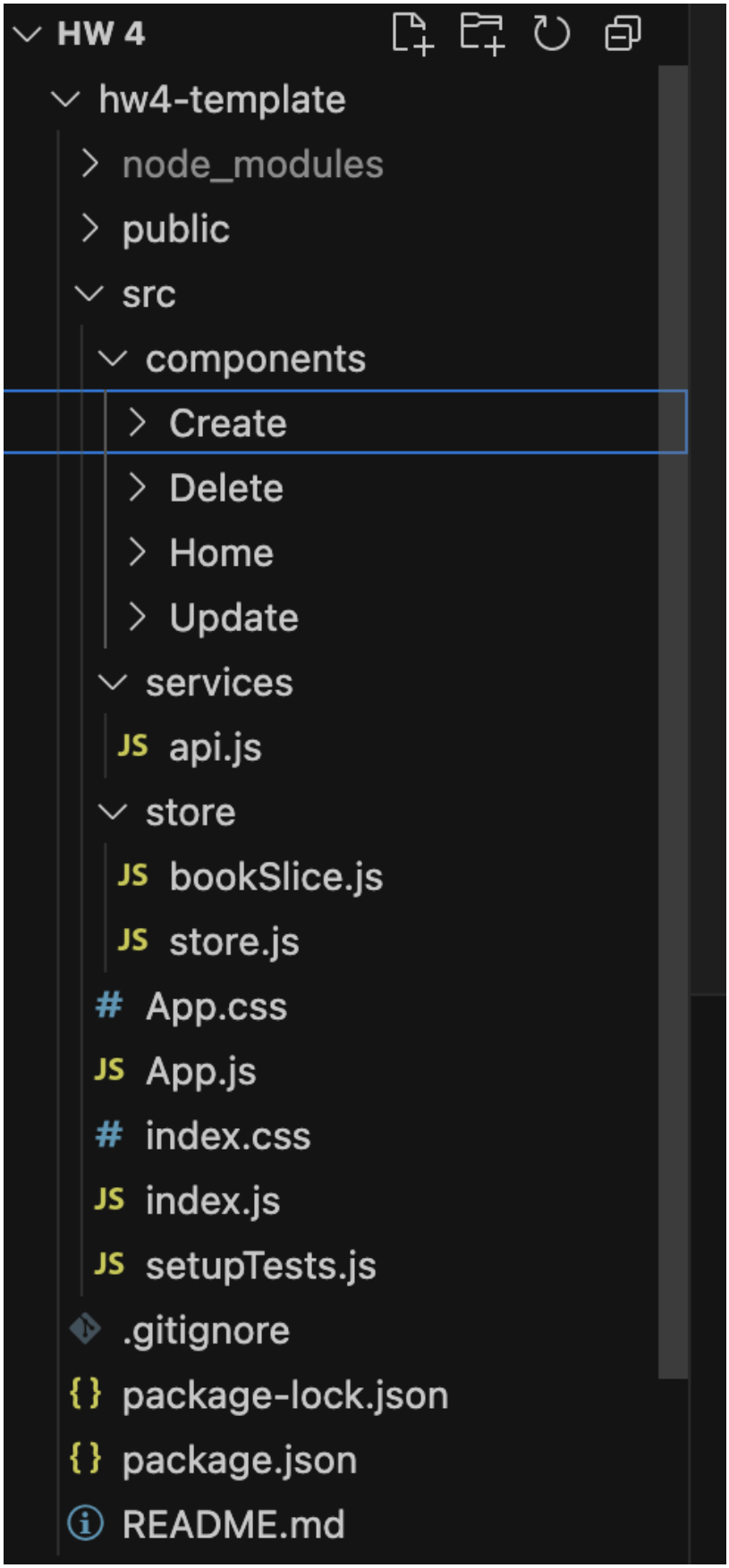
DATA-236 Sec 12 - Distributed Systems for Data Engineering
HOMEWORK 5

