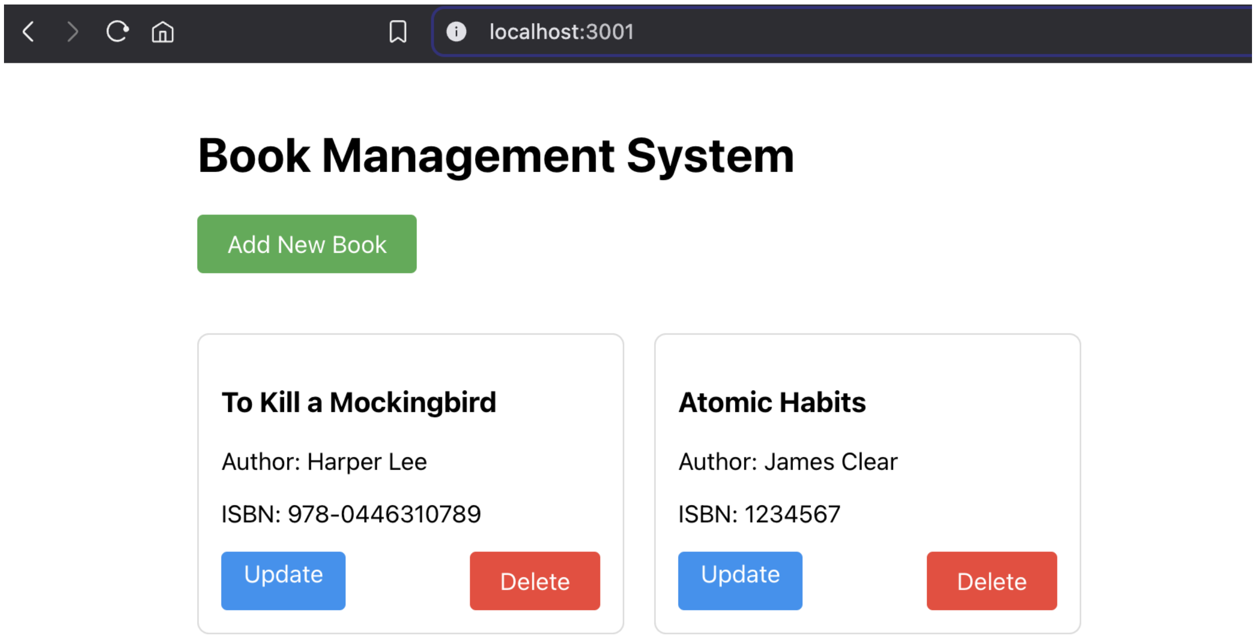
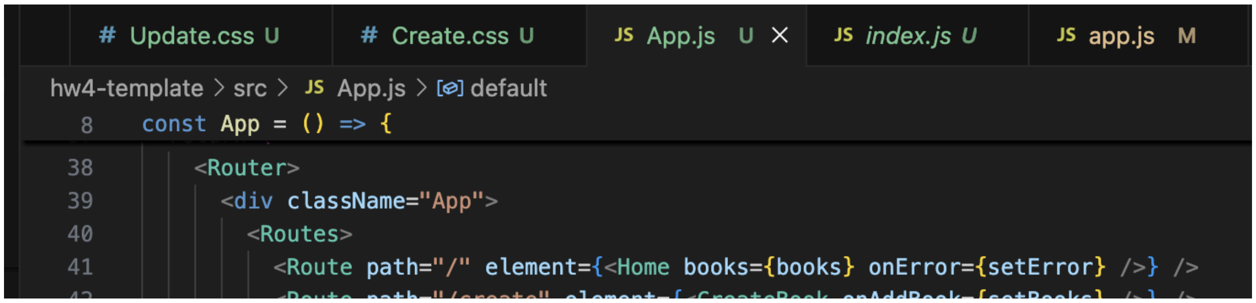
DATA-236 Sec 12 - Distributed Systems for Data Engineering
HOMEWORK 3
Nandhakumar Apparsamy
018190003

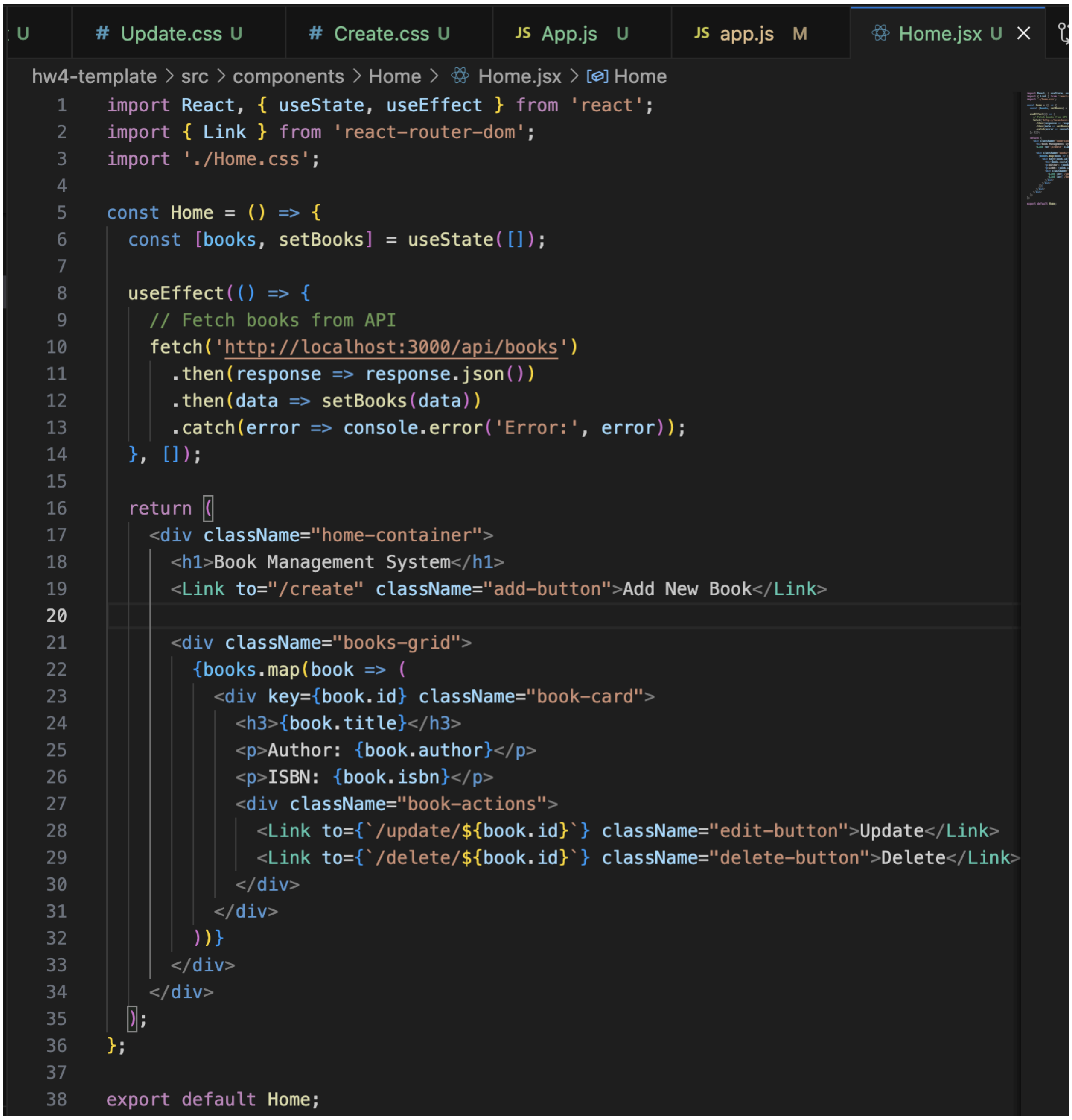
GitHub - https://github.com/Nandha951/DATA-236-HW4-Book-Store-CRUD

Q1. React (10 Points)
Create a Book Management App where users can add, update, and delete books.
I. Home Page **(1 Point)**
Write the necessary code to create the 'Home.jsx' component which will display all the books.
The Home component should be rendered when the user is on the root route '/'.





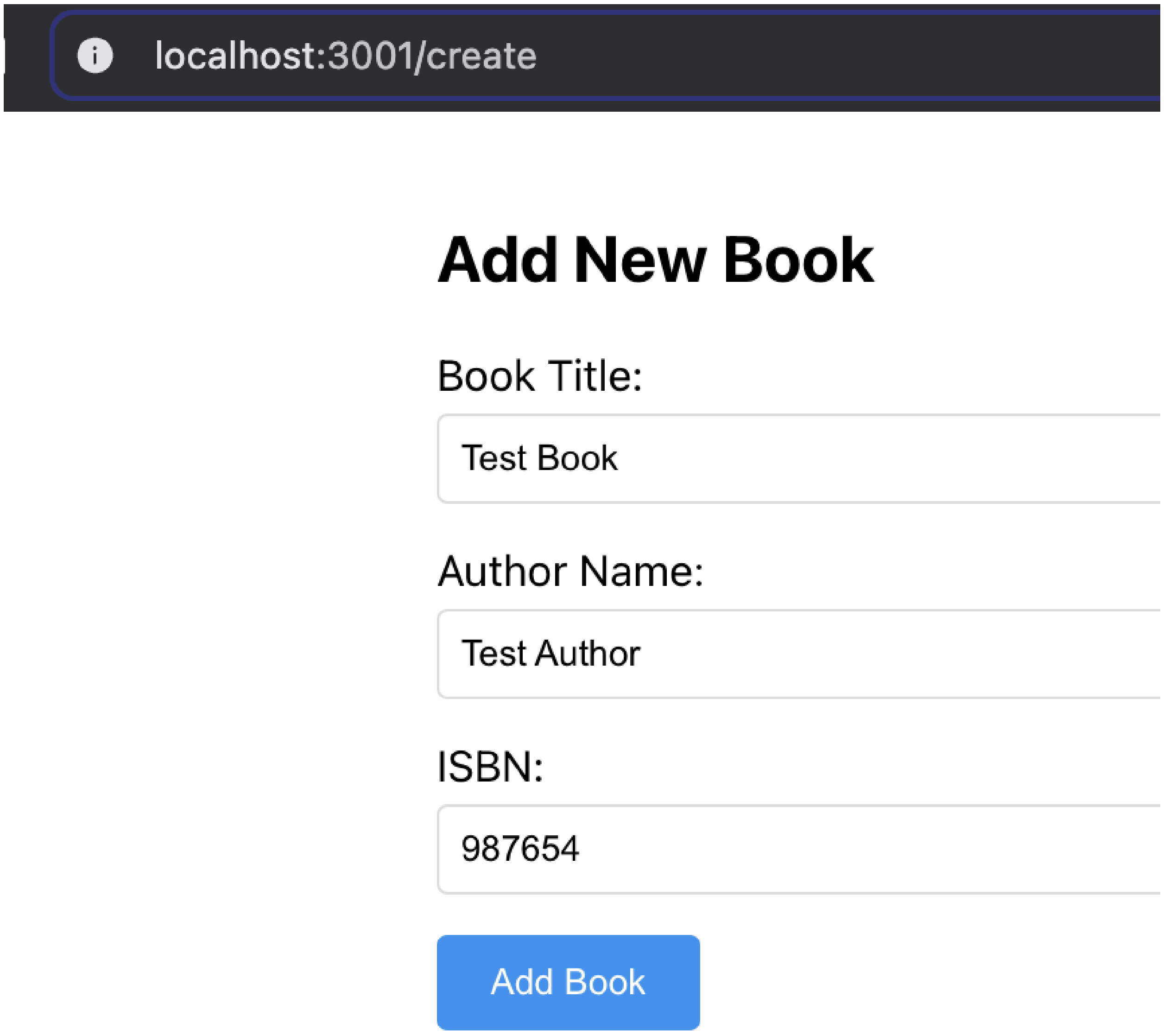
```
const App = () => {
      <Router>
        <div className="App">
          <Routes>
            <Route path="/" element={<Home books={books} onError={setError} />} />
```



```
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import './Home.css';

const Home = () => {
  const [books, setBooks] = useState([]);

  useEffect(() => {
    // Fetch books from API
    fetch('http://localhost:3000/api/books')
      .then(response => response.json())
      .then(data => setBooks(data))
      .catch(error => console.error('Error:', error));
  }, []);

  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/${book.id}`} className="edit-button">Update</Link>
              <Link to={`/delete/${book.id}`} className="delete-button">Delete</Link>
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export default Home;
```

II. Add a New Book **(2 Points)**
Write the necessary code to create the 'CreateBook.jsx' component. The component should be rendered when the user is on the '/create' route.
The component should accept props (similar to demo) to add the new book and have the following:

● An input field for entering the Book Title.
● An input field for entering the Author Name.
● A submit button labeled "Add Book". When the user clicks the Add book button:
● A new book should be added to the book list with an auto-incremented book ID.
● The user should be redirected to the home page to see the updated book list.

