

Sign in

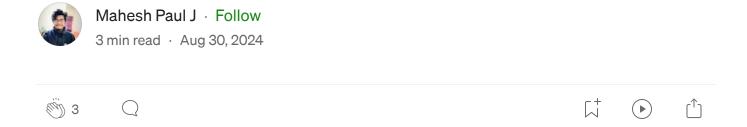
Medium







# Implementing a Free LLM AI Using OpenRouter.ai: A Step-by-Step Guide



The rise of large language models (LLMs) like OpenAI's GPT-4, Meta's Llama 3.1, and Google's Gemini Flash has brought powerful AI capabilities to developers, but accessing these models can often be costly. Luckily, there's a way to integrate these models into your projects for free using OpenRouter API. In this guide, we'll walk you through how to implement a Language Model (LLM) AI using Next.js with TypeScript, OpenRouter API.



# **Prerequisites**

Before we dive into the code, here's what you'll need:

- Next.js: A popular React framework that allows you to build fast, user-friendly web applications.
- **TypeScript:** A strongly typed programming language that builds on JavaScript, adding static types.
- **OpenRouter.ai** Account: Sign up for an account at <u>OpenRouter.ai</u> to get your API token.
- Vercel Account(optional): For deploying your Next.js app, though this is optional if you just want to run it locally.

# **Project Setup**

1. Create a New Next.js Project: Install the openai package, which ovies the tools to interact with OpenRouter.ai's API.

```
npx create-next-app@latest my-llm-app --typescript
cd my-llm-app
```

**2. Install Dependencies:** You need to install the openai package, which provides the tools to interact with OpenRouter.ai's API.

```
npm install openai
```

**3. Set Up Environment Variables:** Create a .env.local file in the root of your project to store your API key securely.

```
NEXT_PUBLIC_OPENROUTER_API_TOKEN=your-openrouter-api-token
```

# **Backend Implementation**

We'll create a simple API route that accepts a prompt and returns a script generated by the LLM.

1. Create the API Route: In your app/api directory, create a file app/api/generate/route.ts with the following code:

```
import { OpenAI } from 'openai';

// Create an OpenAI API client (that's edge friendly!)
const openai = new OpenAI({
```

```
apikey: process.env.NEXT_PUBLIC_OPENROUTER_API_TOKEN,
  baseURL: "https://openrouter.ai/api/v1/",
});
// IMPORTANT! Set the runtime to edge: https://vercel.com/docs/functions/edge-fu
export const runtime = "edge";
export async function POST(req: Request): Promise<Response> {
  const { prompt } = await req.json();
  const response = await openai.chat.completions.create({
    model: "nousresearch/hermes-3-llama-3.1-405b",
    messages: [
        role: "system",
        content: "You are an AI writing assistant that generates a script based
      },
        role: "user",
        content: prompt,
      },
    ],
    temperature: 0.7,
    top_p: 1,
    frequency_penalty: 0,
    presence_penalty: 0,
    n: 1,
  });
  const script = response.choices[0]?.message?.content!.trim() || 'No script gen
  return new Response(JSON.stringify({ script }), {
    headers: { 'Content-Type': 'application/json' },
  });
}
```

# Frontend Implementation

Next, let's create a simple frontend to interact with our API.

1. Create the Frontend Component: In your pages directory, create a file called index.tsx with the following code:

```
import { useState } from 'react';
export default function Home() {
  const [prompt, setPrompt] = useState('');
  const [script, setScript] = useState('');
  const generateScript = async () => {
    const response = await fetch('/api/generate', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json',
      },
      body: JSON.stringify({ prompt }),
    });
    const data = await response.json();
    setScript(data.script);
  };
  return (
    <div style={{ padding: '2rem' }}>
      <h1>LLM AI Script Generator</h1>
      <textarea
        value={prompt}
        onChange={(e) => setPrompt(e.target.value)}
        placeholder="Enter your prompt here..."
        rows={5}
        style={{ width: '100%', marginBottom: '1rem' }}
      <button onClick={generateScript} style={{ padding: '0.5rem 1rem' }}>
        Generate Script
      </button>
      {script && (
        <div style={{ marginTop: '2rem' }}>
          <h2>Generated Script:</h2>
          {script}
        </div>
      )}
    </div>
  );
}
```

# **Running the Application**

To run your application locally, simply execute:

npm run dev

Your application should be available at http://localhost:3000. You can enter a prompt, and the application will generate a script using the nousresearch/hermes-3-llama-3.1-405b model.

# **Deploying the Application**

If you want to deploy your application, Vercel is a great option, as it supports edge functions out of the box. Just push your code to a GitHub repository and connect it to Vercel. The deployment process is straightforward, and you can follow Vercel's documentation <u>here</u>.

### **Conclusion**

By following these steps, you've successfully integrated a powerful LLM into your Next.js application using OpenRouter.ai, all for free! Whether you're using the nousresearch/hermes-3-llama-3.1-405b model, Google's Gemini Flash, or Meta's Llama 3.1, this setup allows you to experiment with cuttingedge AI without worrying about costs.

Happy coding!

Al Open Router Ai Integration Nextjs Web Development