DATA-236 Sec 21 & 71 - Distributed Systems for Data Engineering

HOMEWORK 5

Vimalanandhan Sivanandham

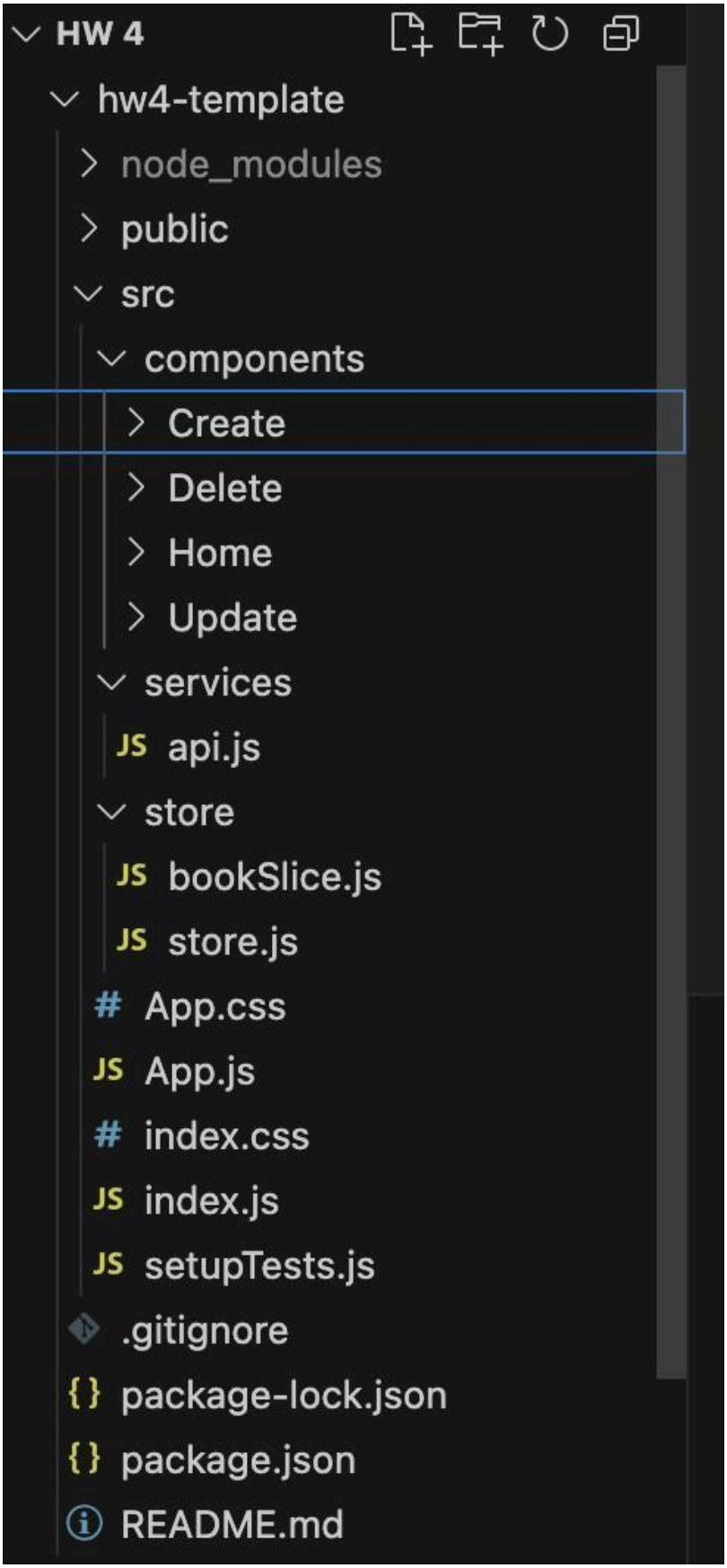
017596436

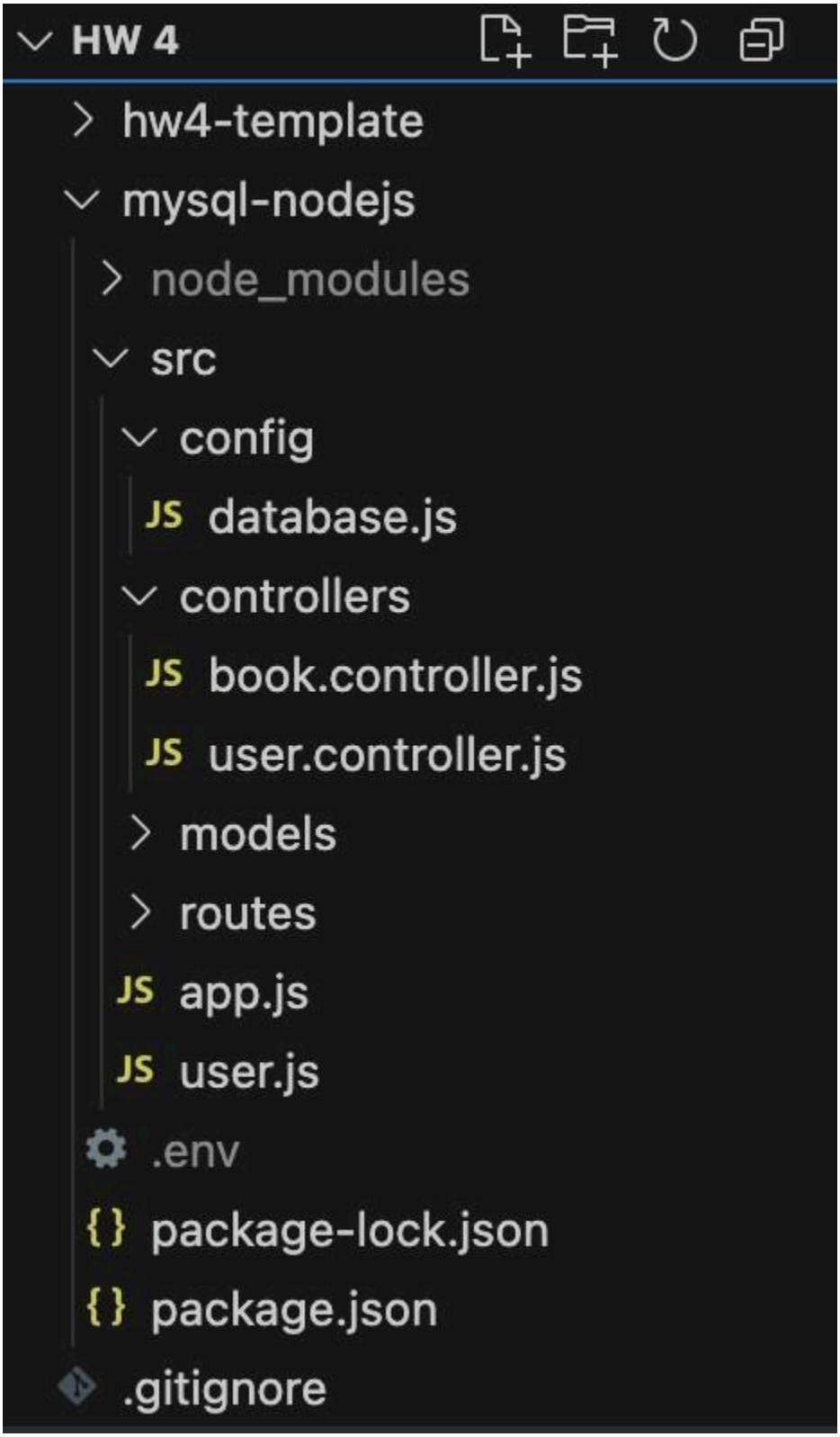
GitHub -

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

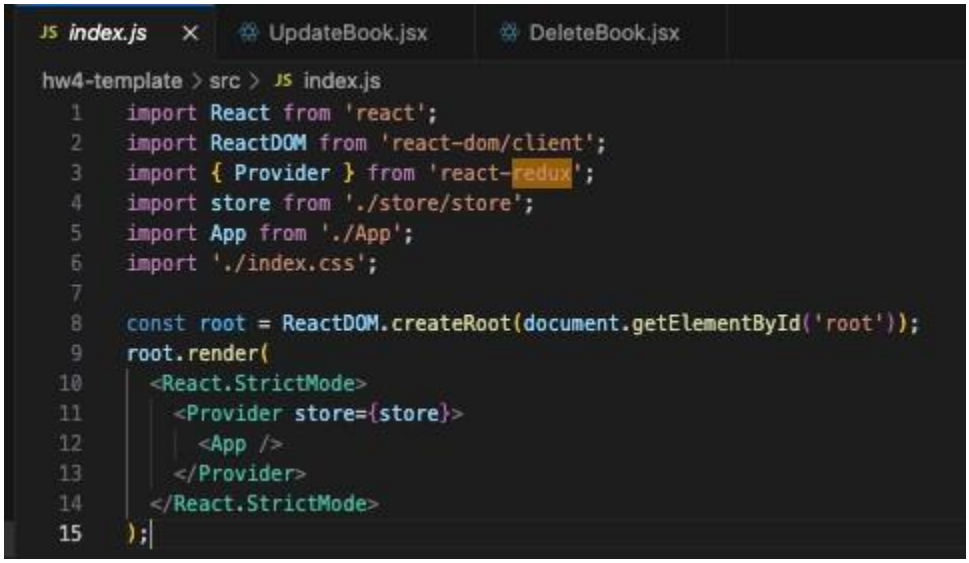
CreateBook.jsx | api.js | UpdateBook.jsx | DeleteBook.jsx

