

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>

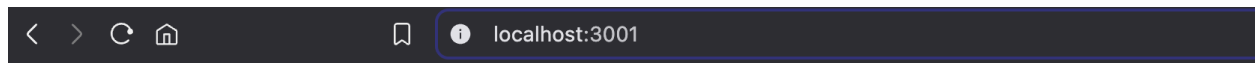
Q1. React (10 Points)

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 '/'.



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

```
# Update.css U # Create.css U JS App.js U X JS index.js U JS app.js M
hw4-template > src > JS App.js > [?] default
8   const App = () => {
38     <Router>
39       <div className="App">
40         <Routes>
41           <Route path="/" element={<Home books={books} onError={setError} />} />
42           <Route path="/create" element={<CreateBook onAddBook={setBooks} />} />
```

```
U # Update.css U # Create.css U JS App.js U JS app.js M Home.jsx U X
hw4-template > src > components > Home > Home.jsx > [?] Home
1   import React, { useState, useEffect } from 'react';
2   import { Link } from 'react-router-dom';
3   import './Home.css';
4
5   const Home = () => {
6     const [books, setBooks] = useState([]);
7
8     useEffect(() => {
9       // Fetch books from API
10      fetch('http://localhost:3000/api/books')
11        .then(response => response.json())
12        .then(data => setBooks(data))
13        .catch(error => console.error('Error:', error));
14    }, []);
15
16    return (
17      <div className="home-container">
18        <h1>Book Management System</h1>
19        <Link to="/create" className="add-button">Add New Book</Link>
20
21        <div className="books-grid">
22          {books.map(book => (
23            <div key={book.id} className="book-card">
24              <h3>{book.title}</h3>
25              <p>Author: {book.author}</p>
26              <p>ISBN: {book.isbn}</p>
27              <div className="book-actions">
28                <Link to={`/${update}/${book.id}`} className="edit-button">Update</Link>
29                <Link to={`/${delete}/${book.id}`} className="delete-button">Delete</Link>
30              </div>
31            </div>
32          ))}
33        </div>
34      </div>
35    );
36  };
37
38  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

Add Book

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 Book

Author: Test Author

ISBN: 987654

Update

Delete

```
CreateBook.jsx U JS App.js U X
hw4-template > src > JS App.js > ...
8   const App = () => {
34   if (loading) return <div>Loading books...</div>;
35   if (error) return <div>Error: {error}</div>;
36
37   return (
38     <Router>
39     <div className="App">
40       <Routes>
41         <Route path="/" element={<Home books={books} onError={setError} />} />
42         <Route path="/create" element={<CreateBook onAddBook={setBooks} />} />
43         <Route
```

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

```
1  import React, { useState } from 'react';
2  import { useNavigate } from 'react-router-dom';
3  import './CreateBook.css';
4
5  const CreateBook = ({ onAddBook }) => {
6    const [title, setTitle] = useState('');
7    const [author, setAuthor] = useState('');
8    const [isbn, setIsbn] = useState('');
9    const [error, setError] = useState(null);
10   const navigate = useNavigate();
11
12   const handleSubmit = async (e) => {
13     e.preventDefault();
14     setError(null);
15
16     try {
17       const response = await fetch('http://localhost:3000/api/books', {
18         method: 'POST',
19         headers: {
20           'Content-Type': 'application/json',
21         },
22         body: JSON.stringify({
23           title,
24           author,
25           isbn
26         }),
27       });
28
29       if (!response.ok) {
30         throw new Error('Network response was not ok');
31       }
32
33       const newBook = await response.json();
34       onAddBook(newBook);
35       navigate('/');
36     } catch (error) {
37       setError('Failed to create book. Please try again.');
```