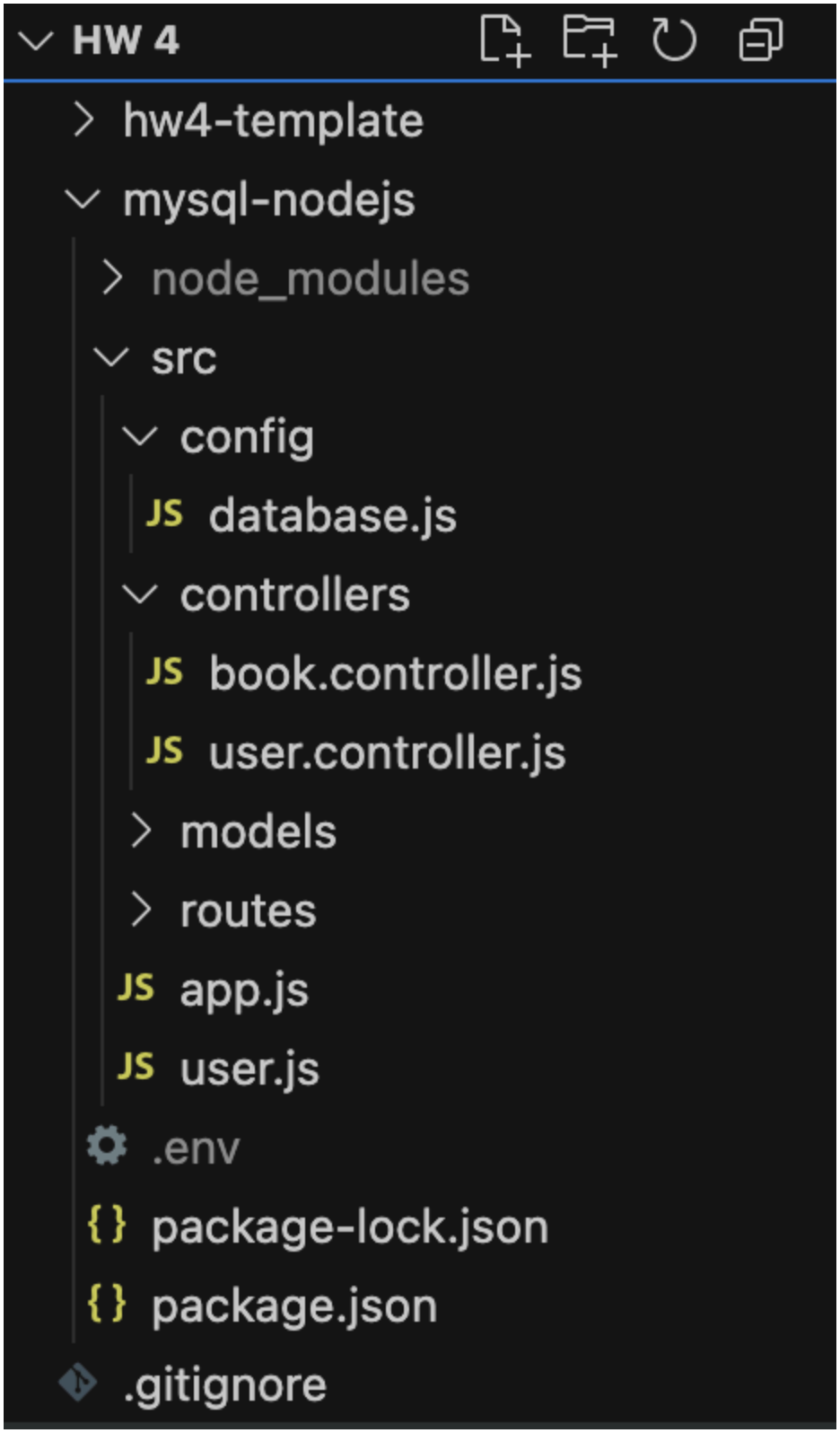
Nandhakumar Apparsamy
018190003

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
api.js
store
bookSlice.js
store.js
App.css
App.js
index.css
index.js
setupTests.js
.gitignore
package-lock.json
package.json
README.md


HW 4
hw4-template
mysql-nodejs
node_modules
src
config
database.js
controllers
book.controller.js
user.controller.js
models
routes
app.js
user.js
.env
package-lock.json
package.json
.gitignore

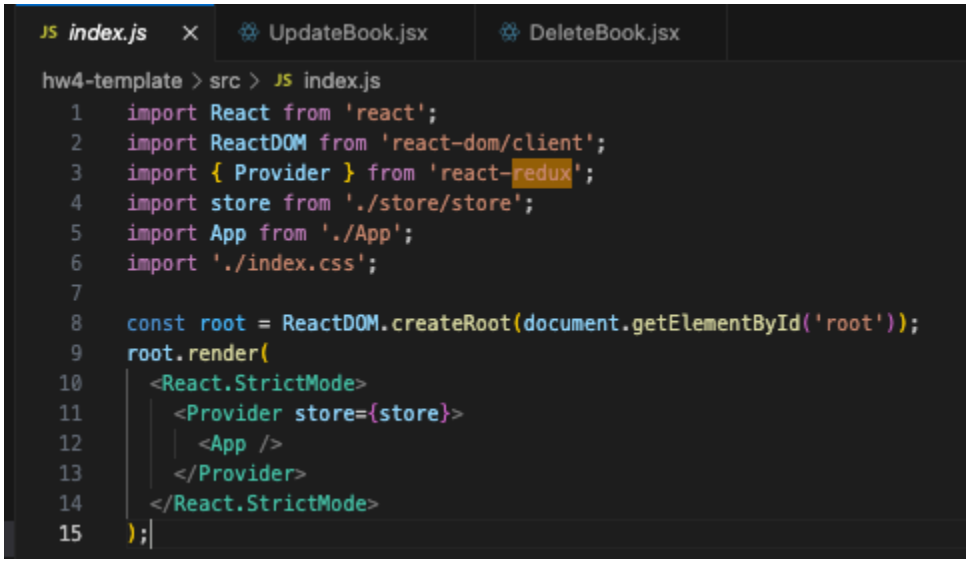
CreateBook.jsx | api.js | UpdateBook.jsx | DeleteBook.jsx