hw4-template > src > services > api.js

```
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.

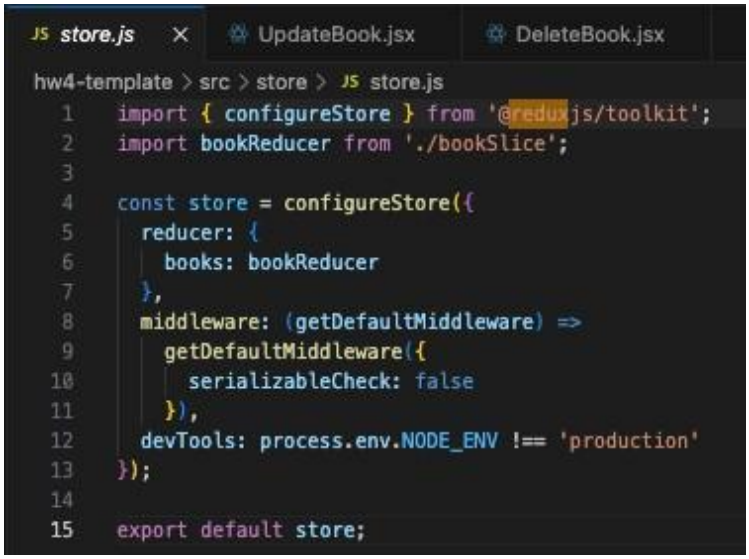
NOTE: I have used Reduxjs Toolkit



index.js | UpdateBook.jsx | DeleteBook.jsx

hw4-template > src > index.js

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import { Provider } from 'react-redux';
import store from './store/store';
import App from './App';
import './index.css';

const root = ReactDOM.createRoot(document.getElementById('root'));
root.render(
  <React.StrictMode>
    <Provider store={store}>
      <App />
    </Provider>
  </React.StrictMode>
);
```

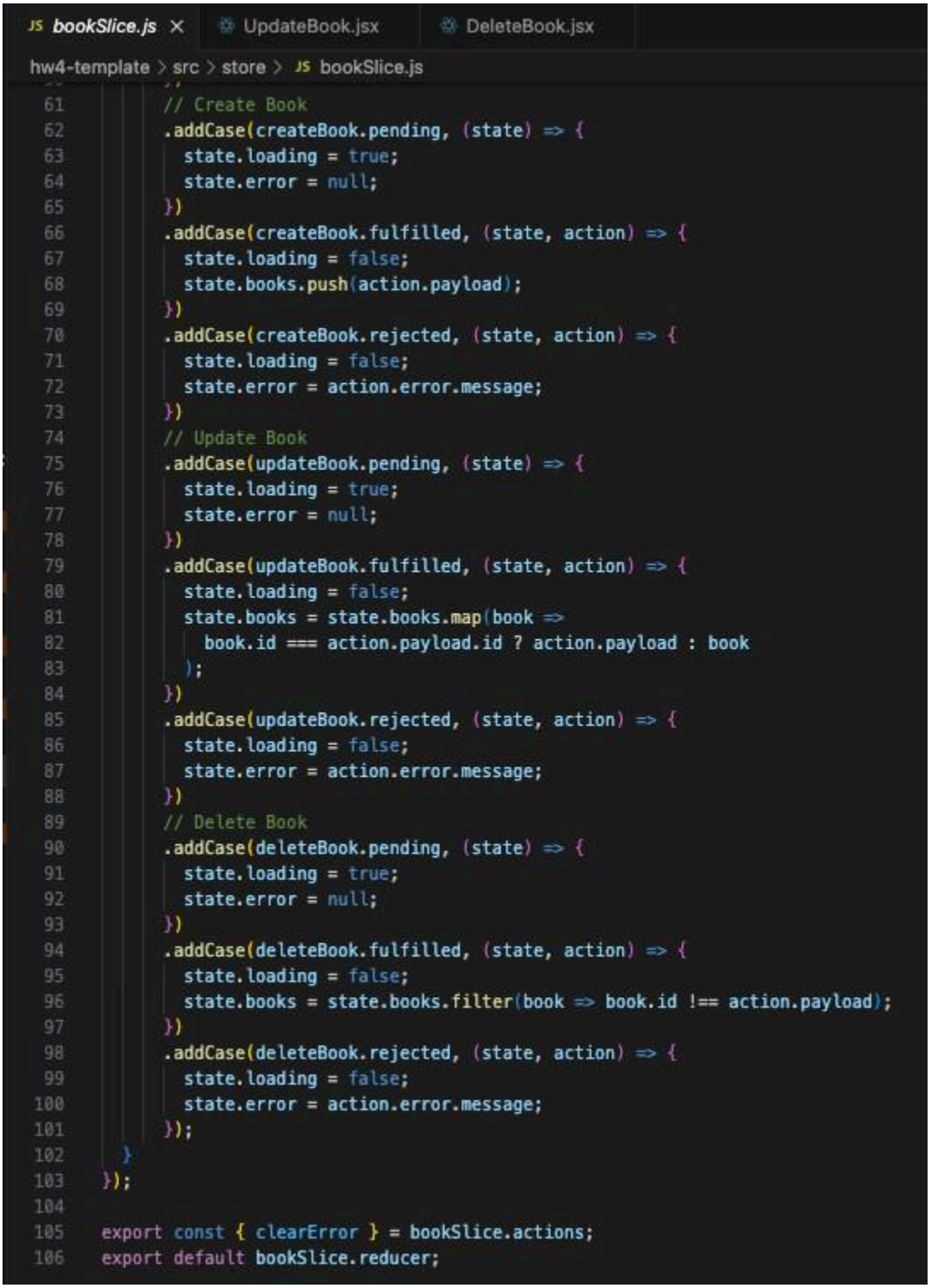


store.js | UpdateBook.jsx | DeleteBook.jsx

hw4-template > src > store > store.js

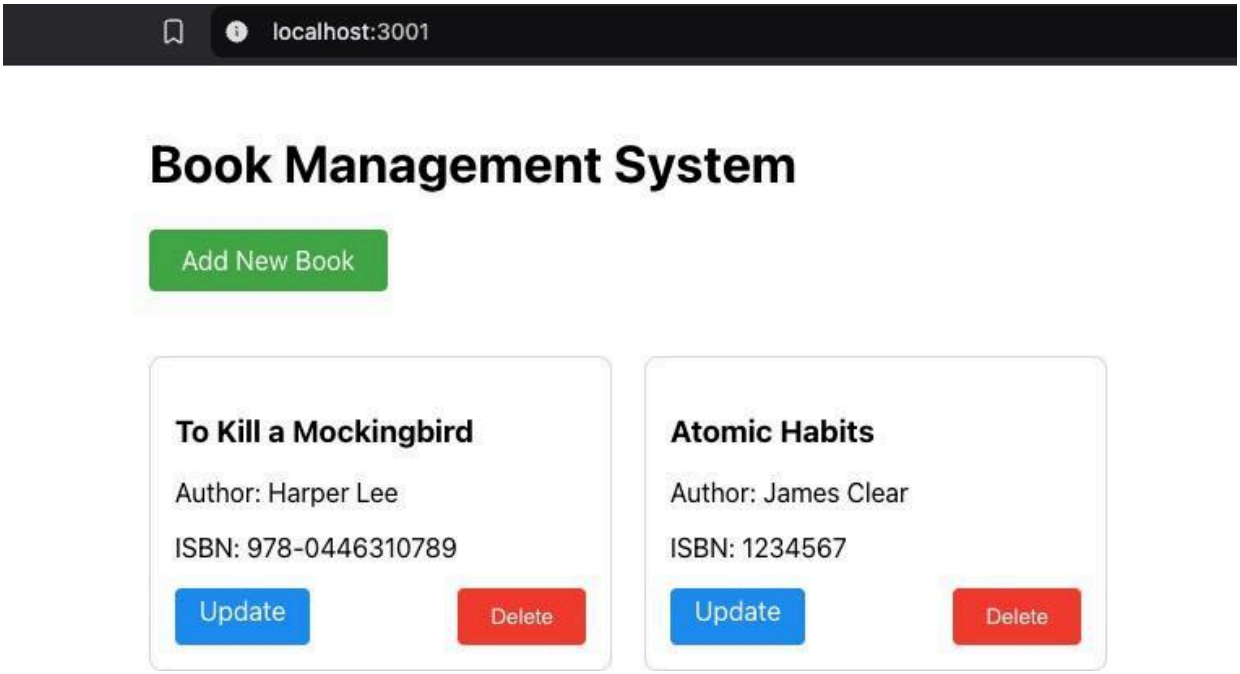
```
import { configureStore } from '@reduxjs/toolkit';
import bookReducer from './bookSlice';

const store = configureStore({
  reducer: {
    books: bookReducer
  },
  middleware: (getDefaultMiddleware) =>
    getDefaultMiddleware({
      serializableCheck: false
    }),
  devTools: process.env.NODE_ENV !== 'production'
});

export default store;
```


bookSlice.js
UpdateBook.jsx
DeleteBook.jsx
hw4-template > src > store > bookSlice.js
import { createSlice, createAsyncThunk } from '@reduxjs/toolkit
import * as api from '../services/api';
// Async Thunks
export const fetchBooks = createAsyncThunk(
'books/fetchBooks',
async () => {
return await api.getBooks();
}
);
export const createBook = createAsyncThunk(
'books/createBook',
async (bookData) => {
return await api.createBook(bookData);
}
);
export const updateBook = createAsyncThunk(
'books/updateBook',
async ({ id, bookData }) => {
return await api.updateBook(id, bookData);
}
);
export const deleteBook = createAsyncThunk(
'books/deleteBook',
async (id) => {
await api.deleteBook(id);
return id;
}
);
const bookSlice = createSlice({
name: 'books',
initialState: {
books: [],
loading: false,
error: null
},
reducers: {
clearError: (state) => {
state.error = null;
}
},
extraReducers: (builder) => {
builder
// Fetch Books
.addCase(fetchBooks.pending, (state) => {
state.loading = true;
state.error = null;
})
.addCase(fetchBooks.fulfilled, (state, action) => {
state.loading = false;
state.books = action.payload;
})
.addCase(fetchBooks.rejected, (state, action) => {
state.loading = false;
state.error = action.error.message;
})



