Full-Stack Intern Assignment; NestJS + Postgres + Prisma + React (React Native optional)

Objective (single full-stack task):
Build a **full-stack** library system: a **NestJS** backend (Postgres + Prisma) exposing REST APIs and a **React.js (TypeScript)** frontend that consumes those APIs. Optionally, candidates may implement a **React Native (Android)** mobile app instead of the web frontend.

What to build

A single integrated app that supports:

- CRUD for **Books** (create / read / update / delete; list with filters).
- CRUD for **Authors** (create / read / update / delete; list).
- CRUD for **Users** (create, list).
- **Borrowing** flows:
    - Mark a book as borrowed by a user.
    - Return a borrowed book.
    - List borrowed books for a user.
- Simple **JWT** authentication for protected operations.

Candidates design exact routes, Prisma schema, and UI flows; document choices in README.

Functional (API) expectations

Implement REST endpoints for:

1. **Books**
    - Add a new book.
    - Update book details.
    - Delete a book.
    - List all books with optional filters (e.g., by author, borrowed status).
2. **Authors**
    - Add a new author.
    - Update author details.
    - Delete an author.
    - List all authors.

3. **Users**
    - Add a new user.
    - List all users.
4. **Borrowed Books**
    - Mark a book as borrowed by a user.
    - Return a borrowed book.
    - Fetch all borrowed books for a user.
5. **Authentication**
    - Simple JWT authentication

Technical stack (required)

- **Backend:** NestJS (TypeScript)
- **ORM:** Prisma (migrations) + Postgres (Supabase free tier or any other)
- **Frontend:** React.js (TypeScript) - primary requirement
    - **Alternately:** React Native (Android) as an alternate frontend
- **API style:** REST
- **Auth:** JWT
- **Containerization:** `Dockerfile` (and optional `docker-compose.yml with postgress` - good to have)
- Use `.env` with `.env.example`

Frontend expectations (React.js)

- Web app in **React (TypeScript)** that:
    - Logs in and persists JWT
    - Lists/adds/edits/deletes authors and books
    - Allows creating users
    - Lets a user borrow and return books; shows user's borrowed list
- If the candidate prefers mobile, a **React Native (Android)** app may be provided instead.

Good-to-have (bonus)

- Dockerized dev environment (backend + Postgres)
- Swagger/OpenAPI docs for the backend
- Tests for core backend flows
- Advanced server-side filtering for books (combinations of search, author, availability, date ranges, pagination)
- Clean pixel perfect ui

- Use of proper state management

Deliverables

- A GitHub repo containing:
  - Backend source: NestJS + Prisma + migrations
  - Frontend source: React (TypeScript) app (or optional React Native app)
  - `README.md` with:
    - How to run backend and frontend locally (with and without Docker/Supabase)
    - How to run migrations and seed data
    - How to get a token and test protected routes
    - Assumptions and any design notes
  - `.env.example`
- Demo video recording
- APK or build/run instructions for Android
- Optional: Swagger/OpenAPI or Postman collection

Evaluation criteria

- Correctness: APIs work and are used by the frontend
- Schema design: sensible Prisma models and relations
- Code quality and TypeScript usage
- Auth: JWT protection for write operations
- Documentation: clear README and run instructions
- UI: functional frontend; polished UI is a plus
- Bonus: Docker, tests, API docs, pixel perfect ui and improved filtering

Very Important Note

We encourage using AI tools in our team, but **“vibe coding” is not encouraged for this assignment**.
We want to understand your actual coding ability, problem-solving, and your approach to designing a working system.

You can absolutely use AI to help you think, structure, or debug; but the code should be **yours**, understandable, and maintainable.

Suggested timebox

Suggested 5-7 days. Focus on a working app with clear README over many unfinished features.