Feedback Form:

import React, { useState } from "react";

// HOC: Authentication

const withAuth = (WrappedComponent) => {

  return (props) => {

    if (!props.isLoggedIn) {

      return <h2>Please login to access the Feedback Form</h2>;

    }

    return <WrappedComponent {...props} />;

  };

};

// HOC: Validation

const withValidation = (WrappedComponent) => {

  return (props) => {

    const validate = (feedback) => {

      if (!feedback.comment || !feedback.choice) {

        alert("⚠️ Please fill out all required fields");

        return false;

      }

      return true;

    };

    return <WrappedComponent {...props} validate={validate} />;

  };

};

// HOC: Logging

const withLogger = (WrappedComponent) => {

  return (props) => {

    const log = (message) => {

      console.log(`[Logger]: ${message}`);

    };

    return <WrappedComponent {...props} log={log} />;

  };

};

// FeedbackForm UI

const FeedbackForm = ({ user, validate, log }) => {

  const [feedback, setFeedback] = useState({ comment: "", choice: "" });

  const handleChange = (e) => {

    setFeedback({ ...feedback, [e.target.name]: e.target.value });

  };

  const handleSubmit = (e) => {

    e.preventDefault();

    log("Submit clicked");

    if (!validate(feedback)) {

      log("Validation failed");

      return;

    }

    log(`${user.name} submitted feedback: ${JSON.stringify(feedback)}`);

    alert("✅ Feedback submitted successfully!");

  };

  return (

    <form

      onSubmit={handleSubmit}

      style={{

        padding: "20px",

        maxWidth: "400px",

        margin: "20px auto",

        border: "1px solid #ccc",

        borderRadius: "10px",

      }}

    >

      <h2>Employee Feedback Form</h2>

      <p>Employee: {user.name}</p>

      <label>

        Comments:

        <input

          type="text"

          name="comment"

          value={feedback.comment}

          onChange={handleChange}

          style={{ width: "100%", marginBottom: "10px" }}

        />

      </label>

      <label>

        MileStone2 Completed?

        <select

          name="choice"

          value={feedback.choice}

          onChange={handleChange}

          style={{ width: "100%", marginBottom: "10px" }}

        >

          <option value="">--Select--</option>

          <option value="Yes">Yes</option>

          <option value="No">No</option>

        </select>

      </label>

      <button type="submit">Submit</button>

    </form>

  );

};

// Compose HOCs (order matters)

const EnhancedFeedbackForm = withAuth(

  withValidation(withLogger(FeedbackForm))

);

// Root component

const FeedbackHOCApp = () => {

  const [user] = useState({

    isLoggedIn: true,

    name: "Ravi",

  });

  return (

    <div>

      <EnhancedFeedbackForm isLoggedIn={user.isLoggedIn} user={user} />

    </div>

  );

};

export default FeedbackHOCApp; import React, { useState, useEffect, useCallback } from 'react';

const FeedbackForm = () => {

// State for form data and validation errors

const [formData, setFormData] = useState({

name: '',

email: '',

feedbackMessage: '',

rating: '',

});

const [errors, setErrors] = useState({});

const [isSubmitted, setIsSubmitted] = useState(false);

// A single function to handle changes for all input fields

const handleChange = (e) => {

const { name, value } = e.target;

setFormData({ ...formData, [name]: value });

};

// Main validation logic wrapped in useCallback

const validateForm = useCallback(() => {

let newErrors = {};

// Required Field Validation

if (!formData.name.trim()) {

newErrors.name = 'Name is required';

}

if (!formData.email.trim()) {

newErrors.email = 'Email is required';

}

if (!formData.feedbackMessage.trim()) {

newErrors.feedbackMessage = 'Feedback message is required';

}

if (!formData.rating) {

newErrors.rating = 'Rating is required';

}

// Format and Pattern Validation

if (formData.email && !/^[^\s@]+@[^\s@]+\.[^\s@]+$/.test(formData.email)) {

newErrors.email = 'Invalid email format';

