Next.js Senior Technical Test – Clean Architecture

Objective

Build a **Next.js application "Task Manager"** that allows collaborative task management.

The goal is to evaluate your ability to:

- Design a **scalable and modular architecture**.
- Properly separate **Domain / Data / Presentation layers**.
- Apply **Clean Architecture principles** (dependency rule, testability).
- Write clean, maintainable, and evolvable code.

Required Features

1. **Authentication**
    - User login with email & password.
    - User signup.
2. **Task Management**
    - Create a task (title, description, status: "to do / in progress / done").
    - Edit / Delete a task.
    - List all tasks.
    - Filter tasks by status.
3. **Future Scalability**
   The code should be designed in a way that it can easily evolve in the future, for example:
    - Support multiple projects (each containing several tasks).
    - Add a notification system.

⚙ Technical Requirements

- **Architecture** must follow **Clean Architecture** principles:
    - **Domain Layer**: Entities, use cases.
    - **Data Layer**: Repositories (Use server action).
    - **Presentation Layer**: UI + State Management.
    - **Unit tests** are required on **Domain use cases**.

Deliverables

1. A **public GitHub or GitLab repository link** containing the full source code.
    - /!\ The git repos should have a clean commits history /!\
    - /!\ No VIBE CODING ALLOWED /!\
    - The project must be runnable with docker-compose
    - The repository must include a **clear README** explaining:
        - The project structure.
        - The technical choices made.
        - Instructions to run the application.
2. **Screenshots of all main website pages**
   (authentication, task list, create/edit task).
    Screenshots should be included in the README or in a `/screenshots` folder.
3. **OR a short demo video (screen recording)**
    - Showcasing the application running.
    - Explaining briefly the architectural and technical choices.

Bonus (optional but highly valued)

- Strict application of the **Dependency Inversion Principle** (inner layers never depend on outer ones).
- Test coverage > 70% on Domain + Data layers.
- CI/CD (e.g. GitHub Actions to build and run tests automatically).
- Additional documentation (e.g. architecture diagram).
- Smart UI/UX

⏱ Suggested Time: 3h–4h From the first to the last commit

We are not expecting a “perfect” production-ready app, but we will evaluate your ability to:

- Deliver a **clear and extensible architecture**.
- Properly **separate responsibilities**.
- Anticipate the **future evolution of the project**.

/!\ The git repos should have a clean commits history /!
/!\ No VIBE CODING ALLOWED /!\