

DATA-236 Sec 12 - Distributed Systems for Data Engineering
HOMEWORK 5
Nandhakumar Apparsamy
018190003

GitHub - <https://github.com/Nandha951/DATA-236-HW4-Book-Store-CRUD>

Integrate the REST API's created in HW4 in your frontend using redux.

- I. Redux and Axios (2 points)axis
- Use Axios to make the API calls.

✓ HW 4



✓ hw4-template

> node_modules

> public

✓ src

✓ components

> Create

> Delete

> Home

> Update

✓ services

JS api.js

✓ store

JS bookSlice.js

JS store.js

App.css

JS App.js

index.css

JS index.js

JS setupTests.js

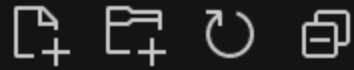
🔍 .gitignore

{ } package-lock.json

{ } package.json

📘 README.md

✓ HW 4



> hw4-template

✓ mysql-nodejs

> node_modules

✓ src

✓ config

JS database.js

✓ controllers

JS book.controller.js

JS user.controller.js

> models

> routes

JS app.js

JS user.js

⚙ .env

{ } package-lock.json

{ } package.json

📁 .gitignore

CreateBook.jsx

JS api.js

X

UpdateBook.jsx

DeleteBook.jsx

hw4-template > src > services > JS api.js

```
1  import axios from 'axios';
2
3  const api = axios.create({
4    baseURL: 'http://localhost:3000/api'
5  });
6
7  // Request interceptor for error handling
8  api.interceptors.request.use(
9    config => {
10     return config;
11   },
12   error => {
13     return Promise.reject(error);
14   }
15 );
16
17 // Response interceptor for error handling
18 api.interceptors.response.use(
19   response => response,
20   error => {
21     const message = error.response?.data?.message || 'An error occurred';
22     return Promise.reject(message);
23   }
24 );
25
26 // Get all books
27 export const getBooks = async () => {
28   const response = await api.get('/books');
29   return response.data;
30 };
31
32 // Get book by ID
33 export const getBookById = async (id) => {
34   const response = await api.get(`/books/${id}`);
35   return response.data;
36 };
37
38 // Create new book
39 export const createBook = async (bookData) => {
40   const response = await api.post('/books', bookData);
41   return response.data;
42 };
43
44 // Update existing book
45 export const updateBook = async (id, bookData) => {
46   const response = await api.put(`/books/${id}`, bookData);
47   return response.data;
48 };
49
50 // Delete book
51 export const deleteBook = async (id) => {
52   const response = await api.delete(`/books/${id}`);
53   return response.data;
54 };
55
56 export default api;
```

- Use redux to manage the state of the application. Make sure to make necessary constants, actions, reducers, and store and include screenshots of code as well as output.

NOTE: I have used Reduxjs Toolkit

```
JS index.js  X  UpdateBook.jsx  DeleteBook.jsx
hw4-template > src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import { Provider } from 'react-redux';
4  import store from './store/store';
5  import App from './App';
6  import './index.css';
7
8  const root = ReactDOM.createRoot(document.getElementById('root'));
9  root.render(
10   <React.StrictMode>
11     <Provider store={store}>
12       <App />
13     </Provider>
14   </React.StrictMode>
15 );
```

```
JS store.js  X  UpdateBook.jsx  DeleteBook.jsx
hw4-template > src > store > JS store.js
1  import { configureStore } from '@reduxjs/toolkit';
2  import bookReducer from './bookSlice';
3
4  const store = configureStore({
5    reducer: {
6      books: bookReducer
7    },
8    middleware: (getDefaultMiddleware) =>
9      getDefaultMiddleware({
10        serializableCheck: false
11      }),
12    devTools: process.env.NODE_ENV !== 'production'
13  });
14
15  export default store;
```

