

DATA-236 Sec 12 - Distributed Systems for Data Engineering

HOMEWORK 4

1. React

Create a Book Management App where users can add, update, and delete books.

I. Home Page

Write the necessary code to create the 'Home.jsx' component which will display all the books.

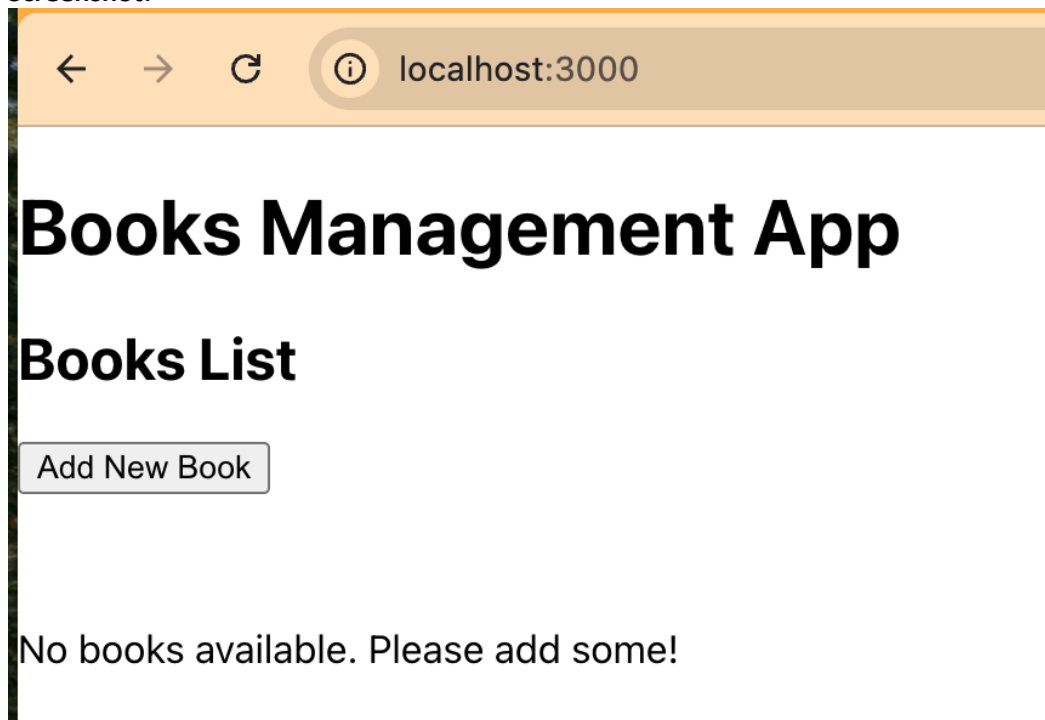
The Home component should be rendered when the user is on the root route '/'.

Code: Home.jsx

```
src > components > @ Home.jsx > @ Home > @ books.map() callback
1  import React, { useContext, useEffect } from "react";
2  //import { Link } from "react-router-dom";
3  import { useNavigate } from "react-router-dom";
4  import { BookContext } from "../context/BookContext";
5  import BookList from "../BooksList";
6
7  const Home = () => {
8    const {books} = useContext(BookContext);
9    const navigate = useNavigate();
10
11    //useEffect
12    useEffect(() => {
13      console.log("Books List updated:", books);
14    }, [books]);
15
16    return (
17      <div className="container mt-4">
18        <h1 className="text-center mb-4">Books Management App</h1>
19        <h2 className="text-center">Books List</h2>
20
21        <div className="text-center mb-4">
22          <button className="btn btn-success" onClick={() => navigate("/create")}>
23            Add New Book
24          </button>
25        </div>
26        <br/>
27        <br/>
28        {books && books.length > 0 ? (
29          <div className="table-responsive">
30            <table className="table table-bordered">
31              <thead className="thead-light">
32                <tr>
33                  <th scope="col">ID</th>
34                  <th scope="col">Title</th>
35                  <th scope="col">Author</th>
36                  <th scope="col">Actions</th>
37                </tr>
38              </thead>
39              <tbody>
40                {books.map((book) => (
41                  <tr key={book.id}>
42                    <td>{book.id}</td>
43                    <td>{book.title}</td>
44                    <td>{book.author}</td>
45                    <td>
46                      <button
47                        className="btn btn-warning me-2"
48                        onClick={() => navigate(`/update/${book.id}`)}
49                      >
50                        Update Book
51                      </button>
52                      <button
53                        className="btn btn-danger"
54                        onClick={() => navigate(`/delete/${book.id}`)}
55                      >
56                        Delete Book
57                      </button>
58                    </td>
59                  </tr>
60                ))}
61              </tbody>
62            </table>
63          </div>
64        ) : (
65          <p>No books available. Please add some!</p>
66        )}
67      </div>
68    );
69  };