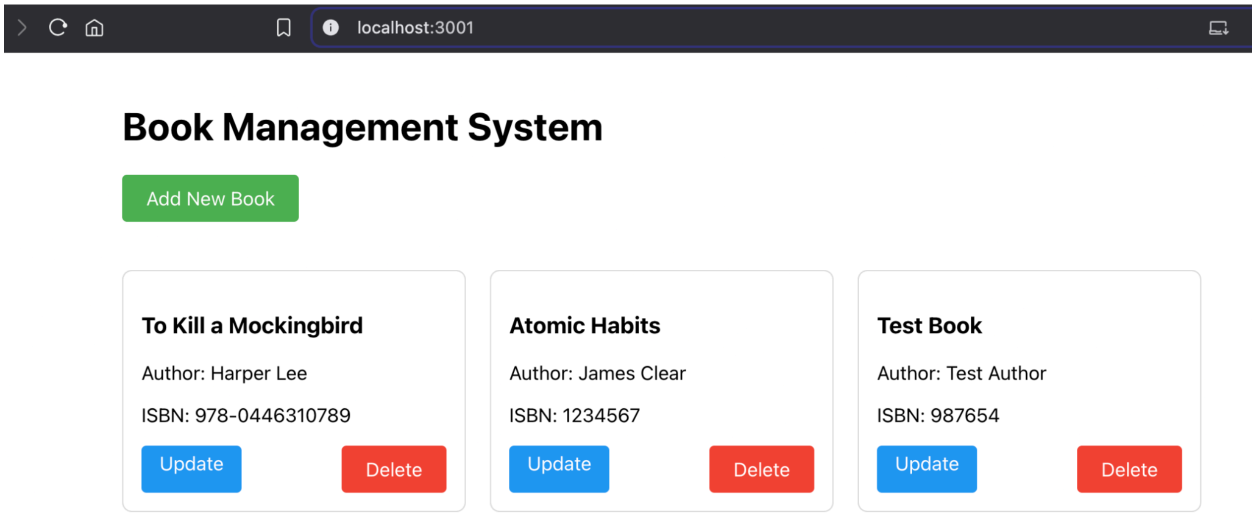

localhost:3001/create

Add New Book

Book Title:
Test Book

Author Name:
Test Author

ISBN:
987654

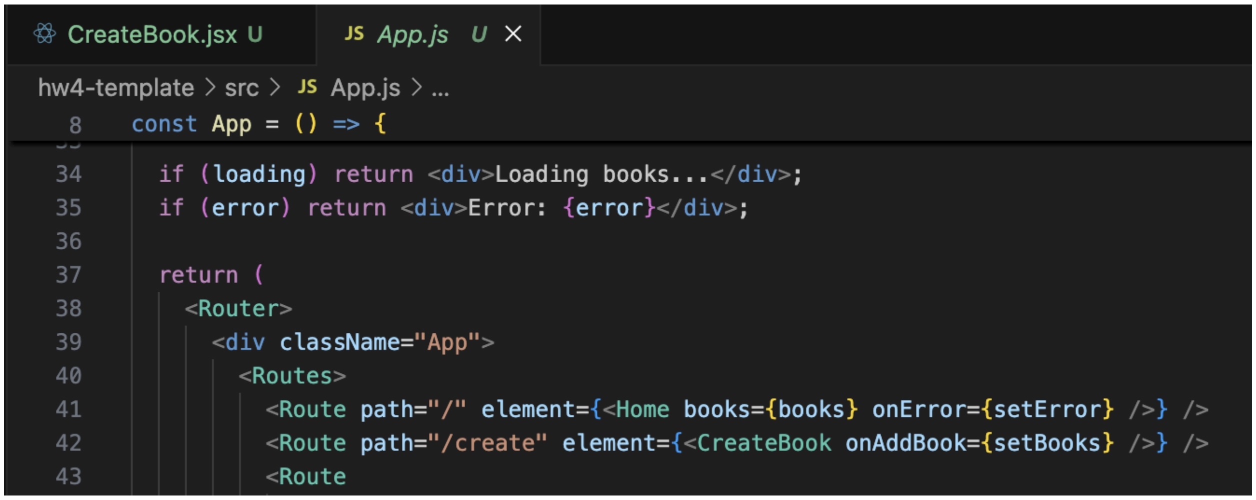

localhost:3001

Book Management System

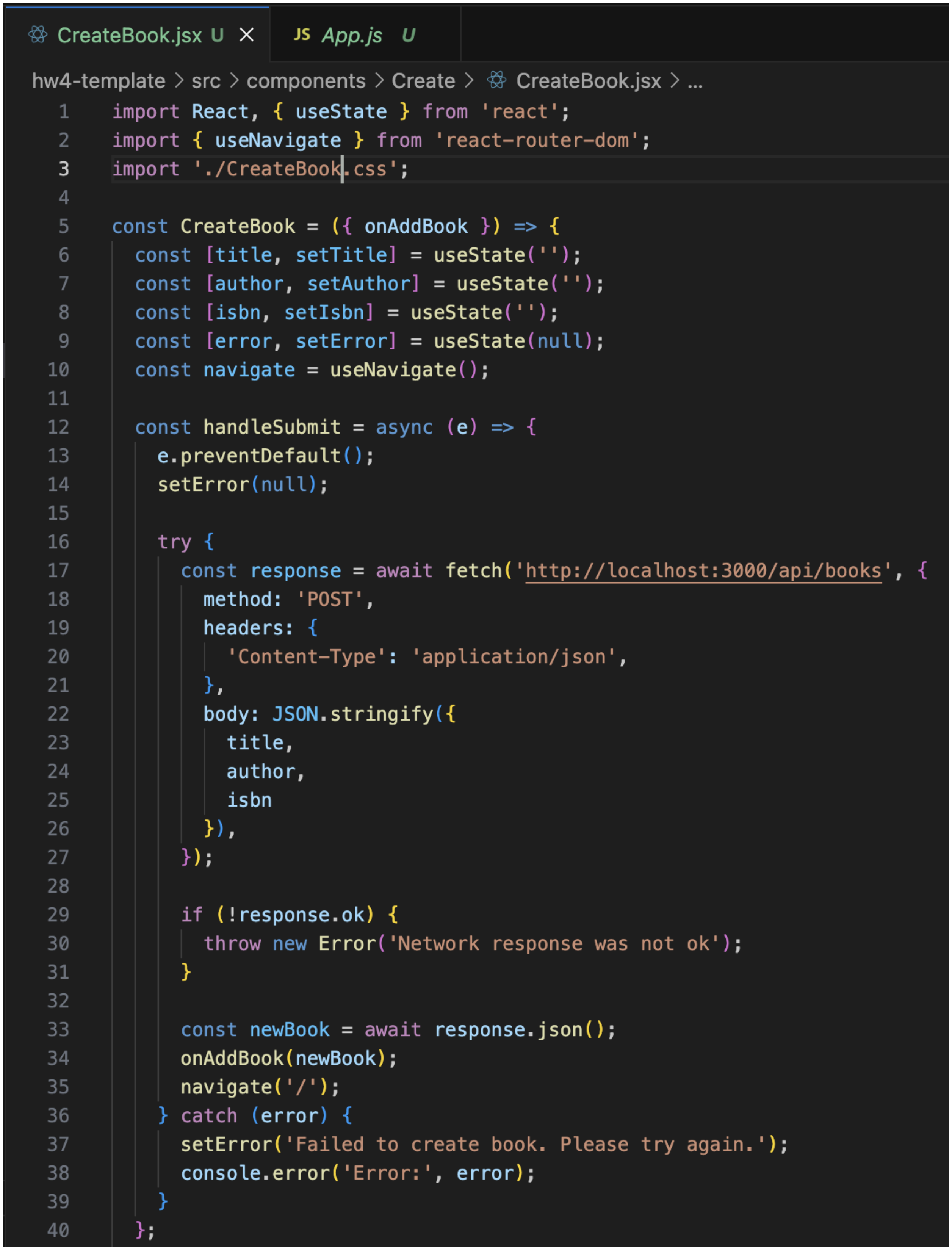
To Kill a Mockingbird
Author: Harper Lee
ISBN: 978-0446310789

Atomic Habits
Author: James Clear
ISBN: 1234567

Test Book
Author: Test Author
ISBN: 987654



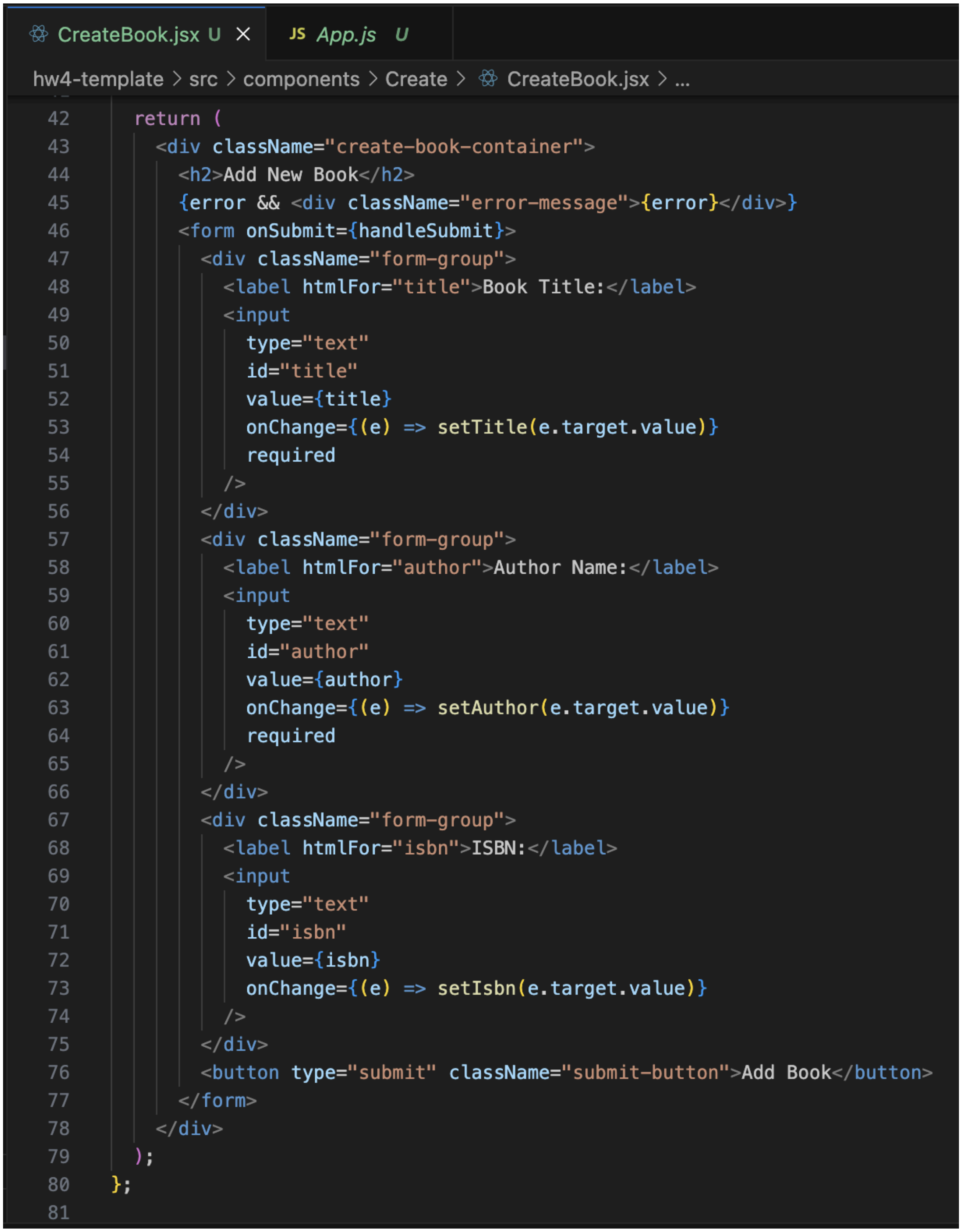
```
const App = () => {
  if (loading) return <div>Loading books...</div>;
  if (error) return <div>Error: {error}</div>;

  return (
    <Router>
      <div className="App">
        <Routes>
          <Route path="/" element={<Home books={books} onError={setError} />} />
          <Route path="/create" element={<CreateBook onAddBook={setBooks} />} />
          <Route
```



CreateBook.jsx U × JS App.js U

hw4-template > src > components > Create > CreateBook.jsx > ...

```jsx
import React, { useState } from 'react';
import { useNavigate } from 'react-router-dom';
import './CreateBook.css';

const CreateBook = ({ onAddBook }) => {
  const [title, setTitle] = useState('');
  const [author, setAuthor] = useState('');
  const [isbn, setIsbn] = useState('');
  const [error, setError] = useState(null);
  const navigate = useNavigate();

  const handleSubmit = async (e) => {
    e.preventDefault();
    setError(null);

    try {
      const response = await fetch('http://localhost:3000/api/books', {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({
          title,
          author,
          isbn
        }),
      });

      if (!response.ok) {
        throw new Error('Network response was not ok');
      }

      const newBook = await response.json();
      onAddBook(newBook);
      navigate('/');
    } catch (error) {
      setError('Failed to create book. Please try again.');
      console.error('Error:', error);
    }
  };
```

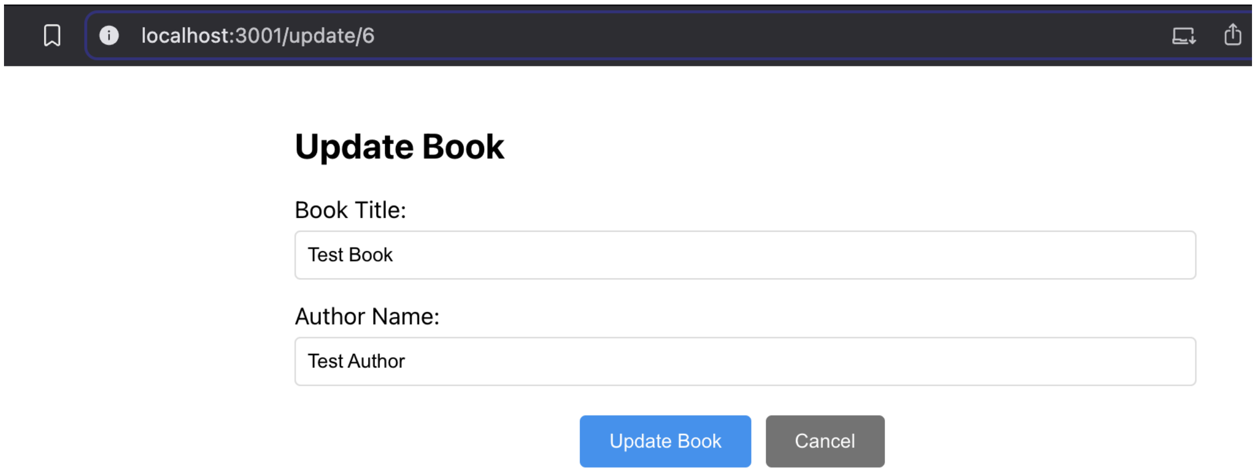

CreateBook.jsx U
App.js U
hw4-template > src > components > Create > CreateBook.jsx > ...

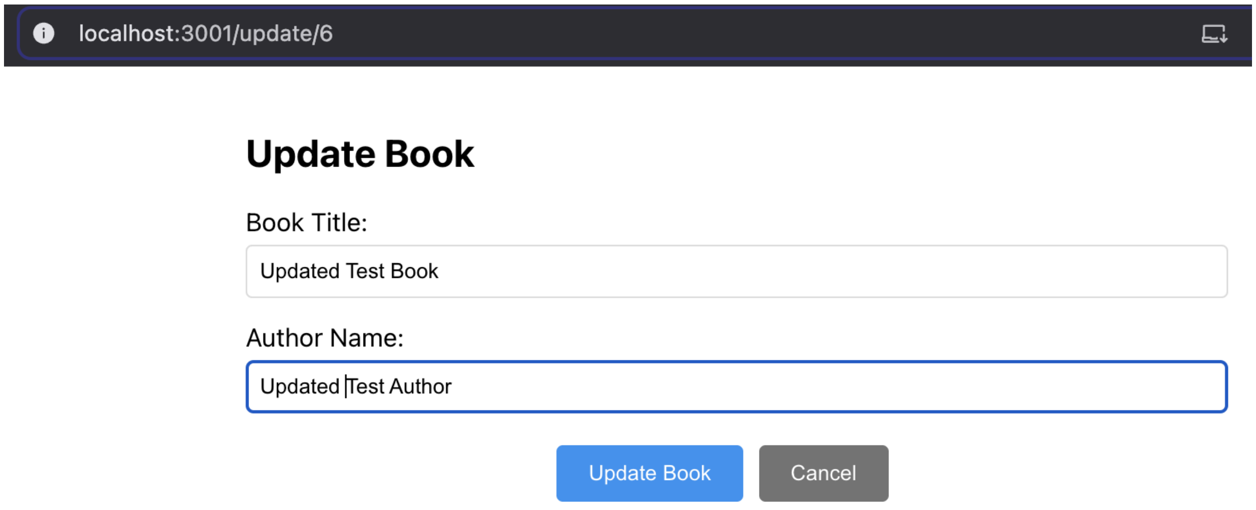
```jsx
  return (
    <div className="create-book-container">
      <h2>Add New Book</h2>
      {error && <div className="error-message">{error}</div>}
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="title">Book Title:</label>
          <input
            type="text"
            id="title"
            value={title}
            onChange={(e) => setTitle(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="author">Author Name:</label>
          <input
            type="text"
            id="author"
            value={author}
            onChange={(e) => setAuthor(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="isbn">ISBN:</label>
          <input
            type="text"
            id="isbn"
            value={isbn}
            onChange={(e) => setIsbn(e.target.value)}
          />
        </div>
        <button type="submit" className="submit-button">Add Book</button>
      </form>
    </div>
  );
};

```

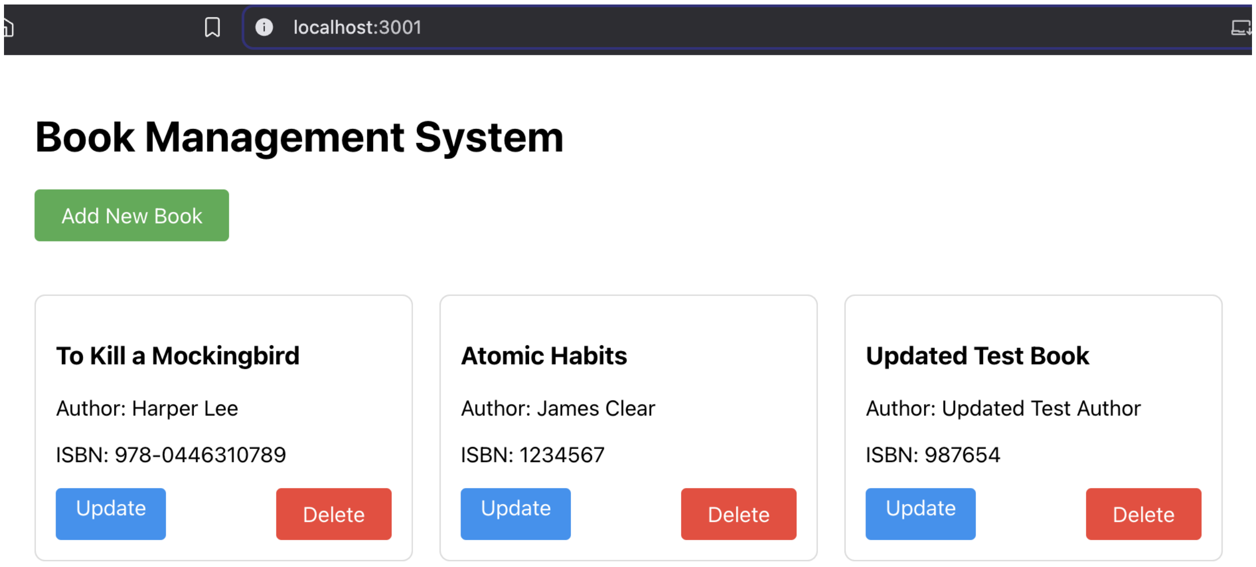
III. Update Book **(2 Points)**
Write the necessary code to create the 'UpdateBook.jsx' component. The component should be rendered when the user is on the '/update' route.
The component should accept props (similar to demo) to update the new book and have the following:

● An input field for Book Title.
● An input field for Author Name.
● A submit button labeled "Update Book".

