Reflection: Docker and Strapi Experience

Working with Docker and Strapi for this assignment was both a challenge and a valuable learning experience. Initially, I aimed to containerize both the Nuxt frontend and Strapi backend , but I encountered several issues:

- The first major hurdle was understanding how Docker works — especially how Dockerfiles and docker-compose.yml work together.
- I struggled with database configuration , switching from the default SQLite (which worked locally) to PostgreSQL for Docker compatibility. This led to errors like Cannot find module 'pg' and better-sqlite3 not working inside containers.
- Another issue was line-ending warnings (LF will be replaced by CRLF) on Windows, which caused Git confusion when pushing code.
- Git also mistakenly treated my frontend folder as a nested repository due to past actions, causing pathspec did not match errors.

Through trial and error, I learned:

1. How to configure Strapi to work with different databases
2. Best practices for separating frontend and backend in Docker
3. The importance of .gitignore and clean repo structure
4. How to debug container logs and environment variables

Eventually, I chose to revert to running both apps locally using npm run develop and npm run dev, avoiding Docker altogether for now
.

How to run:

Step 1:

- Start the Strapi backend
- cd blog-backend
- npm install
- npm run develop

Step 2:
In a new terminal, start the Nuxt frontend:

- cd ../blog_frontend
- npm install
- npm run dev

The blog will be available at:
Frontend : http://localhost:3000
Strapi Admin : http://localhost:1337/admin

This project taught me a lot about full-stack development, dependency management, and version control — and most importantly, when to simplify and focus on what works .