Full-Stack Developer Assignment: E-commerce Product Management System

This assignment evaluates a candidate's full-stack development skills by having them build a product management system. The project now includes a many-to-many relationship and requires advanced features, along with using the `shadcn/ui` component library.

Project Overview

The goal is to build a web application that allows an administrator to manage a catalog of products and their associated categories. The application should have the following core functionalities:

1. **View Products**: Display a list of all products, including their assigned categories.
2. **Add/Edit Product**: A form to add or update a product. This form should allow the user to select one or more categories for the product.
3. **Manage Categories**: A separate section to create, edit, and delete categories.
4. **Delete Product**: A way to remove a product. Deleting a product should not delete its associated categories.

Product Entity	Data Type	Constraints
id	UUID or integer	Primary Key, Auto-generated
name	string	Required, max 255 chars
description	text	Optional
price	numeric (e.g., `DECIMAL(10, 2)`)	Required, must be greater than 0
stockQuantity	integer	Required, must be a non-negative integer

imageUrl	string	Optional, a valid URL

Category Entity	Data Type	Constraints
id	UUID or integer	Primary Key, Auto-generated
name	string	Required, max 255 chars

Relationship: A **many-to-many relationship** between Product and Category. This will require a third table, often called ProductCategory, to link products and categories.

Technical Requirements

1. Back-end

- **Language & Framework**: Node.js with Express.
- **Database**: PostgreSQL. The candidate should define the schemas for Product, Category, and the join table.
- **ORM**: Use **Prisma** or **TypeORM** to manage the database schema and relationships.
- **API**:
    - Create **RESTful API** endpoints for **CRUD** operations on both **Product** and **Category** entities.
    - The Product endpoints must handle the many-to-many relationship, allowing the front-end to associate categories with products.
    - Use **TypeScript** throughout for type safety.
    - Implement robust **input validation** and **error handling**.
    - **Advanced Feature**: Implement **pagination** and **filtering** on the product list API endpoint (e.g., filter by category).

2. Front-end

- **Library & Language**: React with **TypeScript**.
- **UI Components**: Use **shadcn/ui** for all UI components (e.g., Table, Dialog, Form, Select). The candidate must demonstrate proficiency in installing and configuring this library.

- **State Management**: Use React Hooks (`useState`, `useEffect`) and the **React Query** library for managing server-side data (fetching, caching, and mutations). This is a key requirement to evaluate handling of asynchronous data.
- **Styling**: Use **Tailwind CSS**, as required by `shadcn/ui`.
- **UI Features**:
    - A **product list page** with a data table displaying products, including their categories. Implement a search bar or filters to narrow down the list.
    - A **modal form** to add or edit a product. This form must include a multi-select component (using `shadcn/ui`'s `Select` or a similar component) for categories.
    - A separate **category management page** with its own CRUD interface.
    - Use `shadcn/ui`'s `Dialog` for modals and `Form` for form handling.

Deliverables and Evaluation

The candidate should deliver a single Git repository. Evaluation will focus on:

- **Code Quality**: Clean, well-structured, and maintainable code.
- **Functionality**: All CRUD operations must be fully functional for both products and categories. Pagination and filtering must work correctly.
- **Technical Proficiency**:
    - Correct implementation of the **many-to-many relationship** in the database and API.
    - Effective use of **TypeScript** for type safety on both the front-end and back-end.
    - Appropriate use of **React Query** for data fetching and mutations.
    - Proper integration and customization of **`shadcn/ui`** components.
- **Project Setup**: Clear instructions to get the project running.
- **Bonus Points**:
    - Integration of **environment variables** for database credentials.
    - Simple **authentication/authorization**.
    - Writing **unit or integration tests**.
    - Clear Git commit history.
    - Implementation of **optimistic UI updates** using React Query for a better user experience.