Bookish shelf Api

* A RESTful API implementation using React as the frontend
* In-memory storage for the book data
* A beautiful UI that showcases all CRUD operations
* Clean forms for adding and editing books
* Responsive design that works well on all devices

Let's use colors that evoke a cozy library feel: warm browns, creams, and a touch of teal for accents. The typography will be elegant yet readable, perfect for a book management system.

I've created a beautiful book management application with a full REST API implemented in React. The app includes all the requested features: adding, viewing, updating, and deleting books, with in-memory storage. The interface is clean and responsive with a warm teal color scheme reminiscent of a modern library.

# Step 1: Clone the repository using the project's Git URL.

git clone <YOUR\_GIT\_URL>

# Step 2: Navigate to the project directory.

cd <YOUR\_PROJECT\_NAME>

# Step 3: Install the necessary dependencies.

npm i

# Step 4: Start the development server with auto-reloading and an instant preview.

npm run dev

**What technologies are used for this project?**

This project is built with:

* Vite
* TypeScript
* React
* shadcn-ui
* Tailwind CSS

A PostgreSQL REST API is a web service that uses HTTP requests to enable an application to interact with a PostgreSQL database. It uses standard HTTP methods such as GET, POST, PUT, and DELETE instead of traditional database querying languages such as SQL.

PostgreSQL is a powerful, open-source, object-relational database system known for its reliability, features, and performance. It's designed for enterprise-level use and supports both SQL (relational) and JSON (non-relational) querying. PostgreSQL is a popular choice for many applications, including web applications, data warehousing, and analytical reporting.

## REST API Architecture

Representational State Transfer (REST) employs stateless client-server communication using standard HTTP methods (GET, POST, PUT, DELETE)[3](https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest). A RESTful service returns data in JSON (83% usage) or XML format, with caching mechanisms to optimize performance[3](https://www.redhat.com/en/topics/integration/whats-the-difference-between-soap-rest)[7](https://dev.to/anticoder03/building-restful-apis-with-nodejs-and-express-step-by-step-tutorial-2oc6). Its uniform interface constraint requires resource identification through URIs and hypermedia controls for discoverability

Example REST request flow:

text

Client -> GET /users/123 HTTP/1.1

Server <- 200 OK {"id":123,"name":"Alice"}

## GraphQL Query Language

GraphQL introduces a typed schema system that lets clients request exactly the data they need through single endpoints[4](https://graphql.org/learn/). Unlike REST's multiple endpoints, GraphQL uses a query language:

graphql

**query** {

user(id:123) {

name

posts(limit:5) {

title

}

}

}

This returns nested data in one request, reducing over-fetching. Apollo Server benchmarks show GraphQL reducing mobile data usage by 40% compared to REST

bookshelf@123

Mohana.s123

C:\Users\User\Desktop\Project-1\project-bolt-sb1-biwphyoj\project