**Project: Global Book Catalog RESTful API**

**Overview**

**Build a secure, scalable RESTful API using Node.js and Express.js for managing a global catalog of books. The API supports user registration, login, book management, searching, reviewing, and admin moderation. No UI development will be required.**

**Features & Requirements**

**1. User Management**

**- Register: Users can sign up with email and password.**

**- Login: JWT-based authentication.**

**- Profile: View and update user profile.**

**- Roles: User and Admin roles.**

**2. Book Management**

**- Add Books: Authenticated users can add books (title, author, genre, year, description).**

**- Edit/Delete Books: Only the user who added the book or an admin can edit/delete.**

**- List/Search Books: Search by title, author, genre, or year.**

**3. Reviews**

**- Add Review: Authenticated users can review books (rating, comment).**

**- Edit/Delete Review: Only the review author or admin can edit/delete.**

**- List Reviews: View all reviews for a book.**

**4. Admin Features**

**- Manage Users: List, update, or delete users.**

**- Moderate Content: Remove inappropriate books/reviews.**

**Admin Panel Features**

* Implement searching and sorting in admin panel listings
* Configure and manage Access Control Lists (ACL) within the admin panel
* Manage users and books, including image upload functionality
* Implement filtering, searching, and sorting combined for data lists
* Update and manage user access permissions through ACL

**Security Concepts**

* Implement password security using salting techniques

**5. Security**

**- JWT Authentication: Secure endpoints.**

**- Password Hashing: Use bcrypt.**

**- Input Validation: Prevent SQL/NoSQL injection, XSS.**

**- Rate Limiting: Protect against brute force.**

**6. Scalability & Performance**

**- Pagination: For book/review lists.**

**- Caching: Use Redis for popular searches.**

**- Clustering: Use Node.js cluster for scaling.**

**7. Testing**

**- Unit & Integration Tests: For all endpoints.**

**8. Documentation**

**- Swagger/OpenAPI: API documentation.**

**9. Deployment**

**- Docker: Containerize the app.**

**- Cloud Deployment: Deploy to GCP Cloud Run.**

**---**

**Suggested Tech Stack**

**- Backend: Node.js, Express.js**

**- Database: MongoDB (Mongoose) or PostgreSQL (Sequelize)**

**- Auth: JWT, bcrypt**

**- Cache: Redis**

**- Testing: Jest, Supertest**

**- Docs: Swagger**

**- Deployment: Docker, GCP Cloud Run**

**---**

**Project Structure Example**

**```**

**/src**

**/controllers**

**/models**

**/routes**

**/middleware**

**/utils**

**/tests**

**app.js**

**Dockerfile**

**swagger.yaml**

**.env**

**```**

**---**

**Milestones**

**1. Setup & Auth: Project scaffolding, user registration/login, JWT.**

**2. Book CRUD: Add, edit, delete, list, search books.**

**3. Reviews: Add, edit, delete, list reviews.**

**4. Admin: User management, content moderation.**

**5. Security: Validation, rate limiting.**

**6. Testing: Write unit/integration tests.**

**7. Docs & Deployment: Swagger docs, Dockerize, deploy to GCP.**

**---**

**Example Endpoints**

**- `POST /api/register` – Register user**

**- `POST /api/login` – Login user**

**- `GET /api/books` – List/search books**

**- `POST /api/books` – Add book**

**- `PUT /api/books/:id` – Edit book**

**- `DELETE /api/books/:id` – Delete book**

**- `POST /api/books/:id/reviews` – Add review**

**- `GET /api/books/:id/reviews` – List reviews**

**- `GET /api/admin/users` – List users (admin)**

**- `DELETE /api/admin/reviews/:id` – Delete review (admin)**

**---**