Setup

React Client Setup	npm create vite@latest client -- --template react
	cd client
	npm install axios bootstrap react-icons react-router-dom
	src > copy styles.css
	src / app.jsx > import './styles.css'
Node/Express Server setup	root > create a new folder named server
	cd server
	npm init -y
	create server.js file
Concurrently	root > npm i concurrently

server > package.json

```
"scripts": {
"test": "echo \"Error: no test specified\" && exit 1",
"start": "node server.js"
},
"type": "module"
```

root > package.json

```
"scripts": {
"start": "concurrently \"npm run server\" \"npm run client\"",
"server": "cd server && npm start",
"client": "cd client && npm run dev"
}
```

Client > src / config / api.js

```
export const API_BASE_URL = (import.meta.env.VITE_API_B-
ASE_URL || "http://localhost:3000").replace(/\/$/, "");
export const API_URLS = {
books: ${API_ BAS E_U RL} /books,
book: (bookId) => ${API_ BAS E_U RL} /bo oks /${ bookId
},
};
export const BOOKS_API_URL = API_URLS.books;
export const getBookUrl = API_URLS.book;
```

Client > main.jsx

Client > App.jsx

```
import { BOOKS_API_URL, getBookUrl } from "./config/api";
import './styles.css';
const logError = (action, error) => {
console.log(ERROR in ${action}: ${erro r.m essage});
};
function App() {
const [books, setBooks] = useState([]);
useEffect(() => {
const fetchBooks = async () => {
try {
const { data } = await axios.get(BOOKS_API_URL);
setBooks(data);
} catch (err) {
logError("fetching books", err);
}
};
fetchBooks();
}, []);
const handleDelete = async (idToDelete) => {
try {
await axios.delete(getBookUrl(idToDelete));
setBooks((currentBooks) => currentBooks.filter((book) => book._id
!== idToDelete));
} catch (err) {
logError("deleting the book", err);
}
};
const handleAdd = async (newBook) => {
try {
const { data } = await axios.post(BOOKS_API_URL, newBook);
setBooks((currentBooks) => [...currentBooks, data]);
} catch (err) {
logError("adding the book", err);
}
};
const handleUpdate = async (updatedBook) => {
try {
const { data } = await axios.put(getBookUrl(updatedBook._id),
updatedBook);
setBooks((currentBooks) => currentBooks.map((book) => (book._id
=== data._id ? data : book)));
} catch (err) {
logError("updating the book", err);
}
};
```

```
import { StrictMode } from 'react'
import { createRoot } from 'react-dom/client'
import './index.css'
import App from './App.jsx'
import {BrowserRouter as Router} from "react-router-dom"
import "bootstrap/dist/css/bootstrap.css"
createRoot(document.getElementById('root')).render(
<StrictMode>
<Router>
<App />
</Router>
</StrictMode>,
)
```

sally sung

By **sally sung**
cheatography.com/sally-sung/

Not published yet.
Last updated 21st April, 2026.

Client > App.jsx (cont)

```
const handleToggleLike = (bookToToggle) => {
handleUpdate({ ...bookToToggle, like: !bookToToggle.like });
};
return (
<div className="App">
<NavBar />
<Header />
<Routes>
<Route
path="/"
element={<Books books={books} onDelete={handleDelete} onTogg-
leLike={handleToggleLike} />}
/>
<Route path="/addbook" element={<AddBook onAdd={handleAdd}
/>} />
<Route path="/updatebook/:id" element={<UpdateBook onUpdate=-
{handleUpdate} />} />
</Routes>
<Footer />
</div>
);
}
export default App;
```

Client > components / books.jsx

Client > components / book.jsx (cont)

```
<td>{book.author}</td>
<td>{book.numberInStock}</td>
<td>{book.price}</td>
<td>{book.rating}</td>
<td>{formatYear()}</td>
<td>
<button
type="button"
className="btn btn-link p-0 border-0"
onClick={() => onToggleLike(book)}
aria-label={book.like ? "Unlike book" : "Like book"}
>
{book.like ? <FaHeart color="red" /> : <FaRegHeart />}
</button>
</td>
<td>
<Link to={`/updat ebo ok/ ${b ook._id}`}>
<FaEdit />
</Link>
</td>
<td>
<FaTrash onClick={() => onDelete(book._id)} />
</td>
</tr>
);
};
export default Book;
```

