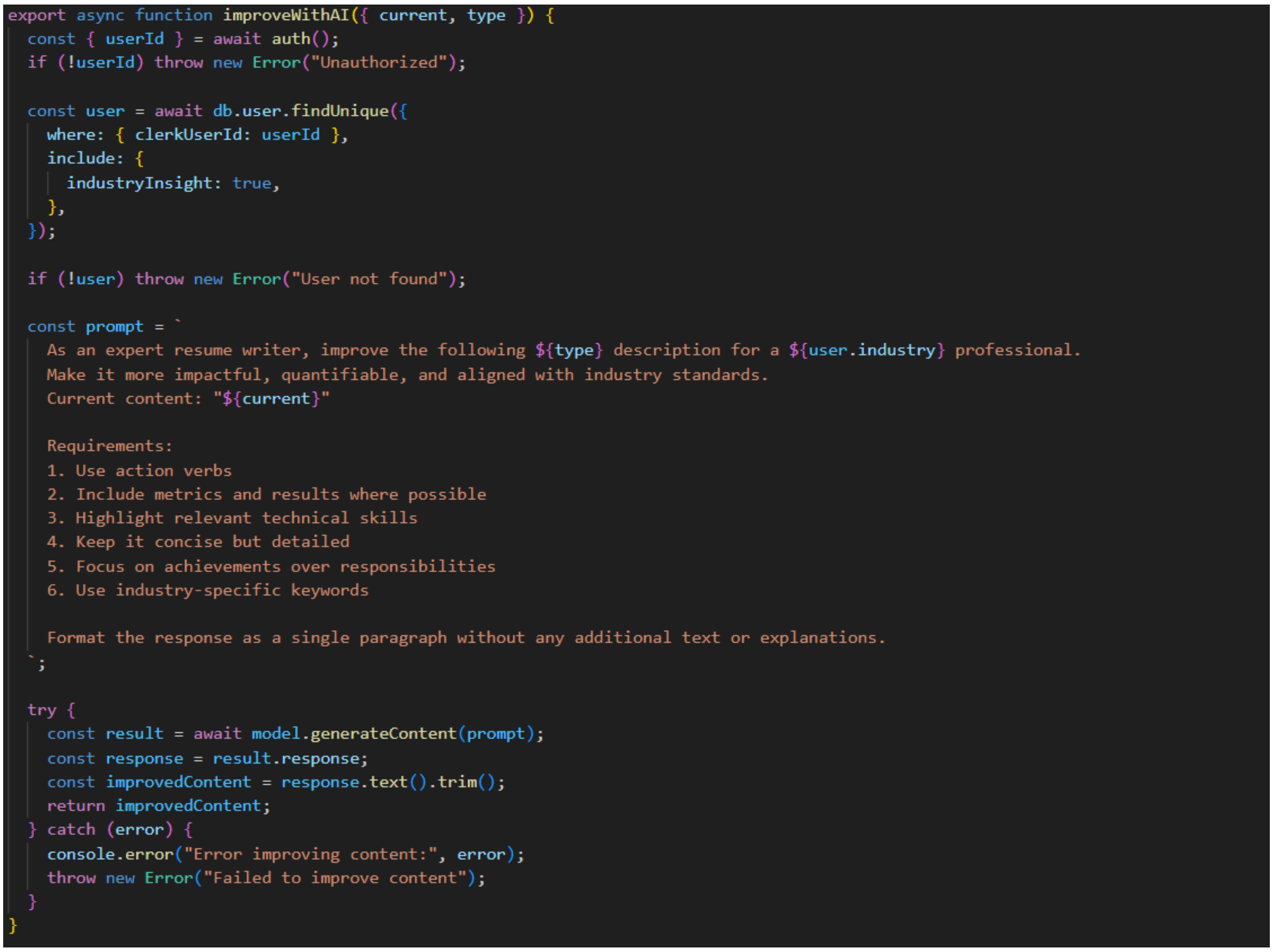
4. RESUME



```
"use server";

import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { GoogleGenerativeAI } from "@google/generative-ai";
import { revalidatePath } from "next/cache";

const genAI = new GoogleGenerativeAI(process.env.GEMINI_API_KEY);
const model = genAI.getGenerativeModel({ model: "gemini-1.5-flash" });

export async function saveResume(content) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  try {
    const resume = await db.resume.upsert({
      where: {
        userId: user.id,
      },
      update: {
        content,
      },
      create: {
        userId: user.id,
        content,
      },
    });

    revalidatePath("/resume");
    return resume;
  } catch (error) {
    console.error("Error saving resume:", error);
    throw new Error("Failed to save resume");
  }
}
```