```
42   return (  
43     <div className="create-book-container">  
44       <h2>Add New Book</h2>  
45       {error && <div className="error-message">{error}</div>}  
46       <form onSubmit={handleSubmit}>  
47         <div className="form-group">  
48           <label htmlFor="title">Book Title:</label>  
49           <input  
50             type="text"  
51             id="title"  
52             value={title}  
53             onChange={(e) => setTitle(e.target.value)}  
54             required  
55           />  
56         </div>  
57         <div className="form-group">  
58           <label htmlFor="author">Author Name:</label>  
59           <input  
60             type="text"  
61             id="author"  
62             value={author}  
63             onChange={(e) => setAuthor(e.target.value)}  
64             required  
65           />  
66         </div>  
67         <div className="form-group">  
68           <label htmlFor="isbn">ISBN:</label>  
69           <input  
70             type="text"  
71             id="isbn"  
72             value={isbn}  
73             onChange={(e) => setIsbn(e.target.value)}  
74           />  
75         </div>  
76         <button type="submit" className="submit-button">Add Book</button>  
77       </form>  
78     </div>  
79   );  
80 };  
81
```

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.



Update Book

Book Title:

Author Name:

Update Book

Cancel



Update Book

Book Title:

Author Name:

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

Updated Test Book

Author: Updated Test Author

ISBN: 987654

Update

Delete

UpdateBook.jsx U X

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

```
1  import React, { useState, useEffect } from 'react';
2  import { useNavigate, useParams } from 'react-router-dom';
3  import './UpdateBook.css';
4
5  const UpdateBook = ({ onUpdateBook }) => {
6    const [title, setTitle] = useState('');
7    const [author, setAuthor] = useState('');
8    const [loading, setLoading] = useState(true);
9    const [error, setError] = useState(null);
10   const navigate = useNavigate();
11   const { id } = useParams();
12
13   useEffect(() => {
14     fetchBook();
15   }, [id]);
16
17   const fetchBook = async () => {
18     try {
19       setLoading(true);
20       const response = await fetch(`http://localhost:3000/api/books/${id}`);
21       if (response.ok) {
22         const book = await response.json();
23         setTitle(book.title);
24         setAuthor(book.author);
25       } else {
26         throw new Error('Book not found');
27       }
28     } catch (error) {
29       setError(error.message);
30       console.error('Error fetching book:', error);
31     } finally {
32       setLoading(false);
33     }
34   };
35
```

UpdateBook.jsx U ×

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

```
5  const UpdateBook = ({ onUpdateBook }) => {
36  const handleSubmit = async (e) => {
37    e.preventDefault();
38    try {
39      const response = await fetch(`http://localhost:3000/api/books/${id}`, {
40        method: 'PUT',
41        headers: {
42          'Content-Type': 'application/json',
43        },
44        body: JSON.stringify({ title, author }),
45      });
46
47      if (response.ok) {
48        const updatedBook = await response.json();
49        onUpdateBook && onUpdateBook(updatedBook);
50        navigate('/');
51      } else {
52        throw new Error('Failed to update book');
53      }
54    } catch (error) {
55      setError(error.message);
56      console.error('Error:', error);
57    }
58  };
59
60  if (loading) return <div>Loading...</div>;
61  if (error) return <div className="error-message">{error}</div>;
62
```

```
UpdateBook.jsx U X
hw4-template > src > components > Update > UpdateBook.jsx > UpdateBook > fetchBook
5  const UpdateBook = ({ onUpdateBook }) => {
63    return (
64      <div className="update-book-container">
65        <h2>Update Book</h2>
66        <form onSubmit={handleSubmit}>
67          <div className="form-group">
68            <label htmlFor="title">Book Title:</label>
69            <input
70              type="text"
71              id="title"
72              value={title}
73              onChange={(e) => setTitle(e.target.value)}
74              required
75            />
76          </div>
77          <div className="form-group">
78            <label htmlFor="author">Author Name:</label>
79            <input
80              type="text"
81              id="author"
82              value={author}
83              onChange={(e) => setAuthor(e.target.value)}
84              required
85            />
86          </div>
87          <div className="button-group">
88            <button type="submit" className="submit-button">Update Book</button>
89            <button type="button" onClick={() => navigate('/')}>Cancel</button>
90          </div>
91        </form>
92      </div>
93    );
94  };
95
96  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.

