

**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

JS api.js

✓ store

JS bookSlice.js

JS store.js

# App.css

JS App.js

# index.css

JS index.js

JS setupTests.js

📄 .gitignore

{ } package-lock.json

{ } package.json

📖 README.md

✓ HW 4



> hw4-template

✓ mysql-nodejs

> node\_modules

✓ src

✓ config

JS database.js

✓ controllers

JS book.controller.js

JS user.controller.js

> models

> routes

JS app.js

JS user.js

⚙ .env

{ } package-lock.json

{ } package.json

📄 .gitignore

CreateBook.jsx

JS api.js

×

UpdateBook.jsx

DeleteBook.jsx

```

hw4-template > src > services > JS 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**

```
JS index.js  X  UpdateBook.jsx  DeleteBook.jsx
hw4-template > src > JS index.js
1  import React from 'react';
2  import ReactDOM from 'react-dom/client';
3  import { Provider } from 'react-redux';
4  import store from './store/store';
5  import App from './App';
6  import './index.css';
7
8  const root = ReactDOM.createRoot(document.getElementById('root'));
9  root.render(
10   <React.StrictMode>
11     <Provider store={store}>
12       <App />
13     </Provider>
14   </React.StrictMode>
15 );
```

```
JS store.js  X  UpdateBook.jsx  DeleteBook.jsx
hw4-template > src > store > JS store.js
1  import { configureStore } from '@reduxjs/toolkit';
2  import bookReducer from './bookSlice';
3
4  const store = configureStore({
5    reducer: {
6      books: bookReducer
7    },
8    middleware: (getDefaultMiddleware) =>
9      getDefaultMiddleware({
10        serializableCheck: false
11      }),
12    devTools: process.env.NODE_ENV !== 'production'
13  });
14
15  export default store;
```

```

JS bookSlice.js x UpdateBook.jsx DeleteBook.jsx
hw4-template > src > store > JS bookSlice.js
1  import { createSlice, createAsyncThunk } from '@reduxjs/toolkit'
2  import * as api from '../services/api';
3
4  // Async Thunks
5  export const fetchBooks = createAsyncThunk(
6    'books/fetchBooks',
7    async () => {
8      return await api.getBooks();
9    }
10 );
11
12 export const createBook = createAsyncThunk(
13   'books/createBook',
14   async (bookData) => {
15     return await api.createBook(bookData);
16   }
17 );
18
19 export const updateBook = createAsyncThunk(
20   'books/updateBook',
21   async ({ id, bookData }) => {
22     return await api.updateBook(id, bookData);
23   }
24 );
25
26 export const deleteBook = createAsyncThunk(
27   'books/deleteBook',
28   async (id) => {
29     await api.deleteBook(id);
30     return id;
31   }
32 );
33
34 const bookSlice = createSlice({
35   name: 'books',
36   initialState: {
37     books: [],
38     loading: false,
39     error: null
40   },
41   reducers: {
42     clearError: (state) => {
43       state.error = null;
44     }
45   },
46   extraReducers: (builder) => {
47     builder
48       // Fetch Books
49       .addCase(fetchBooks.pending, (state) => {
50         state.loading = true;
51         state.error = null;
52       })
53       .addCase(fetchBooks.fulfilled, (state, action) => {
54         state.loading = false;
55         state.books = action.payload;
56       })
57       .addCase(fetchBooks.rejected, (state, action) => {
58         state.loading = false;
59         state.error = action.error.message;
60       })

```