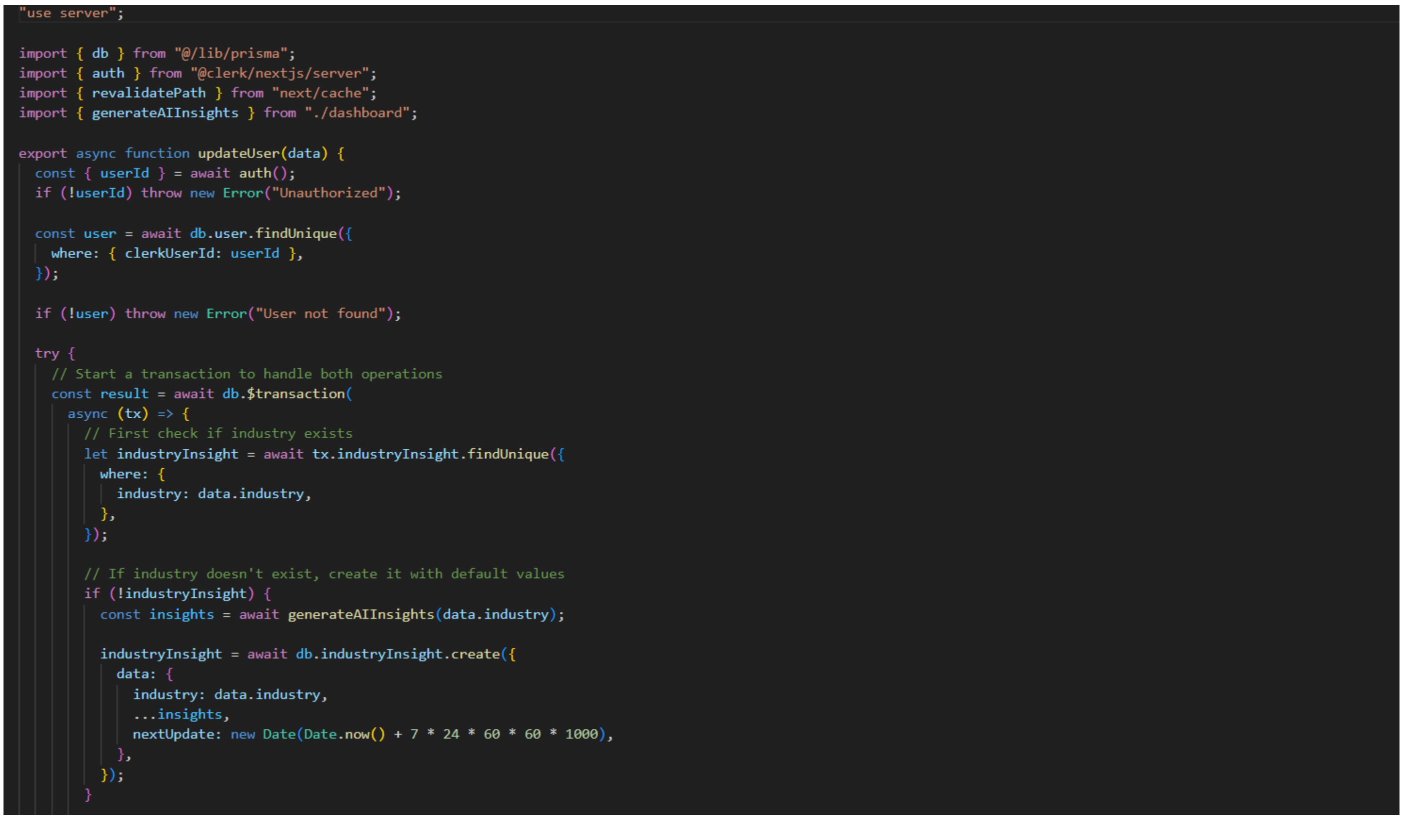


```javascript
  try {
    const resume = await db.resume.upsert({
      where: {
        userId: user.id,
      },
      update: {
        content,
      },
      create: {
        userId: user.id,
        content,
      },
    });

    revalidatePath("/resume");
    return resume;
  } catch (error) {
    console.error("Error saving resume:", error);
    throw new Error("Failed to save resume");
  }
}

export async function getResume() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  return await db.resume.findUnique({
    where: {
      userId: user.id,
    },
  });
}
```