```

Screenshot:

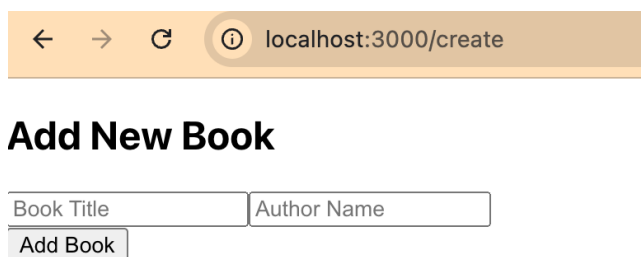


II. Add a New Book

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.

Screenshot:





Books Management App

Books List

Add New Book

ID	Title	Author	Actions	
1	Harry Potter	JK Rowling	Update Book	Delete Book
2	Psychology of Money	Morgan Housel	Update Book	Delete Book

Code: CreateBook.jsx

```
src > components > CreateBook.jsx > CreateBook
1  import React, { useState, useContext } from "react";
2  import { useNavigate } from "react-router-dom";
3  import { BookContext } from "../context/BookContext";
4
5  const CreateBook = () => {
6    const [title, setTitle] = useState("");
7    const [author, setAuthor] = useState("");
8    const { addBook } = useContext(BookContext);
9    const navigate = useNavigate();
10
11    const handleSubmit = (e) => {
12      e.preventDefault();
13      addBook({ title, author });
14      navigate("/");
15    };
16
17    return (
18      <div className="container mt-4">
19        <h2 className="text-center">Add New Book</h2>
20        <form onSubmit={handleSubmit} className="w-50 mx-auto">
21          <div className="mb-3">
22            <input type="text" className="form-control" placeholder="Book Title"
23              value={title} onChange={(e) => setTitle(e.target.value)} required />
24            <input type="text" className="form-control" placeholder="Author Name"
25              value={author} onChange={(e) => setAuthor(e.target.value)} required />
26          </div>
27          <button type="submit" className="btn btn-success w-100">Add Book</button>
28        </form>
29      </div>
30    );
31  };
32
33  export default CreateBook;
```

III. Update Book

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.

Code: UpdateBook.jsx

```
c > components > UpdateBook.jsx > UpdateBook
1  import React, { useState, useContext } from "react";
2  import { useParams, useNavigate } from "react-router-dom";
3  import { BookContext } from "../context/BookContext";
4
5  const UpdateBook = () => {
6    const { id } = useParams();
7    const navigate = useNavigate();
8    const { books, updateBook } = useContext(BookContext);
9    const book = books.find((b) => b.id === parseInt(id));
10
11    const [title, setTitle] = useState(book ? book.title : "");
12    const [author, setAuthor] = useState(book ? book.author : "");
13
14    const handleSubmit = (e) => {
15      e.preventDefault();
16      updateBook(parseInt(id), { id: parseInt(id), title, author });
17      navigate("/");
18    };
19
20    return (
21      <div className="container mt-4">
22        <h2 className="text-center">Update Book</h2>
23        <form onSubmit={handleSubmit} className="w-50 mx-auto">
24          <div className="mb-3">
25            <input type="text" className="form-control" placeholder="Enter new book's title"
26              value={title} onChange={(e) => setTitle(e.target.value)} required />
27            <input type="text" className="form-control" placeholder="Enter new book's author"
28              value={author} onChange={(e) => setAuthor(e.target.value)} required />
29          </div>
30          <button type="submit" className="btn btn-warning w-100">Update Book</button>
31        </form>
32      </div>
33    );
34  };
35
36  export default UpdateBook;
```

Screenshot:



Update Book

Harry Potter	JK Rowling
<input type="button" value="Update Book"/>	



Books Management App

Books List

ID	Title	Author	Actions	
1	Harry Pott	JK Rowli	<input type="button" value="Update Book"/>	<input type="button" value="Delete Book"/>
2	Psychology of Money	Morgan Housel	<input type="button" value="Update Book"/>	<input type="button" value="Delete Book"/>

IV. Delete the Book

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.