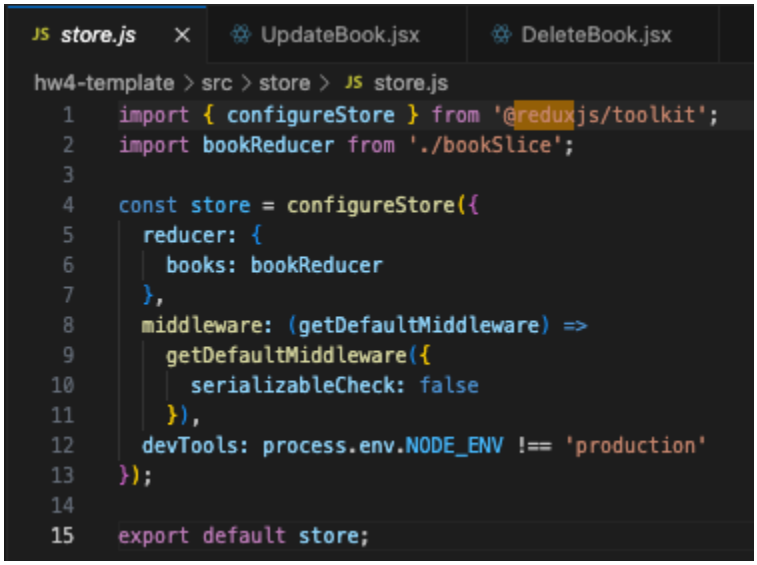
hw4-template > src > services > api.js

```
import axios from 'axios';

const api = axios.create({
  baseURL: 'http://localhost:3000/api'
});

// Request interceptor for error handling
api.interceptors.request.use(
  config => {
    return config;
  },
  error => {
    return Promise.reject(error);
  }
);

// Response interceptor for error handling
api.interceptors.response.use(
  response => response,
  error => {
    const message = error.response?.data?.message || 'An error occurred';
    return Promise.reject(message);
  }
);

// Get all books
export const getBooks = async () => {
  const response = await api.get('/books');
  return response.data;
};

// Get book by ID
export const getBookById = async (id) => {
  const response = await api.get(`/books/${id}`);
  return response.data;
};

// Create new book
export const createBook = async (bookData) => {
  const response = await api.post('/books', bookData);
  return response.data;
};

// Update existing book
export const updateBook = async (id, bookData) => {
  const response = await api.put(`/books/${id}`, bookData);
  return response.data;
};

// Delete book
export const deleteBook = async (id) => {
  const response = await api.delete(`/books/${id}`);
  return response.data;
};

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 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        })
```

II. Home Screen (2 points)

● Integrate the Get API to fetch all books and display them 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

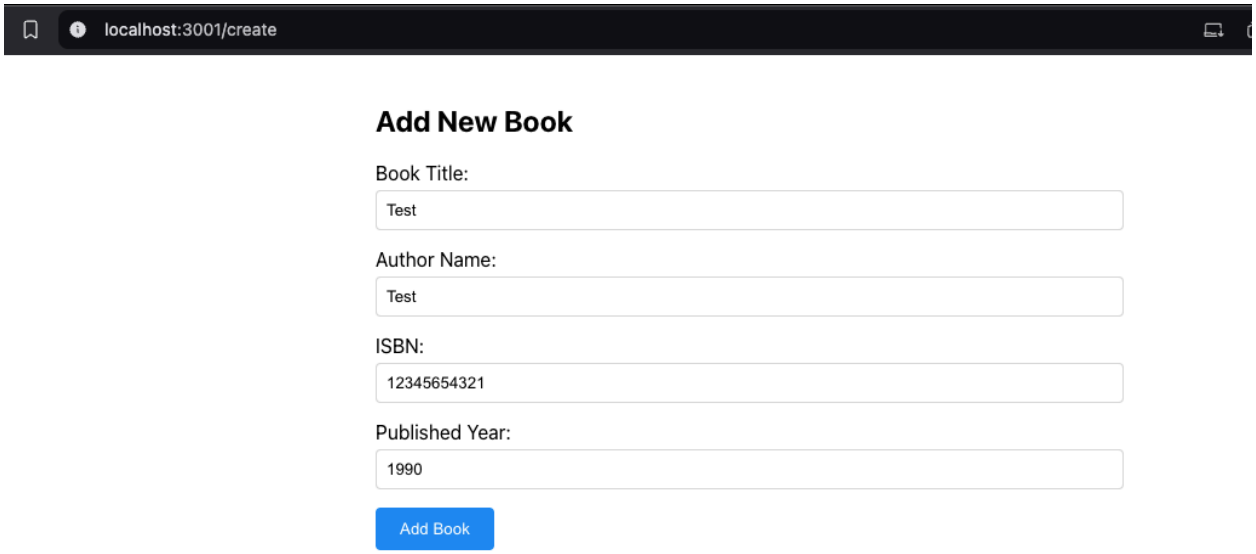
Home.jsx | UpdateBook.jsx | DeleteBook.jsx