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

Updated Test Book

Author: Updated Test Author
ISBN: 987654

Update

Delete

test

Author: test
ISBN: 123

Update

Delete

test

Author: test
ISBN: 1234

Update

Delete

Delete Book

test

Author: test

Are you sure you want to delete this book?

Delete 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

Updated Test Book

Author: Updated Test Author
ISBN: 987654

Update

Delete

test

Author: test
ISBN: 123

Update

Delete

CreateBook.jsx U

DeleteBook.jsx U X

JS book.controller.js M

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

```
1  import React, { useState, useEffect } from 'react';
2  import { useNavigate, useParams } from 'react-router-dom';
3  import './DeleteBook.css';
4
5  const DeleteBook = ({ onDelete }) => {
6    const [book, setBook] = useState(null);
7    const [loading, setLoading] = useState(true);
8    const [error, setError] = useState(null);
9    const navigate = useNavigate();
10   const { id } = useParams();
11
12   useEffect(() => {
13     fetchBookDetails();
14   }, [id]);
15
16   const fetchBookDetails = async () => {
17     try {
18       const response = await fetch(`http://localhost:3000/api/books/${id}`);
19       if (response.ok) {
20         const data = await response.json();
21         setBook(data);
22       } else {
23         throw new Error('Book not found');
24       }
25     } catch (error) {
26       setError(error.message);
27       console.error('Error fetching book:', error);
28     } finally {
29       setLoading(false);
30     }
31   };
32
33   const handleDelete = async () => {
34     try {
35       const response = await fetch(`http://localhost:3000/api/books/${id}`, {
36         method: 'DELETE',
37       });
38     }
```

```
CreateBook.jsx U DeleteBook.jsx U X JS book.controller.js M
hw4-template > src > components > Delete > DeleteBook.jsx
39     if (response.ok) {
40         onDelete && onDelete(id);
41         navigate('/');
42     } else {
43         throw new Error('Failed to delete book');
44     }
45 } catch (error) {
46     setError(error.message);
47     console.error('Error:', error);
48 }
49 };
50
51 if (loading) return <div>Loading...</div>;
52 if (error) return <div className="error-message">{error}</div>;
53
54 return (
55     <div className="delete-book-container">
56         <h2>Delete Book</h2>
57         {book && (
58             <div className="book-details">
59                 <h3>{book.title}</h3>
60                 <p>Author: {book.author}</p>
61                 <p className="warning-text">Are you sure you want to delete this book?</p>
62             </div>
63         )}
64         <div className="button-group">
65             <button onClick={handleDelete} className="delete-button">Delete Book</button>
66             <button onClick={() => navigate('/')} className="cancel-button">Cancel</button>
67         </div>
68     </div>
69 );
70 };
71
72 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)

```
Home.jsx U X
hw4-template > src > components > Home > Home.jsx
1  import React, { useState, useEffect } from 'react';
2  import { Link } from 'react-router-dom';
3  import './Home.css';
4
5  const Home = () => {
6    const [books, setBooks] = useState([]);
7
8    useEffect(() => {
9      // Fetch books from API
10     fetch('http://localhost:3000/api/books')
11       .then(response => response.json())
12       .then(data => setBooks(data))
13       .catch(error => console.error('Error:', error));
14   }, []);
15
16   return (
17     <div className="home-container">
18       <h1>Book Management System</h1>
19       <Link to="/create" className="add-button">Add New Book</Link>
20
21       <div className="books-grid">
22         {books.map(book => (
23           <div key={book.id} className="book-card">
24             <h3>{book.title}</h3>
25             <p>Author: {book.author}</p>
26             <p>ISBN: {book.isbn}</p>
27             <div className="book-actions">
28               <Link to={` /update/${book.id}`} className="edit-button">Update</Link>
29               <Link to={` /delete/${book.id}`} className="delete-button">Delete</Link>
30             </div>
31           </div>
32         ))}
33       </div>
34     </div>
35   );
36 };
37
38 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)

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

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

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "title": "The Great Gatsby",
3   "author": "F. Scott Fitzgerald",
4   "isbn": "978-0743273565",
5   "publishedYear": 1925
6 }
```

Body Cookies Headers (7) Test Results **201 Created**

{} JSON ▾ Preview Visualize ▾