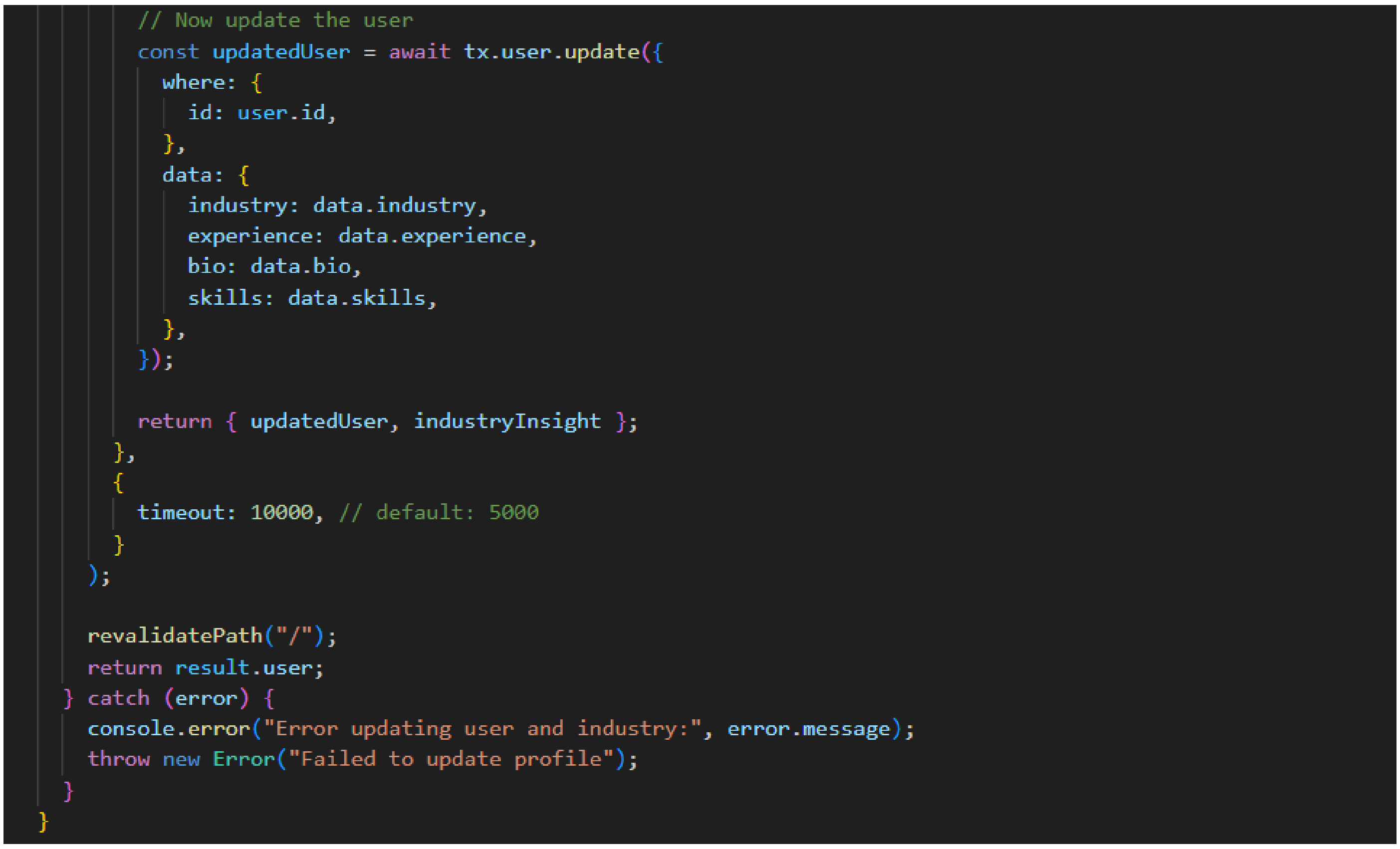


```javascript
export async function improveWithAI({ current, type }) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
    include: {
      industryInsight: true,
    },
  });

  if (!user) throw new Error("User not found");

  const prompt = `
    As an expert resume writer, improve the following ${type} description for a ${user.industry} professional.
    Make it more impactful, quantifiable, and aligned with industry standards.
    Current content: "${current}"

    Requirements:
    1. Use action verbs
    2. Include metrics and results where possible
    3. Highlight relevant technical skills
    4. Keep it concise but detailed
    5. Focus on achievements over responsibilities
    6. Use industry-specific keywords

    Format the response as a single paragraph without any additional text or explanations.
  `;

  try {
    const result = await model.generateContent(prompt);
    const response = result.response;
    const improvedContent = response.text().trim();
    return improvedContent;
  } catch (error) {
    console.error("Error improving content:", error);
    throw new Error("Failed to improve content");
  }
}
```