```
JS bookSlice.js x UpdateBook.jsx DeleteBook.jsx
hw4-template > src > store > JS bookSlice.js
61 // Create Book
62 .addCase(createBook.pending, (state) => {
63   state.loading = true;
64   state.error = null;
65 })
66 .addCase(createBook.fulfilled, (state, action) => {
67   state.loading = false;
68   state.books.push(action.payload);
69 })
70 .addCase(createBook.rejected, (state, action) => {
71   state.loading = false;
72   state.error = action.error.message;
73 })
74 // Update Book
75 .addCase(updateBook.pending, (state) => {
76   state.loading = true;
77   state.error = null;
78 })
79 .addCase(updateBook.fulfilled, (state, action) => {
80   state.loading = false;
81   state.books = state.books.map(book =>
82     book.id === action.payload.id ? action.payload : book
83   );
84 })
85 .addCase(updateBook.rejected, (state, action) => {
86   state.loading = false;
87   state.error = action.error.message;
88 })
89 // Delete Book
90 .addCase(deleteBook.pending, (state) => {
91   state.loading = true;
92   state.error = null;
93 })
94 .addCase(deleteBook.fulfilled, (state, action) => {
95   state.loading = false;
96   state.books = state.books.filter(book => book.id !== action.payload);
97 })
98 .addCase(deleteBook.rejected, (state, action) => {
99   state.loading = false;
100   state.error = action.error.message;
101 });
102 }
103 });
104
105 export const { clearError } = bookSlice.actions;
106 export default bookSlice.reducer;
```

## 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

```

1  import React, { useEffect } from 'react';
2  import { Link } from 'react-router-dom';
3  import { useDispatch, useSelector } from 'react-redux';
4  import { fetchBooks, deleteBook } from '../store/bookSlice';
5  import './Home.css';
6
7  const Home = () => {
8    const dispatch = useDispatch();
9    const { books, loading, error } = useSelector(state => state.books);
10
11    useEffect(() => {
12      dispatch(fetchBooks());
13    }, [dispatch]);
14
15    const handleDelete = async (id) => {
16      try {
17        await dispatch(deleteBook(id)).unwrap();
18      } catch (err) {
19        console.error('Failed to delete book:', err);
20      }
21    };
22
23    if (loading) return <div>Loading...</div>;
24    if (error) return <div>Error: {error}</div>;
25
26    return (
27      <div className="home-container">
28        <h1>Book Management System</h1>
29        <Link to="/create" className="add-button">Add New Book</Link>
30
31        <div className="books-grid">
32          {books.map(book => (
33            <div key={book.id} className="book-card">
34              <h3>{book.title}</h3>
35              <p>Author: {book.author}</p>
36              <p>ISBN: {book.isbn}</p>
37              <div className="book-actions">
38                <Link to={` /update/${book.id}`} className="edit-button">Update</Link>
39                <button
40                  onClick={() => handleDelete(book.id)}
41                  className="delete-button"
42                >
43                  Delete
44                </button>
45              </div>
46            </div>
47          ))}
48        </div>
49      </div>
50    );
51  };
52
53  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.

localhost:3001/create

## Add New Book

Book Title:

Test

Author Name:

Test

ISBN:

12345654321

Published Year:

1990

Add Book

localhost:3001

## Book Management System

Add New Book

To Kill a Mockingbird

Author: Harper Lee

ISBN: 978-0446310789

UpdateDelete

Atomic Habits

Author: James Clear

ISBN: 1234567

UpdateDelete

Test

Author: Test

ISBN: 12345654321

UpdateDelete

ElementsConsoleSourcesReduxNetworkPerformanceMemoryApplicationPrivacy and security>>

2

⚙️⋮✕

ActionsSettings

🔴📌🔒

ResetRevertSweepCommit

1068381183/1

⌵↺

filter...