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

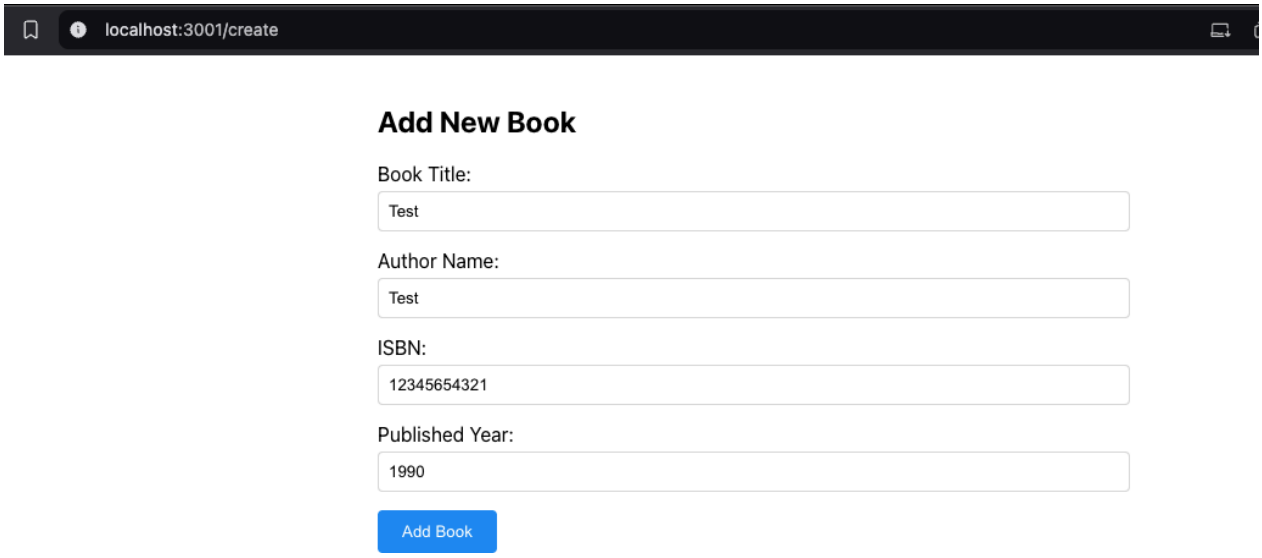
Home.jsx | UpdateBook.jsx | DeleteBook.jsx

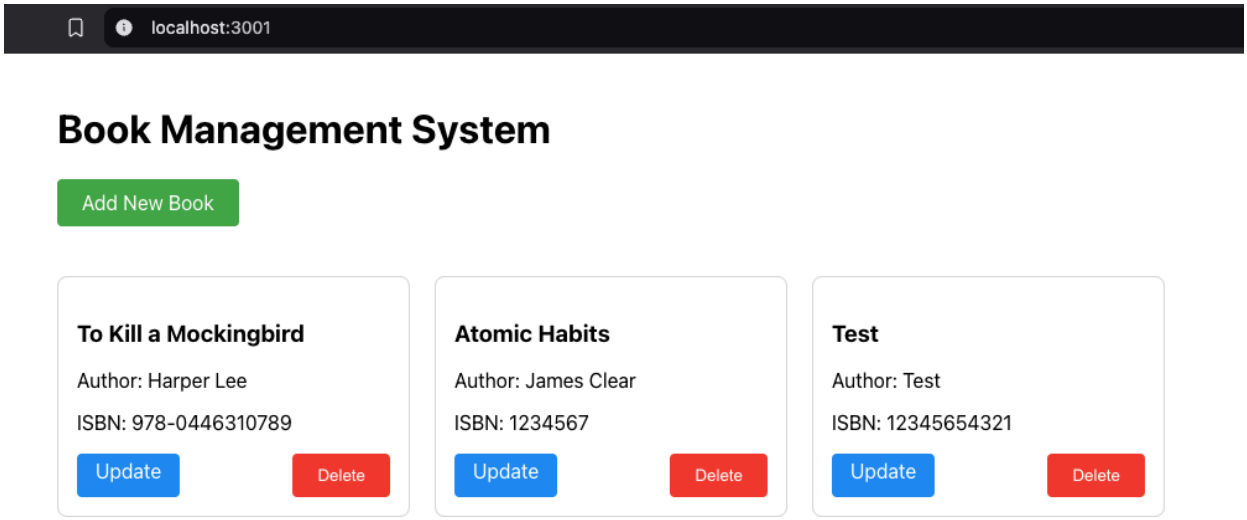
hw4-template > src > components > Home > Home.jsx