When the user clicks the Update Book button the book within context should be updated and the user should be redirected to the home page showing updated book list.






localhost:3001

Book Management System

To Kill a Mockingbird

Author: Harper Lee

ISBN: 978-0446310789

Atomic Habits

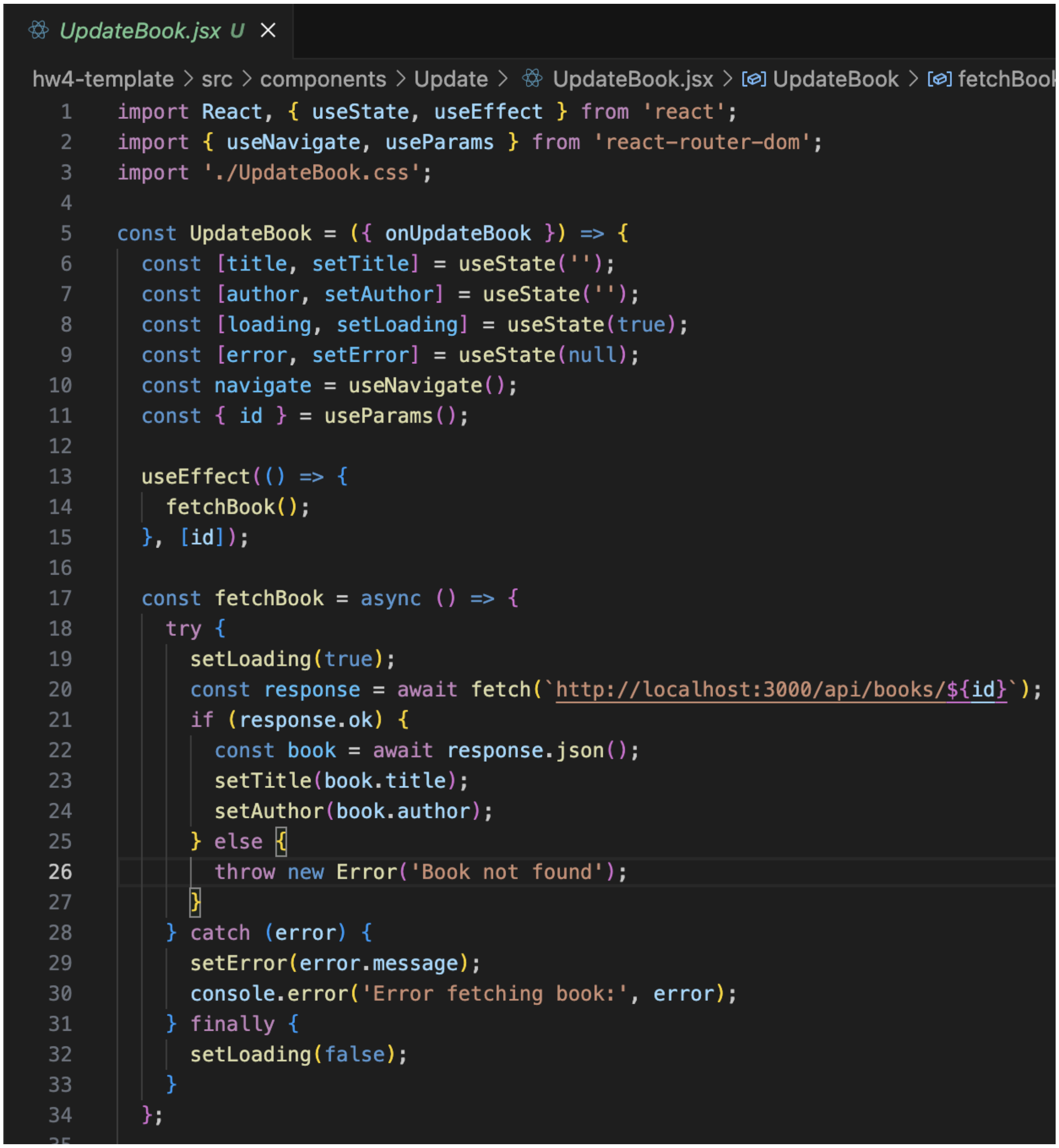
Author: James Clear

ISBN: 1234567

Updated Test Book

Author: Updated Test Author

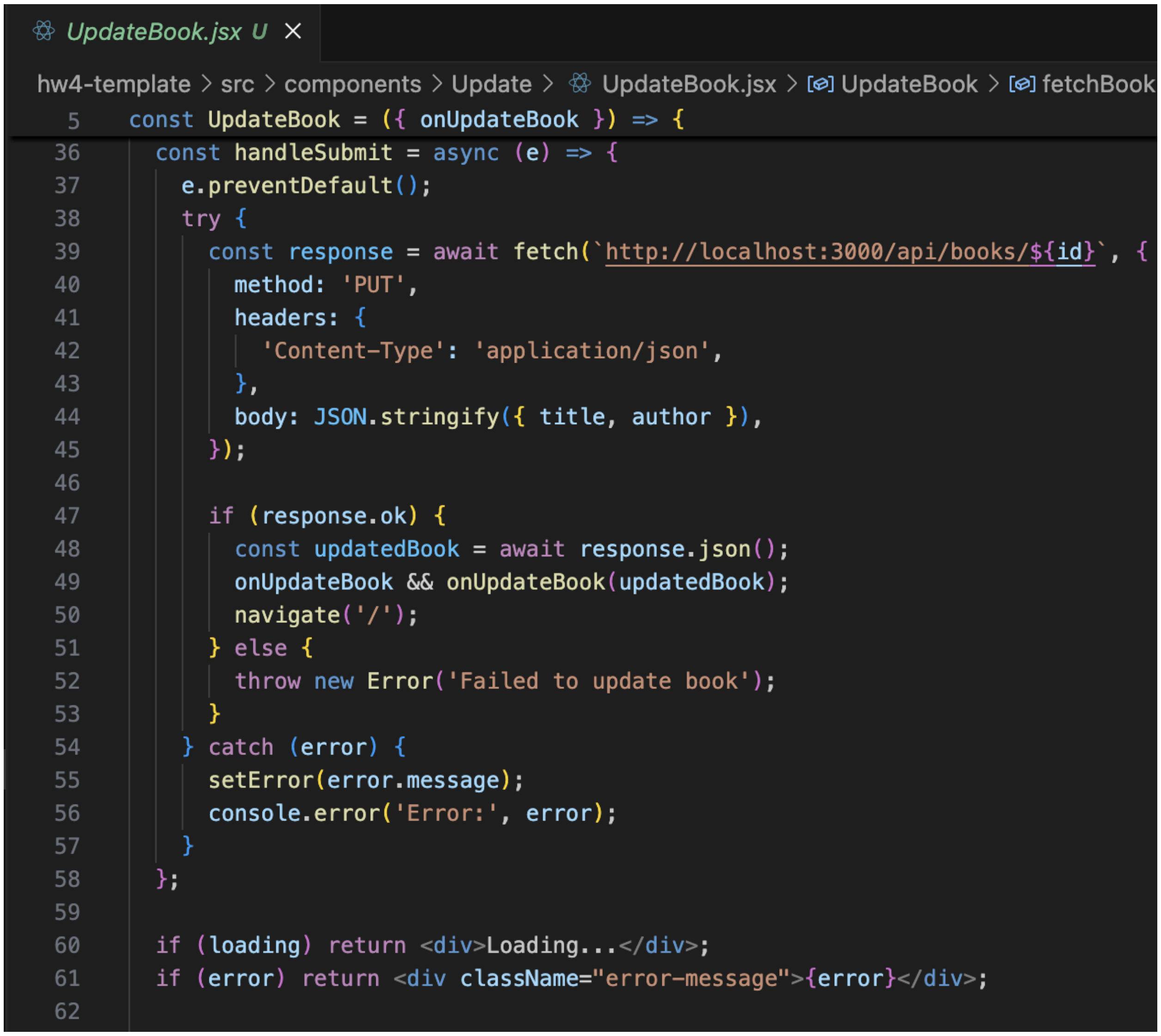
ISBN: 987654



UpdateBook.jsx U

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

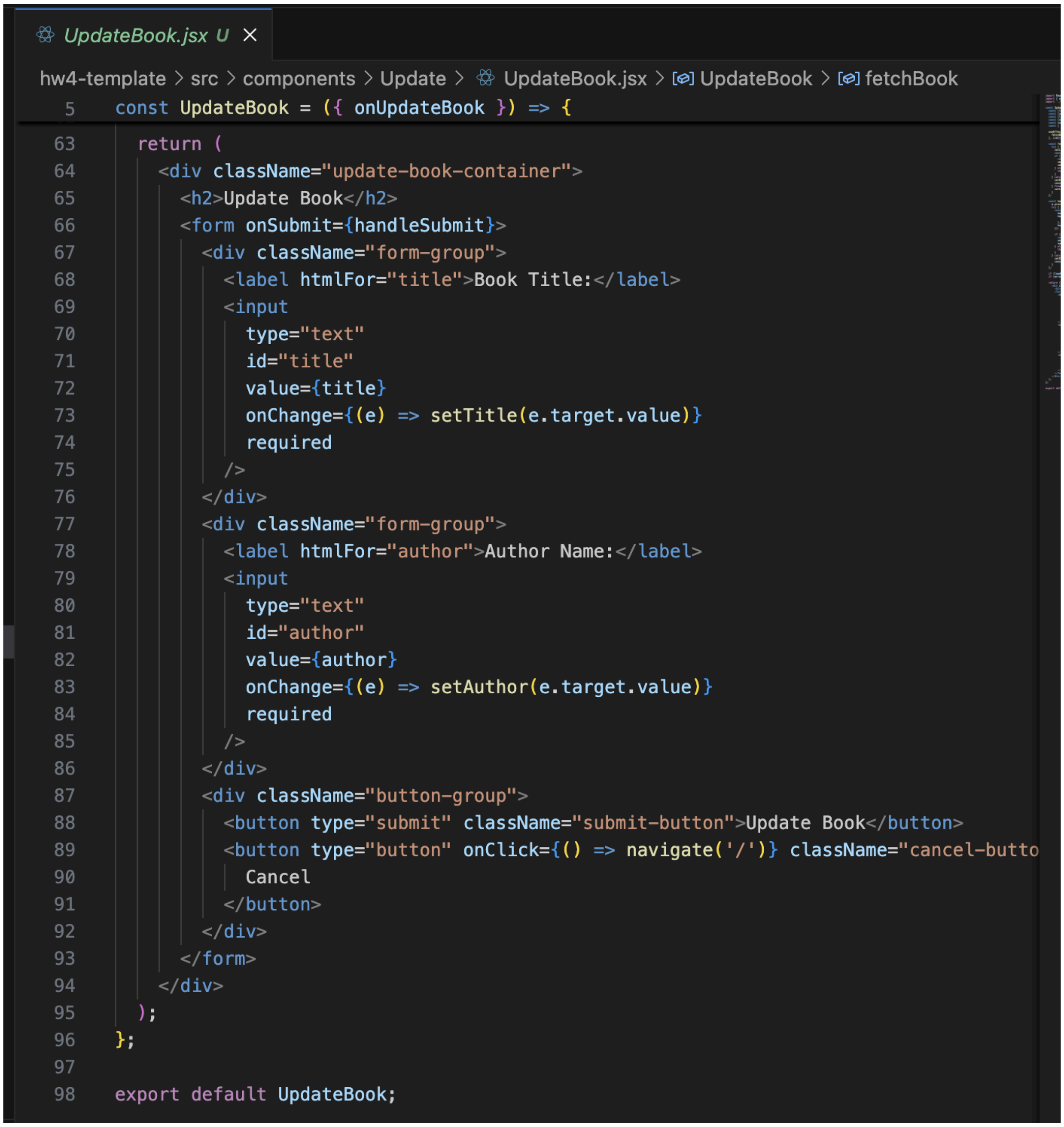
```jsx
import React, { useState, useEffect } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import './UpdateBook.css';

const UpdateBook = ({ onUpdateBook }) => {
  const [title, setTitle] = useState('');
  const [author, setAuthor] = useState('');
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    fetchBook();
  }, [id]);

  const fetchBook = async () => {
    try {
      setLoading(true);
      const response = await fetch(`http://localhost:3000/api/books/${id}`);
      if (response.ok) {
        const book = await response.json();
        setTitle(book.title);
        setAuthor(book.author);
      } else {
        throw new Error('Book not found');
      }
    } catch (error) {
      setError(error.message);
      console.error('Error fetching book:', error);
    } finally {
      setLoading(false);
    }
  };
```



UpdateBook.jsx U ×

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

```jsx
const UpdateBook = ({ onUpdateBook }) => {
  const handleSubmit = async (e) => {
    e.preventDefault();
    try {
      const response = await fetch(`http://localhost:3000/api/books/${id}`, {
        method: 'PUT',
        headers: {
          'Content-Type': 'application/json',
        },
        body: JSON.stringify({ title, author }),
      });

      if (response.ok) {
        const updatedBook = await response.json();
        onUpdateBook && onUpdateBook(updatedBook);
        navigate('/');
      } else {
        throw new Error('Failed to update book');
      }
    } catch (error) {
      setError(error.message);
      console.error('Error:', error);
    }
  };

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

```



UpdateBook.jsx U ×

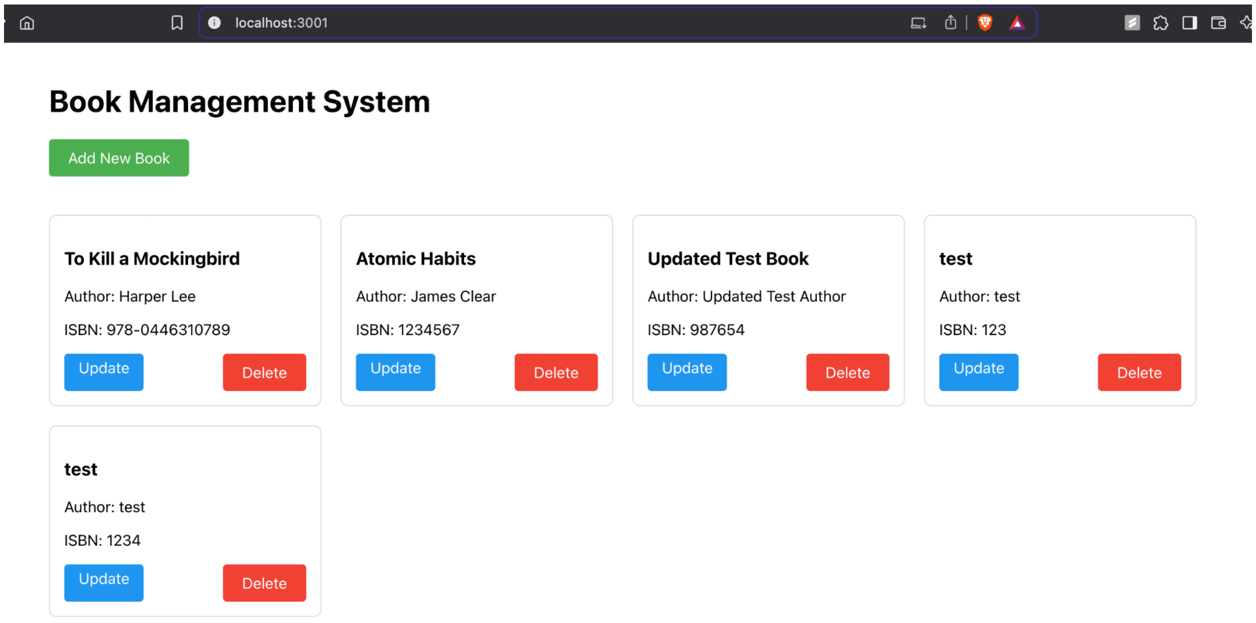
hw4-template > src > components > Update > UpdateBook.jsx > UpdateBook > fetchBook

```jsx
const UpdateBook = ({ onUpdateBook }) => {

  return (
    <div className="update-book-container">
      <h2>Update Book</h2>
      <form onSubmit={handleSubmit}>
        <div className="form-group">
          <label htmlFor="title">Book Title:</label>
          <input
            type="text"
            id="title"
            value={title}
            onChange={(e) => setTitle(e.target.value)}
            required
          />
        </div>
        <div className="form-group">
          <label htmlFor="author">Author Name:</label>
          <input
            type="text"
            id="author"
            value={author}
            onChange={(e) => setAuthor(e.target.value)}
            required
          />
        </div>
        <div className="button-group">
          <button type="submit" className="submit-button">Update Book</button>
          <button type="button" onClick={() => navigate('/')} className="cancel-butto
            Cancel
          </button>
        </div>
      </form>
    </div>
  );
};

export default UpdateBook;
```

IV . Delete the Book **(2 Points)**
Write the necessary code to create the 'DeleteBook.jsx' component. The component should be rendered when the user is on the '/delete' route.
The component should accept props (similar to demo) to delete a book and have the following:

● A button labeled "Delete Book".