@@INIT5:56:31.43

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

ActionStateDiffTraceTest

TreeRaw

▼ 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

```

CreateBook.jsx x UpdateBook.jsx DeleteBook.jsx
hw4-template > src > components > Create > CreateBook.jsx
1  import React, { useState } from 'react';
2  import { useNavigate } from 'react-router-dom';
3  import { useDispatch, useSelector } from 'react-redux';
4  import { createBook } from '../../store/bookSlice';
5  import './CreateBook.css';
6
7  const CreateBook = () => {
8    const [formData, setFormData] = useState({
9      title: '',
10     author: '',
11     isbn: '',
12     publishedYear: ''
13   });
14   const dispatch = useDispatch();
15   const navigate = useNavigate();
16   const { loading, error } = useSelector(state => state.books);
17
18   const handleChange = (e) => {
19     const { name, value } = e.target;
20     setFormData(prev => ({
21       ...prev,
22       [name]: value
23     }));
24   };
25
26   const handleSubmit = async (e) => {
27     e.preventDefault();
28     try {
29       await dispatch(createBook(formData)).unwrap();
30       navigate('/');
31     } catch (err) {
32       console.error('Failed to create book:', err);
33     }
34   };
35
36   return (
37     <div className="create-book-container">
38       <h2>Add New Book</h2>
39       {error && <div className="error-message">{error}</div>}
40       <form onSubmit={handleSubmit}>
41         <div className="form-group">
42           <label htmlFor="title">Book Title:</label>
43           <input
44             type="text"

```

#### IV. Update Screen (2 points)

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

# Update Book

Book Title:

Test - Updated

Author Name:

Test - Updated

ISBN:

12345654321

Published Year:

1990

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

Test - Updated

Author: Test - Updated

ISBN: 12345654321

Update

Delete

ElementsConsoleSourcesReduxNetworkPerformanceMemoryApplicationPrivacy and security>>

2

⚙️⋮✕

ActionsSettings

🔴📌🔒

ResetRevertSweepCommit

1068381183/1

⌵↺

filter...

books3/fetchBooks/pending

+00:00.00

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/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

ActionStateDiffTraceTest

TreeRaw

▼ 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)

▶

◀▶

Live1x2x



The screenshot shows the Redux DevTools interface. The left pane displays a list of actions, including 'books/fetchBooks/pending', 'books/fetchBooks/fulfilled', 'books/createBook/pending', 'books/createBook/fulfilled', and 'books/updateBook/pending'. The right pane shows the state diff for the 'books' object, which is updated with 'author', 'title', and 'updatedAt' fields. The bottom status bar indicates the current state is 'books/fetchBooks/fulfilled (15)'.

CreateBook.jsx

UpdateBook.jsx X

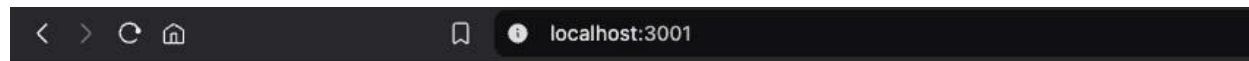
DeleteBook.jsx

hw4-template &gt; src &gt; components &gt; Update &gt; UpdateBook.jsx

```
1  import React, { useState, useEffect } from 'react';
2  import { useNavigate, useParams } from 'react-router-dom';
3  import { useDispatch, useSelector } from 'react-redux';
4  import { updateBook } from '../../store/bookSlice';
5  import { getBookById } from '../../services/api';
6  import './UpdateBook.css';
7
8  const UpdateBook = () => {
9    const [formData, setFormData] = useState({
10      title: '',
11      author: '',
12      isbn: '',
13      publishedYear: ''
14    });
15
16    const navigate = useNavigate();
17    const dispatch = useDispatch();
18    const { id } = useParams();
19    const { loading, error } = useSelector(state => state.books);
20    const [fetchError, setFetchError] = useState(null);
21
22    useEffect(() => {
23      const fetchBook = async () => {
24        try {
25          const book = await getBookById(id);
26          setFormData(book);
27        } catch (error) {
28          setFetchError(error.message || 'Failed to fetch book');
29        }
30      };
31
32      fetchBook();
33    }, [id]);
34
35    const handleChange = (e) => {
36      const { name, value } = e.target;
37      setFormData(prev => ({
38        ...prev,
39        [name]: value
40      }));
41    };
42
43    const handleSubmit = async (e) => {
44      e.preventDefault();
45      try {
46        await dispatch(updateBook({ id, bookData: formData })).unwrap();
47        navigate('/');
48      } catch (err) {
49        console.error('Failed to update book:', err);
50      }
51    };
52
53    if (loading) return <div>Loading...</div>;
54    if (fetchError) return <div>Error: {fetchError}</div>;
55
56    return (
57      <div className="update-book-container">
58        <h2>Update Book</h2>
59        {error && <div className="error-message">{error}</div>}
60        <form onSubmit={handleSubmit}>
61          <div className="form-group">
62            <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.



## 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

The screenshot shows the Redux DevTools interface. The top bar includes navigation icons and the user profile 'SJSU'. The main toolbar has tabs for 'Elements', 'Console', 'Sources', 'Redux', 'Network', 'Performance', 'Memory', 'Application', 'Privacy and security', and a search icon. The 'Redux' tab is active, showing a list of actions on the left and a diff view on the right. The actions list includes 'books/fetchBooks/fulfilled', 'books/createBook/pending', 'books/createBook/fulfilled', 'books/fetchBooks/pending', 'books/fetchBooks/fulfilled', 'books/updateBook/pending', 'books/updateBook/fulfilled', 'books/deleteBook/pending', and 'books/deleteBook/fulfilled'. The 'fetchBooks/fulfilled' action is selected, showing a diff of the state. The state is an object with 'books' and 'loading' properties. The 'books' array contains an object with 'id', 'title', 'author', 'isbn', 'publishedYear', 'createdAt', and 'updatedAt'.

filter...	Diff	Action	State	Diff	Trace	Test
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/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					