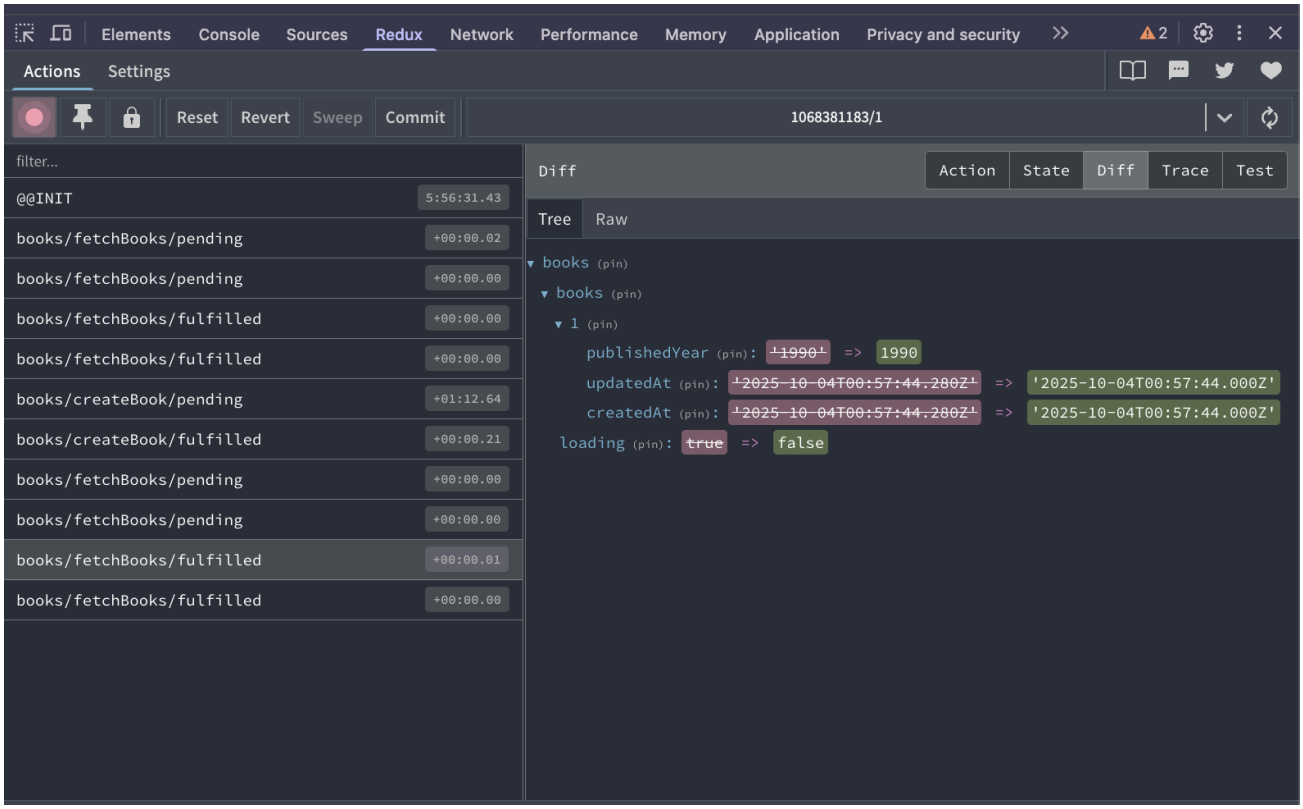
```jsx
import React, { useEffect } from 'react';
import { Link } from 'react-router-dom';
import { useDispatch, useSelector } from 'react-redux';
import { fetchBooks, deleteBook } from '../../store/bookSlice';
import './Home.css';

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.

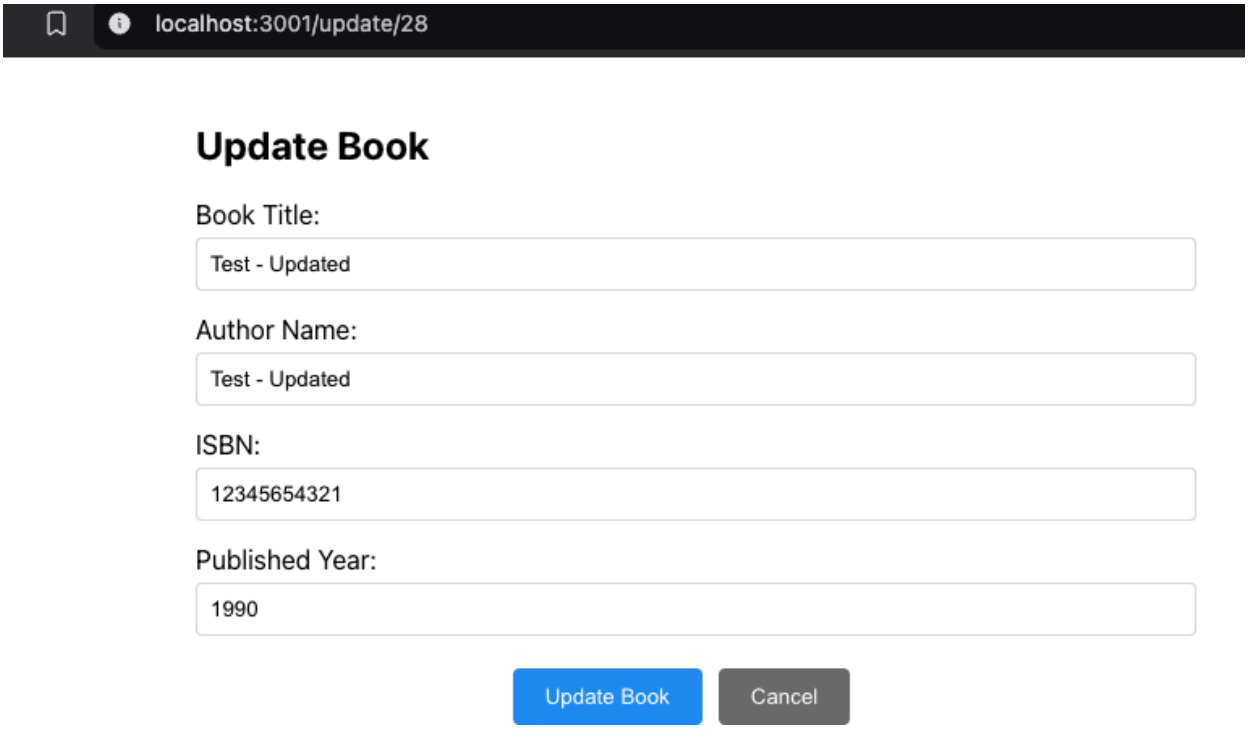


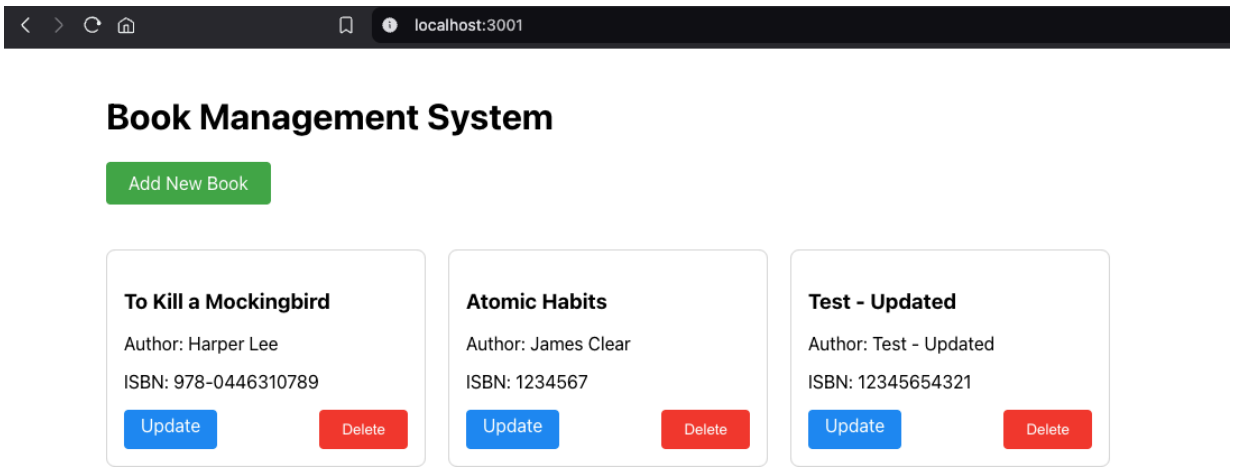


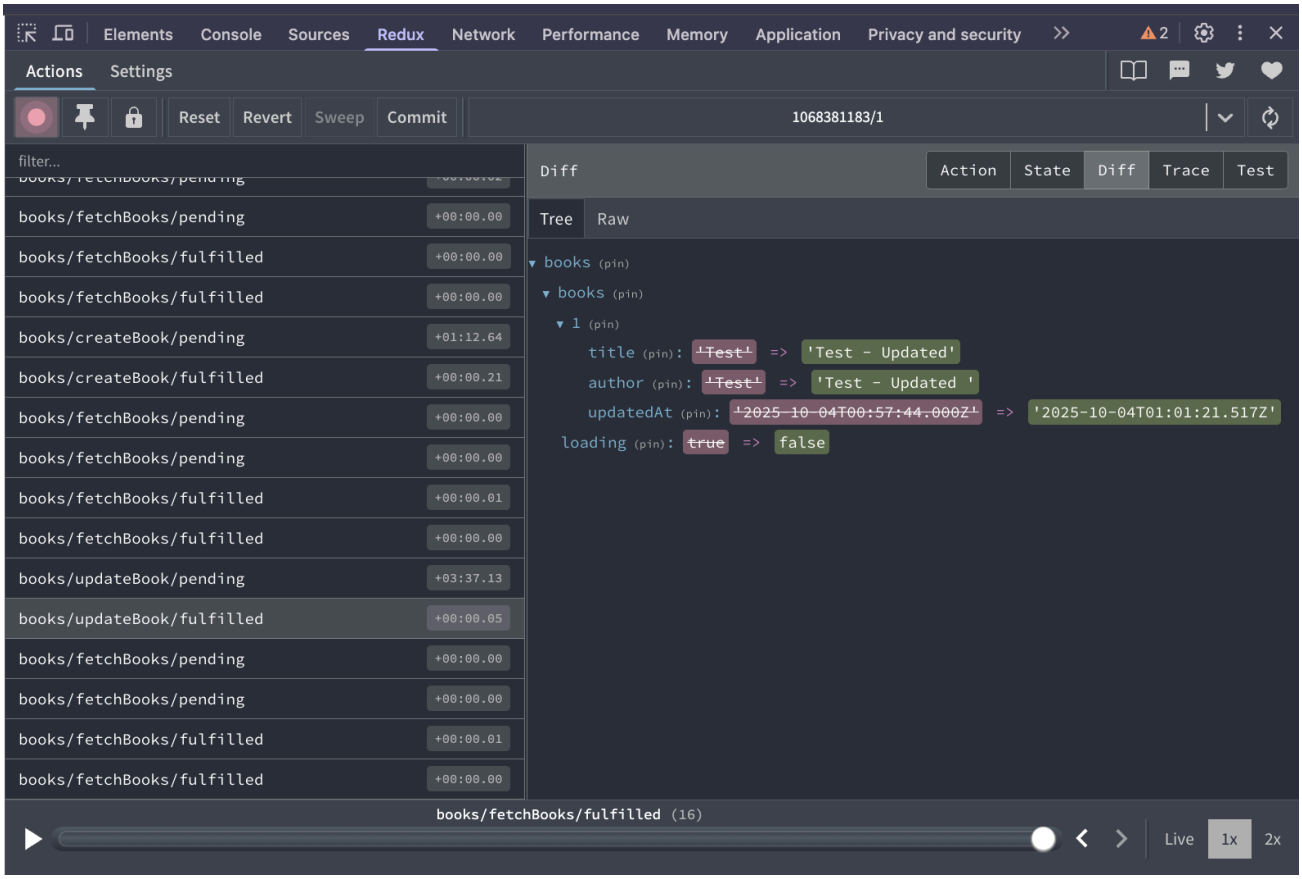

Elements
Console
Sources
Redux
Network
Performance
Memory
Application
Privacy and security
2
Actions
Settings
Reset
Revert
Sweep
Commit
1068381183/1
filter...
@@INIT 5:56:31.43
books/fetchBooks/pending +00:00.02
books/fetchBooks/pending +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/createBook/pending +01:12.64
books/createBook/fulfilled +00:00.21
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)
1 (pin)
publishedYear (pin): 1990 => 1990
updatedAt (pin): '2025-10-04T00:57:44.280Z' => '2025-10-04T00:57:44.000Z'
createdAt (pin): '2025-10-04T00:57:44.280Z' => '2025-10-04T00:57:44.000Z'
loading (pin): true => false

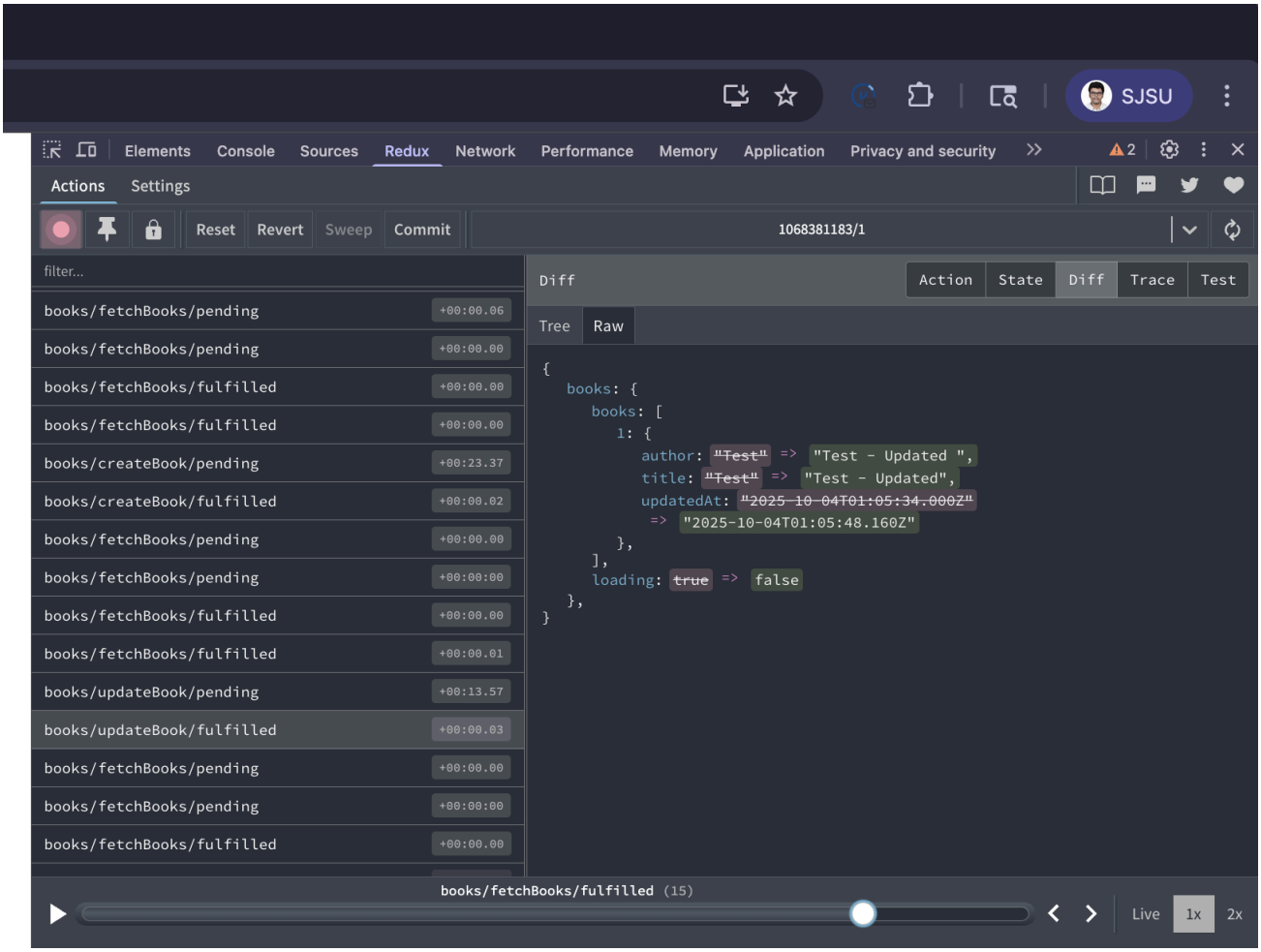
IV. Update Screen (2 points)

- 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
Network
Performance
Memory
Application
Privacy and security
Actions
Settings
Reset
Revert
Sweep
Commit
1068381183/1
filter...
books/fetchBooks/pending +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/createBook/pending +01:12.64
books/createBook/fulfilled +00:00.21