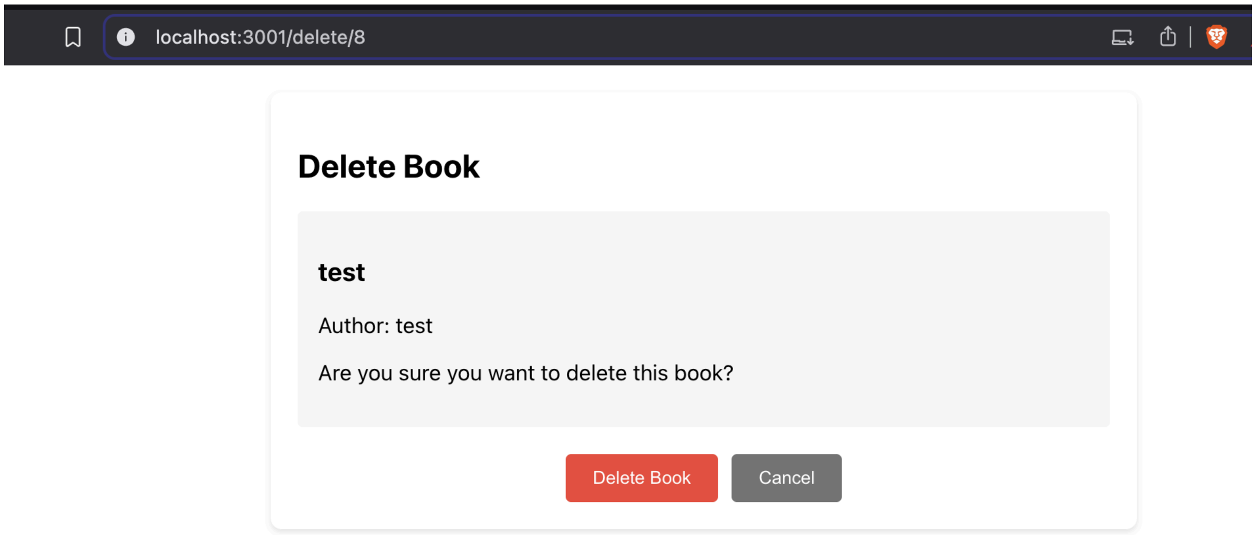
When the user clicks the Delete Book button the book in context should be deleted and the user should be redirected to the home page showing the updated book list.


localhost:3001

Book Management System

Add New Book

To Kill a Mockingbird	**Atomic Habits**	**Updated Test Book**	**test**
Author: Harper Lee	Author: James Clear	Author: Updated Test Author	Author: test
ISBN: 978-0446310789	ISBN: 1234567	ISBN: 987654	ISBN: 123
Update Delete	Update Delete	Update Delete	Update Delete
test			
Author: test			
ISBN: 1234			
Update Delete			


localhost:3001/delete/8

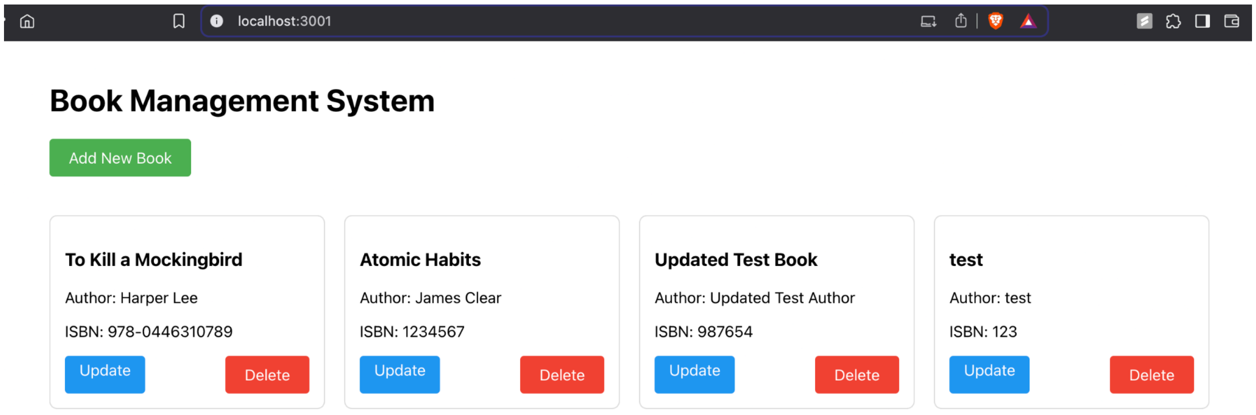
Delete Book

test

Author: test

Are you sure you want to delete this book?

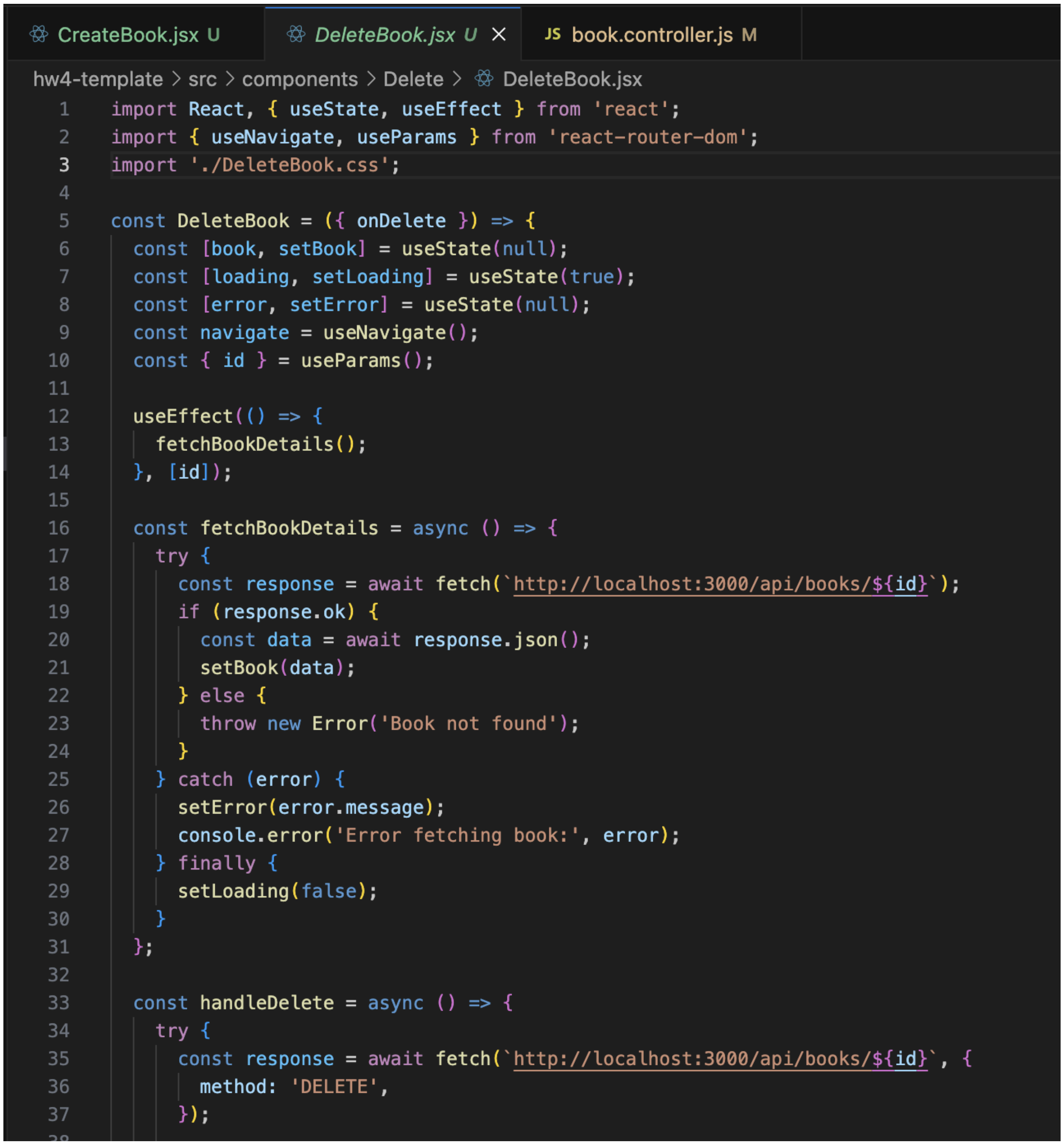
Delete Book Cancel


localhost:3001

Book Management System

Add New Book

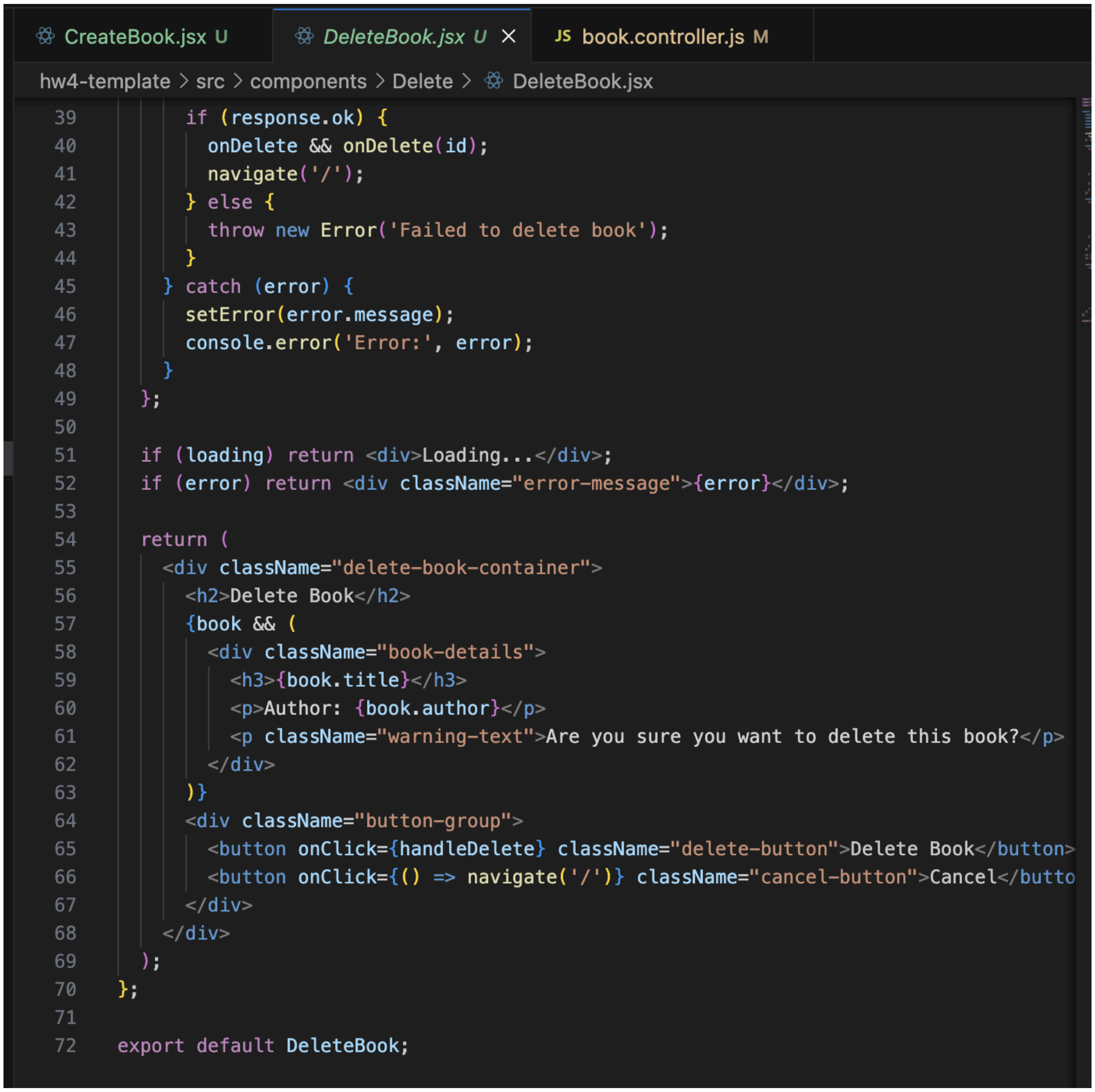
To Kill a Mockingbird	**Atomic Habits**	**Updated Test Book**	**test**
Author: Harper Lee	Author: James Clear	Author: Updated Test Author	Author: test
ISBN: 978-0446310789	ISBN: 1234567	ISBN: 987654	ISBN: 123
Update Delete	Update Delete	Update Delete	Update Delete



CreateBook.jsx U   DeleteBook.jsx U ×   JS book.controller.js M

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

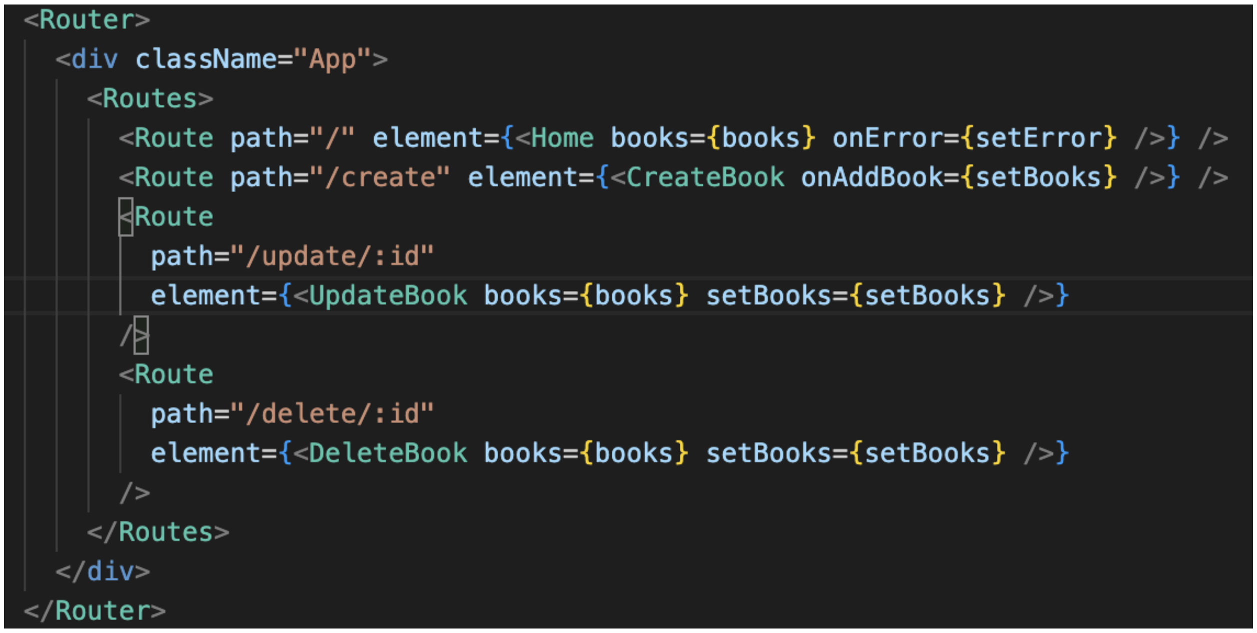
```
import React, { useState, useEffect } from 'react';
import { useNavigate, useParams } from 'react-router-dom';
import './DeleteBook.css';

const DeleteBook = ({ onDelete }) => {
  const [book, setBook] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    fetchBookDetails();
  }, [id]);

  const fetchBookDetails = async () => {
    try {
      const response = await fetch(`http://localhost:3000/api/books/${id}`);
      if (response.ok) {
        const data = await response.json();
        setBook(data);
      } else {
        throw new Error('Book not found');
      }
    } catch (error) {
      setError(error.message);
      console.error('Error fetching book:', error);
    } finally {
      setLoading(false);
    }
  };

  const handleDelete = async () => {
    try {
      const response = await fetch(`http://localhost:3000/api/books/${id}`, {
        method: 'DELETE',
      });
```



```
        if (response.ok) {
          onDelete && onDelete(id);
          navigate('/');
        } else {
          throw new Error('Failed to delete book');
        }
      } 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 className="warning-text">Are you sure you want to delete this book?</p>
          </div>
        )}
        <div className="button-group">
          <button onClick={handleDelete} className="delete-button">Delete Book</button>
          <button onClick={() => navigate('/')} className="cancel-button">Cancel</butto
        </div>
      </div>
    );
  };

  export default DeleteBook;
```