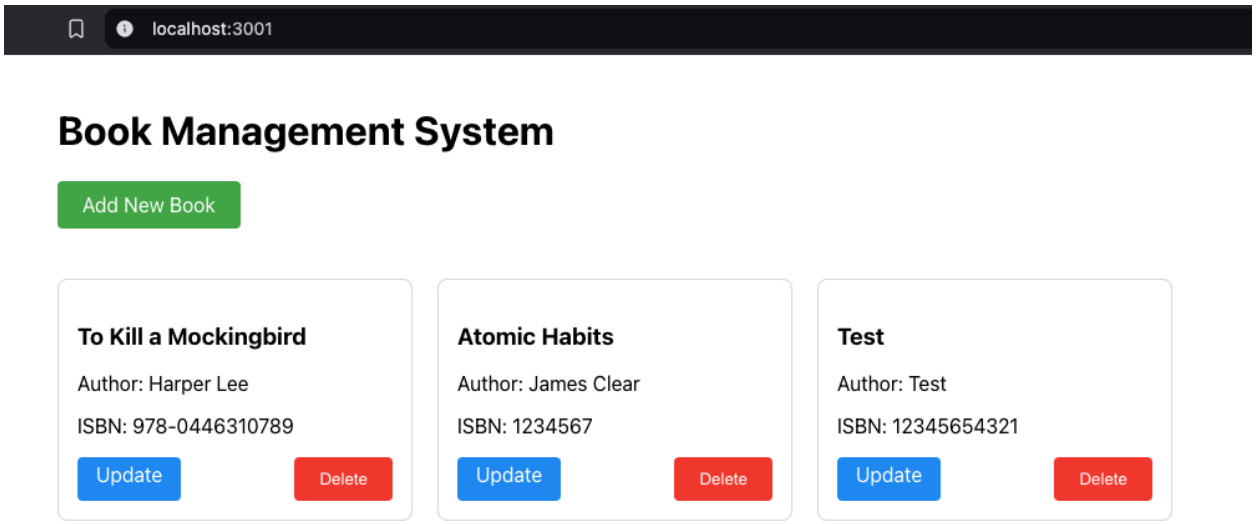
hw4-template > src > components > Home > Home.jsx

