Frontend Developer Assignment

Objective: Build a small admin dashboard to manage Products and Orders. Data must be mocked on the frontend (JSON + simulated API calls) and persisted using Redux (rehydrated from localStorage).

Tech stack

- React + TypeScript
- Redux Toolkit (slices + async thunks)
- Tailwind CSS (only styling method; no plain CSS or UI frameworks)
- Mock API using JSON + Promise/setTimeout (frontend-only)
- Persistence: Redux state must rehydrate from localStorage on reload

What to build

Create routes/pages for the following:

1) Dashboard (/)

Show summary + charts (data sourced from Redux, not hardcoded):

- Summary cards: Total Products, Active Products, Total Orders, Pending Orders, Revenue (sum of completed orders)
- Charts (use any chart library):
- - Bar/Line: Orders per day (last 7 or 14 days)
- - Pie/Donut: Order status distribution (Pending/Completed/Cancelled)

2) Products (/products)

Responsive UI: desktop table, mobile cards list.

Features:

- Search by name
- Filter by category and status (Active/Inactive)

- Sort by price OR updatedAt
- Pagination (10 per page)

Actions:

- Create product (/products/new)
- Edit product (modal or route)
- Toggle status (Active/Inactive)
- Delete product (soft delete preferred)

Validation:

- name: min 3 characters
- price: > 0
- stock: >= 0
- category: required
- rating: 1-5

3) Orders (/orders)

Create orders from products:

- Add products to a cart
- Quantity controls
- Place order with: id, items, total, createdAt, status, customerName

Orders list:

- Filter by status
- View order details
- Update status (Pending -> Completed/Cancelled)

Mock data + API simulation (must)

Create:

- src/mock/products.json
- src/mock/orders.json
- src/mock/api.ts

Rules:

- Initial seed data must come from the JSON files.
- Simulate API calls using Promise + setTimeout (delay ~300-800ms).
- Thunks call API, then update Redux. API must not mutate Redux directly.

Redux Toolkit + persistence (must)

- Use slices: productsSlice, ordersSlice (uiSlice optional).
- Persist products + orders to localStorage.
- Rehydrate store on app start.
- Do NOT persist loading flags or transient UI state.

Custom hooks (must)

Create and use a few reusable, typed custom hooks. Examples (pick at least 2):

- useDebounce(value, delay) for search
- usePagination(data, pageSize)
- useFilters(products) and/or useSort(data)
- useLocalStorageState(key, initialValue) OR a small persistence helper used by the store setup

Code quality expectations

- Desktop-first responsive UI with Tailwind breakpoints (sm/md/lg).
- Optimize derived lists using useMemo and stable callbacks with useCallback where needed.
- Use typed hooks: useAppDispatch/useAppSelector.
- Add comments for non-obvious logic (persistence, selectors, chart data transformation).

Deployment (required)

Deploy the app and share the live URL in your submission and in README. Use any one: Vercel or Netlify.

Deliverables

- GitHub repository link
- Deployed live URL (Vercel/Netlify)