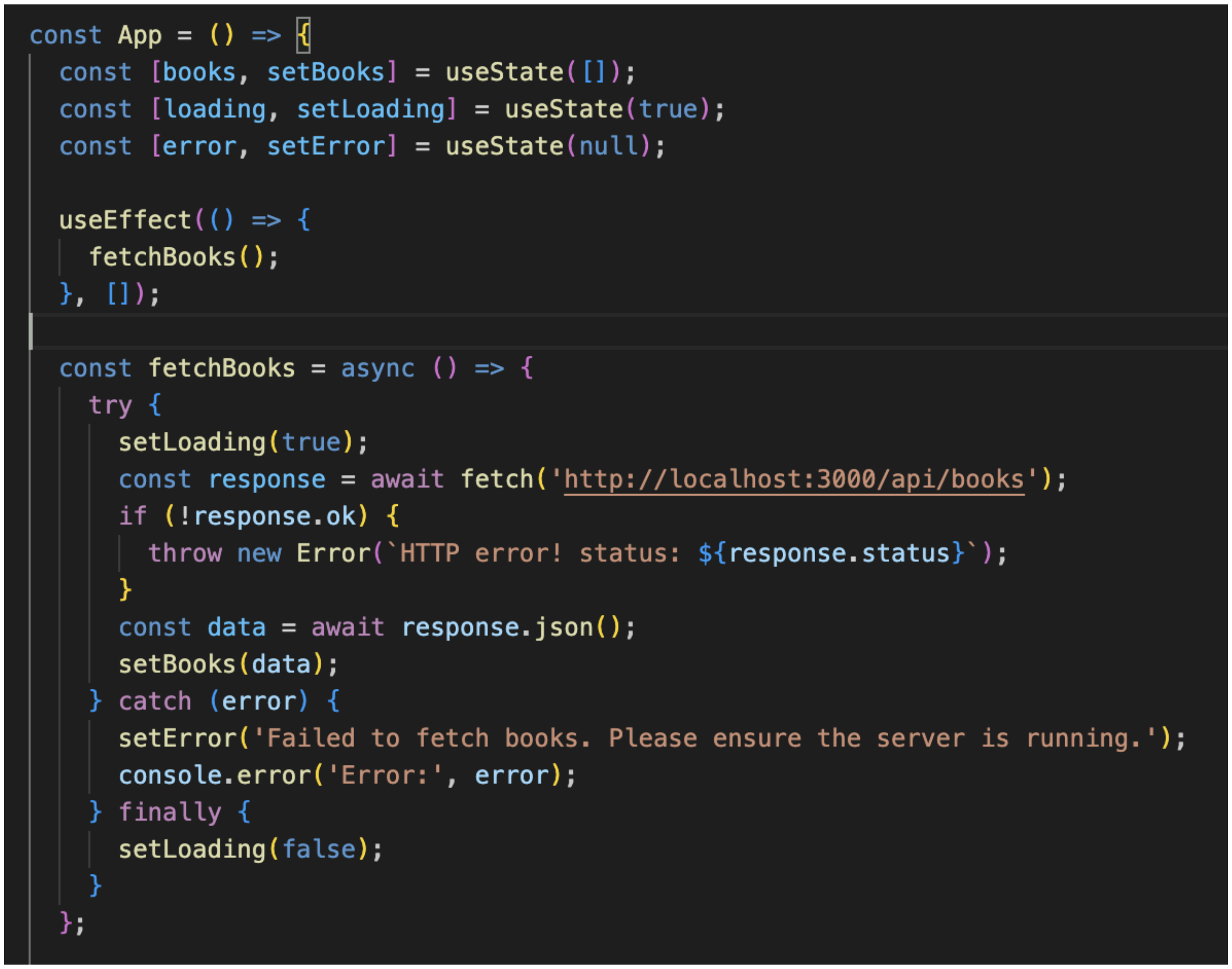
V . For all the above questions make sure to:

● Pass props for the Create, Update, and Delete components. **(1 Point)**

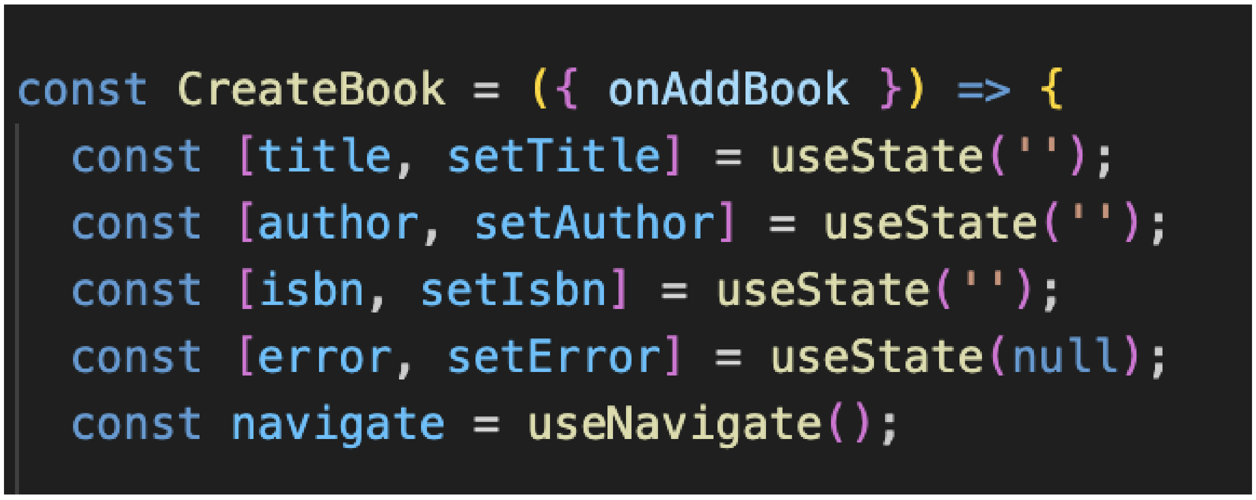


```
<Router>
  <div className="App">
    <Routes>
      <Route path="/" element={<Home books={books} onError={setError} />} />
      <Route path="/create" element={<CreateBook onAddBook={setBooks} />} />
      <Route
        path="/update/:id"
        element={<UpdateBook books={books} setBooks={setBooks} />}
      />
      <Route
        path="/delete/:id"
        element={<DeleteBook books={books} setBooks={setBooks} />}
      />
    </Routes>
  </div>
</Router>
```

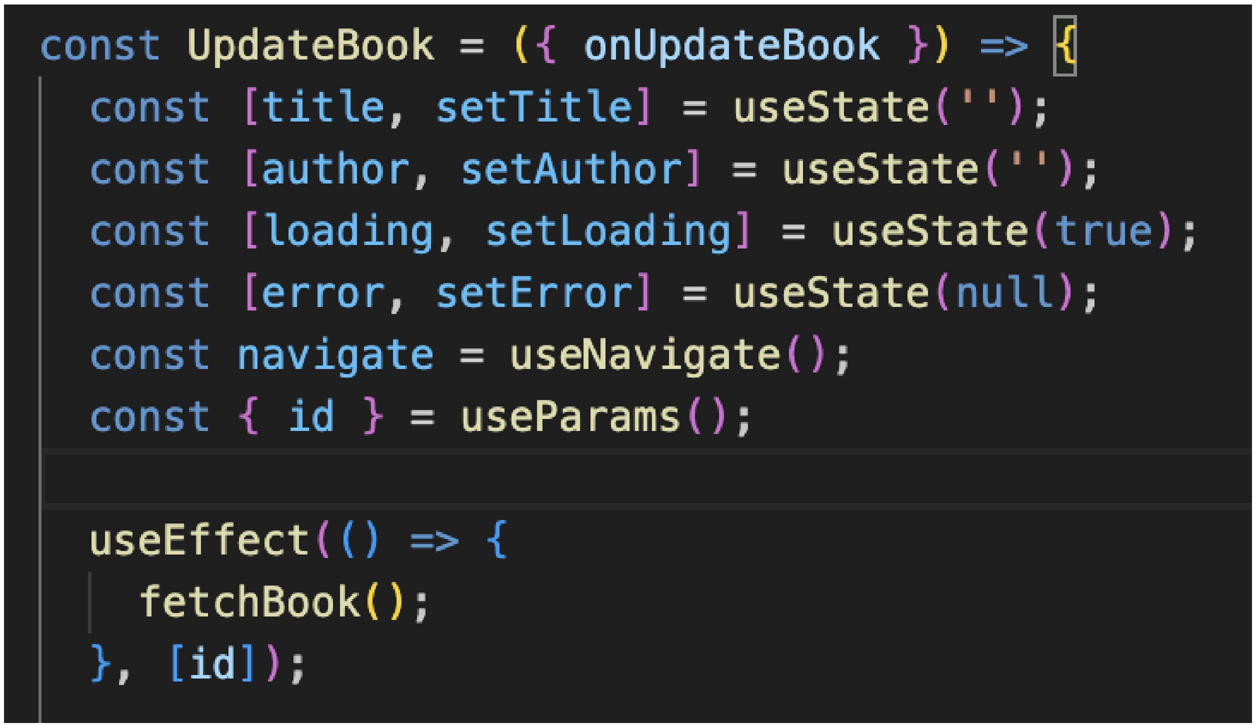
- Use hooks like useState and useEffect wherever necessary. **(1 Point)**



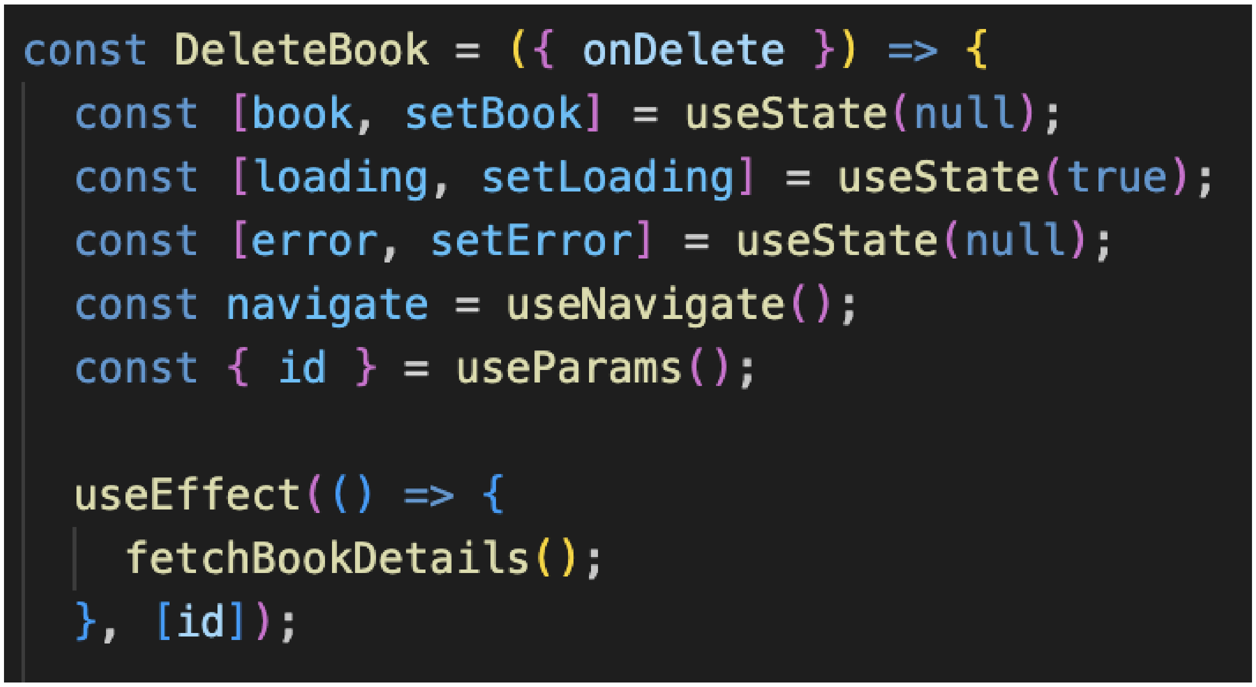
```
const App = () => {
  const [books, setBooks] = useState([]);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);

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

  const fetchBooks = async () => {
    try {
      setLoading(true);
      const response = await fetch('http://localhost:3000/api/books');
      if (!response.ok) {
        throw new Error(`HTTP error! status: ${response.status}`);
      }
      const data = await response.json();
      setBooks(data);
    } catch (error) {
      setError('Failed to fetch books. Please ensure the server is running.');
      console.error('Error:', error);
    } finally {
      setLoading(false);
    }
  };
```



```
const CreateBook = ({ onAddBook }) => {
  const [title, setTitle] = useState('');
  const [author, setAuthor] = useState('');
  const [isbn, setIsbn] = useState('');
  const [error, setError] = useState(null);
  const navigate = useNavigate();
```

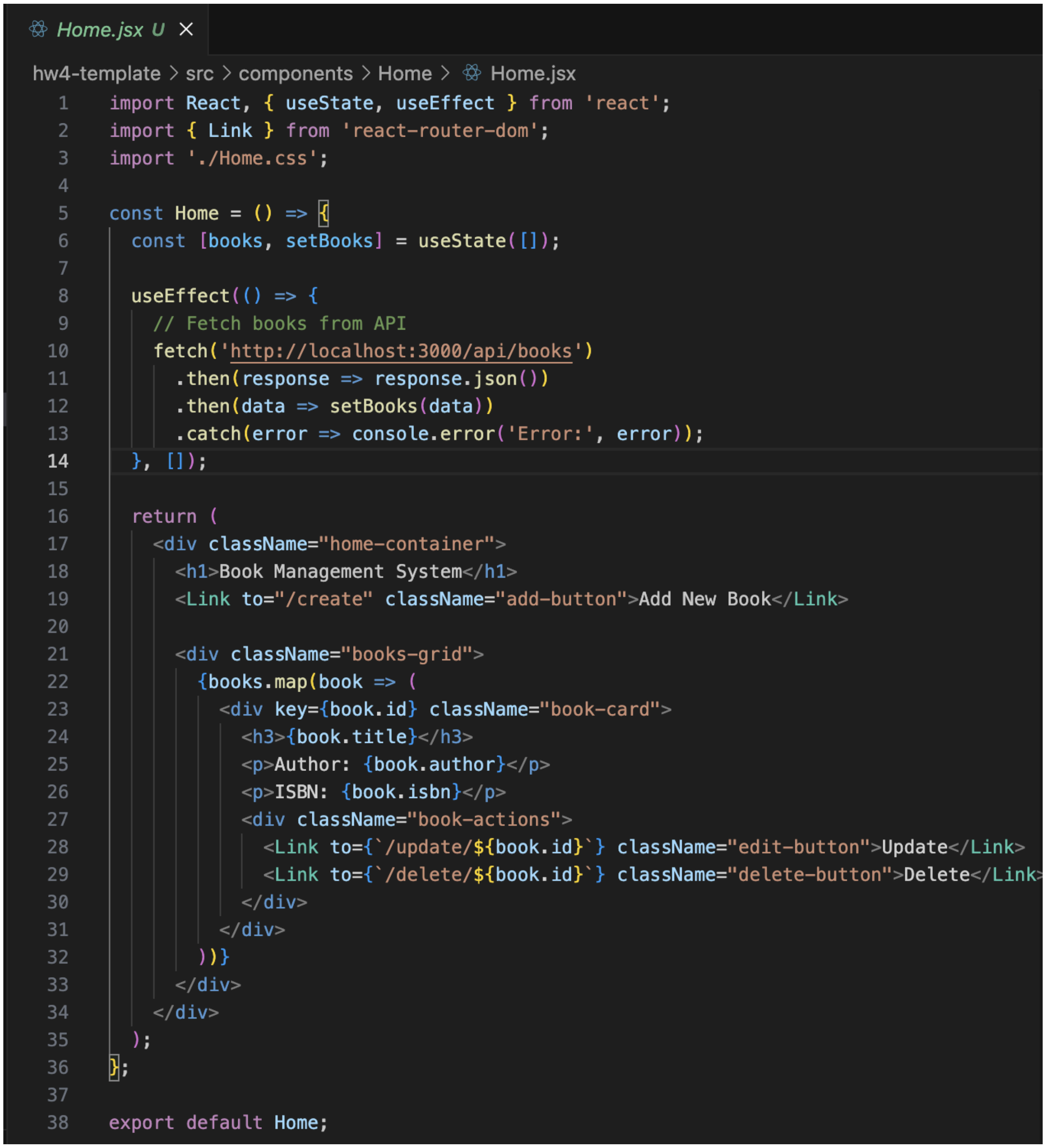


```
const UpdateBook = ({ onUpdateBook }) => {
  const [title, setTitle] = useState('');
  const [author, setAuthor] = useState('');
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    fetchBook();
  }, [id]);
```



```
const DeleteBook = ({ onDelete }) => {
  const [book, setBook] = useState(null);
  const [loading, setLoading] = useState(true);
  const [error, setError] = useState(null);
  const navigate = useNavigate();
  const { id } = useParams();

  useEffect(() => {
    fetchBookDetails();
  }, [id]);
```