```
1 {
2   "id": 2,
3   "title": "The Great Gatsby",
4   "author": "F. Scott Fitzgerald",
5   "isbn": "978-0743273565",
6   "publishedYear": 1925,
7   "updatedAt": "2025-02-20T03:51:27.523Z",
8   "createdAt": "2025-02-20T03:51:27.523Z"
9 }
```

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

POST

http://localhost:3000/api/books

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {  
2   "title": "To Kill a Mockingbird",  
3   "author": "Harper Lee",  
4   "isbn": "978-0446310789",  
5   "publishedYear": 1960  
6 }
```

Body Cookies Headers (7) Test Results ↻

201 Created

{ } JSON ▾ ▶ Preview 🖼 Visualize ▾

```
1 {  
2   "id": 3,  
3   "title": "To Kill a Mockingbird",  
4   "author": "Harper Lee",  
5   "isbn": "978-0446310789",  
6   "publishedYear": 1960,  
7   "updatedAt": "2025-02-20T03:52:15.544Z",  
8   "createdAt": "2025-02-20T03:52:15.544Z"  
9 }
```

- View all books (GET)

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

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

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (7) Test Results 200 OK

{ } JSON Preview Visualize

```
1  [
2    {
3      "id": 2,
4      "title": "The Great Gatsby",
5      "author": "F. Scott Fitzgerald",
6      "isbn": "978-0743273565",
7      "publishedYear": 1925,
8      "createdAt": "2025-02-20T03:51:27.000Z",
9      "updatedAt": "2025-02-20T03:51:27.000Z"
10   },
11   {
12     "id": 3,
13     "title": "To Kill a Mockingbird",
14     "author": "Harper Lee",
15     "isbn": "978-0446310789",
16     "publishedYear": 1960,
17     "createdAt": "2025-02-20T03:52:15.000Z",
18     "updatedAt": "2025-02-20T03:52:15.000Z"
19   }
20 ]
```

- View a book by ID (GET)

HTTP 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 ⌵

```
1  {
2    "id": 2,
3    "title": "The Great Gatsby",
4    "author": "F. Scott Fitzgerald",
5    "isbn": "978-0743273565",
6    "publishedYear": 1925,
7    "createdAt": "2025-02-20T03:51:27.000Z",
8    "updatedAt": "2025-02-20T03:51:27.000Z"
9  }
```

- Update book details (PUT)


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


PUT <http://localhost:3000/api/books/2>

Params Authorization Headers (8) **Body** Scripts Settings

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▾

```
1 {
2   "title": "Updated Title",
3   "author": "Updated Author",
4   "isbn": "978-0743273565",
5   "publishedYear": 2024
6 }
```

Body Cookies Headers (7) Test Results  **200 OK**

{} JSON ▾ ▶ Preview  Visualize ▾

```
1 {
2   "id": 2,
3   "title": "Updated Title",
4   "author": "Updated Author",
5   "isbn": "978-0743273565",
6   "publishedYear": 2024,
7   "createdAt": "2025-02-20T03:51:27.000Z",
8   "updatedAt": "2025-02-20T03:53:42.994Z"
9 }
```

HTTP 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	Description
	Key	Value	Description

Body Cookies Headers (7) Test Results

200 OK

{ } JSON

Preview

Visualize

```
1 {
2   "id": 2,
3   "title": "Updated Title",
4   "author": "Updated Author",
5   "isbn": "978-0743273565",
6   "publishedYear": 2024,
7   "createdAt": "2025-02-20T03:51:27.000Z",
8   "updatedAt": "2025-02-20T03:53:42.000Z"
9 }
```


- Delete a book (DELETE)

HTTP 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	Description
	Key	Value	Description

