Here's a complete and clean **recap** of how you integrated **Next.js** into your **Django + Gunicorn + Nginx** stack for the GooseGooseGo (GGG) project, step by step.

## 🧱 Initial Setup: Django + Gunicorn + Nginx

Before Next.js:

* **Django** was running your backend
* Served via **Gunicorn** using a Unix socket:

bash

CopyEdit

/home/adm\_linux/GooseGooseGo/goosegoosego.sock

* **Nginx** was configured to:
  + Serve /api/ → Django
  + Serve static files → /static/
  + Serve / → Django templates or nothing

## 🧪 Goal: Add Next.js as the Frontend (SSR SPA)

You wanted:

* A **Next.js app** to render the user interface
* Keep Django as the **API backend**
* Use Nginx to smartly route between them
* Support future apps like sollottery.t21.uk

## ✅ What You Did (Recap)

### 1. 🔧 Deleted old frontend (CRA)

You initially created a React app with CRA (create-react-app), then deleted it to start fresh with:

bash

CopyEdit

npx create-next-app@latest frontend

You chose:

* App Router ✅
* Tailwind ✅
* No TypeScript ❌ (or optional)

### 2. 🧩 Built the Frontend MVP

We scaffolded:

* /app/page.tsx → DCA replay homepage
* /app/login, /register, /dashboard
* Recharts chart + mock data table
* Tailwind styling
* Google Analytics optional setup

### 3. 🚀 Built and Started the Next.js App

You ran:

bash

CopyEdit

cd ~/GooseGooseGo/frontend

npm run build

npm start

Then upgraded to use pm2:

bash

CopyEdit

pm2 start npm --name ggg-frontend -- start

Next.js was now running on port 3000.

### 4. 🔁 Updated Nginx to Proxy to Both Django and Next.js

You edited:

bash

CopyEdit

/etc/nginx/sites-available/goosegoosego

And added:

nginx

CopyEdit

upstream ggg\_frontend {

server 127.0.0.1:3000;

}

Then you split traffic like this:

| Path | Target |
| --- | --- |
| /api/ | Django via Gunicorn socket |
| /static/ | Django static files |
| / (default) | Next.js frontend (SSR) |

### 5. 🔒 SSL (Let's Encrypt) Already Handled

You already had:

nginx

CopyEdit

listen 443 ssl;

ssl\_certificate /etc/letsencrypt/...

So all traffic is HTTPS by default. Certbot handled the SSL setup.

### 6. 🏷 Bonus: Named Upstreams for Clarity

You asked:

Instead of using 127.0.0.1:3000, can we use names?

✅ Yes, we introduced:

nginx

CopyEdit

upstream ggg\_frontend {

server 127.0.0.1:3000;

}

And now you can do the same for future apps like:

nginx

CopyEdit

upstream sollottery\_frontend {

server 127.0.0.1:3002;

}

## ✅ What You Now Have

| Component | Stack |
| --- | --- |
| API backend | Django + Gunicorn |
| Frontend | Next.js (SSR) |
| Web server | Nginx |
| Process manager | PM2 |
| Domain | goosegoosego.co.uk (live) |

Traffic flows like this:

csharp

CopyEdit

[User] ──▶ [Nginx]

├─ /api/ → Gunicorn/Django

├─ /static/ → Django static files

└─ / → Next.js app (SSR via pm2 at :3000)

Would you like a shell script that automatically sets up Nginx, pm2, and SSL for a new app like sollottery.t21.uk?

Ask ChatGPT

##### You said:

Django was running your backend ... is that now not the case ?

###### ChatGPT said:

Yes — **Django is** still **running your backend.** That has **not changed**.

Here’s the clarified breakdown:

## ✅ Current Architecture Summary

### 🔧 ****Django**** (Unchanged):

* **Handles your API** — e.g. /api/dca24
* Runs via **Gunicorn**, socket-based:

bash

CopyEdit

/home/adm\_linux/GooseGooseGo/goosegoosego.sock

* Serves static files if needed (e.g. /static/)
* You can still use Django admin, authentication, models, etc.
* Accessed through **Nginx via** /api/ **routes**

### 🌐 ****Next.js**** (Now Added):

* Handles all **frontend UI and routing**
* Built separately inside ~/GooseGooseGo/frontend
* Runs on **port 3000**, managed with pm2
* Fetches data from Django via /api/ (e.g. fetch('/api/dca24'))
* Replaces Django's template rendering (you no longer need Django templates for public pages)

## ✅ How Traffic Works Now

text

CopyEdit

+-------------+

| Browser |

+------+------+

|

v

+----+----+ (Nginx listens on 443)

| Nginx |

+----+----+

|

+-------------+--------------+

| |

v v

[Django Backend] [Next.js Frontend]

@ unix socket @ http://127.0.0.1:3000

via Gunicorn via pm2 or npm start

## 🔁 Summary

* ✅ **Django is still the backend**
* ✅ **Next.js is now the frontend**
* 🛑 Django no longer handles UI rendering — only API/data
* ✅ You now have a modern separation of concerns:
  + backend = Django
  + frontend = Next.js

This is a common modern full-stack setup — clean, performant, and scalable.

Would you like to start integrating /api/dca24 into the frontend next?