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/updateBook/pending +03:37.13
books/updateBook/fulfilled +00:00.05
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)
1 (pin)
title (pin): 'Test' => 'Test - Updated'
author (pin): 'Test' => 'Test - Updated '
updatedAt (pin): '2025-10-04T00:57:44.000Z' => '2025-10-04T01:01:21.517Z'
loading (pin): true => false
books/fetchBooks/fulfilled (16)
Live
1x
2x


SJSU
Elements
Console
Sources
Redux
Network
Performance
Memory
Application
Privacy and security
Actions
Settings
Reset
Revert
Sweep
Commit
1068381183/1
filter...
books/fetchBooks/pending +00:00.06
books/fetchBooks/pending +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/createBook/pending +00:23.37
books/createBook/fulfilled +00:00.02
books/fetchBooks/pending +00:00.00
books/fetchBooks/pending +00:00:00
books/fetchBooks/fulfilled +00:00.00
books/fetchBooks/fulfilled +00:00.01
books/updateBook/pending +00:13.57
books/updateBook/fulfilled +00:00.03
books/fetchBooks/pending +00:00.00
books/fetchBooks/pending +00:00:00
books/fetchBooks/fulfilled +00:00.00
Diff
Action
State
Diff
Trace
Test
Tree
Raw
{
books: {
books: [
1: {
author: "Test" => "Test - Updated ",
title: "Test" => "Test - Updated",
updatedAt: "2025-10-04T01:05:34.000Z"
=> "2025-10-04T01:05:48.160Z"
},
],
loading: true => false
},
}
books/fetchBooks/fulfilled (15)
Live
1x
2x

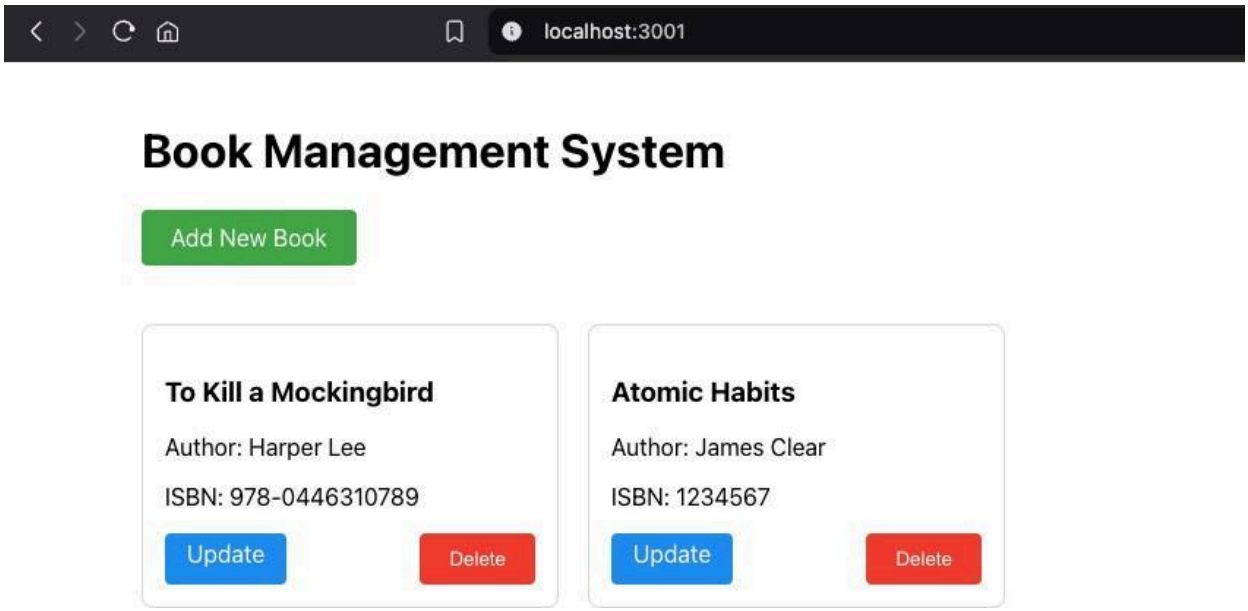
CreateBook.jsx | UpdateBook.jsx | DeleteBook.jsx

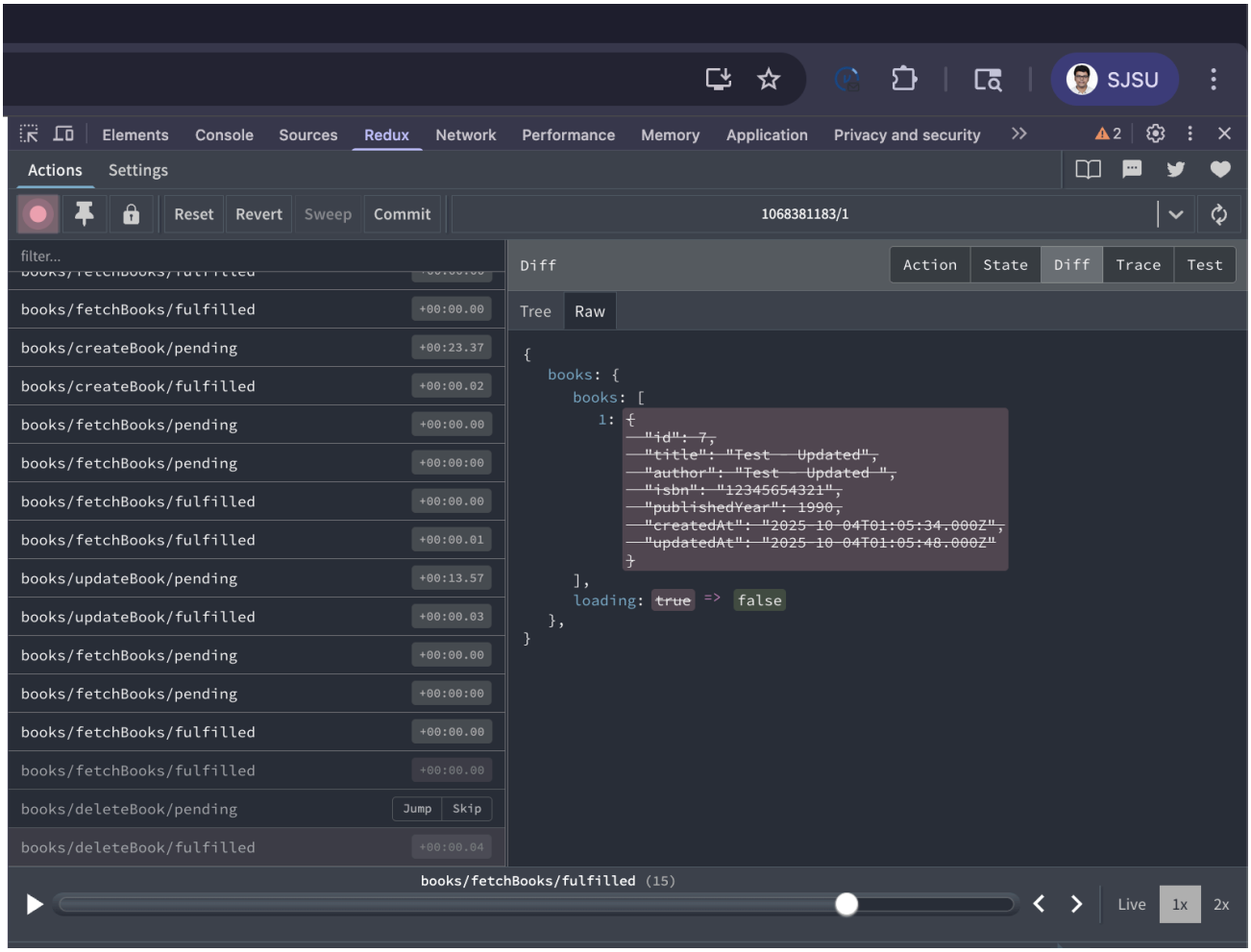
hw4-template > src > components > Update > UpdateBook.jsx