Body Cookies Headers (4) Test Results ↺ **204 No Content**

Raw ▼ ▶ Preview 🔗 Visualize ▼

1

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

GET

http://localhost:3000/api/books

Params Authorization Headers (6) Body Scripts Settings

Query Params

	Key	Value	Description
	Key	Value	Description

Body Cookies Headers (7) Test Results

200 OK

{ } JSON

Preview

Visualize

```
1  [
2    {
3      "id": 3,
4      "title": "To Kill a Mockingbird",
5      "author": "Harper Lee",
6      "isbn": "978-0446310789",
7      "publishedYear": 1960,
8      "createdAt": "2025-02-20T03:52:15.000Z",
9      "updatedAt": "2025-02-20T03:52:15.000Z"
10   }
11 ]
```

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 books1
```

```
    -> ;
```

```
ERROR 1146 (42S02): Table 'book_db.books1' doesn't exist
```

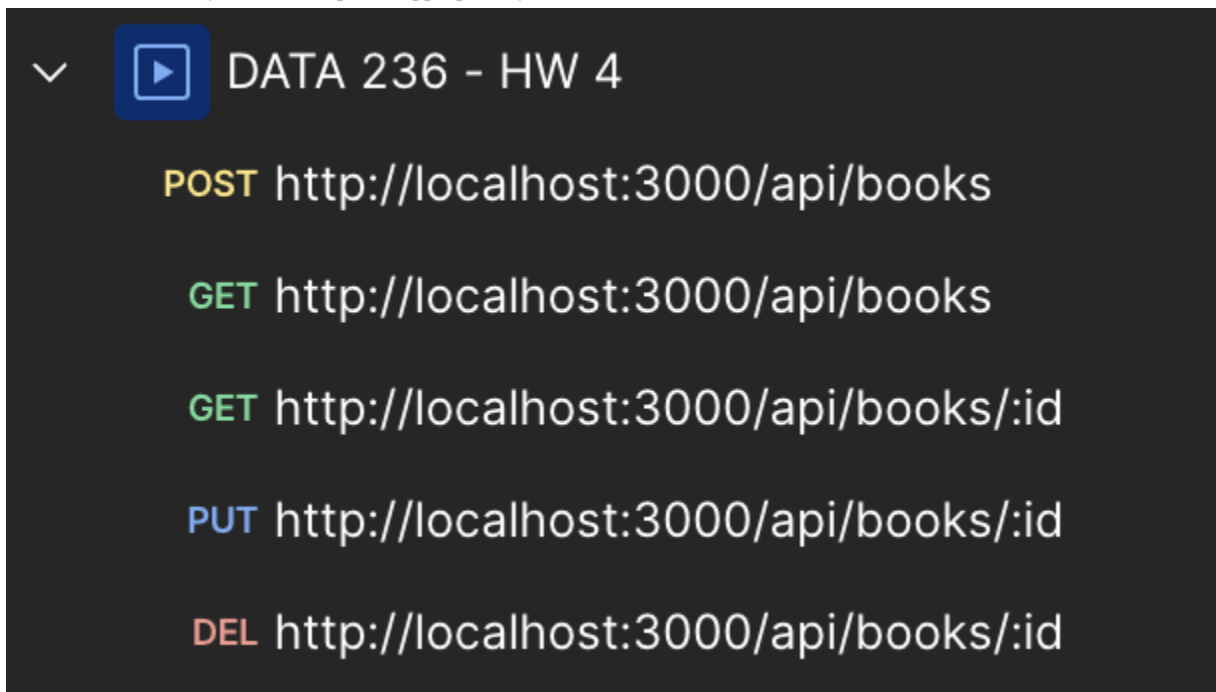
```
mysql> describe books;
```

```
+-----+-----+-----+-----+-----+-----+
| 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)
```

mysql-nodajs > JS book.model.js > ...

```
1  const { DataTypes } = require('sequelize');
2  const sequelize = require('./config/database');
3
4  const Book = sequelize.define('Book', {
5    title: {
6      type: DataTypes.STRING,
7      allowNull: false
8    },
9    author: {
10     type: DataTypes.STRING,
11     allowNull: false
12   },
13   isbn: {
14     type: DataTypes.STRING,
15     unique: true
16   },
17   publishedYear: {
18     type: DataTypes.INTEGER
19   }
20 });
21
22 module.exports = Book;
```

You are free to name your API endpoints appropriately.



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.