```

{
  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

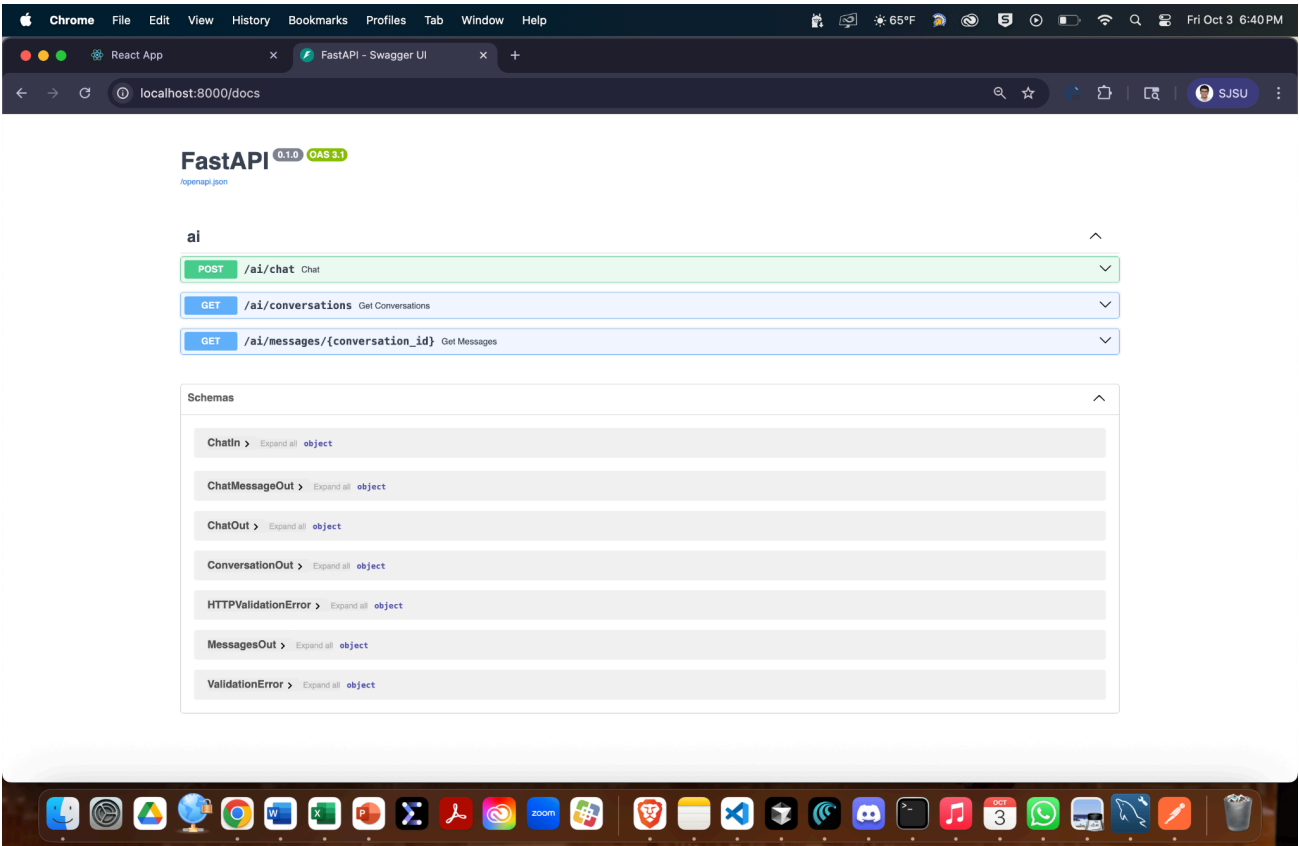
CreateBook.jsx

UpdateBook.jsx

DeleteBook.jsx X

hw4-template &gt; src &gt; components &gt; Delete &gt; DeleteBook.jsx

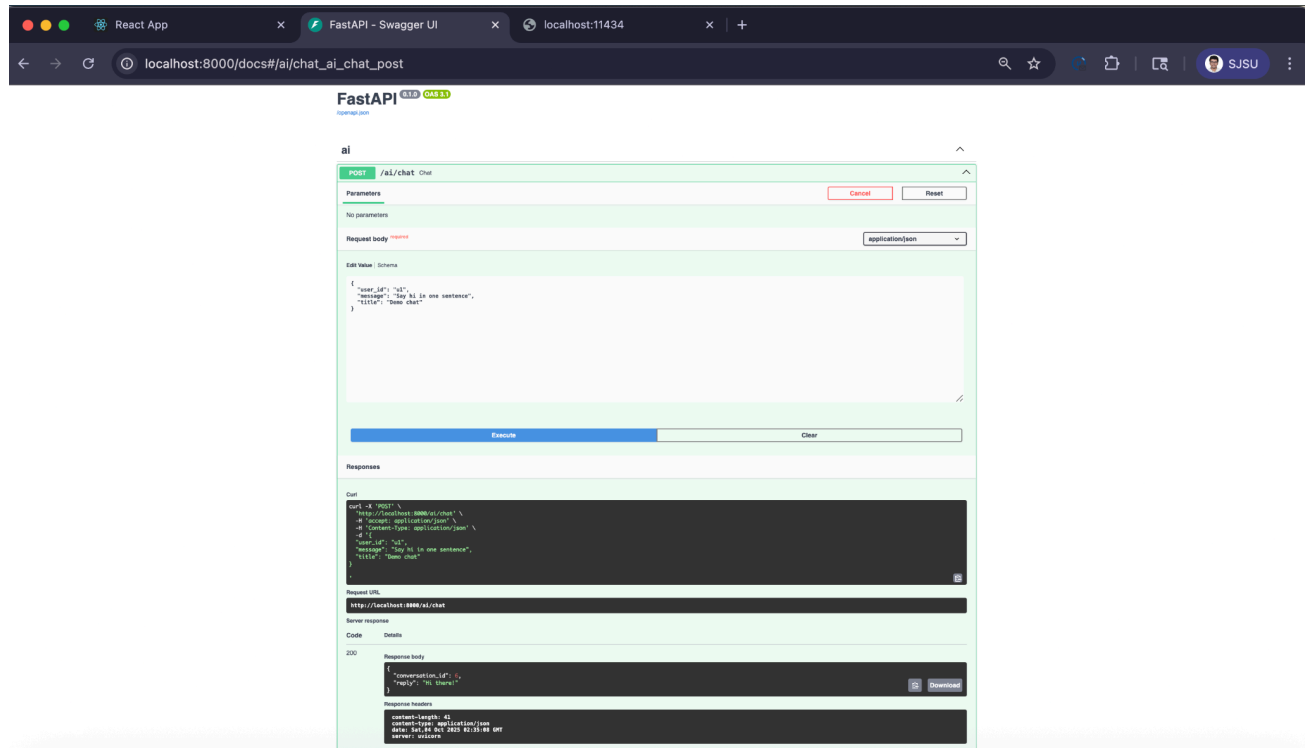
```
1 import React, { useState, useEffect } from 'react';
2 import { useNavigate, useParams } from 'react-router-dom';
3 import { useDispatch, useSelector } from 'react-redux';
4 import { deleteBook } from '../../store/bookSlice';
5 import { getBookById } from '../../services/api';
6 import './DeleteBook.css';
7
8 const DeleteBook = () => {
9   const [book, setBook] = useState(null);
10  const [error, setError] = useState(null);
11  const navigate = useNavigate();
12  const dispatch = useDispatch();
13  const { id } = useParams();
14  const { loading } = useSelector(state => state.books);
15
16  useEffect(() => {
17    const fetchBook = async () => {
18      try {
19        const bookData = await getBookById(id);
20        setBook(bookData);
21      } catch (error) {
22        setError('Failed to fetch book details');
23        console.error('Error:', error);
24      }
25    };
26
27    fetchBook();
28  }, [id]);
29
30  const handleDelete = async () => {
31    try {
32      await dispatch(deleteBook(id)).unwrap();
33      navigate('/');
34    } catch (error) {
35      setError(error.message);
36      console.error('Error:', error);
37    }
38  };
39
40  if (loading) return <div>Loading...</div>;
41  if (error) return <div className="error-message">{error}</div>;
42
43  return (
44    <div className="delete-book-container">
45      <h2>Delete Book</h2>
46      {book && (
47        <div className="book-details">
48          <h3>{book.title}</h3>
49          <p>Author: {book.author}</p>
50          <p>ISBN: {book.isbn}</p>
51          {book.publishedYear && <p>Published Year: {book.publishedYear}</p>}
52          <p className="warning-text">Are you sure you want to delete this book?</p>
53        </div>
54      )}
```