```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.

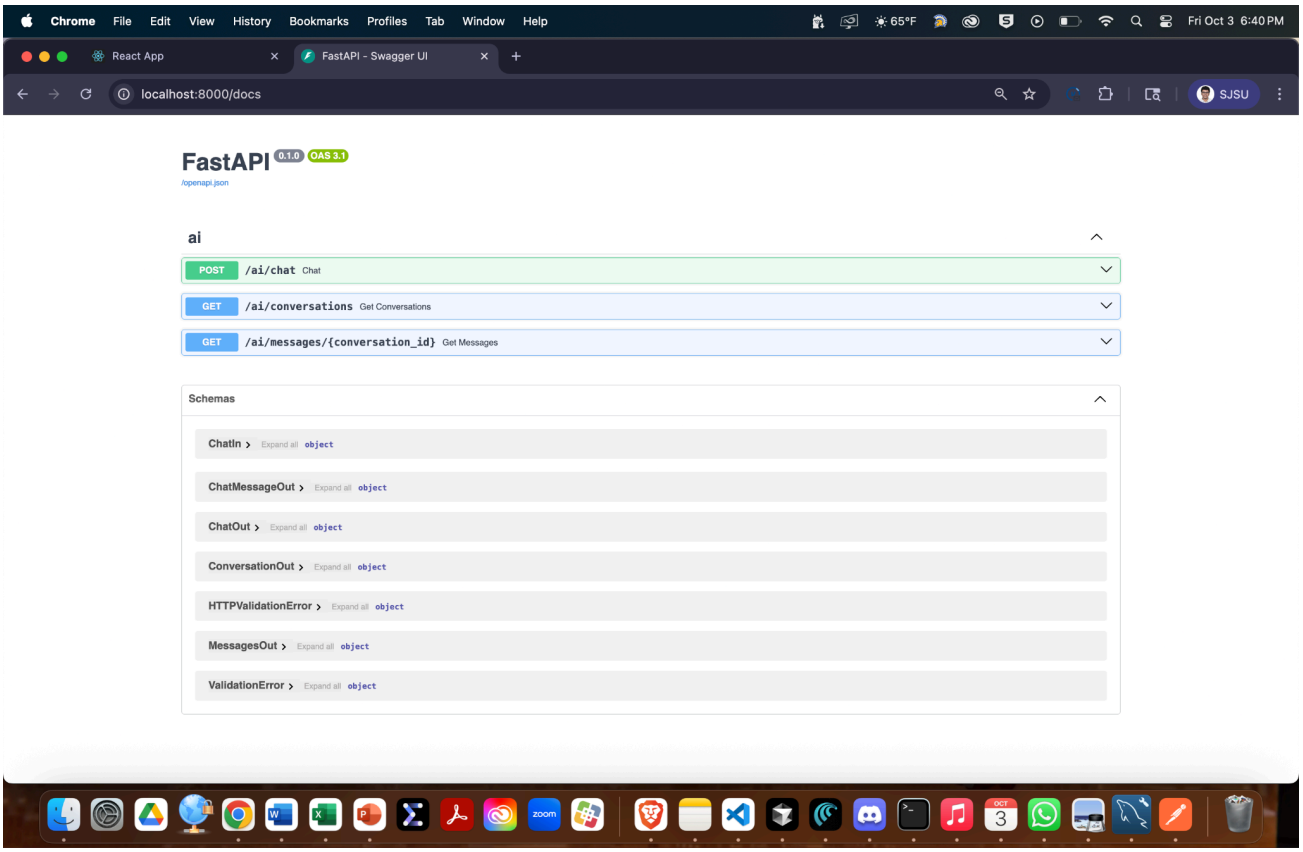



SJSU
Elements
Console
Sources
Redux
Network
Performance
Memory
Application
Privacy and security
Actions
Settings
Reset
Revert
Sweep
Commit
1068381183/1
filter...
books/fetchBooks/fulfilled +00:00.00
books/createBook/pending +00:23.37
books/createBook/fulfilled +00:00.02
books/fetchBooks/pending +00:00.00
books/fetchBooks/pending +00:00:00
books/fetchBooks/fulfilled +00:00.00
books/fetchBooks/fulfilled +00:00.01
books/updateBook/pending +00:13.57
books/updateBook/fulfilled +00:00.03
books/fetchBooks/pending +00:00.00
books/fetchBooks/pending +00:00:00
books/fetchBooks/fulfilled +00:00.00
books/fetchBooks/fulfilled +00:00.00
books/deleteBook/pending Jump Skip
books/deleteBook/fulfilled +00:00.04
Diff
Action
State
Diff
Trace
Test
Tree
Raw
{
books: {
books: [
1: {
"id": 7,
"title": "Test - Updated",
"author": "Test - Updated ",
"isbn": "12345654321",
"publishedYear": 1990,
"createdAt": "2025-10-04T01:05:34.000Z",
"updatedAt": "2025-10-04T01:05:48.000Z"
}
],
loading: true => false
},
}
books/fetchBooks/fulfilled (15)
Live
1x
2x

CreateBook.jsx | UpdateBook.jsx | DeleteBook.jsx

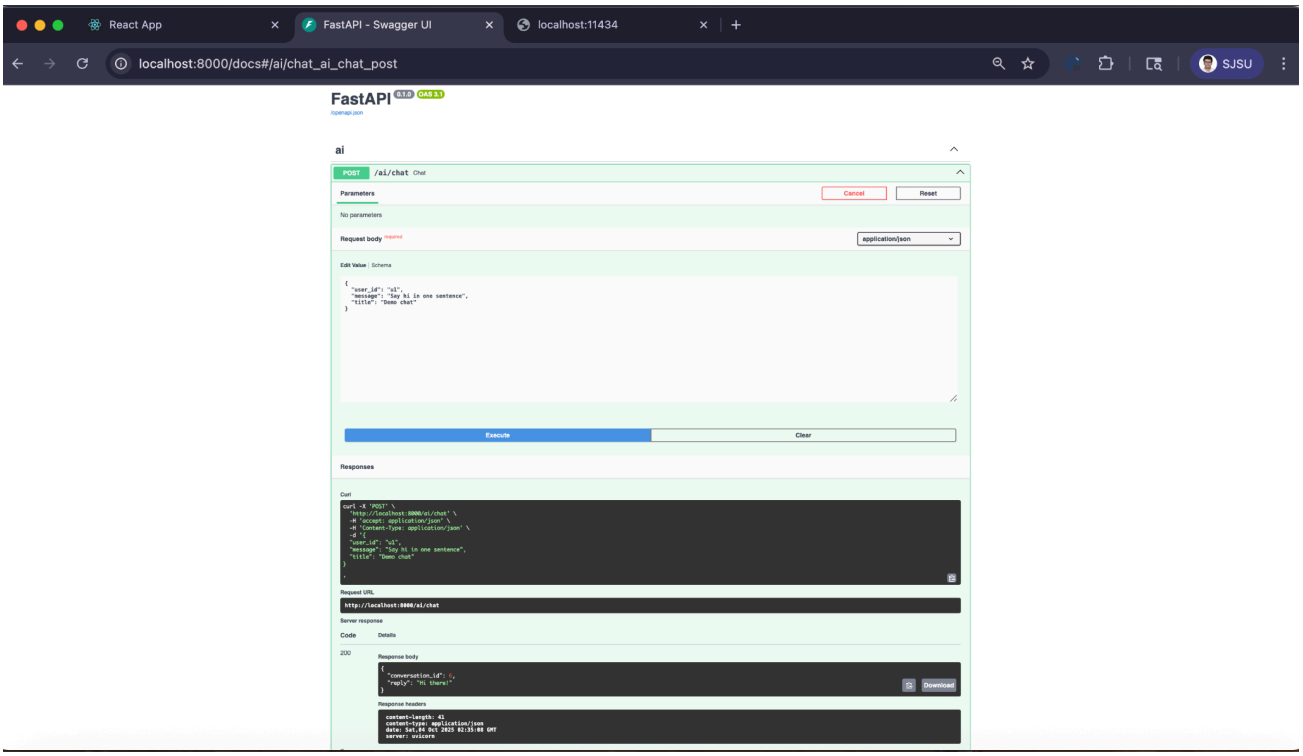
hw4-template > src > components > Delete > DeleteBook.jsx

```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>
      )}
```