Client > components / addBook.jsx

```
import Book from "./book";
const Books = ({ books = [], onDelete = (f) => f, onToggleLike = (f) =>
f }) => {
return (
<section>
{!books.length ? (
<h2>There are no books available.</h2>
) : (
<>
<h3>Showing {books.length} books.</h3>
<table className="table">
<thead>
<tr>
<th>Title</th>
<th>Author</th>
<th>Number in Stock</th>
<th>Price</th>
<th>Rating</th>
<th>Publish Year</th>
<th>Like</th>
<th>Actions(s)</th>
</tr>
</thead>
<tbody>
{books.map((book) => (
<Book key={book._id} book={book} onDelete={onDelete} onToggleL-
ike={onToggleLike} />
))}
</tbody>
</table>
</>
)}
</section>
);
};
export default Books;
```

Client > components / book.jsx

```
import { FaHeart, FaRegHeart, FaTrash, FaEdit } from "react-icons/-
fa";
import { Link } from "react-router-dom";
const Book = ({ book, onDelete = (f) => f, onToggleLike = (f) => f }) =>
{
const formatYear = () => {
return new Date(book.publishYear).toISOString().slice(0, 10);
};
return (
<tr>
<td>{book.title}</td>
```

```
import { useState } from "react";
import { useNavigate } from "react-router-dom";
const AddBook = ({ onAdd = (f) => f }) => {
const [book, setBook] = useState({
title: "",
author: "",
numberInStock: "",
price: "",
rating: "",
publishYear: "",
like: "",
});
const navigate = useNavigate();
const submitForm = (e) => {
e.preventDefault();
onAdd(book);
setBook({
title: "",
author: "",
numberInStock: "",
price: "",
rating: "",
publishYear: "",
like: "",
});
navigate("/");
```

Client > components / addBook.jsx (cont)

```
};
return (
<div>
<h1>Please enter the details of the new Book here..</h1>
<form onSubmit={submitForm} method="post">
<div className="form-group">
<label htmlFor="title" className="p-3">
Book Title
</label>
<input
type="text"
name="title"
id="title"
className="form-control"
placeholder="Enter a Title"
onChange={(event) => setBook({ ...book, title: event.target.value })}
value={book.title}
required
/>
</div>
<div className="form-group">
<label htmlFor="author" className="p-3">
Book Author
</label>
<input
type="text"
name="author"
id="author"
className="form-control"
placeholder="Enter an Author"
onChange={(event) => setBook({ ...book, author: event.target.value
})}
value={book.author}
required
/>
</div>
<div className="form-group">
<label htmlFor="numberInStock" className="p-3">
Number In Stock
</label>
<input
type="number"
name="numberInStock"
id="numberInStock"
className="form-control"
placeholder="Number In Stock"
onChange={(event) => setBook({ ...book, numberInStock: event.tar-
get.value })}
value={book.numberInStock}
required
/>
</div>
```

Client > components / addBook.jsx (cont)

```
/>
</div>
<div className="form-group">
<label htmlFor="rating" className="p-3">
Book Rating
</label>
<input
type="number"
name="rating"
id="rating"
className="form-control"
placeholder="Enter a Rating"
onChange={(event) => setBook({ ...book, rating: event.target.value
})}
value={book.rating}
required
/>
</div>
<div className="form-group">
<label htmlFor="publishYear" className="p-3">
Book Publish Year
</label>
<input
type="date"
name="publishYear"
id="publishYear"
className="form-control"
placeholder="Enter a Publish Year"
onChange={(event) => setBook({ ...book, publishYear: event.tar-
get.value })}
value={book.publishYear}
required
/>
</div>
<div className="form-group">
<label className="p-3 d-block">
Like Status (True/False)
</label>
<div className="form-check form-check-inline">
<input
type="radio"
name="like"
id="like-true"
className="form-check-input"
checked={book.like === true}
onChange={() => setBook({ ...book, like: true })}
required
/>
<label htmlFor="like-true" className="form-check-label">
True
</label>
</div>
```

```
<div className="form-group">
<label htmlFor="price" className="p-3">
Book Price
</label>
<input
type="text"
name="price"
id="price"
className="form-control"
placeholder="Enter a Price"
onChange={(event) => setBook({ ...book, price: event.target.value })}
value={book.price}
required
```

```
<div className="form-check form-check-inline">
<input
type="radio"
name="like"
id="like-false"
className="form-check-input"
checked={book.like === false}
onChange={() => setBook({ ...book, like: false })}
required
/>
<label htmlFor="like-false" className="form-check-label">
False
</label>
</div>
</div>
<button className="m-5 p-3 w-25">ADD THE BOOK</button>
</form>
</div>
);
```