JS bookSlice.js ×

UpdateBook.jsx

DeleteBook.jsx

hw4-template > src > store > JS bookSlice.js

```
1  import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'
2  import * as api from '../services/api';
3
4  // Async Thunks
5  export const fetchBooks = createAsyncThunk(
6    'books/fetchBooks',
7    async () => {
8      return await api.getBooks();
9    }
10 );
11
12 export const createBook = createAsyncThunk(
13   'books/createBook',
14   async (bookData) => {
15     return await api.createBook(bookData);
16   }
17 );
18
19 export const updateBook = createAsyncThunk(
20   'books/updateBook',
21   async ({ id, bookData }) => {
22     return await api.updateBook(id, bookData);
23   }
24 );
25
26 export const deleteBook = createAsyncThunk(
27   'books/deleteBook',
28   async (id) => {
29     await api.deleteBook(id);
30     return id;
31   }
32 );
33
34 const bookSlice = createSlice({
35   name: 'books',
36   initialState: {
37     books: [],
38     loading: false,
39     error: null
40   },
41   reducers: {
42     clearError: (state) => {
43       state.error = null;
44     }
45   },
46   extraReducers: (builder) => {
47     builder
48       // Fetch Books
49       .addCase(fetchBooks.pending, (state) => {
50         state.loading = true;
51         state.error = null;
52       })
53       .addCase(fetchBooks.fulfilled, (state, action) => {
54         state.loading = false;
55         state.books = action.payload;
56       })
57       .addCase(fetchBooks.rejected, (state, action) => {
58         state.loading = false;
59         state.error = action.error.message;
60       })
61   }
62 });
```

```
JS bookSlice.js x UpdateBook.jsx DeleteBook.jsx
hw4-template > src > store > JS bookSlice.js
61 // Create Book
62 .addCase(createBook.pending, (state) => {
63   state.loading = true;
64   state.error = null;
65 })
66 .addCase(createBook.fulfilled, (state, action) => {
67   state.loading = false;
68   state.books.push(action.payload);
69 })
70 .addCase(createBook.rejected, (state, action) => {
71   state.loading = false;
72   state.error = action.error.message;
73 })
74 // Update Book
75 .addCase(updateBook.pending, (state) => {
76   state.loading = true;
77   state.error = null;
78 })
79 .addCase(updateBook.fulfilled, (state, action) => {
80   state.loading = false;
81   state.books = state.books.map(book =>
82     book.id === action.payload.id ? action.payload : book
83   );
84 })
85 .addCase(updateBook.rejected, (state, action) => {
86   state.loading = false;
87   state.error = action.error.message;
88 })
89 // Delete Book
90 .addCase(deleteBook.pending, (state) => {
91   state.loading = true;
92   state.error = null;
93 })
94 .addCase(deleteBook.fulfilled, (state, action) => {
95   state.loading = false;
96   state.books = state.books.filter(book => book.id !== action.payload);
97 })
98 .addCase(deleteBook.rejected, (state, action) => {
99   state.loading = false;
100   state.error = action.error.message;
101 });
102 }
103 });
104
105 export const { clearError } = bookSlice.actions;
106 export default bookSlice.reducer;
```

II. Home Screen (2 points)

- Integrate the Get API to fetch all books and display them on the home screen.



localhost:3001

Book Management System

Add New Book

To Kill a Mockingbird

Author: Harper Lee

ISBN: 978-0446310789

Update

Delete

Atomic Habits

Author: James Clear

ISBN: 1234567

Update

Delete

Home.jsx ×

UpdateBook.jsx

DeleteBook.jsx

hw4-template > src > components > Home > Home.jsx

```
1  import React, { useEffect } from 'react';
2  import { Link } from 'react-router-dom';
3  import { useDispatch, useSelector } from 'react-redux';
4  import { fetchBooks, deleteBook } from '../store/bookSlice';
5  import './Home.css';
6
7  const Home = () => {
8    const dispatch = useDispatch();
9    const { books, loading, error } = useSelector(state => state.books);
10
11    useEffect(() => {
12      dispatch(fetchBooks());
13    }, [dispatch]);
14
15    const handleDelete = async (id) => {
16      try {
17        await dispatch(deleteBook(id)).unwrap();
18      } catch (err) {
19        console.error('Failed to delete book:', err);
20      }
21    };
22
23    if (loading) return <div>Loading...</div>;
24    if (error) return <div>Error: {error}</div>;
25
26    return (
27      <div className="home-container">
28        <h1>Book Management System</h1>
29        <Link to="/create" className="add-button">Add New Book</Link>
30
31        <div className="books-grid">
32          {books.map(book => (
33            <div key={book.id} className="book-card">
34              <h3>{book.title}</h3>
35              <p>Author: {book.author}</p>
36              <p>ISBN: {book.isbn}</p>
37              <div className="book-actions">
38                <Link to={` /update/${book.id}`} className="edit-button">Update</Link>
39                <button
40                  onClick={() => handleDelete(book.id)}
41                  className="delete-button"
42                >
43                  Delete
44                </button>
45              </div>
46            </div>
47          ))}
48        </div>
49      </div>
50    );
51  };
52
53  export default Home;
```

III. Create Screen (2 points)

- Integrate the Post API to create a new book and display the updated results on the home Screen.

localhost:3001/create

Add New Book

Book Title:

Author Name:

ISBN:

Published Year:

Add Book

localhost:3001

Book Management System

Add New Book

To Kill a Mockingbird Author: Harper Lee ISBN: 978-0446310789 <div>UpdateDelete</div>	Atomic Habits Author: James Clear ISBN: 1234567 <div>UpdateDelete</div>	Test Author: Test ISBN: 12345654321 <div>UpdateDelete</div>
---	---	---

Redux >> 2

Actions Settings

filter...

