Bài Tập Cuối Chương 5

Bài tập thực hành: Xây dựng ứng dụng Job Board với React Router DOM v7

Tổng quan

Bạn sẽ xây dựng một **ứng dụng Job Board** để quản lý danh sách công việc (jobs). Người dùng có thể xem danh sách công việc, chi tiết công việc, ứng tuyển, và quản lý công việc qua trang Admin. Ứng dụng sử dụng **React Router DOM v7** để định tuyến và **Context API** để quản lý state toàn cục.

Mục tiêu

- Ôn lại React cơ bản (component, state, props) và Context API (quản lý state toàn cục).
- Thành thạo React Router DOM v7: route tĩnh/động, nested routes, điều hướng (`Link`, `NavLink`, `Navigate`), xử lý lỗi.
- Xây dựng ứng dụng hoàn chỉnh với giao diện đơn giản nhưng đầy đủ tính năng.

Cấu trúc thư mục

Tạo cấu trúc thư mục như sau trong thư mục `src/`:

```
src/
├── components/          # Các component dùng chung
│   ├── Header/          # Thanh điều hướng
│   │   ├── Header.jsx
│   │   └── Header.css
│   └── JobApplicationForm/  # Form ứng tuyển
│       ├── JobApplicationForm.jsx
│       └── JobApplicationForm.css
├── screens/             # Các màn hình chính
│   ├── Home/            # Trang chủ
│   │   └── Home.jsx
```

```
│   ├── JobList/          # Danh sách công việc
│   │   └── JobList.jsx
│   ├── JobDetail/        # Chi tiết công việc
│   │   └── JobDetail.jsx
│   ├── Admin/            # Trang quản lý (nested routes)
│   │   ├── Admin.jsx
│   │   ├── JobManager/   # Quản lý danh sách công việc
│   │   │   └── JobManager.jsx
│   │   └── AddJob/       # Thêm công việc mới
│   │       └── AddJob.jsx
│   └── NotFound/         # Trang lỗi 404
│       └── NotFound.jsx
├── context/              # Context để quản lý state
│   └── JobContext.jsx
├── App.jsx               # File chính kết nối routes
└── index.css             # CSS global (tùy chọn)
```

Yêu cầu bài tập

Ứng dụng Job Board cần có các tính năng sau, được chia thành 4 phần:

Phần 1: Cấu hình cơ bản

Mục tiêu: Thiết lập định tuyến cơ bản và điều hướng giữa các trang.

Hướng dẫn:

1. Cài đặt React Router DOM v7 bằng lệnh npm trong terminal (bạn tự tìm lệnh phù hợp).
2. Trong `App.jsx`:
   - Sử dụng `<BrowserRouter>` để bao bọc toàn bộ ứng dụng.
   - Thêm `<Routes>` với 3 route:
     - `/` → Hiển thị màn hình `Home`.
     - `/jobs` → Hiển thị màn hình `JobList`.
     - `/jobs/:id` → Hiển thị màn hình `JobDetail` với `id` là tham số động.

3. Tạo `Header.jsx` trong `components/Header/`:
    - Thêm thanh điều hướng với 2 liên kết (`<Link>`): "Home" tới `/` và "Jobs" tới `/jobs`.
    - Tạo file CSS (`Header.css`) để style cơ bản (màu nền xanh dương, chữ trắng, khoảng cách giữa các link).
4. Tạo các file màn hình trong `screens/`:
    - `Home.jsx`: Hiển thị tiêu đề "Welcome to Job Board".
    - `JobList.jsx`: Hiển thị placeholder "Job Listings (Coming Soon)".
    - `JobDetail.jsx`: Dùng `useParams` để lấy `id` từ URL và hiển thị placeholder "Job Detail - ID: [id] (Coming Soon)".

Kết quả mong đợi:

- Truy cập `http://localhost:3000/` thấy tiêu đề "Welcome to Job Board".
- Truy cập `/jobs` thấy "Job Listings (Coming Soon)".
- Truy cập `/jobs/1` thấy "Job Detail - ID: 1 (Coming Soon)".
- Thanh điều hướng (`Header`) hiển thị ở đầu mỗi trang với 2 liên kết hoạt động.

Phần 2: Quản lý state với Context API

Mục tiêu: Sử dụng Context API để quản lý danh sách công việc và hiển thị dữ liệu thực tế.

Hướng dẫn:

1. Tạo `JobContext.jsx` trong `context/`:
    - Tạo Context với `createContext` và export trực tiếp dưới tên `JobContext`.
    - Tạo `JobProvider` component với state `jobs` chứa danh sách công việc mẫu (ít nhất 2 công việc, mỗi công việc có `id`, `title`, `company`, `description`).
2. Trong `App.jsx`:
    - Bao bọc toàn bộ ứng dụng bằng `<JobProvider>` (đặt bên ngoài `<BrowserRouter>`).
3. Cập nhật `JobList.jsx`:

- Dùng `useContext(JobContext)` để lấy danh sách `jobs`.
- Hiển thị danh sách dưới dạng `<ul>` với mỗi `<li>` chứa `<Link>` dẫn tới `/jobs/[id]` và nội dung là "title - company".

4. Cập nhật `JobDetail.jsx`:
   - Dùng `useContext(JobContext)` để lấy danh sách `jobs`.
   - Tìm công việc theo `id` từ `useParams` và hiển thị `title`, `company`, `description`.
   - Nếu không tìm thấy công việc, hiển thị "Job not found".