Client > components / addBook.jsx (cont)

```
};
export default AddBook;
```

Client > components / updateBook.jsx

```
import { getBookUrl } from "../config/api";
const UpdateBook = ({ onUpdate = (f) => f }) => {
const [book, setBook] = useState({
title: "",
author: "",
numberInStock: "",
price: "",
rating: "",
publishYear: "",
like: "",
});
const { id } = useParams();
useEffect(() => {
const fetchData = async () => {
try {
const { data } = await axios.get(getBookUrl(id));
const date = new Date(data.publishYear).toISOString().slice(0, 10);
setBook({ ...data, publishYear: date }); //data is the existing book in
the DB
} catch (err) {
console.log(There is an ERROR in fetching ${err.m es s
age});
}
};
fetchData();
return () => {
//clean up
};
}, [id]);
const navigate = useNavigate();
const submitForm = (e) => {
e.preventDefault();
onUpdate(book);
setBook({
title: "",
author: "",
numberInStock: "",
price: "",
rating: "",
publishYear: "",
like: "",
```

Client > components / updateBook.jsx (cont)

```
});
navigate("/");
};
return (
<div>
<h1>Please enter the details of the updated Book here..</h1>
<form onSubmit={submitForm} method="post">
<div className="form-group">
<label htmlFor="title" className="p-3">
Book Title
</label>
<input
type="text"
name="title"
id="title"
className="form-control"
placeholder="Enter a Title"
onChange={(event) => setBook({ ...book, title: event.target.value })}
value={book.title}
required
/>
</div>
<div className="form-group">
<label htmlFor="author" className="p-3">
Book Author
</label>
<input
type="text"
name="author"
id="author"
className="form-control"
placeholder="Enter an Author"
onChange={(event) => setBook({ ...book, author: event.target.value
})}
value={book.author}
required
/>
</div>
<div className="form-group">
<label htmlFor="numberInStock" className="p-3">
Number In Stock
</label>
<input
type="number"
name="numberInStock"
id="numberInStock"
className="form-control"
placeholder="Number In Stock"
onChange={(event) => setBook({ ...book, numberInStock: event.tar-
get.value })}
value={book.numberInStock}
required
```

```
/>
</div>
<div className="form-group">
<label htmlFor="price" className="p-3">
Book Price
</label>
<input
type="text"
name="price"
id="price"
className="form-control"
placeholder="Enter a Price"
onChange={(event) => setBook({ ...book, price: event.target.value })}
```

Client > components / updateBook.jsx (cont)

```
value={book.price}
required
/>
</div>
<div className="form-group">
<label htmlFor="rating" className="p-3">
Book Rating
</label>
<input
type="number"
name="rating"
id="rating"
className="form-control"
placeholder="Enter a Rating"
onChange={(event) => setBook({ ...book, rating: event.target.value
})}
value={book.rating}
required
/>
</div>
<div className="form-group">
<label htmlFor="publishYear" className="p-3">
Book Publish Year
</label>
<input
type="date"
name="publishYear"
id="publishYear"
className="form-control"
placeholder="Enter a Publish Year"
onChange={(event) => setBook({ ...book, publishYear: event.tar-
get.value })}
value={book.publishYear}
required
/>
</div>
<div className="form-group">
<label className="p-3 d-block">Like Status (True/False)</label>
<div className="form-check form-check-inline">
<input
type="radio"
name="like"
id="like-true"
className="form-check-input"
checked={book.like === true}
onChange={() => setBook({ ...book, like: true })}
required
/>
<label htmlFor="like-true" className="form-check-label">
True
</label>
</div>
```

Client > components / updateBook.jsx (cont)

```
<button className="m-5 p-3 w-25">UPDATE THE BOOK</button>
</form>
</div>
);
};
export default UpdateBook;
```

Client > navBar.jsx

```
import { Link } from "react-router-dom";
const NavBar = () => {
return (
<nav className="navbar navbar-expand-lg navbar-light bg-light">
<div className="collapse navbar-collapse" id="navbarNav">
<ul className="navbar-nav">
<li className="nav-item active">
<Link className="nav-link" to="/">
My Book Library
</Link>
</li>
<li className="nav-item">
<Link className="nav-link" to="/">
Books
</Link>
</li>
<li className="nav-item">
<Link className="nav-link" to="addbook">
Add a Book
</Link>
</li>
</ul>
</div>
</nav>
);
};
export default NavBar;
```