@@@INIT 4:53:10.81

books/createBook/pending +00:25.21

books/createBook/fulfilled +00:00.02

books/fetchBooks/pending +00:00.00

books/fetchBooks/pending +00:00.00

books/fetchBooks/fulfilled +00:00.01

books/fetchBooks/fulfilled +00:00.00

Diff

Action State Diff Trace Test

Tree Raw

books (pin)

books (pin)

2 (pin)

publishedYear (pin): 1990 => 1990

updatedAt (pin): 2025-03-02T00:00:00.000Z => 2025-03-02T00:00:00.000Z

createdAt (pin): 2025-03-02T00:00:00.000Z => 2025-03-02T00:00:00.000Z

loading (pin): true => false

```
⚙️ CreateBook.jsx × ⚙️ UpdateBook.jsx ⚙️ DeleteBook.jsx
hw4-template > src > components > Create > ⚙️ CreateBook.jsx
1  import React, { useState } from 'react';
2  import { useNavigate } from 'react-router-dom';
3  import { useDispatch, useSelector } from 'react-redux';
4  import { createBook } from '../../store/bookSlice';
5  import './CreateBook.css';
6
7  const CreateBook = () => {
8    const [formData, setFormData] = useState({
9      title: '',
10     author: '',
11     isbn: '',
12     publishedYear: ''
13   });
14   const dispatch = useDispatch();
15   const navigate = useNavigate();
16   const { loading, error } = useSelector(state => state.books);
17
18   const handleChange = (e) => {
19     const { name, value } = e.target;
20     setFormData(prev => ({
21       ...prev,
22       [name]: value
23     }));
24   };
25
26   const handleSubmit = async (e) => {
27     e.preventDefault();
28     try {
29       await dispatch(createBook(formData)).unwrap();
30       navigate('/');
31     } catch (err) {
32       console.error('Failed to create book:', err);
33     }
34   };
35
36   return (
37     <div className="create-book-container">
38       <h2>Add New Book</h2>
39       {error && <div className="error-message">{error}</div>}
40       <form onSubmit={handleSubmit}>
41         <div className="form-group">
42           <label htmlFor="title">Book Title:</label>
43           <input
44             type="text"
```

IV. Update Screen (2 points)

- Integrate the Put API to update an existing book and display the updated results on the home screen.

Update Book

Book Title:

Test - Updated

Author Name:

Test - Updated

ISBN:

12345654321

Published Year:

1990

Update Book

Cancel

Book Management System

Add New Book

To Kill a Mockingbird

Author: Harper Lee

ISBN: 978-0446310789

Update

Delete

Atomic Habits

Author: James Clear

ISBN: 1234567

Update

Delete

Test - Updated

Author: Test - Updated

ISBN: 12345654321

Update

Delete

Redux >> 2

Actions Settings

Reset Revert Sweep Commit

823167... ▾

🔄

filter...

books/fetchBooks/pending +00:00.00

books/fetchBooks/fulfilled +00:00.01

books/fetchBooks/fulfilled +00:00.00

books/updateBook/pending +01:45.39

books/updateBook/fulfilled +00:00.03

books/fetchBooks/pending +00:00.00

books/fetchBooks/pending +00:00.00

books/fetchBooks/fulfilled +00:00.01

books/fetchBooks/fulfilled +00:00.00

Diff

Action State Diff Trace Test

Tree Raw

```
{
  books: {
    books: [
      2: {
        author: "Test" => "Test - Updated",
        title: "Test" => "Test - Updated",
        updatedAt: "2025-03-02T00:53:36.000Z"
          => "2025-03-02T00:55:21.486Z"
      },
    ],
    loading: true => false
  },
}
```

books/fetchBooks/fulfilled (12)