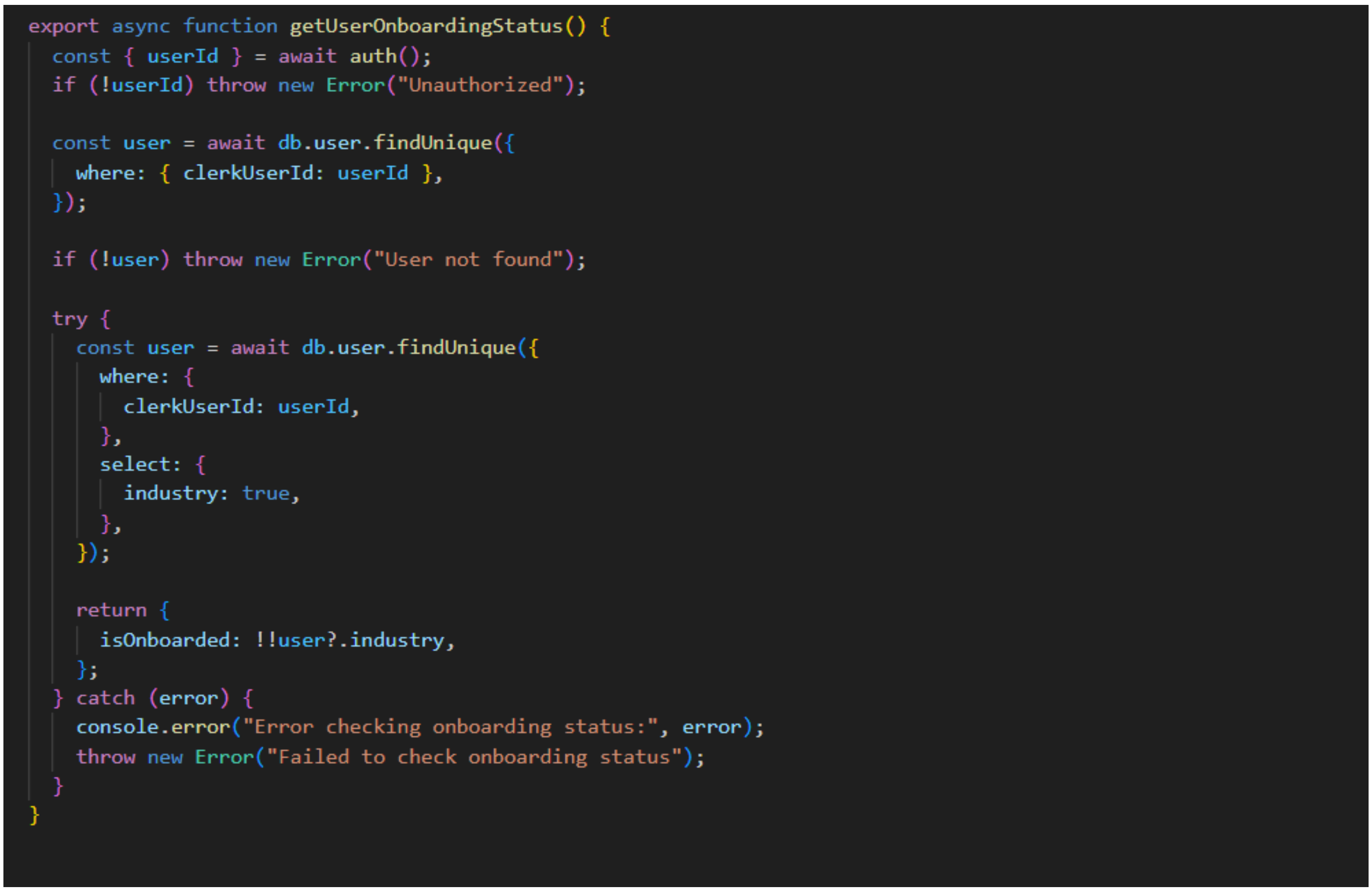
5. USER



```javascript
"use server";

import { db } from "@/lib/prisma";
import { auth } from "@clerk/nextjs/server";
import { revalidatePath } from "next/cache";
import { generateAIInsights } from "./dashboard";

export async function updateUser(data) {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  try {
    // Start a transaction to handle both operations
    const result = await db.$transaction(
      async (tx) => {
        // First check if industry exists
        let industryInsight = await tx.industryInsight.findUnique({
          where: {
            industry: data.industry,
          },
        });

        // If industry doesn't exist, create it with default values
        if (!industryInsight) {
          const insights = await generateAIInsights(data.industry);

          industryInsight = await db.industryInsight.create({
            data: {
              industry: data.industry,
              ...insights,
              nextUpdate: new Date(Date.now() + 7 * 24 * 60 * 60 * 1000),
            },
          });
        }
```