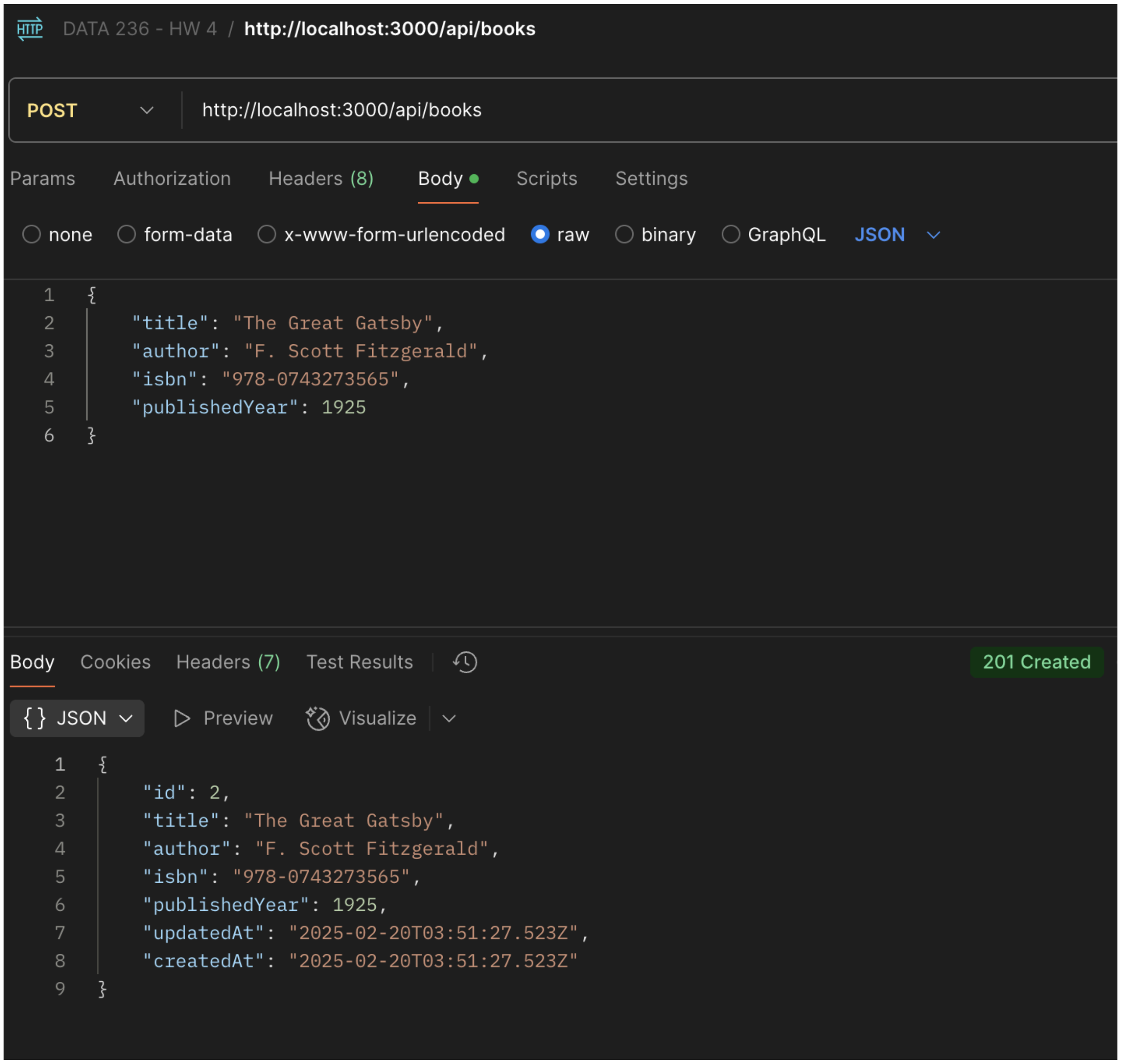
● User React-Router-Dom for routing. **(1 Point)**



```
import React, { useState, useEffect } from 'react';
import { Link } from 'react-router-dom';
import './Home.css';

const Home = () => {
  const [books, setBooks] = useState([]);

  useEffect(() => {
    // Fetch books from API
    fetch('http://localhost:3000/api/books')
      .then(response => response.json())
      .then(data => setBooks(data))
      .catch(error => console.error('Error:', error));
  }, []);

  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/${book.id}`} className="edit-button">Update</Link>
              <Link to={`/delete/${book.id}`} className="delete-button">Delete</Link
            </div>
          </div>
        ))}
      </div>
    </div>
  );
};

export default Home;
```

Q2. MySQL (10 points)
In this question, you will apply the concepts learned in class to build a simple CRUD (Create, Read, Update, Delete) application using Node.js and MySQL using Sequelize
Create an appropriate book Model, *View(Optional)*, and Controller.

You are required to create a Book Management System that allows users to:
● Add a new book (POST)



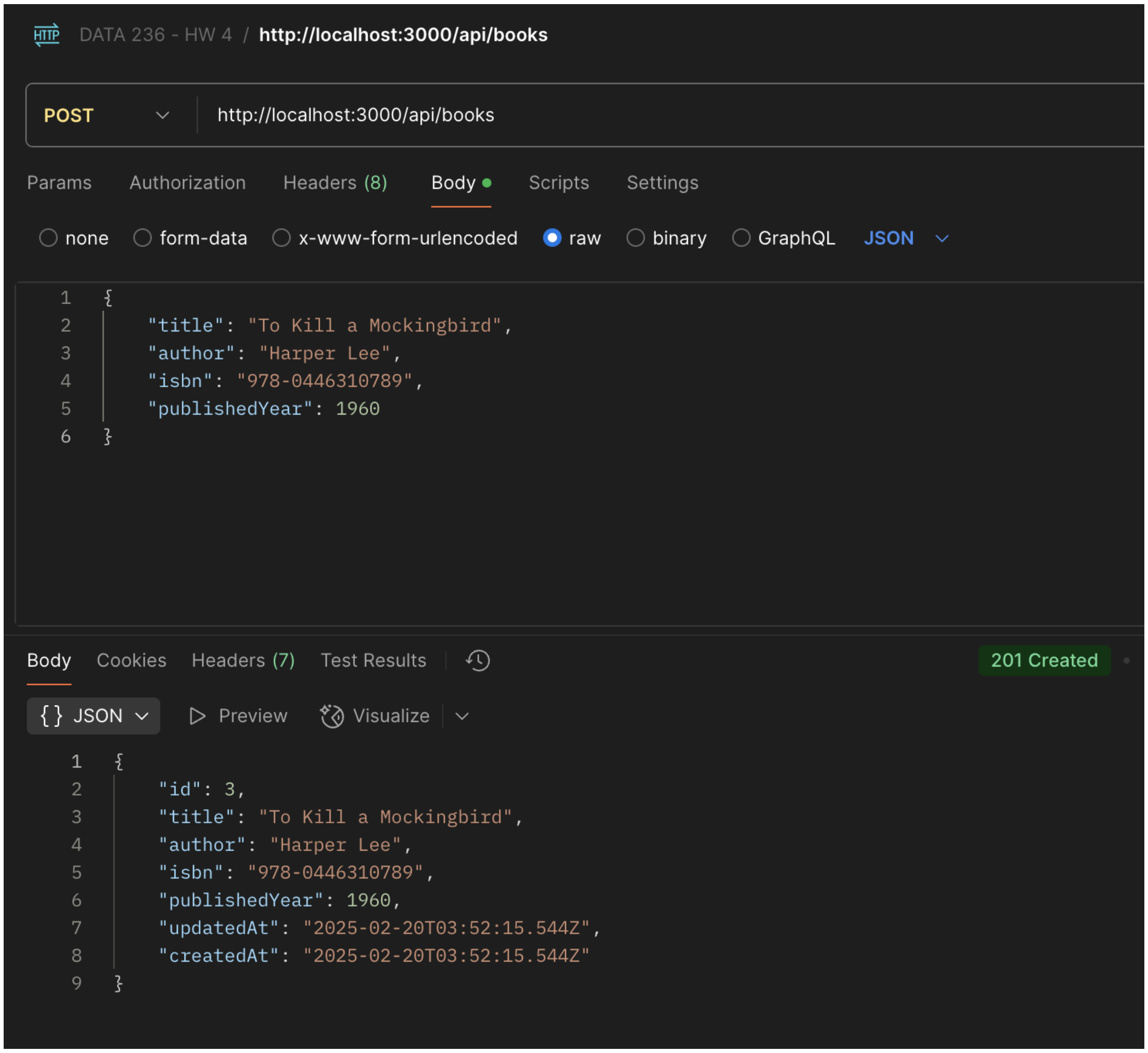
POST http://localhost:3000/api/books

```
{
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "isbn": "978-0743273565",
    "publishedYear": 1925
}
```

201 Created

{} JSON

```
{
    "id": 2,
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "isbn": "978-0743273565",
    "publishedYear": 1925,
    "updatedAt": "2025-02-20T03:51:27.523Z",
    "createdAt": "2025-02-20T03:51:27.523Z"
}
```



DATA 236 - HW 4 / http://localhost:3000/api/books

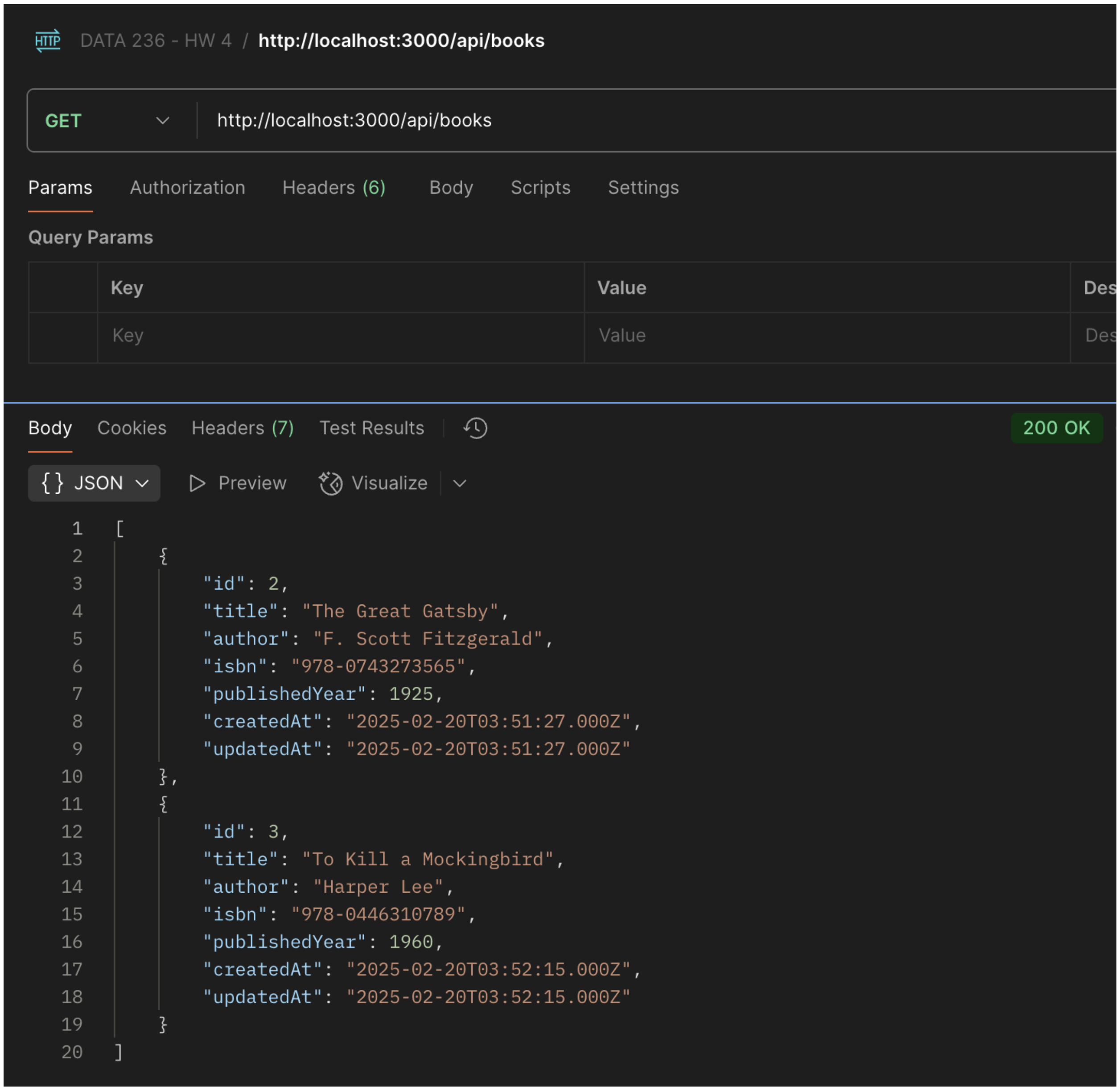
POST http://localhost:3000/api/books

```
{
    "title": "To Kill a Mockingbird",
    "author": "Harper Lee",
    "isbn": "978-0446310789",
    "publishedYear": 1960
}
```

201 Created

```
{
    "id": 3,
    "title": "To Kill a Mockingbird",
    "author": "Harper Lee",
    "isbn": "978-0446310789",
    "publishedYear": 1960,
    "updatedAt": "2025-02-20T03:52:15.544Z",
    "createdAt": "2025-02-20T03:52:15.544Z"
}
```

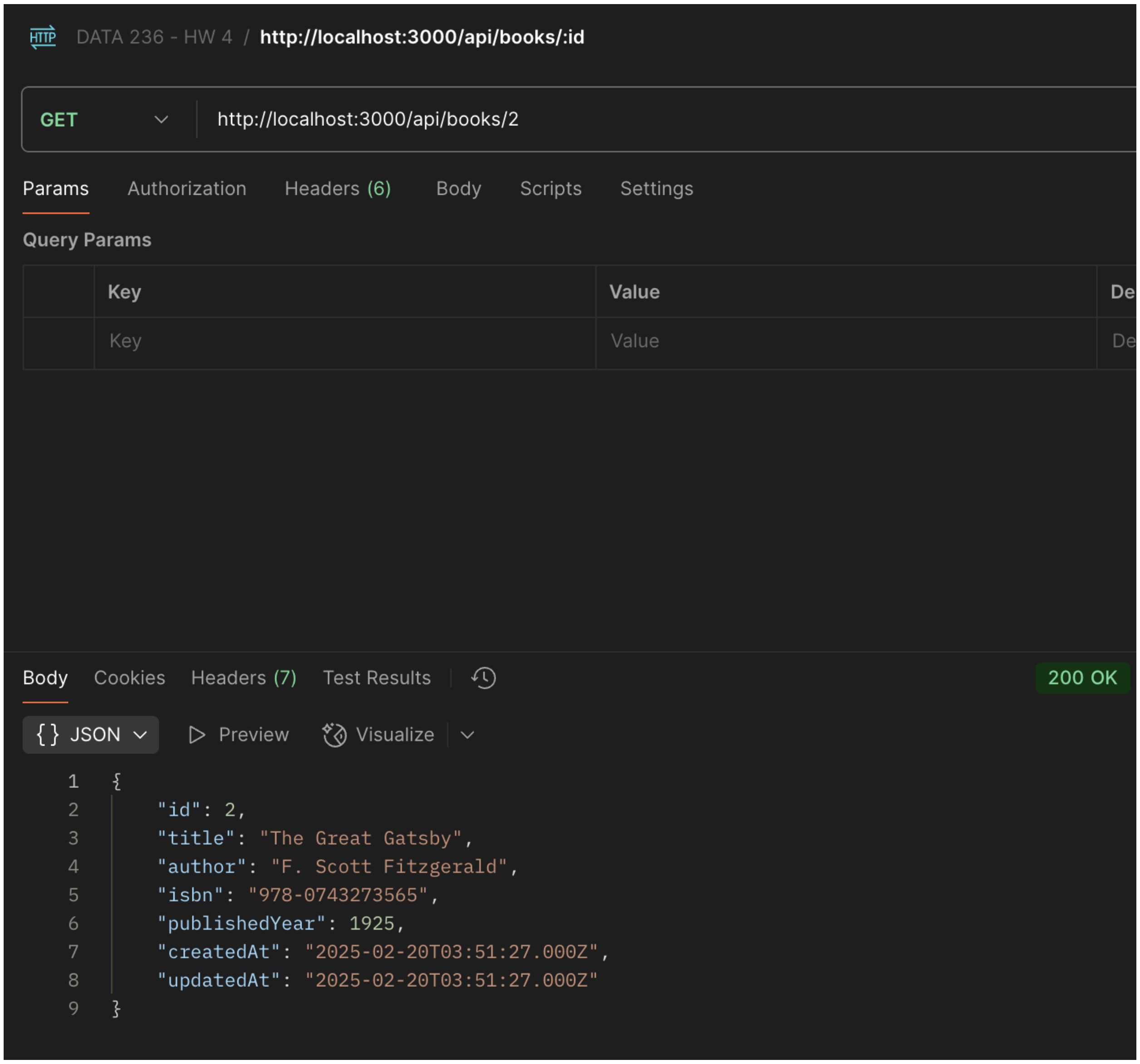
● View all books (GET)



GET http://localhost:3000/api/books

```
[
    {
        "id": 2,
        "title": "The Great Gatsby",
        "author": "F. Scott Fitzgerald",
        "isbn": "978-0743273565",
        "publishedYear": 1925,
        "createdAt": "2025-02-20T03:51:27.000Z",
        "updatedAt": "2025-02-20T03:51:27.000Z"
    },
    {
        "id": 3,
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "isbn": "978-0446310789",
        "publishedYear": 1960,
        "createdAt": "2025-02-20T03:52:15.000Z",
        "updatedAt": "2025-02-20T03:52:15.000Z"
    }
]
```