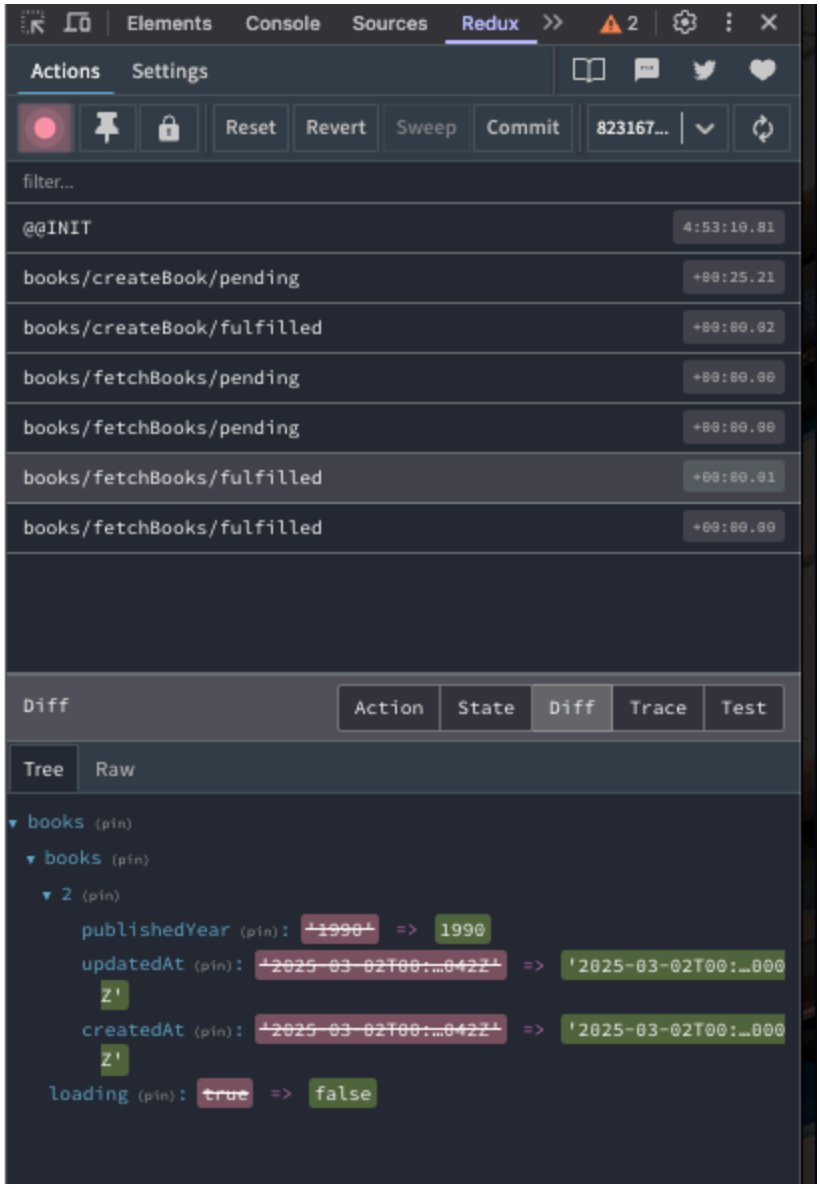
```jsx
import React, { useEffect } from 'react';
import { Link } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { fetchBooks, deleteBook } from '../../store/bookSlice';
import './Home.css';

const Home = () => {
  const dispatch = useDispatch();
  const { books, loading, error } = useSelector(state => state.books);

  useEffect(() => {
    dispatch(fetchBooks());
  }, [dispatch]);

  const handleDelete = async (id) => {
    try {
      await dispatch(deleteBook(id)).unwrap();
    } catch (err) {
      console.error('Failed to delete book:', err);
    }
  };

  if (loading) return <div>Loading...</div>;
  if (error) return <div>Error: {error}</div>;

  return (
    <div className="home-container">
      <h1>Book Management System</h1>
      <Link to="/create" className="add-button">Add New Book</Link>

      <div className="books-grid">
        {books.map(book => (