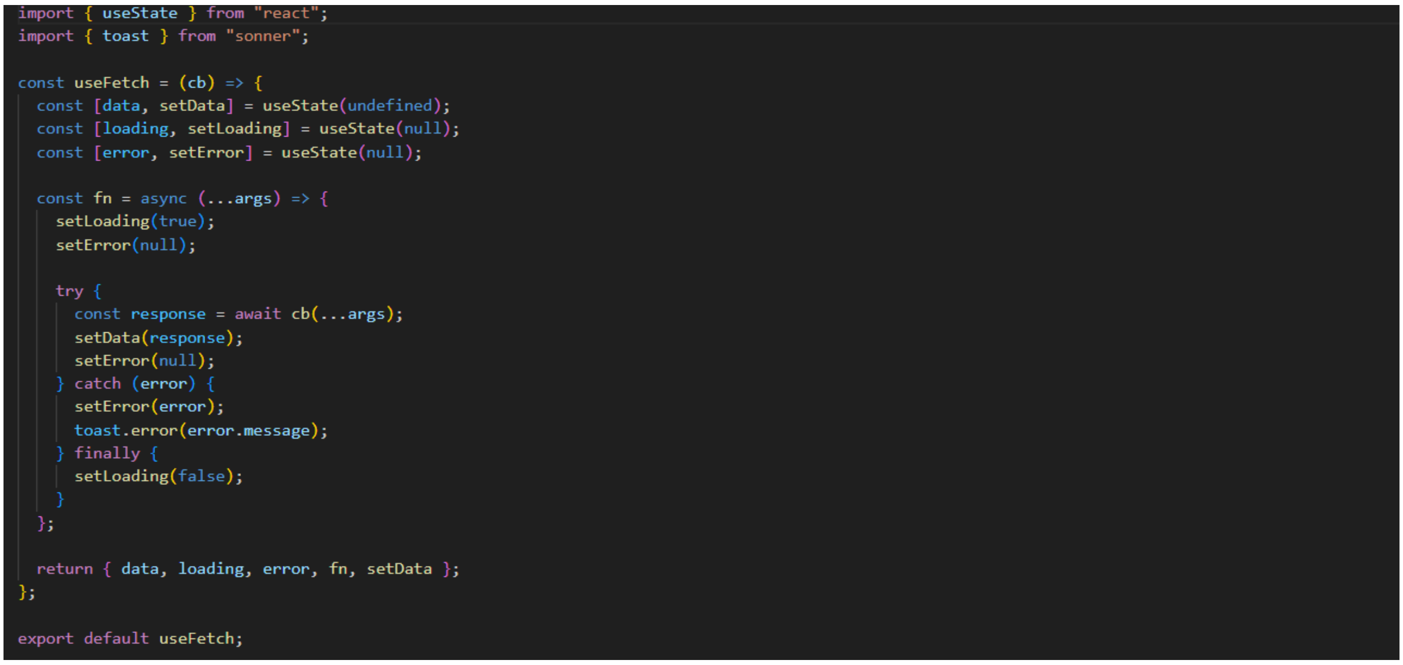


```
        // Now update the user
        const updatedUser = await tx.user.update({
          where: {
            id: user.id,
          },
          data: {
            industry: data.industry,
            experience: data.experience,
            bio: data.bio,
            skills: data.skills,
          },
        });

        return { updatedUser, industryInsight };
      },
      {
        timeout: 10000, // default: 5000
      }
    );

    revalidatePath("/");
    return result.user;
  } catch (error) {
    console.error("Error updating user and industry:", error.message);
    throw new Error("Failed to update profile");
  }
}
```