- View a book by ID (GET)



DATA 236 - HW 4 / http://localhost:3000/api/books/:id

GET http://localhost:3000/api/books/2

Params Authorization Headers (6) Body Scripts Settings

Query Params

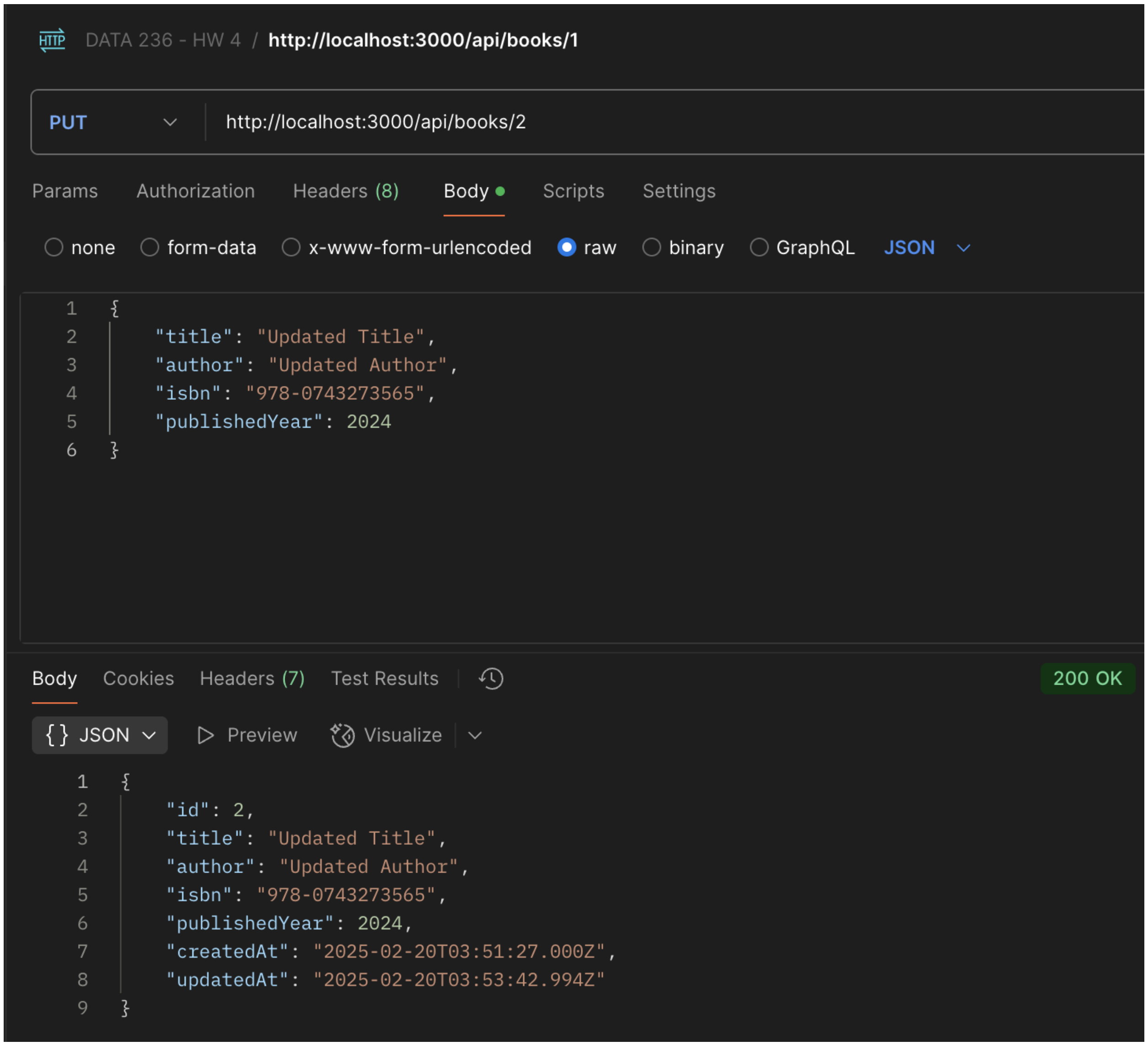
	Key	Value	De
	Key	Value	De

Body Cookies Headers (7) Test Results 200 OK

{} JSON Preview Visualize

```
{
    "id": 2,
    "title": "The Great Gatsby",
    "author": "F. Scott Fitzgerald",
    "isbn": "978-0743273565",
    "publishedYear": 1925,
    "createdAt": "2025-02-20T03:51:27.000Z",
    "updatedAt": "2025-02-20T03:51:27.000Z"
}
```

● Update book details (PUT)



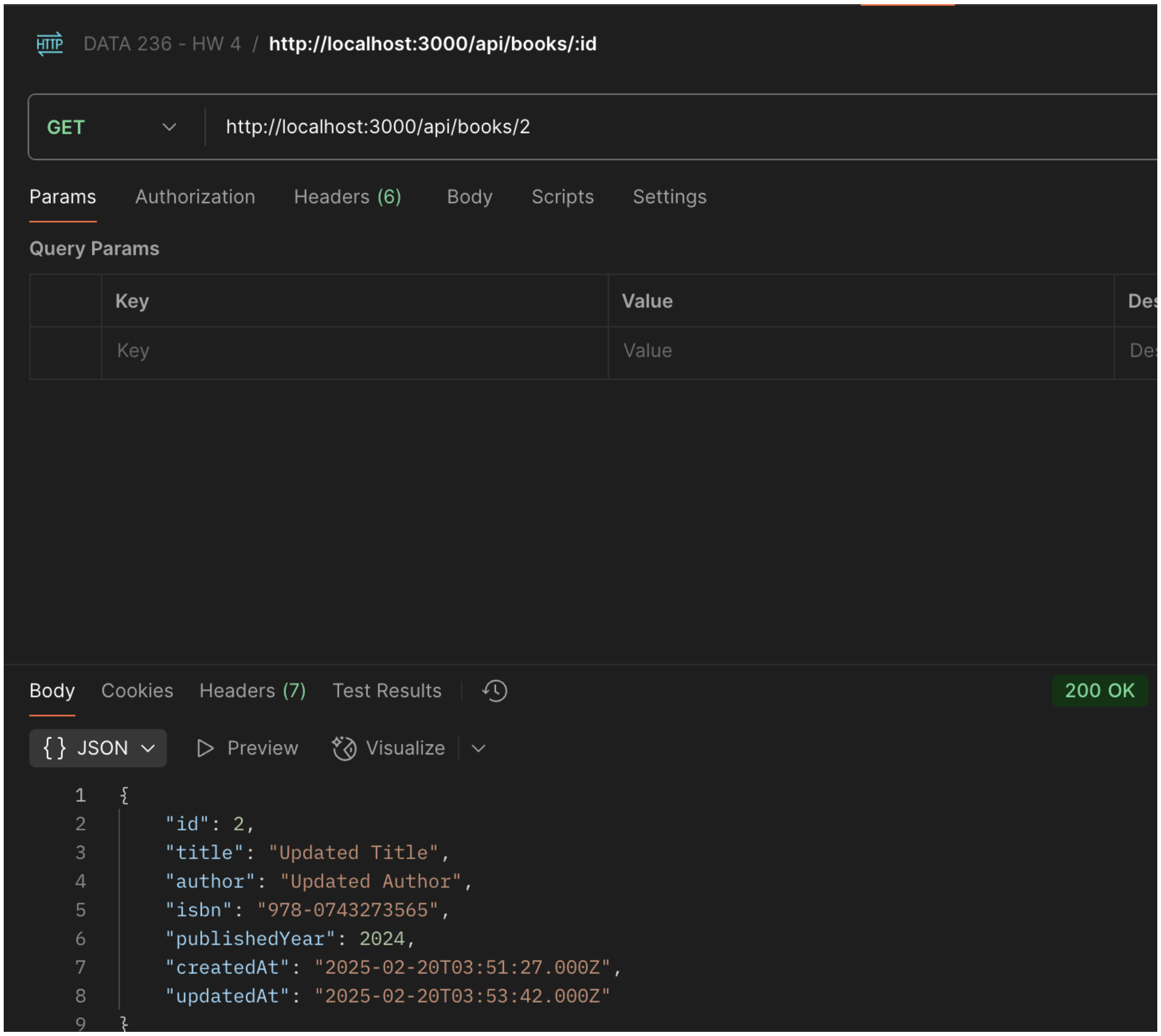
http://localhost:3000/api/books/2

( ) none ( ) form-data ( ) x-www-form-urlencoded [✓ raw] ( ) binary ( ) GraphQL JSON

```
{
    "title": "Updated Title",
    "author": "Updated Author",
    "isbn": "978-0743273565",
    "publishedYear": 2024
}
```

200 OK

```
{
    "id": 2,
    "title": "Updated Title",
    "author": "Updated Author",
    "isbn": "978-0743273565",
    "publishedYear": 2024,
    "createdAt": "2025-02-20T03:51:27.000Z",
    "updatedAt": "2025-02-20T03:53:42.994Z"
}
```



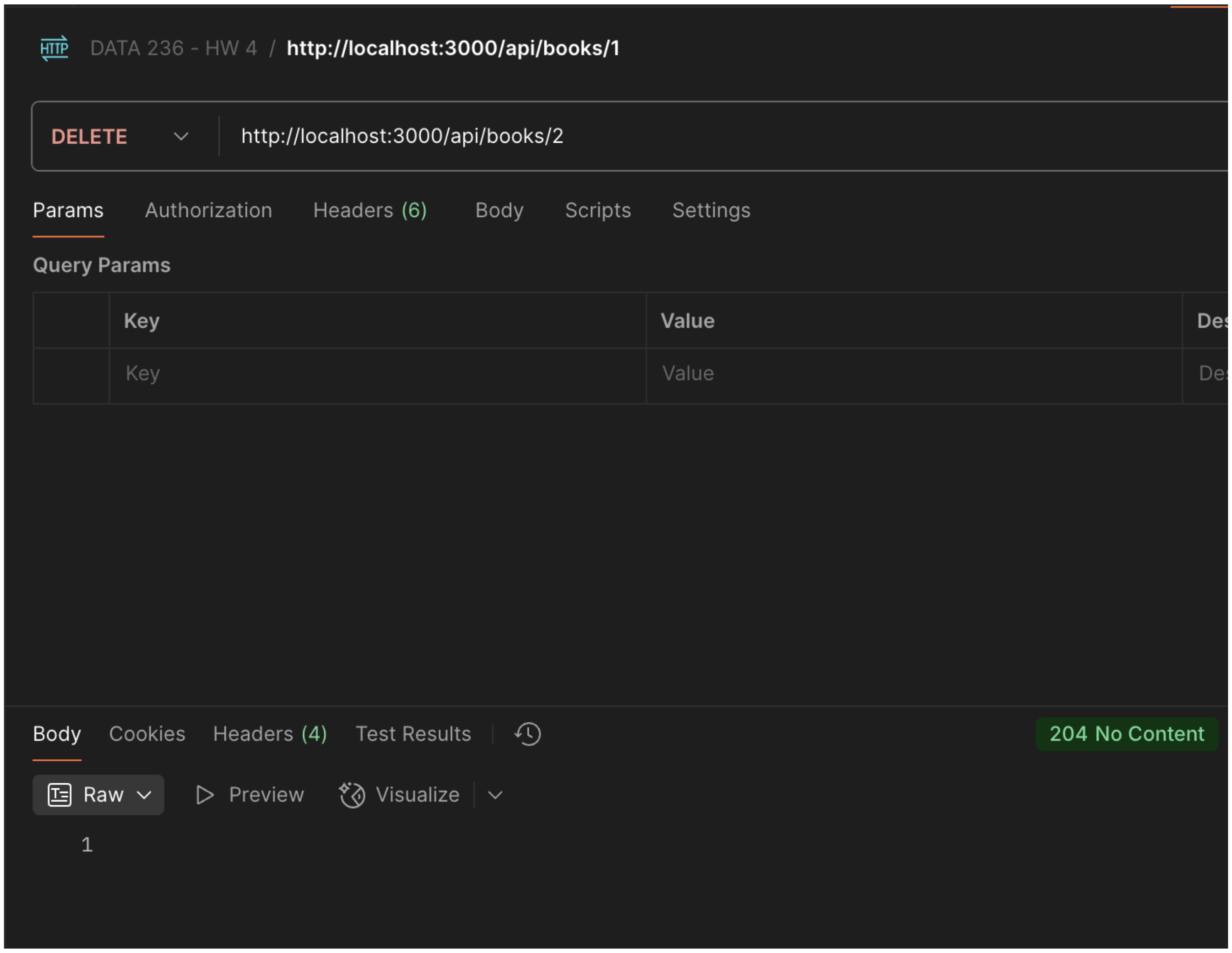
GET http://localhost:3000/api/books/2

Query Params

Key	Value	Des
Key	Value	De

```
{
    "id": 2,
    "title": "Updated Title",
    "author": "Updated Author",
    "isbn": "978-0743273565",
    "publishedYear": 2024,
    "createdAt": "2025-02-20T03:51:27.000Z",
    "updatedAt": "2025-02-20T03:53:42.000Z"
}
```

● Delete a book (DELETE)