          <div key={book.id} className="book-card">
            <h3>{book.title}</h3>
            <p>Author: {book.author}</p>
            <p>ISBN: {book.isbn}</p>
            <div className="book-actions">
              <Link to={`/update/${book.id}`} className="edit-button">Update</Link>
              <button
                onClick={() => handleDelete(book.id)}
                className="delete-button"
              >
                Delete
              </button>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

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.

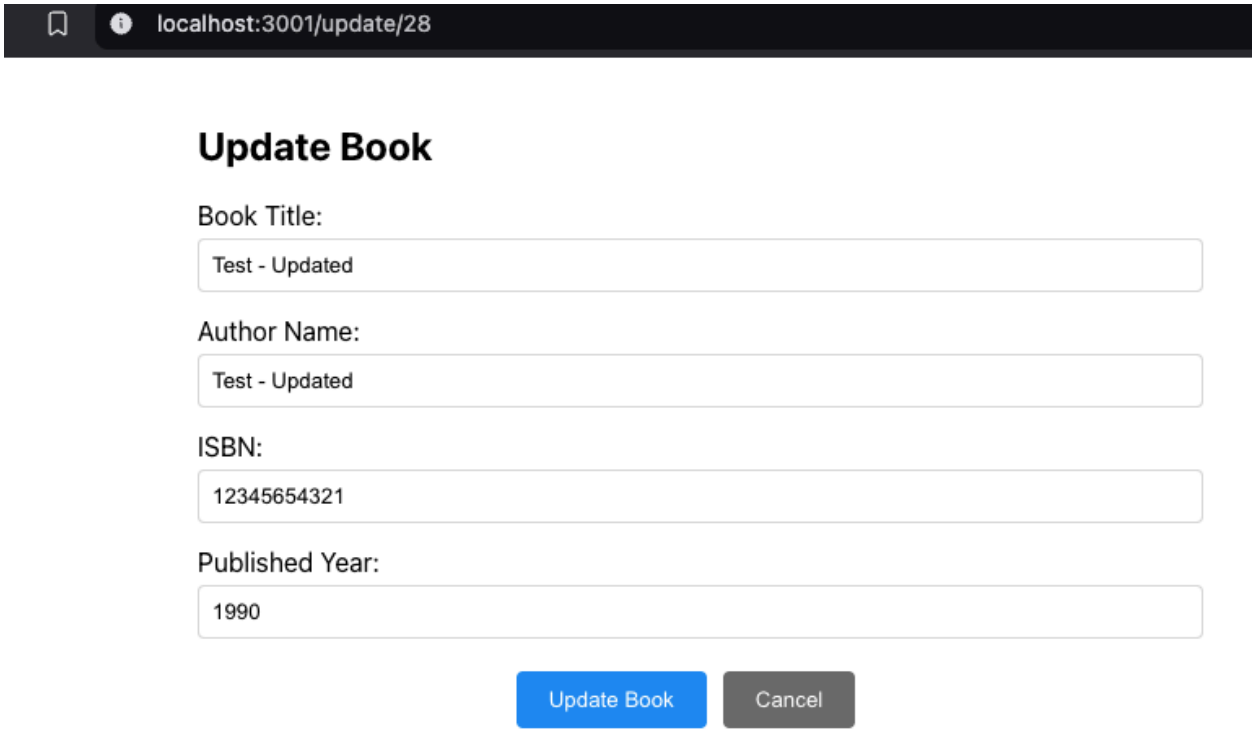


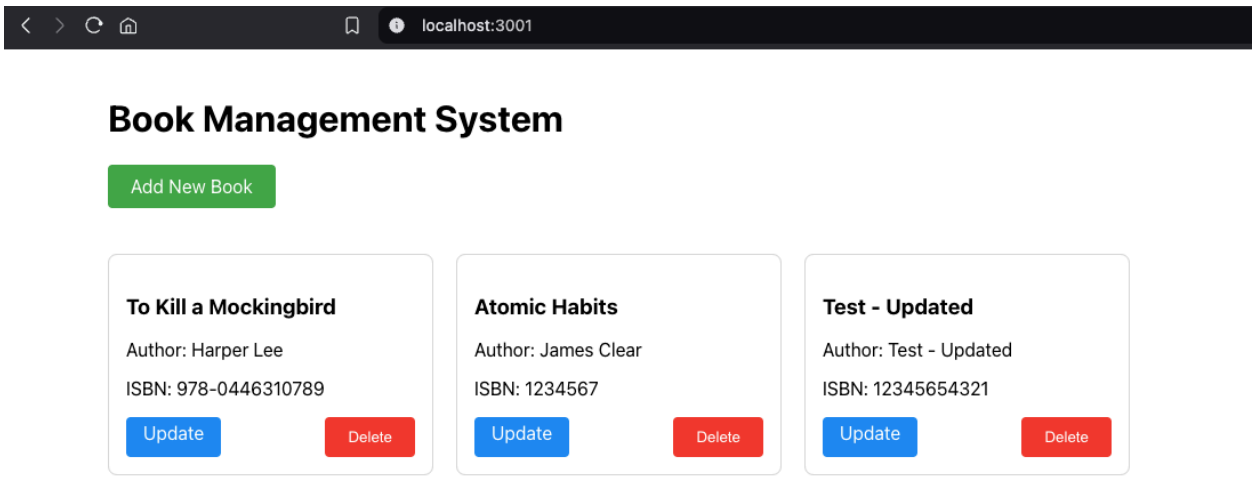


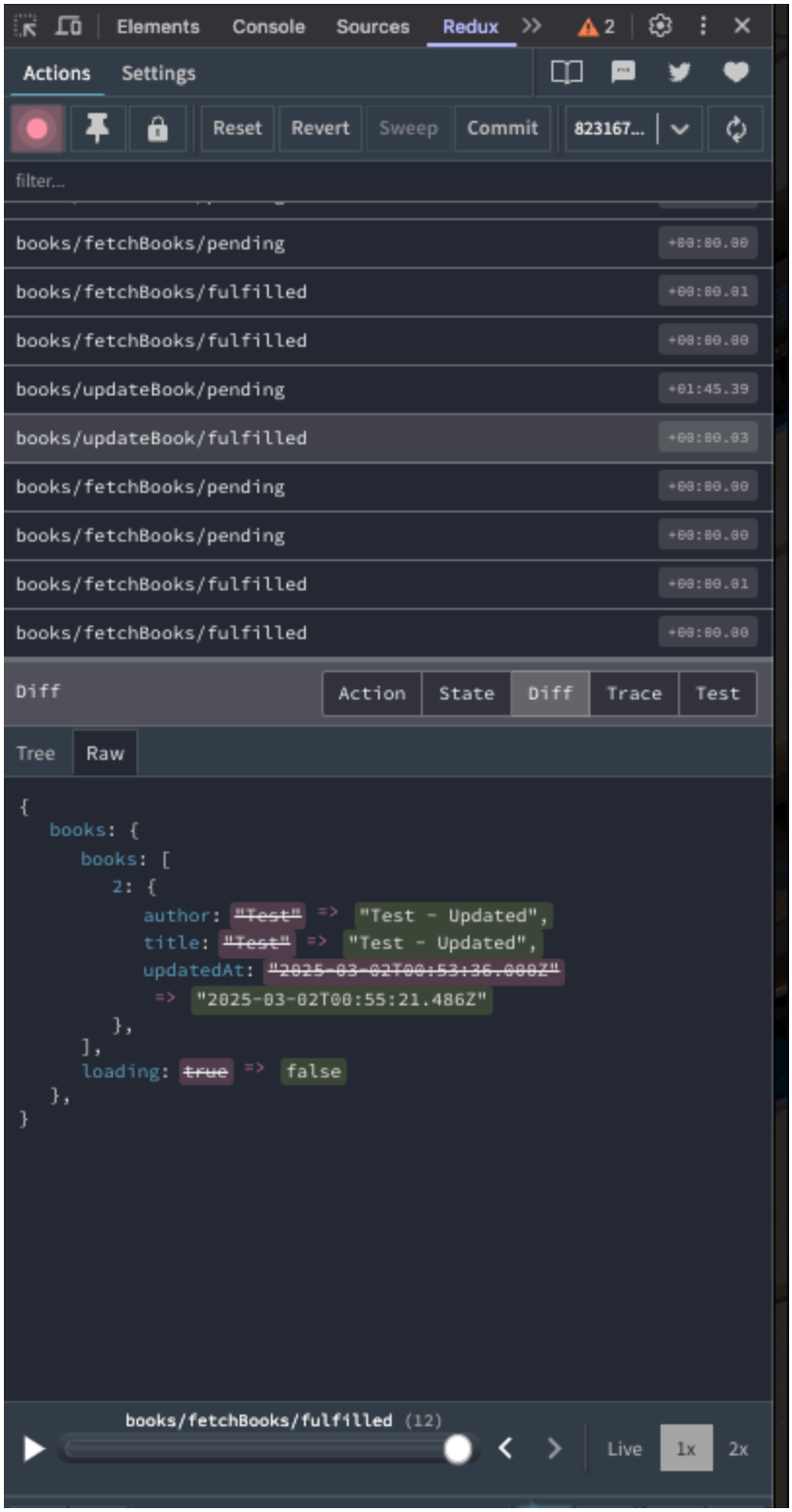

Elements
Console
Sources
Redux
2
Actions
Settings
Reset
Revert
Sweep
Commit
823167...
filter...
@@INIT
4:53:10.81
books/createBook/pending
+00:25.21
books/createBook/fulfilled
+00:00.02
books/fetchBooks/pending