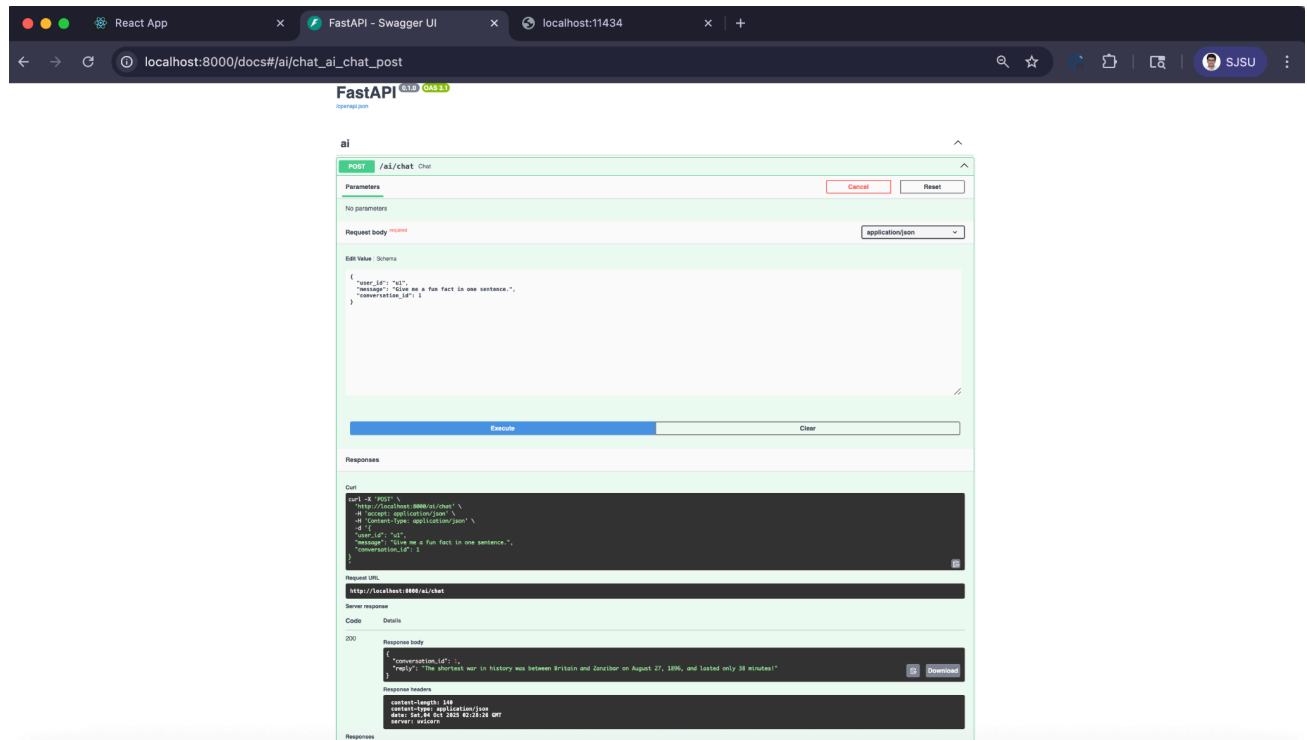


# POST /ai/chat — start a conversation

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

ai

POST /ai/chat Chat

GET /ai/conversations Get Conversations

Parameters

Cancel

Name	Description
user_id * required	
string	
(query)	

u1

Execute Clear

Responses

Curl

```
curl -X 'GET' \
  'http://localhost:8000/ai/conversations?user_id=u1' \
  -H 'accept: application/json'
```

Request URL

http://localhost:8000/ai/conversations?user\_id=u1

Server response

Code Details

200

Response body

```
[
  {
    "id": 3,
    "user_id": "u1",
    "title": "Test chat",
    "created_at": "2025-10-03T19:10:52",
    "updated_at": "2025-10-03T19:10:52"
  },
  {
    "id": 4,
    "user_id": "u1",
    "title": "Test Chat",
    "created_at": "2025-10-03T19:23:58",
    "updated_at": "2025-10-03T19:23:58"
  },
  {
    "id": 6,
    "user_id": "u1",
    "title": "Demo chat",
    "created_at": "2025-10-03T19:35:09",
    "updated_at": "2025-10-03T19:35:09"
  }
]
```

Response headers

```
content-length: 343
content-type: application/json
date: Sat, 04 Oct 2025 02:39:14 GMT
server: uvicorn
```

Download

4) GET /ai/messages/{conversation\_id} — all messages in a chat

Swagger: path param `conversation_id` = 1