Code: DeleteBook.jsx

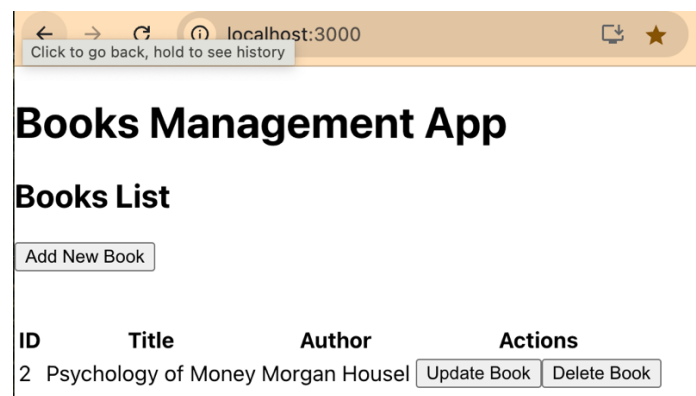
```
1 import React, { useContext } from "react";
2 import { useParams, useNavigate } from "react-router-dom";
3 import { BookContext } from "../context/BookContext";
4
5 const DeleteBook = () => {
6   const { id } = useParams();
7   const navigate = useNavigate();
8   const { deleteBook } = useContext(BookContext);
9
10  const handleDelete = (e) => {
11    e.preventDefault();
12    deleteBook(parseInt(id));
13    navigate("/");
14  };
15
16  return (
17    <div className="container mt-4">
18      <h2 className="text-center">Are you sure you want to delete this book?</h2>
19      <button onClick={handleDelete} className="w-50 mx-auto">Delete Book</button>
20      <button onClick={() => navigate("/")} className="w-50 mx-auto">Cancel</button>
21    </div>
22  );
23 };
24
25 export default DeleteBook;
```

Screenshot:



Are you sure you want to delete this book?

Delete Book Cancel



- V. For all the above questions make sure to:
- Pass props for the Create, Update, and Delete components
 - Use hooks like useState and useEffect wherever necessary
 - User React-Router-Dom for routing.

Code: App.js

```
src > JS App.js > ...
1 //import logo from './logo.svg';
2 import React from "react";
3 import { BrowserRouter as Router, Route, Routes } from "react-router-dom";
4 import Home from "./components/Home";
5 import CreateBook from "./components/CreateBook";
6 import UpdateBook from "./components/UpdateBook";
7 import DeleteBook from "./components/DeleteBook";
8 import { BookProvider } from "./context/BookContext";
9 import './App.css';
10
11 const App = () => {
12   return (
13     <BookProvider>
14       <Router>
15         <Routes>
16           <Route path="/" element={<Home />} />
17           <Route path="/create" element={<CreateBook />} />
18           <Route path="/update/:id" element={<UpdateBook />} />
19           <Route path="/delete/:id" element={<DeleteBook />} />
20         </Routes>
21       </Router>
22     </BookProvider>
23   );
24 };
25
26 export default App;
```

2. MySQL

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)

The screenshot shows a REST client interface with the following details:

- URL:** http://localhost:3000/api/books
- Method:** POST
- Body (raw):**

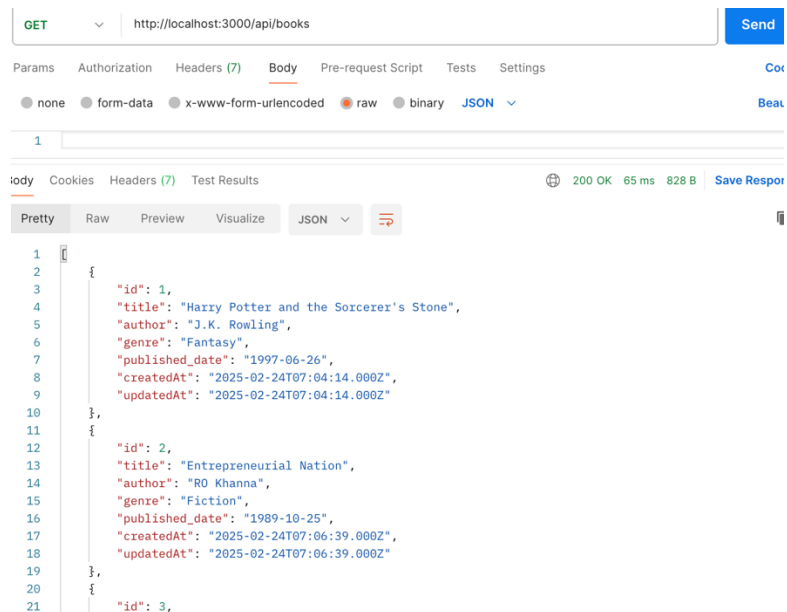
```
{
  "title": "Harry Potter and the Sorcerer's Stone",
  "author": "J.K. Rowling",
  "genre": "Fantasy",
  "published_date": "1997-06-26"
}
```
- Response (raw):**

```
{
  "id": 1,
  "title": "Harry Potter and the Sorcerer's Stone",
  "author": "J.K. Rowling",
  "genre": "Fantasy",
  "published_date": "1997-06-26T00:00:00.000Z",
  "updatedAt": "2025-02-24T07:04:14.815Z",
  "createdAt": "2025-02-24T07:04:14.815Z"
}
```

