Frontend Challenge: Products Dashboard

Goal

Build a small React + TypeScript + MUI v5 web app that fetches “products” (use the products API from DummyJSON) and lets the user browse, search, filter, view details, and create/edit products.

https://dummyjson.com/docs/products

Focus on clean architecture, type safety, UI/UX quality, theming and performance.

Tech Stack (Required)

React 18+

TypeScript

Material UI v5 (@mui/material)

React Router v6+

Optional (nice-to-have)

Form handling with react-hook-form

Form validation with yup

API

Use DummyJSON as your external data source.

- List products
- Search products
- Get product details
- Create a new product (for demo purposes, data is not persisted on API)
- Update a product

Focus on sending correct requests and updating the UI optimistically.

Requirements

1. Products List Page (/)

Display products in a Material UI Table or Card Grid

Include:

- Search bar (debounced 300 ms)
- Filter by category
- Pagination (10 per page)
- Show loading and error states
- Display “No results found” for empty states

2. Product Details Page (/products/:id)

Show full product info: name, price, category, description, etc.

Include a “Back” button that preserves filters/search state

3. Create / Edit product Form (/new and /products/:id/edit)

Use MUI TextField, Select, and Button components

Client-side validation:

- title: required (2–100 chars)
- price: required, number ≥ 0
- category: required

Disable submit until the form is valid

On submit, POST or PUT to the API and navigate back to the detail page

Show a Snackbar on success

Theming & UI

Implement a custom MUI theme with:

Custom primary/secondary colors

Typography customization

Add a light/dark mode toggle in the app bar

Ensure the layout is responsive on mobile and desktop

Performance

Debounce search requests

Avoid unnecessary re-renders (React.memo, useMemo, useCallback)

Lazy-load details and form pages (code splitting)

Deliverables

A GitHub repository (or ZIP) with:

Complete source code

README.md explaining:

- How to run and test the app
- Project structure and design decisions
- Trade-offs and what you'd improve with more time

Working commands:

npm install
npm run dev

Tips

Focus on readable, maintainable code over completeness.

Use TypeScript interfaces/types for all API responses.

Write small, testable components.

Show off your MUI skills (custom styles, responsive layouts, overrides).