}

if (formData.name && !/^[a-zA-Z\s]+$/.test(formData.name)) {

newErrors.name = 'Name should only contain alphabets and spaces';

}

// Length Validation

if (formData.feedbackMessage && (formData.feedbackMessage.length < 20 || formData.feedbackMessage.length > 250)) {

newErrors.feedbackMessage = 'Message must be between 20 and 250 characters';

}

setErrors(newErrors);

return Object.keys(newErrors).length === 0;

}, [formData]); // <-- Dependency array for useCallback

// Real-time validation using useEffect

useEffect(() => {

if (isSubmitted) {

validateForm();

}

}, [isSubmitted, validateForm]); // <-- Added validateForm to the dependency array

// Form submission handler

const handleSubmit = (e) => {

e.preventDefault();

setIsSubmitted(true);

const isValid = validateForm();

if (isValid) {

console.log('Form data to be submitted:', formData);

alert('Form submitted successfully!');

// Reset form fields after successful submission

setFormData({

name: '',

email: '',

feedbackMessage: '',

rating: '',

});

setErrors({});

setIsSubmitted(false);

} else {

console.log('Form has errors. Please correct them.');

}

};

return (

<div style={{ maxWidth: '600px', margin: 'auto', padding: '20px', border: '1px solid #ccc', borderRadius: '8px', boxShadow: '0 0 10px rgba(0,0,0,0.1)' }}>

<h2 style={{ textAlign: 'center' }}>LMS Feedback Form</h2>

<form onSubmit={handleSubmit}>

<div style={{ marginBottom: '15px' }}>

<label style={{ display: 'block', marginBottom: '5px' }}>Name:</label>

<input

type="text"

name="name"

value={formData.name}

onChange={handleChange}

style={{ width: '100%', padding: '8px', boxSizing: 'border-box' }}

/>

{errors.name && <p style={{ color: 'red', fontSize: '12px', marginTop: '5px' }}>{errors.name}</p>}

</div>

<div style={{ marginBottom: '15px' }}>

<label style={{ display: 'block', marginBottom: '5px' }}>Email:</label>

<input

type="email"

name="email"

value={formData.email}

onChange={handleChange}

style={{ width: '100%', padding: '8px', boxSizing: 'border-box' }}

/>

{errors.email && <p style={{ color: 'red', fontSize: '12px', marginTop: '5px' }}>{errors.email}</p>}

</div>

<div style={{ marginBottom: '15px' }}>

<label style={{ display: 'block', marginBottom: '5px' }}>Rating:</label>

<select

name="rating"

value={formData.rating}

onChange={handleChange}

style={{ width: '100%', padding: '8px', boxSizing: 'border-box' }}

>

<option value="">Select a rating</option>

<option value="1">1 (Poor)</option>

<option value="2">2</option>

<option value="3">3 (Average)</option>

<option value="4">4</option>

<option value="5">5 (Excellent)</option>

</select>

{errors.rating && <p style={{ color: 'red', fontSize: '12px', marginTop: '5px' }}>{errors.rating}</p>}

</div>

<div style={{ marginBottom: '15px' }}>

<label style={{ display: 'block', marginBottom: '5px' }}>Feedback Message:</label>

<textarea

name="feedbackMessage"

value={formData.feedbackMessage}

onChange={handleChange}

rows="5"

style={{ width: '100%', padding: '8px', boxSizing: 'border-box' }}

/>

{errors.feedbackMessage && <p style={{ color: 'red', fontSize: '12px', marginTop: '5px' }}>{errors.feedbackMessage}</p>}

</div>

<button

type="submit"

style={{

width: '100%',

padding: '10px',

backgroundColor: '#007bff',

color: 'white',

border: 'none',

borderRadius: '5px',

cursor: 'pointer',

fontSize: '16px',

marginTop: '10px'