+00:00.00
books/fetchBooks/pending
+00:00.00
books/fetchBooks/fulfilled
+00:00.01
books/fetchBooks/fulfilled
+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:…042Z' => '2025-03-02T00:…000Z'
createdAt (pin): '2025-03-02T00:…042Z' => '2025-03-02T00:…000Z'
loading (pin): true => false

IV. Update Screen (2 points)

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


localhost:3001/update/28
Update Book
Book Title:
Test - Updated
Author Name:
Test - Updated
ISBN:
12345654321
Published Year:
1990
Update Book
Cancel


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
Test - Updated
Author: Test - Updated
ISBN: 12345654321
Update
Delete


Elements
Console
Sources
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

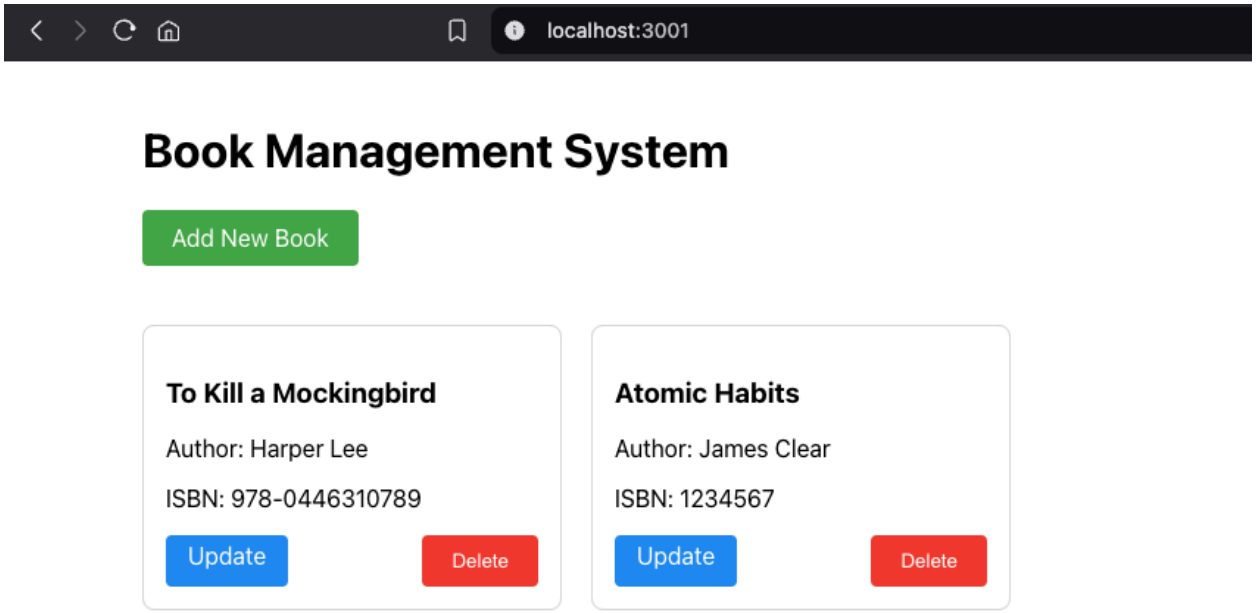
CreateBook.jsx | UpdateBook.jsx | DeleteBook.jsx

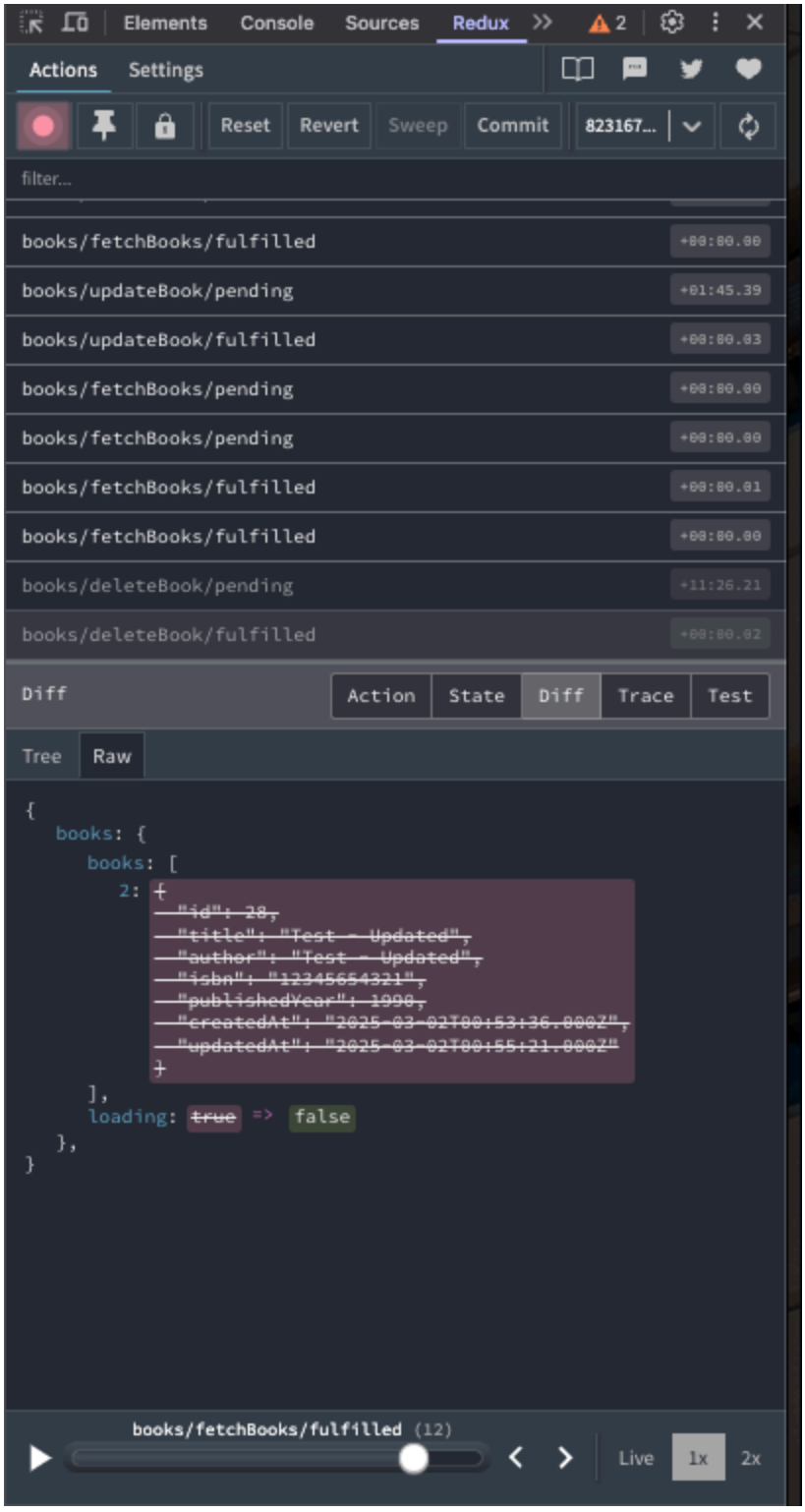
hw4-template > src > components > Update > UpdateBook.jsx