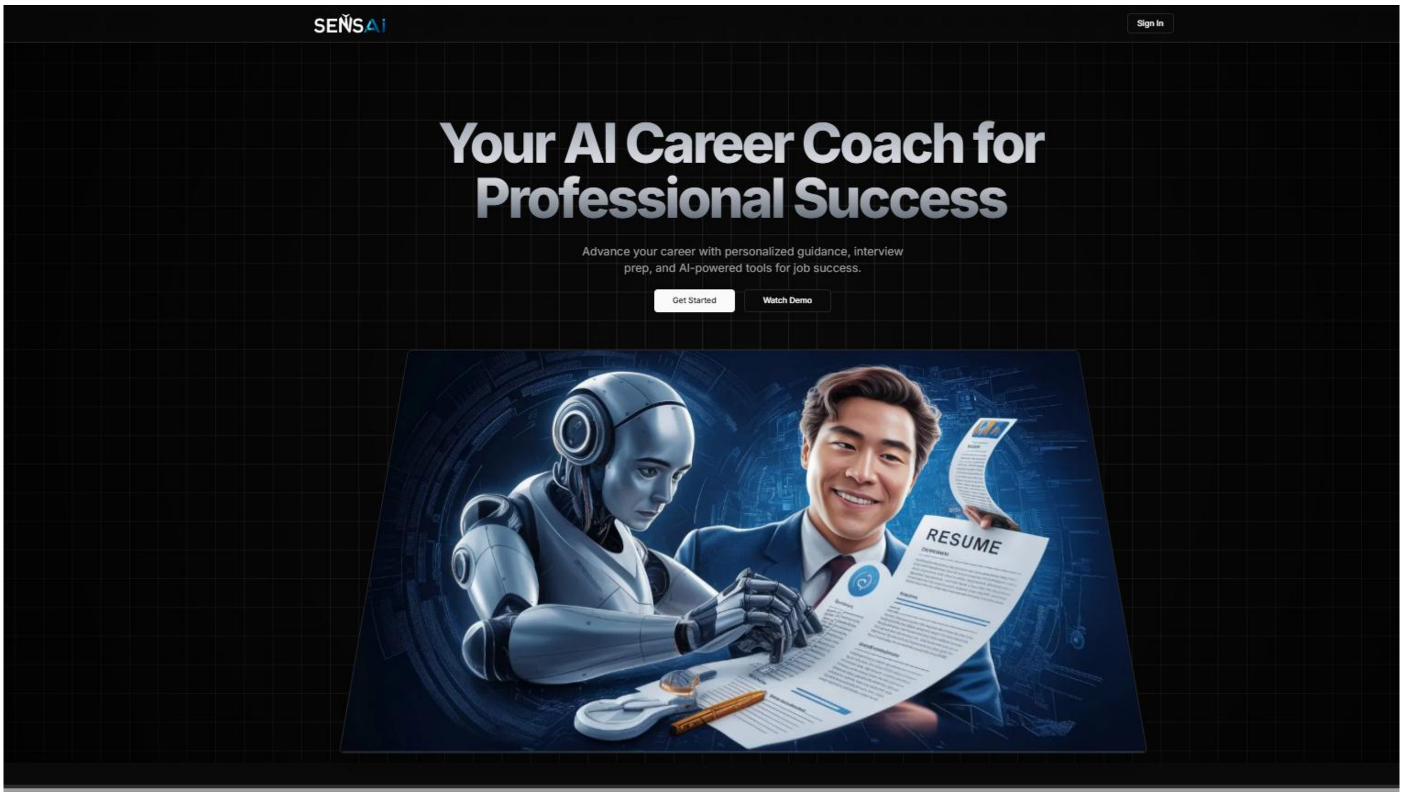
```
export async function getUserOnboardingStatus() {
  const { userId } = await auth();
  if (!userId) throw new Error("Unauthorized");

  const user = await db.user.findUnique({
    where: { clerkUserId: userId },
  });

  if (!user) throw new Error("User not found");

  try {
    const user = await db.user.findUnique({
      where: {
        clerkUserId: userId,
      },
      select: {
        industry: true,
      },
    });

    return {
      isOnboarded: !!user?.industry,
    };
  } catch (error) {
    console.error("Error checking onboarding status:", error);
    throw new Error("Failed to check onboarding status");
  }
}
```