}}

>

Submit Feedback

</button>

</form>

</div>

);

};

export default FeedbackForm;

BookList.jsx:

import React, { useState, useEffect } from 'react';

const BookList = () => {

  const [books, setBooks] = useState([]);

  const [filteredBooks, setFilteredBooks] = useState([]);

  const [languages, setLanguages] = useState([]);

  const [search, setSearch] = useState('');

  const [language, setLanguage] = useState('all');

  const [sort, setSort] = useState('none');

  // Fetch books

  useEffect(() => {

    fetch('https://gutendex.com/books')

      .then(res => res.json())

      .then(data => {

        setBooks(data.results);

        setFilteredBooks(data.results);

        // Extract available languages

        const langs = new Set();

        data.results.forEach(b => {

          b.languages.forEach(l => langs.add(l));

        });

        setLanguages([...langs]);

      });

  }, []);

  // Apply filters

  useEffect(() => {

    let result = [...books];

    if (language !== "all") {

      result = result.filter(b => b.languages.includes(language));

    }

    if (search.trim() !== "") {

      result = result.filter(

        b =>

          b.title.toLowerCase().includes(search.toLowerCase()) ||

          b.authors.some(a => a.name.toLowerCase().includes(search.toLowerCase()))

      );

    }

    if (sort === "low-high") {

      result.sort((a, b) => a.download\_count - b.download\_count);

    } else if (sort === "high-low") {

      result.sort((a, b) => b.download\_count - a.download\_count);

    }

    setFilteredBooks(result);

  }, [search, language, sort, books]);

  return (

    <div className='container my-4'>

      <h1 className='text-center mb-4'> Public Domain Book Explorer</h1>

      {/\* Filters \*/}

      <div className='row mb-4'>

        <div className='col-md-4'>

          <input

            type='text'

            className='form-control'

            placeholder="Search by Title or Author"

            value={search}

            onChange={(e) => setSearch(e.target.value)}

          />

        </div>

        <div className='col-md-4'>

          <select

            className="form-select"

            value={language}

            onChange={(e) => setLanguage(e.target.value)}

          >

            <option value="all">All Languages</option>

            {languages.map((lang, i) => (

              <option key={i} value={lang}>

                {lang.toUpperCase()}

              </option>

            ))}

          </select>

        </div>

        <div className='col-md-4'>

          <select

            className='form-select'

            value={sort}

            onChange={(e) => setSort(e.target.value)}

          >

            <option value="none">Sort by Popularity</option>

            <option value="low-high">Least Popular</option>

            <option value="high-low">Most Popular</option>

          </select>

        </div>

      </div>

      {/\* Book Cards \*/}

      <div className="row">

        {filteredBooks.map((book) => (

          <div key={book.id} className="col-md-4 mb-4">

            <div className='card h-100 shadow-sm'>

              {book.formats["image/jpeg"] && (

                <img

                  src={book.formats["image/jpeg"]}

                  className='card-img-top p-3'

                  alt={book.title}

                  style={{ height: "250px", objectFit: "contain" }}

                />

              )}

              <div className='card-body d-flex flex-column'>

                <h5 className="card-title">{book.title}</h5>

                <p className='text-muted small'>

                  {book.authors.map(a => a.name).join(", ") || "Unknown Author"}

                </p>

                <div className='mt-auto'>

                  <p className='fw-bold text-success'>

                    Downloads: {book.download\_count}

                  </p>

                  {book.formats["text/html"] && (

                    <a

                      href={book.formats["text/html"]}

                      target="\_blank"

                      rel="noopener noreferrer"

                      className="btn btn-primary w-100"

                    >

                      Read Online

                    </a>

                  )}

                </div>

              </div>

            </div>

          </div>

        ))}

        {filteredBooks.length === 0 && (

          <p className='text-center text-muted'>No books found.</p>

        )}

      </div>

    </div>

  );

};

export default BookList;