Next.js migration

We've moved from TanStack Start to Next.js. Here's why it matters

July 28, 2025

🚀 Next.js Migration

We migrated from TanStack Start (older version - alpha) to Next.js — here's why

After building on TanStack Start for a while, we made the leap to **Next.js** — and the results speak for themselves.

We're on GCP.

📈 Performance Gains

We went from solid to stellar:

Metric	TanStack Start	Next.js
Performance	85	✅ 100
Accessibility	85	84
Best Practices	74	78
SEO	80	✅ 100
FCP	0.3s	0.3s
Speed Index	❌ 2.4s	✅ 1.1s
TBT	50ms	60ms
CLS	0	0

🧠 Why We Switched

Better SEO and performance out of the box with Next.js' built-in optimizations

Simpler mental model for routing, layouts, and deployment (yes for real)

Edge-ready and more compatible with our future plans (middleware, streaming, etc.)

Improved observability: CPU and memory usage became more stable and efficient after the switch

🔍 What Changed?

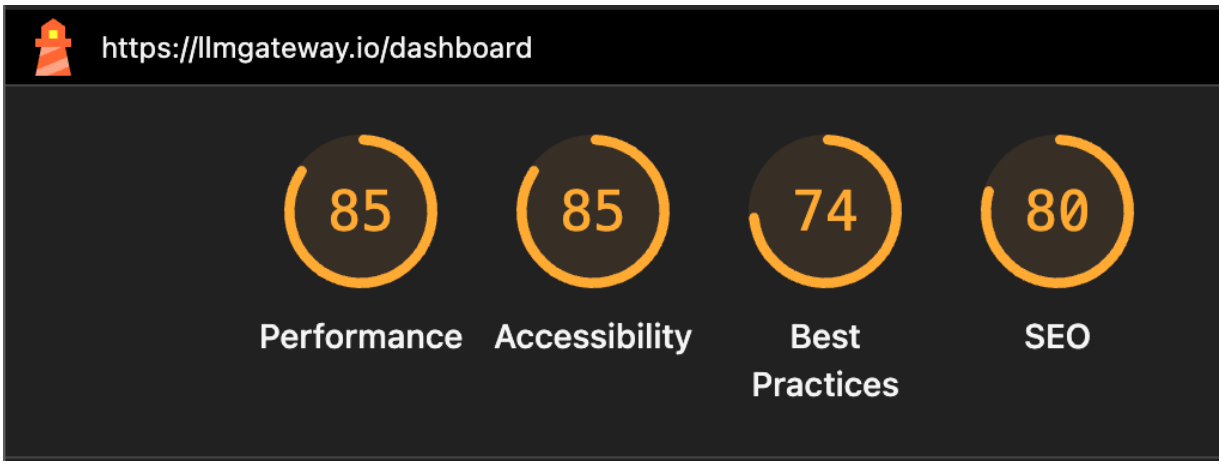
Before: Higher CPU + memory spikes, inconsistent rendering, lower SEO

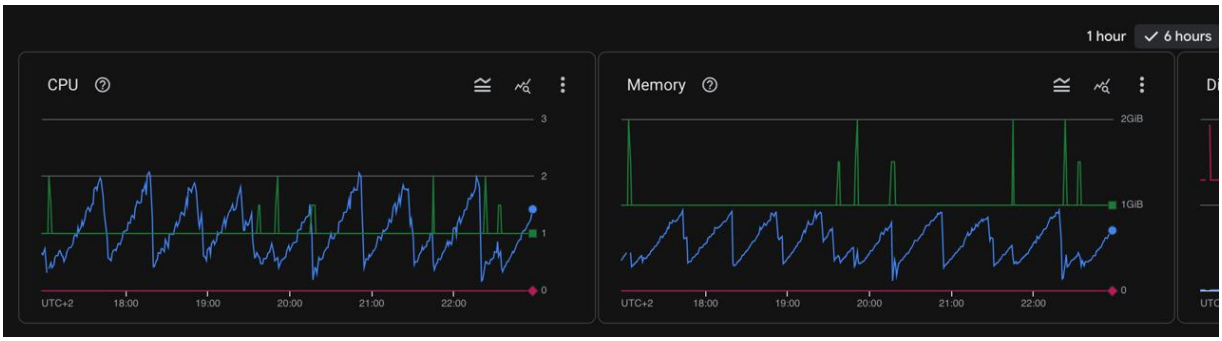
Now: Lower system resource usage, sub-1.2s paint speeds, and perfect Lighthouse scores in key areas

💡 We're just getting started, this migration sets the foundation for faster feature delivery, lower latency, and a smoother developer experience. More updates soon!

TanStack Start

First Contentful Paint 0.3 s	Largest Contentful Paint 1.2 s
Total Blocking Time 50 ms	Cumulative Layout Shift 0
Speed Index 2.4 s	





Next.js

First Contentful Paint	Largest Contentful Paint

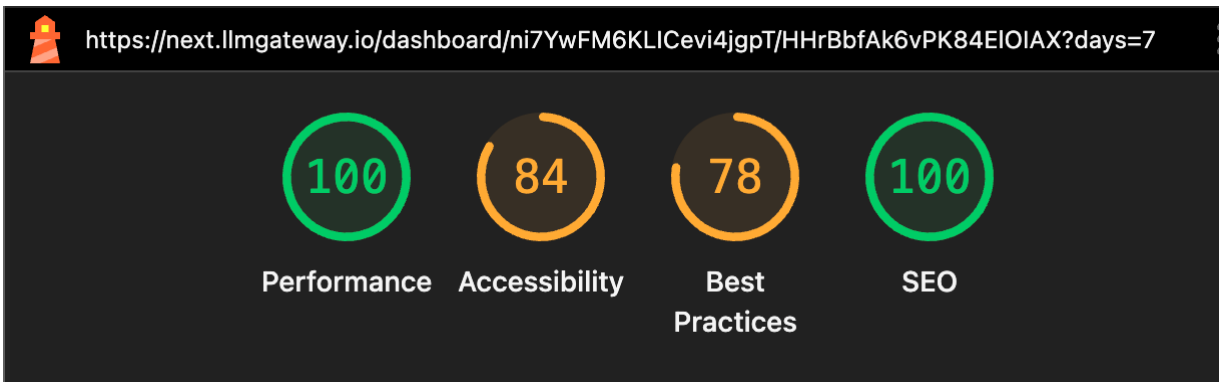
0.3 s

1.1 s

Total Blocking Time
60 ms

Cumulative Layout Shift
0

Speed Index
1.1 s



LLM Gateway

Product

Features

Models

Providers

Playground

Changelog

Resources

Documentation

GitHub

Contact Us

Community

Twitter

Discord

Compare

OpenRouter

Providers

OpenAI

Anthropic

Google Vertex AI

Google AI Studio

Inference.net

Together AI

CloudRift

Mistral AI

Moonshot AI

NovitaAI

xAI

Groq

DeepSeek

Perplexity

Alibaba Cloud

Nebius AI

Z AI

Custom