```jsx
import React, { useState, useEffect } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { updateBook } from '../../store/bookSlice';
import { getBookById } from '../../services/api';
import './UpdateBook.css';

const UpdateBook = () => {
  const [formData, setFormData] = useState({
    title: '',
    author: '',
    isbn: '',
    publishedYear: ''
  });

  const navigate = useNavigate();
  const dispatch = useDispatch();
  const { id } = useParams();
  const { loading, error } = useSelector(state => state.books);
  const [fetchError, setFetchError] = useState(null);

  useEffect(() => {
    const fetchBook = async () => {
      try {
        const book = await getBookById(id);
        setFormData(book);
      } catch (error) {
        setFetchError(error.message || 'Failed to fetch book');
      }
    };

    fetchBook();
  }, [id]);

  const handleChange = (e) => {
    const { name, value } = e.target;
    setFormData(prev => ({
      ...prev,
      [name]: value
    }));
  };

  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      await dispatch(updateBook({ id, bookData: formData })).unwrap();
      navigate('/');
    } catch (err) {
      console.error('Failed to update book:', err);
    }
  };

  if (loading) return <div>Loading...</div>;
  if (fetchError) return <div>Error: {fetchError}</div>;

  return (
    <div className="update-book-container">
      <h2>Update Book</h2>
      {error && <div className="error-message">{error}</div>}
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <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.




Elements
Console
Sources
Redux
2
Actions
Settings
Reset
Revert
Sweep
Commit
823167...
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": 1998,
"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

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

```jsx
import React, { useState, useEffect } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { deleteBook } from '../../store/bookSlice';
import { getBookById } from '../../services/api';
import './DeleteBook.css';

const DeleteBook = () => {
  const [book, setBook] = useState(null);
  const [error, setError] = useState(null);
  const navigate = useNavigate();
  const dispatch = useDispatch();
  const { id } = useParams();
  const { loading } = useSelector(state => state.books);

  useEffect(() => {
    const fetchBook = async () => {
      try {
        const bookData = await getBookById(id);
        setBook(bookData);
      } catch (error) {
        setError('Failed to fetch book details');
        console.error('Error:', error);
      }
    };

    fetchBook();
  }, [id]);

  const handleDelete = async () => {
    try {
      await dispatch(deleteBook(id)).unwrap();
      navigate('/');
    } catch (error) {
      setError(error.message);
      console.error('Error:', error);
    }
  };

  if (loading) return <div>Loading...</div>;
  if (error) return <div className="error-message">{error}</div>;

  return (
    <div className="delete-book-container">
      <h2>Delete Book</h2>
      {book && (
        <div className="book-details">
          <h3>{book.title}</h3>
          <p>Author: {book.author}</p>
          <p>ISBN: {book.isbn}</p>
          {book.publishedYear && <p>Published Year: {book.publishedYear}</p>}
          <p className="warning-text">Are you sure you want to delete this book?</p>
        </div>
      )}
```