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' \n'http://localhost:8000/ai/messages/1' \n-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

Local instance 3306 - Warning - not supported

Administration

Schemas

SCHEMAS

Filter objects

book\_db

Tables

Books

conversations

Columns

id

user\_id

title

created\_at

updated\_at

Indexes

Foreign Keys

Triggers

messages

Columns

id

conversation\_id

role

content

created\_at

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

sys

Query 1

Limit to 1000 rows

1 USE book\_db;

2

3 SELECT id, user\_id, title, created\_at, updated\_at

4 FROM conversations

5 ORDER BY id DESC

6 LIMIT 10;

7

Context Help

Snippets

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

100%

1:7

Result Grid

Filter Rows

Search

Edit

Export/Import

Result Grid

Form Editor

id	user_id	title	created_at	updated_at
7	u1	DB Proof	2025-10-03 19:46:27	2025-10-03 19:46:27
6	u1	Demo chat	2025-10-03 19:35:09	2025-10-03 19:35:09
5	123456789	Test Chat	2025-10-03 19:25:04	2025-10-03 19:25:04
4	u1	Test Chat	2025-10-03 19:23:58	2025-10-03 19:23:58
3	u1	Test chat	2025-10-03 19:10:52	2025-10-03 19:10:52
2	1234	string	2025-10-03 19:09:15	2025-10-03 19:09:15
1	1234	string	2025-10-03 18:44:32	2025-10-03 18:44:32
	NULL		NULL	NULL