Kết quả mong đợi:

- `/jobs` hiển thị danh sách công việc (ví dụ: "Frontend Developer - Tech Co", "Backend Developer - Data Inc").
- Nhấp vào một công việc (ví dụ `/jobs/1`) hiển thị chi tiết: tiêu đề, công ty, mô tả của công việc đó.
- Truy cập `/jobs/3` (giả sử không tồn tại) hiển thị "Job not found".

Phần 3: Nested Routes và thêm công việc

Mục tiêu: Thêm trang Admin với nested routes và form ứng tuyển công việc.

Hướng dẫn:

1. Trong `JobContext.jsx`:
   - Thêm state `applications` (mảng lưu thông tin ứng tuyển: `jobId`, `name`, `email`).
   - Cập nhật giá trị `value` của `JobProvider` để bao gồm `jobs`, `setJobs`, `applications`, và `setApplications`.
2. Trong `App.jsx`:
   - Thêm route `/admin` với component `Admin` và 2 nested routes:
     - `/admin/jobs` → Hiển thị `JobManager`.
     - `/admin/add` → Hiển thị `AddJob`.
3. Tạo `Admin.jsx` trong `screens/Admin/`:
   - Hiển thị tiêu đề "Admin Panel".

- Thêm `<nav>` với 2 `<Link>`: "Manage Jobs" tới `jobs` và "Add Job" tới `add`.
- Thêm `<Outlet>` để render các route con.

4. Tạo `JobManager.jsx` trong `screens/Admin/JobManager/`:
   - Dùng `useContext(JobContext)` để lấy `jobs`.
   - Hiển thị danh sách dưới dạng `<ul>` với mỗi `<li>` là "title - company".
5. Tạo `AddJob.jsx` trong `screens/Admin/AddJob/`:
   - Tạo form với 3 field: `title`, `company`, `description` (dùng state cục bộ).
   - Dùng `useContext(JobContext)` để lấy `jobs` và `setJobs`.
   - Khi submit, thêm công việc mới vào `jobs` (id = số lượng công việc hiện tại + 1) và chuyển hướng về `/admin/jobs` (dùng `useNavigate`).
6. Tạo `JobApplicationForm.jsx` trong `components/JobApplicationForm/`:
   - Tạo form với 2 field: `name` và `email` (dùng state cục bộ).
   - Dùng `useContext(JobContext)` để lấy `applications`.
   - Dùng `useParams` để lấy `id` của công việc, `useNavigate` để chuyển hướng sau khi submit.
   - Khi submit, tạo object `{ jobId: id, name, email }`, thêm vào `applications` bằng `setApplications`, hiển thị alert "Application submitted successfully!" rồi chuyển về `/jobs`.
   - Tạo file CSS để style form (các field xếp dọc, nút màu xanh lá).
7. Cập nhật `Header.jsx`:
   - Thêm `<NavLink>` tới `/admin`.
   - Thêm style active (ví dụ: chữ đậm và gạch chân).
8. Cập nhật `JobDetail.jsx`:
   - Import `useContext` và `JobContext`, dùng `useContext(JobContext)` để lấy `jobs`.
   - Thêm `<h3>Apply Now</h3>` và render `JobApplicationForm` bên dưới thông tin công việc.

Kết quả mong đợi:

- `/admin/add`: Thêm công việc mới (ví dụ: "Designer - Creative Ltd"), sau đó thấy công việc mới ở `/jobs` và `/admin/jobs`.
- `/jobs/1`: Hiển thị chi tiết công việc và form ứng tuyển. Sau khi điền tên/email và submit, thấy alert và chuyển về `/jobs`.
- `/admin`: Hiển thị điều hướng tới "Manage Jobs" và "Add Job", nhấp vào hiển thị nội dung tương ứng.

Phần 4: Điều hướng và xử lý lỗi nâng cao

Mục tiêu: Hoàn thiện ứng dụng với điều hướng nâng cao và trang lỗi chi tiết.

Hướng dẫn:

1. Trong `App.jsx`:
   - Thêm route `/careers` với `<Navigate>` để chuyển hướng về `/jobs`.
   - Thêm route với component `NotFound` để xử lý lỗi 404.
2. Tạo `NotFound.jsx` trong `screens/NotFound/`:
   - Hiển thị tiêu đề "404 - Page Not Found".
   - Dùng `useLocation` để lấy đường dẫn hiện tại (`pathname`) và hiển thị thông báo "Oops! The page '[pathname]' doesn't exist."
   - Thêm `<Link>` với nội dung "Return to Home" dẫn về `/`.
3. Cập nhật `Header.jsx`:
   - Đảm bảo tất cả liên kết dùng `<NavLink>` với style active đã được áp dụng từ Phần 3.

Kết quả mong đợi:

- Truy cập `/careers` tự động chuyển hướng về `/jobs`.
- Truy cập `/xyz` hiển thị:
  - "404 - Page Not Found"
  - "Oops! The page '/xyz' doesn't exist."
  - Nút "Return to Home" dẫn về `/`.

- Thanh điều hướng (`Header`) đánh dấu active khi nhấp vào từng trang.

Kết quả cuối cùng

Ứng dụng Job Board hoàn chỉnh với:

- Trang chủ, danh sách công việc, chi tiết công việc với form ứng tuyển.
- Trang Admin để xem danh sách công việc và thêm công việc mới qua nested routes.
- Điều hướng mượt mà với `<NavLink>` và xử lý lỗi bằng trang 404 chi tiết.

Thử thách bổ sung (Tùy chọn)

- Trong `JobManager.jsx`: Thêm nút "Xóa" cho mỗi công việc để xóa khỏi danh sách `jobs`.
- Hiển thị số lượng ứng tuyển (`applications`) cho mỗi công việc trong `JobManager` (ví dụ: "Frontend Developer - Tech Co (2 applications)").