WEBSITE HOOKS COMPONENTS



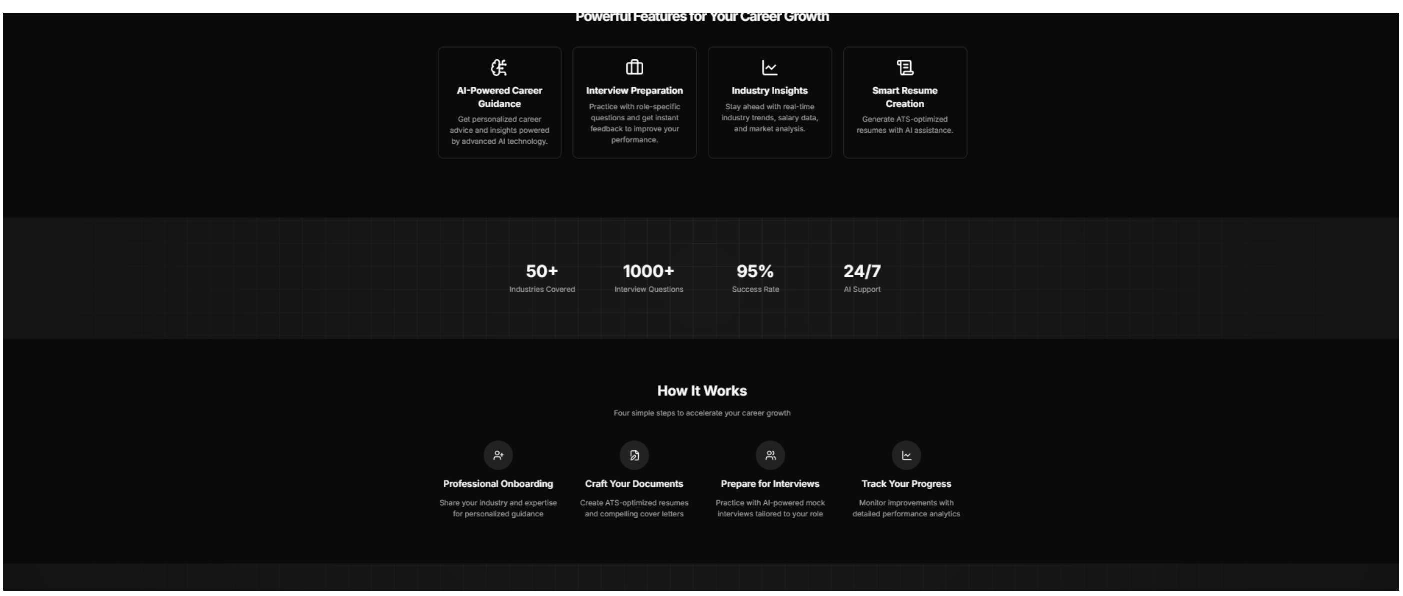
```
import { useState } from "react";
import { toast } from "sonner";

const useFetch = (cb) => {
  const [data, setData] = useState(undefined);
  const [loading, setLoading] = useState(null);
  const [error, setError] = useState(null);

  const fn = async (...args) => {
    setLoading(true);
    setError(null);

    try {
      const response = await cb(...args);
      setData(response);
      setError(null);
    } catch (error) {
      setError(error);
      toast.error(error.message);
    } finally {
      setLoading(false);
    }
  };

  return { data, loading, error, fn, setData };
};

export default useFetch;
```

Project Outputs



Your AI Career Coach for Professional Success

Advance your career with personalized guidance, interview prep, and AI-powered tools for job success.



Powerful Features for Your Career Growth

AI-Powered Career Guidance
Get personalized career advice and insights powered by advanced AI technology.

Interview Preparation
Practice with role-specific questions and get instant feedback to improve your performance.

Industry Insights
Stay ahead with real-time industry trends, salary data, and market analysis.

Smart Resume Creation
Generate ATS-optimized resumes with AI assistance.

50+	1000+	95%	24/7
Industries Covered	Interview Questions	Success Rate	AI Support