▶

◀ ▶

Live 1x 2x

CreateBook.jsx

UpdateBook.jsx X

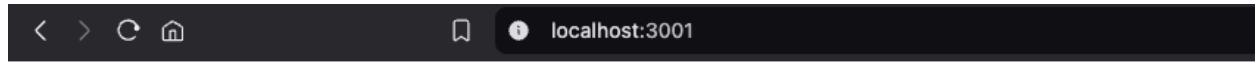
DeleteBook.jsx

hw4-template > src > components > Update > UpdateBook.jsx

```
1  import React, { useState, useEffect } from 'react';
2  import { useNavigate, useParams } from 'react-router-dom';
3  import { useDispatch, useSelector } from 'react-redux';
4  import { updateBook } from '../../../store/bookSlice';
5  import { getBookById } from '../../../services/api';
6  import './UpdateBook.css';
7
8  const UpdateBook = () => {
9    const [formData, setFormData] = useState({
10      title: '',
11      author: '',
12      isbn: '',
13      publishedYear: ''
14    });
15
16    const navigate = useNavigate();
17    const dispatch = useDispatch();
18    const { id } = useParams();
19    const { loading, error } = useSelector(state => state.books);
20    const [fetchError, setFetchError] = useState(null);
21
22    useEffect(() => {
23      const fetchBook = async () => {
24        try {
25          const book = await getBookById(id);
26          setFormData(book);
27        } catch (error) {
28          setFetchError(error.message || 'Failed to fetch book');
29        }
30      };
31
32      fetchBook();
33    }, [id]);
34
35    const handleChange = (e) => {
36      const { name, value } = e.target;
37      setFormData(prev => ({
38        ...prev,
39        [name]: value
40      }));
41    };
42
43    const handleSubmit = async (e) => {
44      e.preventDefault();
45      try {
46        await dispatch(updateBook({ id, bookData: formData })).unwrap();
47        navigate('/');
48      } catch (err) {
49        console.error('Failed to update book:', err);
50      }
51    };
52
53    if (loading) return <div>Loading...</div>;
54    if (fetchError) return <div>Error: {fetchError}</div>;
55
56    return (
57      <div className="update-book-container">
58        <h2>Update Book</h2>
59        {error && <div className="error-message">{error}</div>}
60        <form onSubmit={handleSubmit}>
61          <div className="form-group">
62            <label htmlFor="title">Book Title:</label>
```

V. Delete Screen (2 points)

- Integrate the Delete API to delete an existing book and display the updated results on the home screen.



Book Management System

Add New Book

To Kill a Mockingbird

Author: Harper Lee

ISBN: 978-0446310789

Update

Delete

Atomic Habits

Author: James Clear

ISBN: 1234567

Update

Delete

Redux >> 2

Actions Settings

Reset Revert Sweep Commit

823167... | v

filter...

books/fetchBooks/fulfilled +00:00.00

books/updateBook/pending +01:45.39

books/updateBook/fulfilled +00:00.03

books/fetchBooks/pending +00:00.00

books/fetchBooks/pending +00:00.00

books/fetchBooks/fulfilled +00:00.01

books/fetchBooks/fulfilled +00:00.00

books/deleteBook/pending +11:26.21

books/deleteBook/fulfilled +00:00.02

Diff

Action State Diff Trace Test

Tree Raw

```
{
  books: {
    books: [
      2: {
        "id": 28,
        "title": "Test - Updated",
        "author": "Test - Updated",
        "isbn": "12345654321",
        "publishedYear": 1990,
        "createdAt": "2025-03-02T00:53:36.000Z",
        "updatedAt": "2025-03-02T00:55:21.000Z"
      }
    ],
    loading: true => false
  }
}
```

books/fetchBooks/fulfilled (12)

Live 1x 2x

CreateBook.jsx

UpdateBook.jsx

DeleteBook.jsx X

hw4-template > src > components > Delete > DeleteBook.jsx

```
1 import React, { useState, useEffect } from 'react';
2 import { useNavigate, useParams } from 'react-router-dom';
3 import { useDispatch, useSelector } from 'react-redux';
4 import { deleteBook } from '../../store/bookSlice';
5 import { getBookById } from '../../services/api';
6 import './DeleteBook.css';
7
8 const DeleteBook = () => {
9   const [book, setBook] = useState(null);
10  const [error, setError] = useState(null);
11  const navigate = useNavigate();
12  const dispatch = useDispatch();
13  const { id } = useParams();
14  const { loading } = useSelector(state => state.books);
15
16  useEffect(() => {
17    const fetchBook = async () => {
18      try {
19        const bookData = await getBookById(id);
20        setBook(bookData);
21      } catch (error) {
22        setError('Failed to fetch book details');
23        console.error('Error:', error);
24      }
25    };
26
27    fetchBook();
28  }, [id]);
29
30  const handleDelete = async () => {
31    try {
32      await dispatch(deleteBook(id)).unwrap();
33      navigate('/');
34    } catch (error) {
35      setError(error.message);
36      console.error('Error:', error);
37    }
38  };
39
40  if (loading) return <div>Loading...</div>;
41  if (error) return <div className="error-message">{error}</div>;
42
43  return (
44    <div className="delete-book-container">
45      <h2>Delete Book</h2>
46      {book && (
47        <div className="book-details">
48          <h3>{book.title}</h3>
49          <p>Author: {book.author}</p>
50          <p>ISBN: {book.isbn}</p>
51          {book.publishedYear && <p>Published Year: {book.publishedYear}</p>}
52          <p className="warning-text">Are you sure you want to delete this book?</p>
53        </div>
54      )}
```