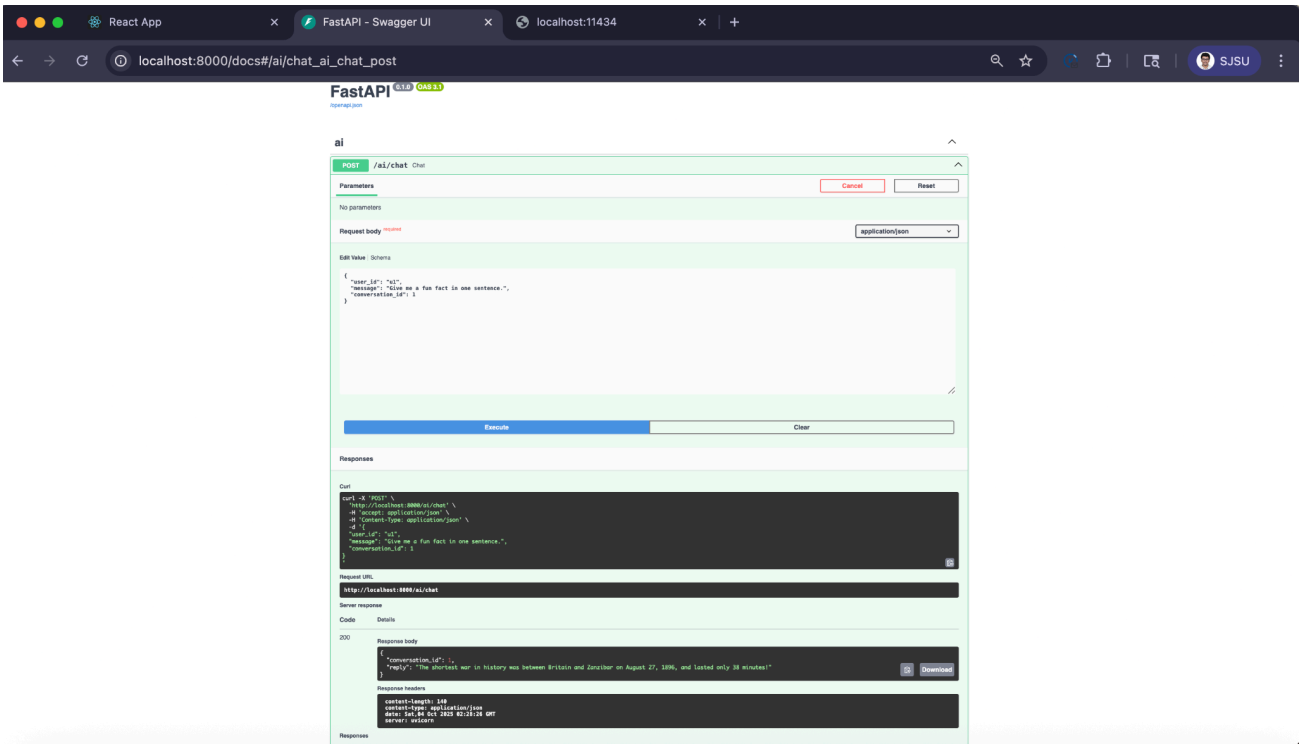

Chrome File Edit View History Bookmarks Profiles Tab Window Help
65°F
Fri Oct 3 6:40 PM
React App
FastAPI - Swagger UI
localhost:8000/docs
SJSU
FastAPI
0.1.0
OAS 3.1
/openapi.json
ai
POST /ai/chat Chat
GET /ai/conversations Get Conversations
GET /ai/messages/{conversation_id} Get Messages
Schemas
ChatIn Expand all object
ChatMessageOut Expand all object
ChatOut Expand all object
ConversationOut Expand all object
HTTPValidationError Expand all object
MessagesOut Expand all object
ValidationError Expand all object

POST /ai/chat — start a conversation

Swagger: body

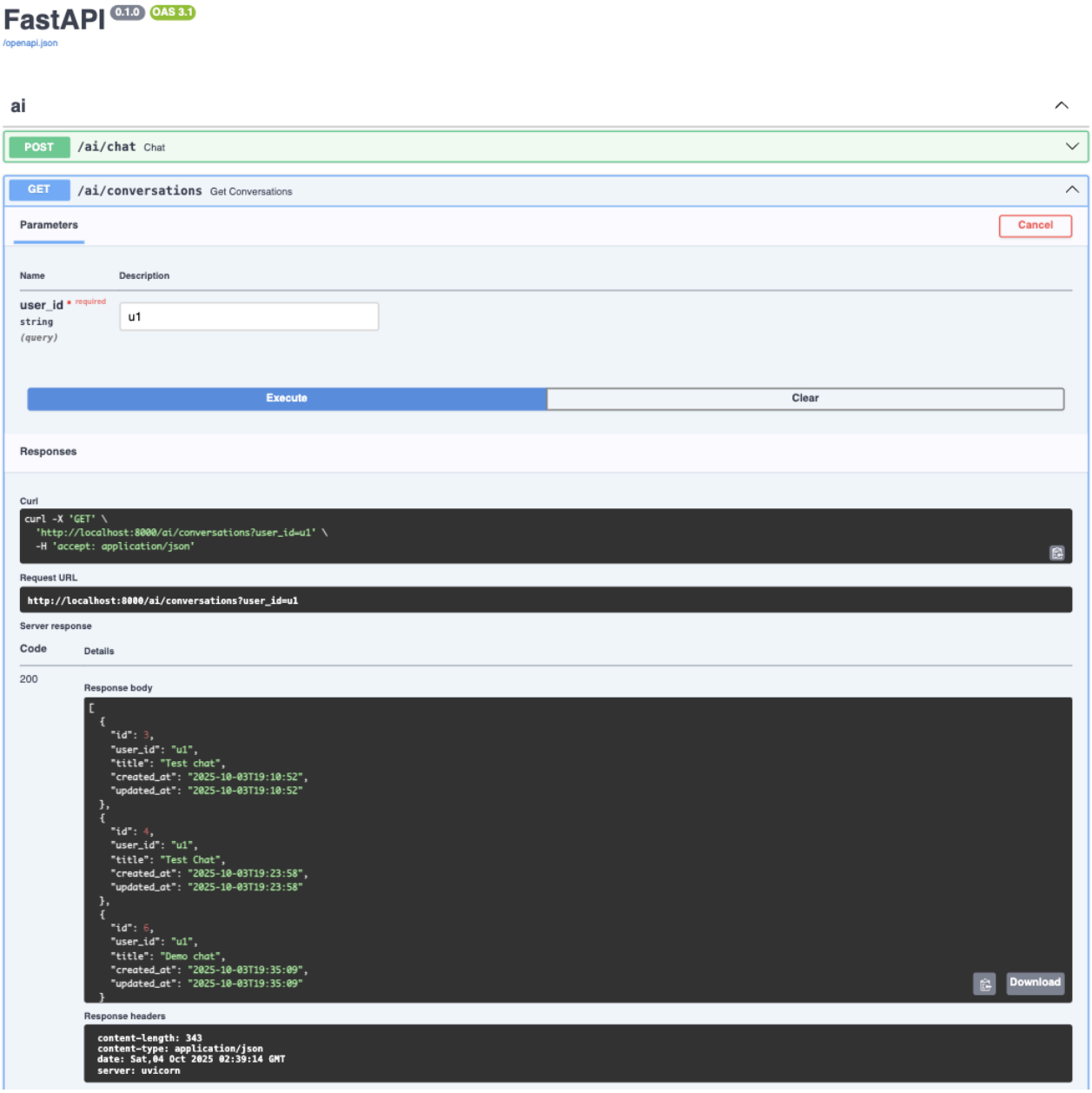


2) POST /ai/chat — continue the same conversation



3) GET /ai/conversations — list the user's chats

Swagger: set `user_id = u1` in the query param



4) GET /ai/messages/{conversation_id} — all messages in a chat

Swagger: path param `conversation_id = 1`

FastAPI 0.1.0 OAS 3.1

/openapi.json

ai

POST /ai/chat Chat

GET /ai/conversations Get Conversations

GET /ai/messages/{conversation_id} Get Messages

Parameters Cancel

Name	Description
conversation_id * required integer (path)	1

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/ai/messages/1' \
  -H 'accept: application/json'
```

Request URL

```
http://localhost:8000/ai/messages/1
```

Server response

Code	Details
200	Response body