DATA 236 - HW 4 / http://localhost:3000/api/books/1

DELETE http://localhost:3000/api/books/2

Params Authorization Headers (6) Body Scripts Settings

Query Params

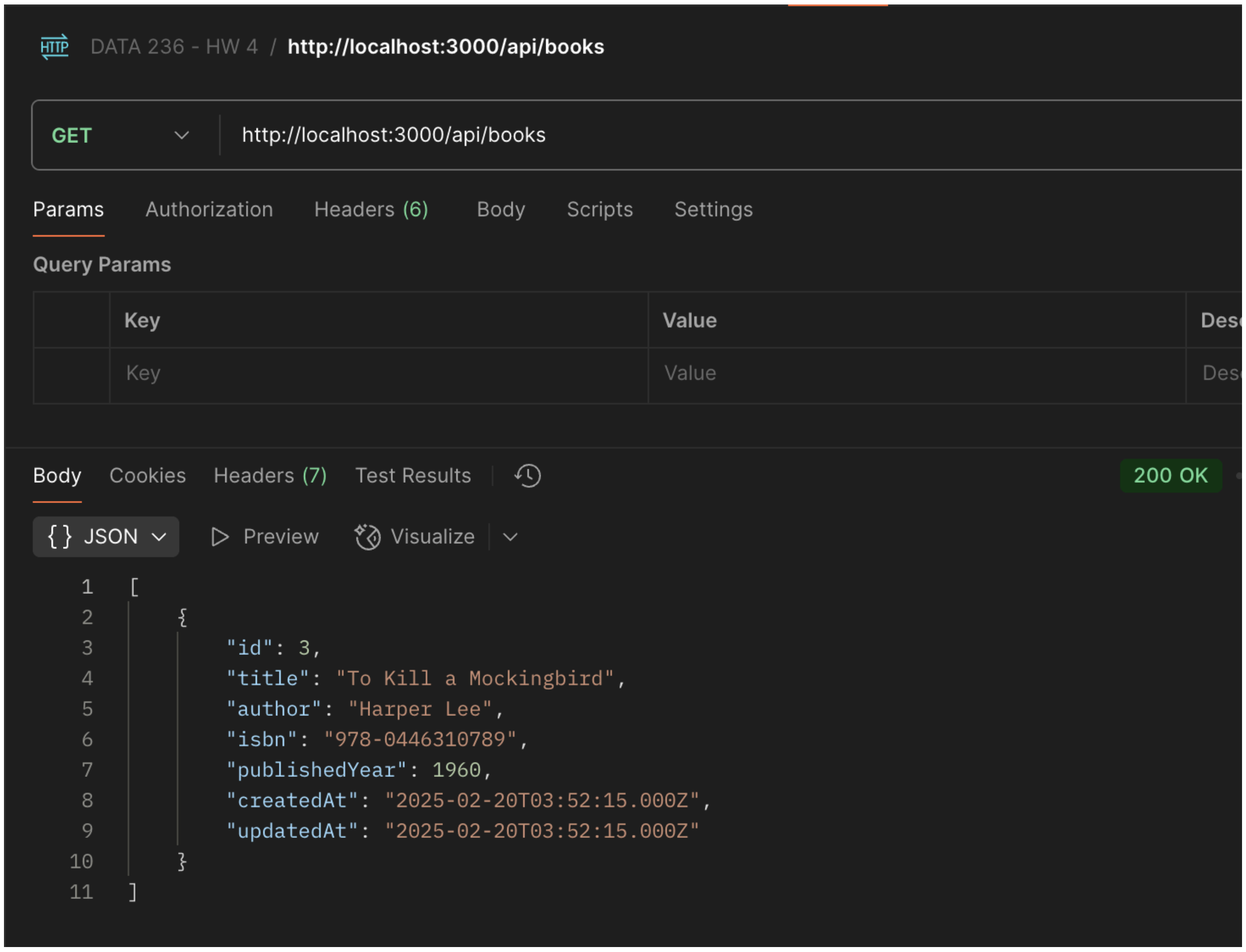
	Key	Value	Des
	Key	Value	Des

Body Cookies Headers (4) Test Results

204 No Content

Raw Preview Visualize

```
1
```



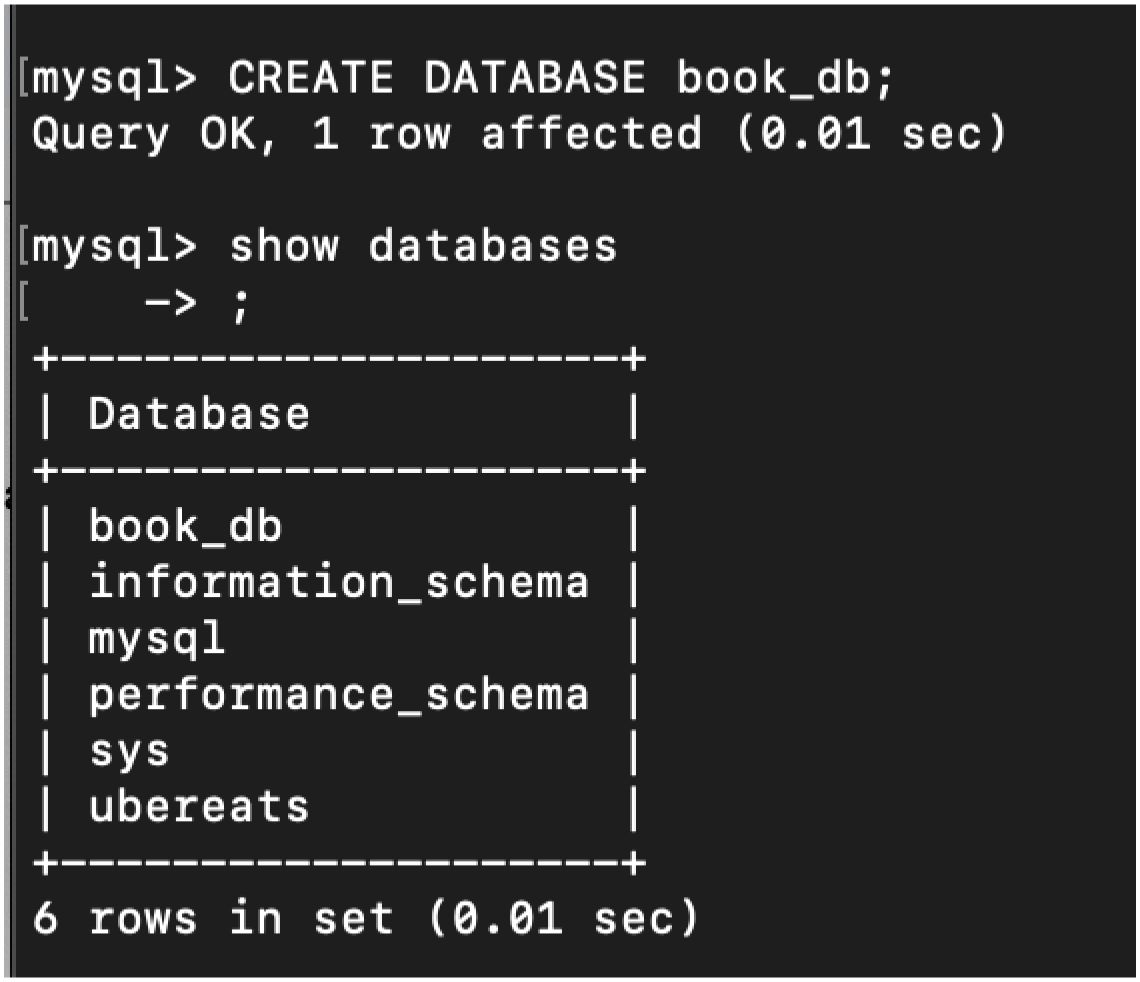
GET http://localhost:3000/api/books

Query Params

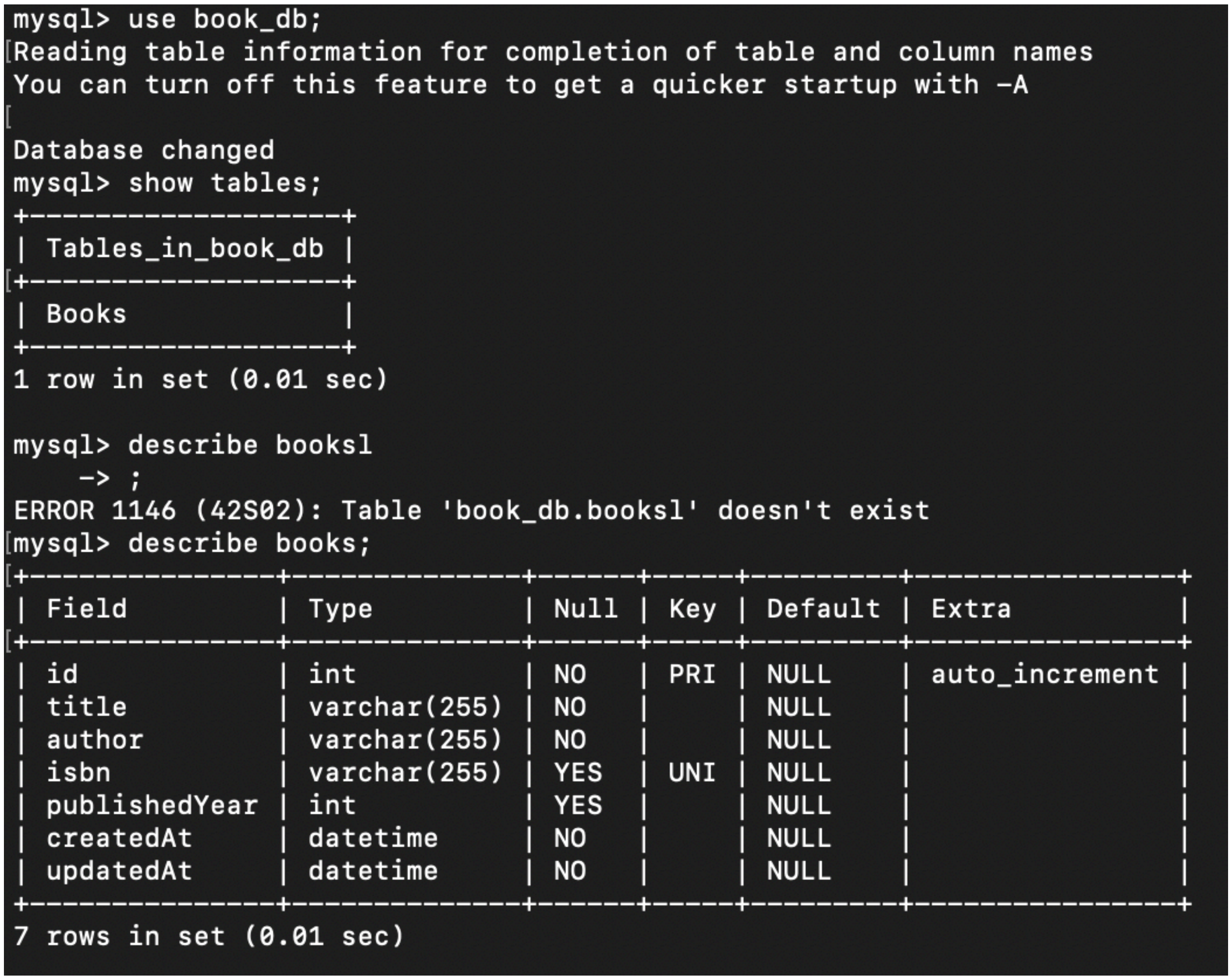
Key	Value	Des
Key	Value	Des

```
[
    {
        "id": 3,
        "title": "To Kill a Mockingbird",
        "author": "Harper Lee",
        "isbn": "978-0446310789",
        "publishedYear": 1960,
        "createdAt": "2025-02-20T03:52:15.000Z",
        "updatedAt": "2025-02-20T03:52:15.000Z"
    }
]
```

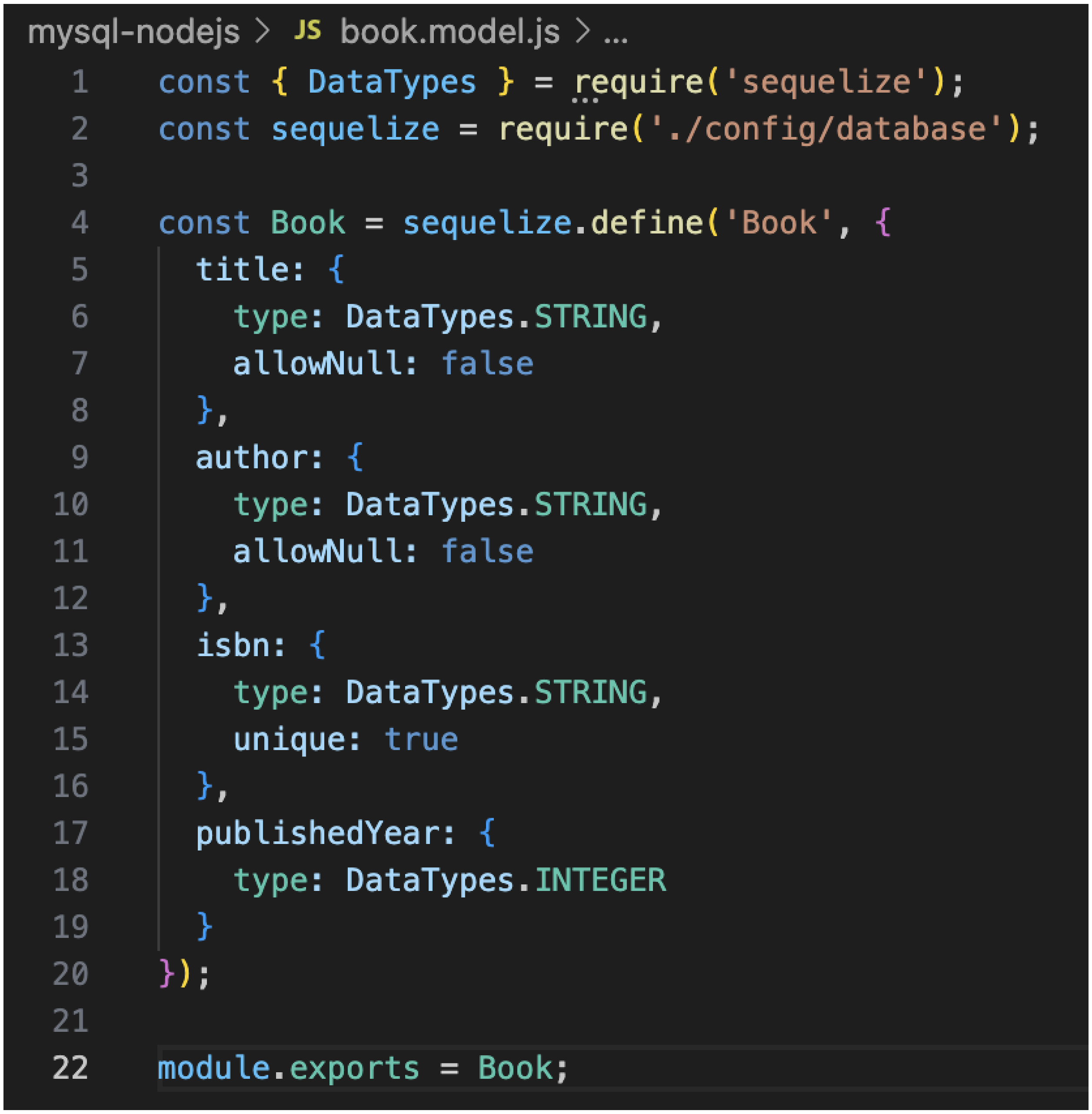
Create a MySQL database named **book_db** with a table called **books** for this task.



```
mysql> CREATE DATABASE book_db;
Query OK, 1 row affected (0.01 sec)

mysql> show databases
    -> ;
+--------------------+
| Database           |
+--------------------+
| book_db            |
| information_schema |
| mysql              |
| performance_schema |
| sys                |
| ubereats           |
+--------------------+
6 rows in set (0.01 sec)
```

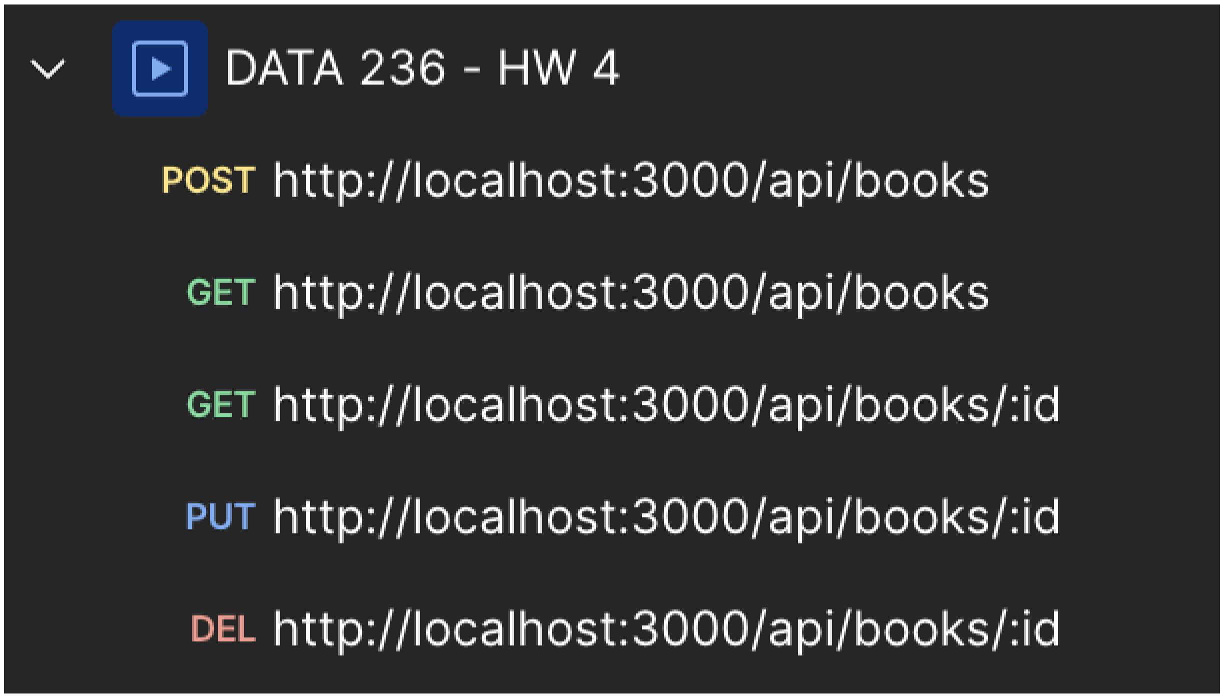


```
mysql> use book_db;
[Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A
[
Database changed
mysql> show tables;
+-------------------+
| Tables_in_book_db |
[+-------------------+
| Books             |
+-------------------+
1 row in set (0.01 sec)

mysql> describe booksl
    -> ;
ERROR 1146 (42S02): Table 'book_db.booksl' doesn't exist
[mysql> describe books;
[+---------------+--------------+------+-----+---------+----------------+
| Field         | Type         | Null | Key | Default | Extra          |
[+---------------+--------------+------+-----+---------+----------------+
| id            | int          | NO   | PRI | NULL    | auto_increment |
| title         | varchar(255) | NO   |     | NULL    |                |
| author        | varchar(255) | NO   |     | NULL    |                |
| isbn          | varchar(255) | YES  | UNI | NULL    |                |
| publishedYear | int          | YES  |     | NULL    |                |
| createdAt     | datetime     | NO   |     | NULL    |                |
| updatedAt     | datetime     | NO   |     | NULL    |                |
+---------------+--------------+------+-----+---------+----------------+
7 rows in set (0.01 sec)
```



```javascript
const { DataTypes } = require('sequelize');
const sequelize = require('./config/database');

const Book = sequelize.define('Book', {
  title: {
    type: DataTypes.STRING,
    allowNull: false
  },
  author: {
    type: DataTypes.STRING,
    allowNull: false
  },
  isbn: {
    type: DataTypes.STRING,
    unique: true
  },
  publishedYear: {
    type: DataTypes.INTEGER
  }
});

module.exports = Book;
```

You are free to name your API endpoints appropriately.


DATA 236 - HW 4

```
POST http://localhost:3000/api/books
GET http://localhost:3000/api/books
GET http://localhost:3000/api/books/:id
PUT http://localhost:3000/api/books/:id
DEL http://localhost:3000/api/books/:id
```

Submit the Postman API response screenshots for each operation. Each completed operation will get 2 points. Also, submit a screenshot of the database and project folder structure.