Aashima Taneja
017536727

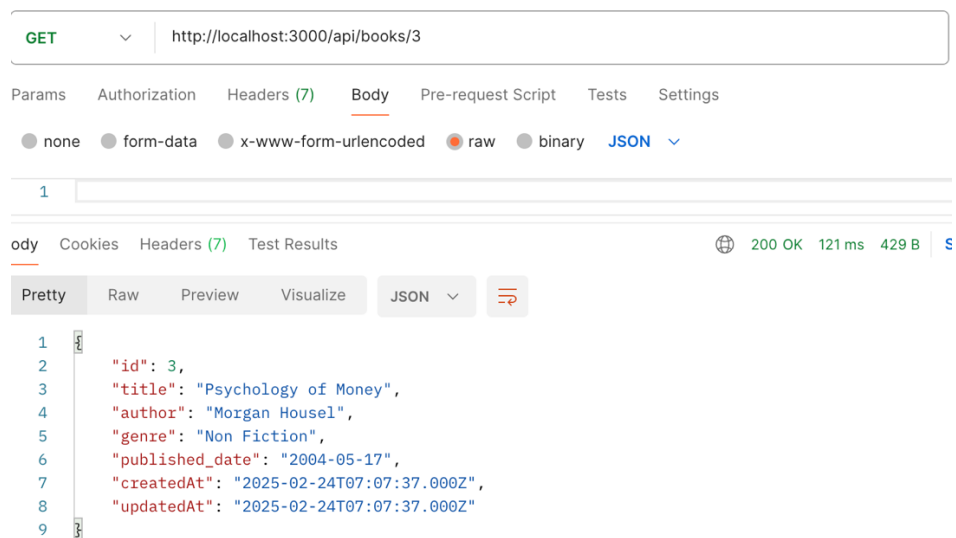
- **View all books (GET)**

Screenshot:



- **View a book by ID (GET)**

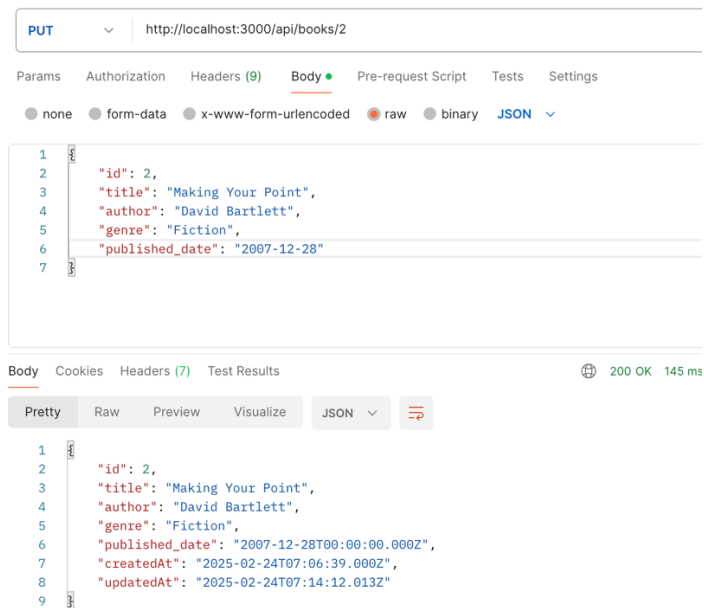
Screenshot:



Aashima Taneja
017536727

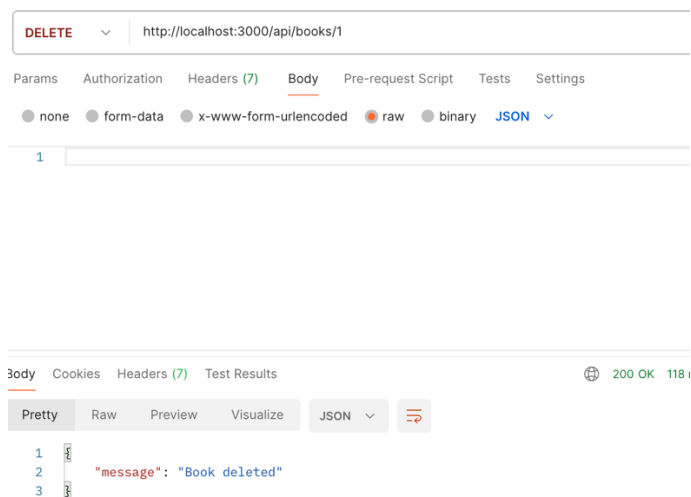
○ Update book details (PUT)