How It Works

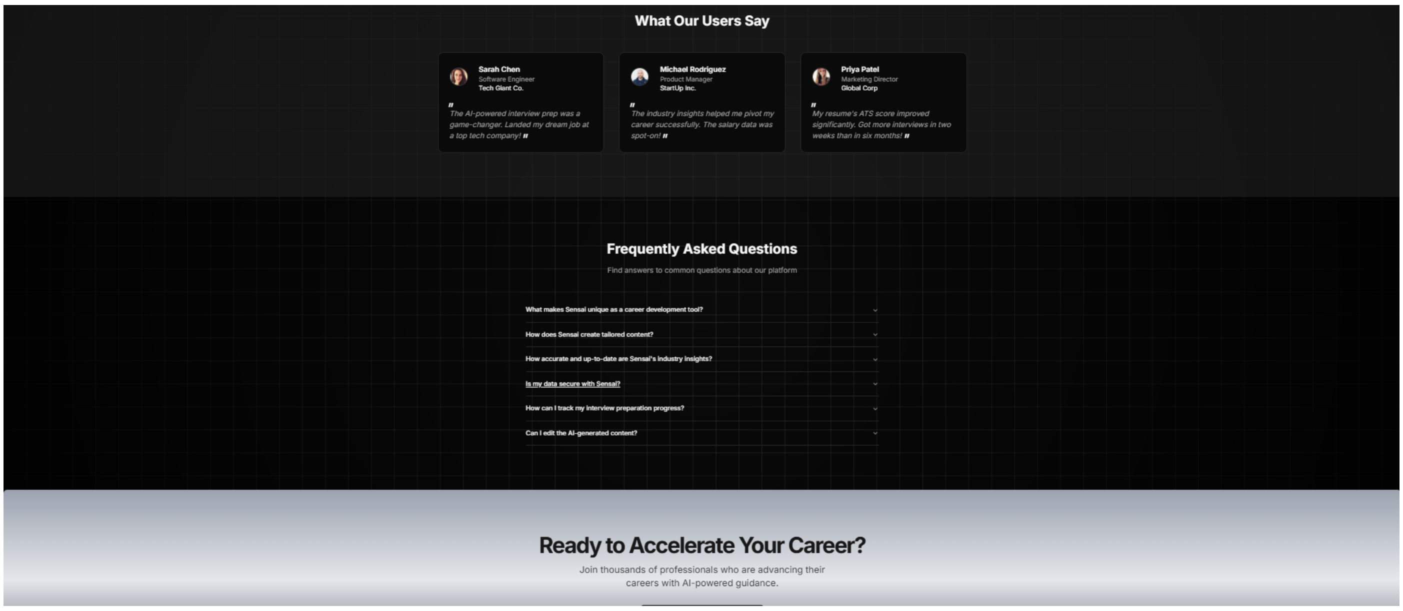
Four simple steps to accelerate your career growth

Professional Onboarding
Share your industry and expertise for personalized guidance

Craft Your Documents
Create ATS-optimized resumes and compelling cover letters

Prepare for Interviews
Practice with AI-powered mock interviews tailored to your role

Track Your Progress
Monitor improvements with detailed performance analytics



What Our Users Say

Sarah Chen
Software Engineer
Tech Giant Co.

"*The AI-powered interview prep was a game-changer. Landed my dream job at a top tech company!*"

Michael Rodriguez
Product Manager
StartUp Inc.

"*The industry insights helped me pivot my career successfully. The salary data was spot-on!*"

Priya Patel
Marketing Director
Global Corp

"*My resume's ATS score improved significantly. Got more interviews in two weeks than in six months!*"

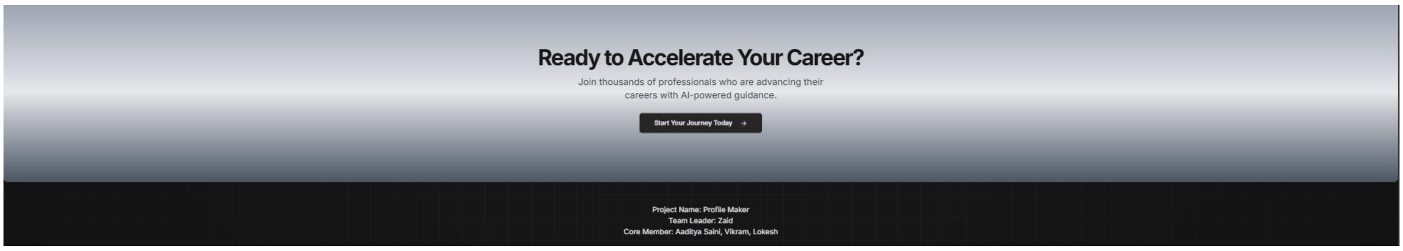
Frequently Asked Questions

Find answers to common questions about our platform

- What makes Sensai unique as a career development tool?
- How does Sensai create tailored content?
- How accurate and up-to-date are Sensai's industry insights?
- Is my data secure with Sensai?
- How can I track my interview preparation progress?
- Can I edit the AI-generated content?

Ready to Accelerate Your Career?

Join thousands of professionals who are advancing their careers with AI-powered guidance.



Ready to Accelerate Your Career?

Join thousands of professionals who are advancing their careers with AI-powered guidance.

Project Name: Profile Maker
Team Leader: Zaid
Core Member: Aaditya Saini, Vikram, Lokesh