Server > config / db.js

```
<div className="form-check form-check-inline">
<input
type="radio"
name="like"
id="like-false"
className="form-check-input"
checked={book.like === false}
onChange={() => setBook({ ...book, like: false })}
required
/>
<label htmlFor="like-false" className="form-check-label">
False
</label>
</div>
</div>
```

```
import mongoose from "mongoose";
import { url } from "../atlas_uri.js";
export const connectDB = async () => {
try {
await mongoose.connect(url);
console.log("Connected to DB");
} catch (err) {
console.log(ERROR in connec ting: ${err.m es sage});
process.exit(1); //exit in case of failure
}
};
export const disconnectDB = async () => {
await mongoose.connection.close();
console.log("Connection closed");
};
```

Server > models / books.js

```
import mongoose from "mongoose";
const bookSchema = new mongoose.Schema({
title: {
type: String,
required: [true, "Please check your entry, no title specified."],
},
author: {
type: String,
required: [true, "Please check your entry, no author specified."],
},
```

Server > models / books.js (cont)

```
numberInStock: {
type: Number,
required: [true, "Number in Stock is missing. Please provide a value."],
},
price: Number,
rating: {
type: Number,
min: 1,
max: 10,
required: [true, "The rating is required, please specify a value between 1-100."],
},
publishYear: {
type: Date,
default: Date.now, // No parentheses to ensure execution at creation time
},
like: Boolean,
});
export const Book = mongoose.model("Book", bookSchema);
```

Server > server.js

```
import express from "express";
import { connectDB } from "./config/db.js";
import router from "./routes/basicRoutes.js";
import cors from "cors";
//1. create an express application
const app = express();
//2. set up all the middlewares
app.use(express.urlencoded({ extended: true }));
app.use(express.json());
app.use(express.static("public"));
app.use(cors());
//3. connect to the database
connectDB();
//4. set up the routes
app.use("/", router);
//5. make your server listen to port 3000
const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
console.log(`The server is up and listening on PORT${ PORT}`);
});
```

Server > routes / basicRoutes.js

```
import express from "express";
import { Book } from "../models/books.js";
const router = express.Router();
//GET ALL BOOKS - READ OPERATION
router.get("/books", async (req, res) => {
try {
const books = await Book.find();
if (books.length === 0) {
return res.status(404).json({ error: "There are no books." });
}
res.json(books);
} catch (err) {
//500 --> Internal Server Error
res.status(500).json({ error: err.message });
}
});
//GET ONE BOOK - READ OPERATION
router.get("/books/:id", async (req, res) => {
const { id } = req.params;
try {
// const book = await Book.findById(id);
// const book = await Book.findOne({"title":"Eloquent JavaScript"});
const book = await Book.findOne({ _id: id });
if (!book) {
return res.status(404).json({ error: "There are no such book." });
}
res.json(book);
} catch (err) {
//500 --> Internal Server Error
res.status(500).json({ error: err.message });
}
});
//POST BOOK - CREATE OPERATION
router.post("/books", async (req, res) => {
try {
const book = new Book(req.body);
const savedBook = await book.save();
res.status(201).json(savedBook);
} catch (err) {
//500 --> Internal Server Error
res.status(500).json({ error: err.message });
}
```

Server > routes / basicRoutes.js (cont)

```
});
//PUT BOOK - UPDATE OPERATION
router.put("/books/:id", async (req, res) => {
const { id } = req.params;
try {
const upatedBook = await Book.findByIdAndUpdate(id, req.body, {
new: true, runValidators: true });
if (!upatedBook) {
return res.status(404).json({ error: "There are no such book." });
}
res.json(upatedBook);
} catch (err) {
//500 --> Internal Server Error
res.status(500).json({ error: err.message });
}
});
//DELETE BOOK - DELETE OPERATION
router.delete("/books/:id", async (req, res) => {
const { id } = req.params;
try {
const deletedBook = await Book.findByIdAndDelete(id);
if (!deletedBook) {
return res.status(404).json({ error: "There are no such book." });
}
res.json(deletedBook);
} catch (err) {
//500 --> Internal Server Error
res.status(500).json({ error: err.message });
}
});
export default router;
```