Screenshot:

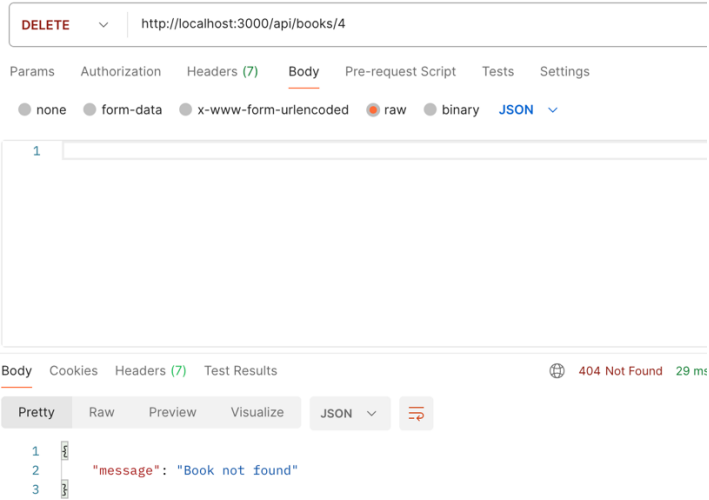


○ Delete a book (DELETE)

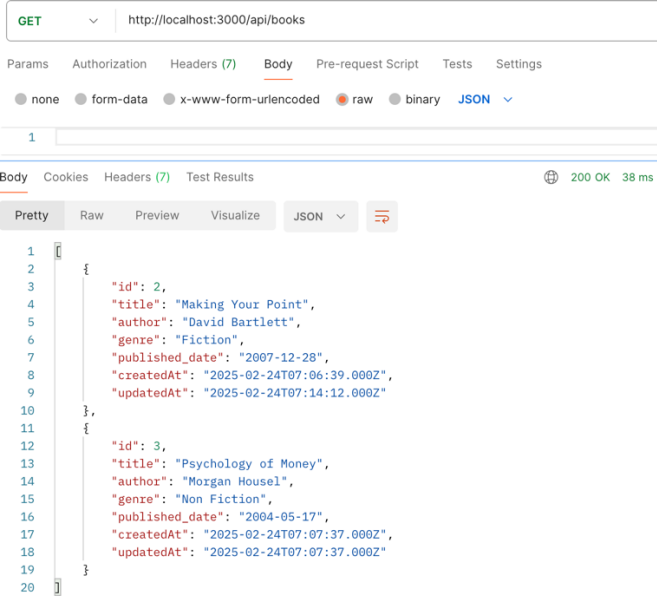
Screenshot:



Screenshot:



After deleting book with Id-1
Screenshot:



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

Database Screenshots:

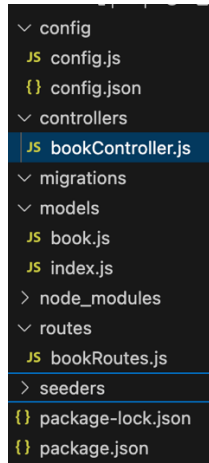
```
create database book_db;
use book_db;
create table books(
  id int auto_increment primary key,
  title varchar(255) not null,
  author varchar(255) not null,
  genre varchar(100),
  published_date Date
);
```

Field	Type	Null	Key	Default	Extra
id	int	NO	PRI	<code>NULL</code>	auto_increment
title	varchar(255)	NO		<code>NULL</code>	
author	varchar(255)	NO		<code>NULL</code>	
genre	varchar(100)	YES		<code>NULL</code>	
published_date	date	YES		<code>NULL</code>	
createdAt	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED
updatedAt	datetime	YES		CURRENT_TIMESTAMP	DEFAULT_GENERATED on update CURRENT...

Aashima Taneja
017536727

Folder structure:

Screenshot:



A screenshot of a file explorer window showing a project folder structure. The folders are expanded to show their contents. The 'controllers' folder is highlighted with a blue background. The files are listed as follows:

- config
 - config.js
 - config.json
- controllers
 - bookController.js
- migrations
- models
 - book.js
 - index.js
- node_modules
- routes
 - bookRoutes.js
- seeders
- package-lock.json
- package.json