```
{
  "messages": [
    {
      "id": 1,
      "role": "user",
      "content": "Hello! Vimal",
      "created_at": "2025-10-03T18:44:32"
    },
    {
      "id": 8,
      "role": "user",
      "content": "Give me a fun fact in one sentence.",
      "created_at": "2025-10-03T19:28:26"
    },
    {
      "id": 9,
      "role": "assistant",
      "content": "The shortest war in history was between Britain and Zanzibar on August 27, 1896, and lasted only 38 minutes!",
      "created_at": "2025-10-03T19:28:40"
    }
  ]
}
```

Download

Response headers

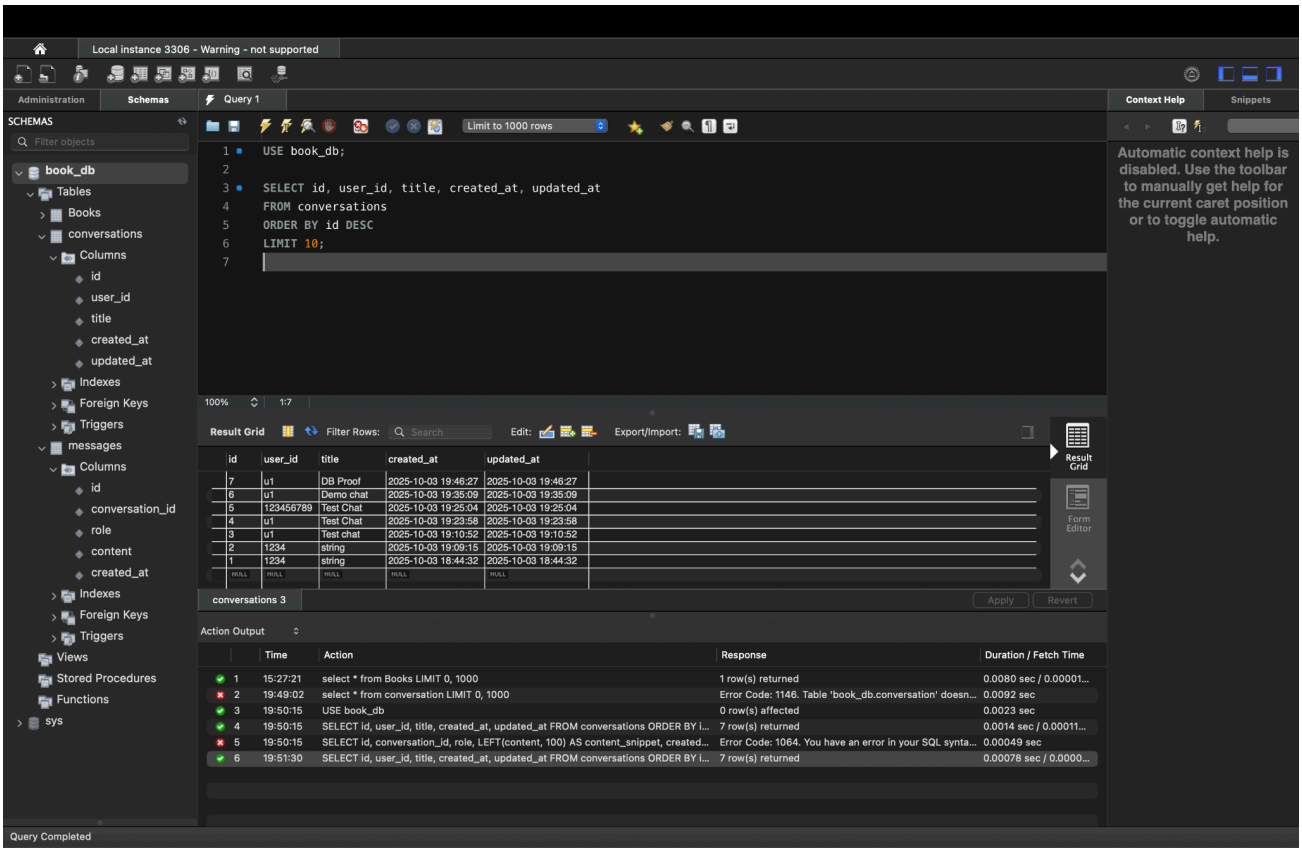
```
content-length: 387
content-type: application/json
date: Sat,04 Oct 2025 02:41:03 GMT
server: uvicorn
```

Responses

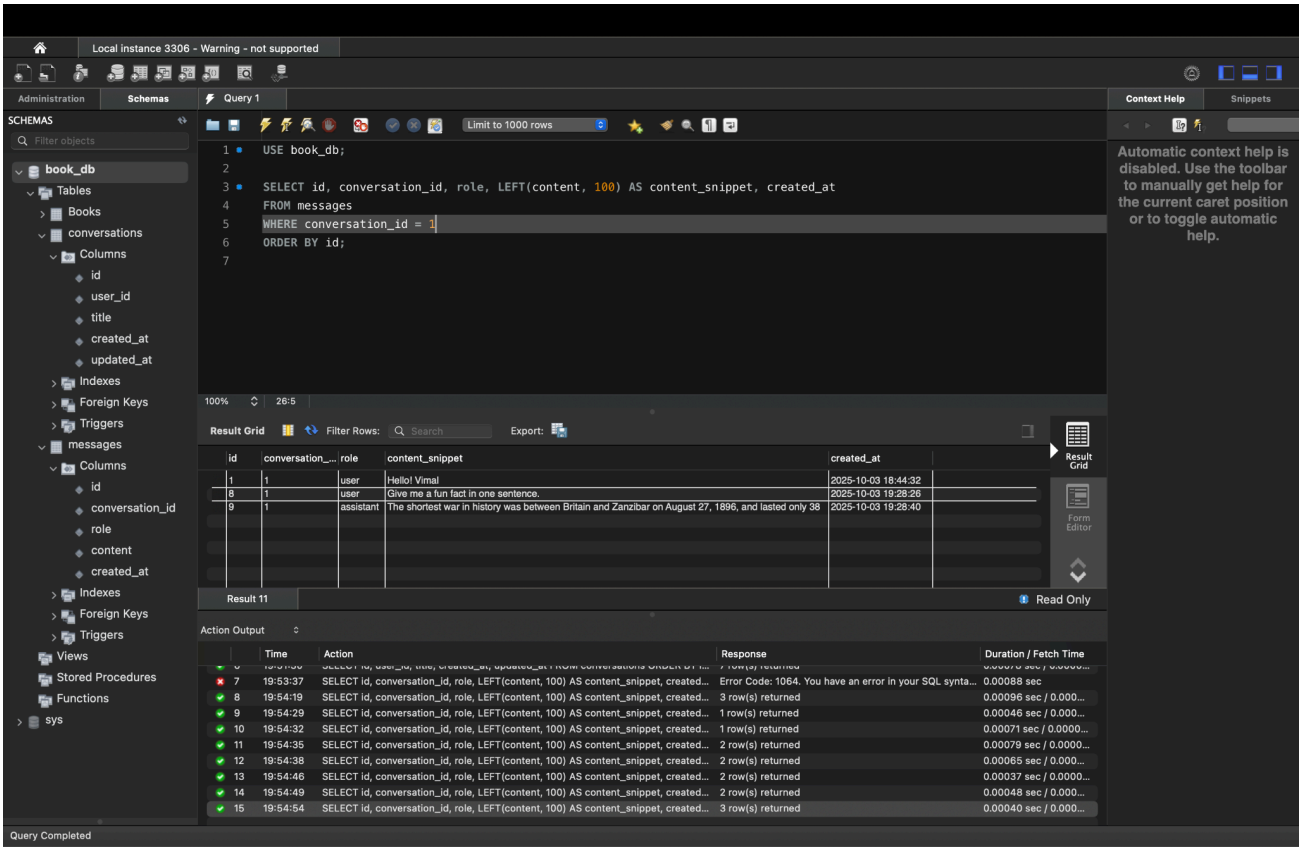
Code	Description	Links
200	Successful Response	*No links*

DB Screenshots:

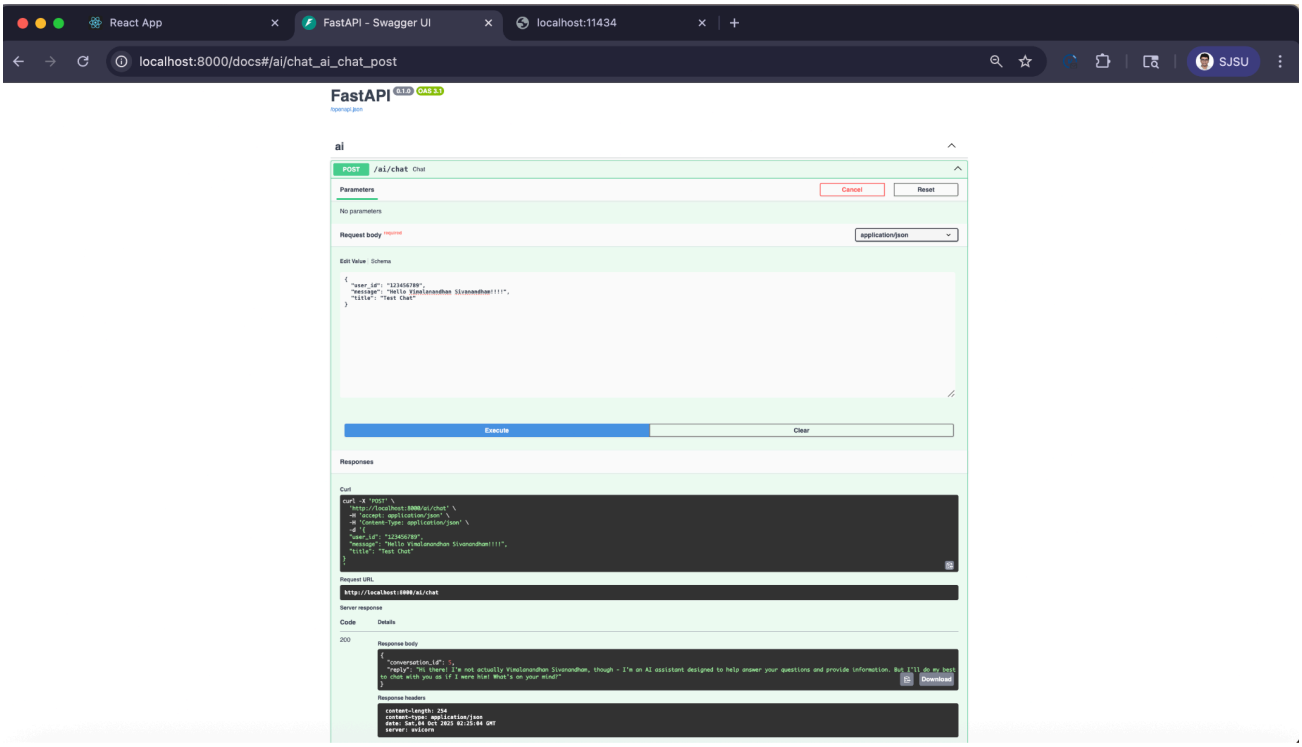
Conversations list



Messages for one conversation



XXXXXXXXXXXXXXXXXX—----------XXXXXXXXXX


React App
FastAPI - Swagger UI
localhost:11434
localhost:8000/docs#/ai/chat_ai_chat_post
SJSU
FastAPI
0.1.0
OAS 3.1
openapi.json
ai
POST
/ai/chat
Chat
Parameters
Cancel
Reset
No parameters
Request body
required
application/json
Edit Value
Schema
{
"user_id": "1234567890",
"message": "Hello VimalanandhanSivanandhan!!!!",
"title": "Test Chat"
}
Execute
Clear
Responses
Curl
curl -X 'POST' \
'http://localhost:8000/ai/chat' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
"user_id": "1234567890",
"message": "Hello VimalanandhanSivanandhan!!!!",
"title": "Test Chat"
}'
Request URL
http://localhost:8000/ai/chat
Server response
Code
Details
200
Response body
{
"conversation_id": "",
"reply": "Hi there! I'm not actually VimalanandhanSivanandhan, though - I'm an AI assistant designed to help answer your questions and provide information. But I'll do my best to chat with you as if I were him! What's on your mind?"
}
Download
Response headers
content-length: 254
content-type: application/json
date: Sat,04 Oct 2025 01:25:04 GMT
server: uvicorn