conversations 3

Apply

Revert

Action Output

	Time	Action	Response	Duration / Fetch Time
1	15:27:21	select * from Books LIMIT 0, 1000	1 row(s) returned	0.0080 sec / 0.00001...
2	19:49:02	select * from conversation LIMIT 0, 1000	Error Code: 1146. Table 'book_db.conversation' doesn't exist	0.0082 sec
3	19:50:15	USE book_db	0 row(s) affected	0.0023 sec
4	19:50:15	SELECT id, user_id, title, created_at, updated_at FROM conversations ORDER BY id DESC	7 row(s) returned	0.0014 sec / 0.00011...
6	19:50:15	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created_at, updated_at FROM conversations ORDER BY id DESC	Error Code: 1064. You have an error in your SQL syntax; check the manual that corresponds to your MySQL server version for the right syntax to use near 'SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created_at, updated_at FROM conversations ORDER BY L...	0.00049 sec
6	19:51:30	SELECT id, user_id, title, created_at, updated_at FROM conversations ORDER BY id DESC	7 row(s) returned	0.00078 sec / 0.0000...

Query Completed

Messages for one conversation

AdministrationSchemasQuery 1Context HelpSnippets

SCHEMAS

Filter objects

book\_db

Tables

Books

conversations

Columns

id

user\_id

title

created\_at

updated\_at

Indexes

Foreign Keys

Triggers

messages

Columns

id

conversation\_id

role

content

created\_at

Indexes

Foreign Keys

Triggers

Views

Stored Procedures

Functions

sys

1USE book\_db;

2

3SELECT id, conversation\_id, role, LEFT(content, 100) AS content\_snippet, created\_at

4FROM messages

5WHERE conversation\_id = 1

6ORDER BY id;

7

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

100%26.5

Result GridFilter Rows: SearchExport

id	conversation_id	role	content_snippet	created_at
1	1	user	Hello Vimal	2025-10-03 18:44:32
8	1	user	Give me a fun fact in one sentence.	2025-10-03 19:28:26
9	1	assistant	The shortest war in history was between Britain and Zanzibar on August 27, 1896, and lasted only 38	2025-10-03 19:28:40

Result 11Read Only

Action Output

	Time	Action	Response	Duration / Fetch Time
7	19:53:37	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	Error Code: 1064. You have an error in your SQL synta...	0.00088 sec
8	19:54:19	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	3 row(s) returned	0.00096 sec / 0.000...
9	19:54:29	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	1 row(s) returned	0.00046 sec / 0.000...
10	19:54:32	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	1 row(s) returned	0.00071 sec / 0.0000...
11	19:54:35	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	2 row(s) returned	0.00079 sec / 0.0000...
12	19:54:38	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	2 row(s) returned	0.00065 sec / 0.000...
13	19:54:46	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	2 row(s) returned	0.00037 sec / 0.0000...
14	19:54:49	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	2 row(s) returned	0.00048 sec / 0.000...
15	19:54:54	SELECT id, conversation_id, role, LEFT(content, 100) AS content_snippet, created...	3 row(s) returned	0.00040 sec / 0.000...

Query Completed

XXXXXXXXXXXXXXXXXXXX-----XXXXXXXXXXXX



**FastAPI** 0.100.0 0.103.3

ai

**POST** /ai/chat Chat

Parameters

No parameters

Request body required application/json

Edit Value - Schema

```
{
  "user_id": "123456789",
  "message": "Hello Vigneshan Sivaraman!!!",
  "title": "Test Chat"
}
```

Execute Clear

Responses

Curl

```
curl -d '{
  "user_id": "123456789",
  "message": "Hello Vigneshan Sivaraman!!!",
  "title": "Test Chat"
}' -X POST http://localhost:8000/ai/chat
```

Request URL

<http://localhost:8000/ai/chat>

Server response

Code Details

200

Response body

```
{
  "conversation_id": 1,
  "reply": "Hi there! I'm not actually Vigneshan Sivaraman, 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?"
}
```

Response headers

```
content-length: 264
content-type: application/json
date: 06 Oct 2023 02:55:04 GMT
server: uvicorn
```