Book Inventory management

Introduction

📚 Welcome to "Efficient Inventory Management System: A Real-time Spring Boot Project." This book is designed to be your comprehensive guide to developing a cutting-edge inventory management system using the powerful and versatile Spring Boot framework. Whether you are a seasoned developer, a tech enthusiast, or a business professional looking to optimize your organization's inventory processes, this book will equip you with the knowledge and skills needed to create a reliable and scalable solution.

Objective

📌 Development of Book inventory Management Portal -The portal is designed to address the challenges associated with manual bookkeeping, offering a comprehensive solution that caters to the specific needs of book-related businesses, libraries, and educational institutions.

Database Schema:

▼ ER Diagram

https://prod-files-secure.s3.us-west-2.amazonaws.com/31282dbc-a6a3-454c-930f-bedcf45363ec/ee369a4c-1b25-4f4c-9f25-8d7c0026c008/BookInventoryManagement.pdf

▼ MySQL Script file

createBOOK.sql

table_wise.sql

REST API Design:

All REST API requests and responses follow the JSON format. The project employs standard HTTP status codes (200, 201, 400, 404, 401, 500, 505) appropriately in response objects to convey the outcome of API requests.

Http Status Code:

HTTP Status Code	Error Response
400	Bad Request
401	Unauthorized
404	Not Found
500	Internal Server Error
505	HTTP Version Not Supported

Working with GitHub:

▼ **Create a New Repository on GitHub:**

- Log in to your GitHub account.
- Click on the "+" icon in the top right corner and select "New repository."
- Provide a name, description, and other relevant details.

▼ **Fork the Repository:**

- Go to the GitHub page of the repository you want to contribute to.
- Click on the "Fork" button in the upper right corner of the page.

▼ **Clone Your Forked Repository:**

Clone your forked repository to your local machine.

```
git clone https://github.com/your-username/repository.git
```

▼ **Make Changes, Commit, and Push:**

- Make the necessary changes to the code.
- Stage and commit your changes.

```
git add .
git commit -m "Your descriptive commit message"
git push
```

▼ **Create a Pull Request:**

- Go to the GitHub page of your forked repository.
- Switch to the branch you just pushed.
- Click on the "New Pull Request" button.

▼ **Review and Merge:**

- Describe your changes and submit the pull request.
- The repository owner or maintainers will review your changes.

▼ **Update Your Fork (Optional):**

To keep your fork up-to-date with the original repository, fetch and merge changes.

```
git fetch upstream
git merge upstream/main
```

Endpoints

Endpoints

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
_ /api/author/post _	POST	Add new Author object in DB	{ "code": "POSTSUCCESS", "message": "Author added successfully" }	{ "code": ADDFAILS", "message":Author already exist" }
_ /api/author/{authorId} _	GET	Get author object with given Author ID	Author object	
_ /api/author/firstname/{firstname} _	GET	Get author with given first name	Author object	
_ /api/author/lastname/{lastname} _	GET	Get author with given last name	Author object	
_	PUT	Update first		

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
/api/author/update/firstname/{authorId}		name for the given Author ID		
/api/author/update/lastname/{authorId}	PUT	Update last name for the given Author ID		
/api/author/books/{authorId}	GET	Get all books of an author	{Books}	
/api/publisher/post	POST	Add new publisher object	{ "code": "POSTSUCCESS", "message": "Publisher added successfully" }	{ "code": ADDFAILS", "message":Publisher already exist" }
/api/publishers	GET	Get all publishers	{Publisher}	
/api/publisher/{publisherId}	GET	Get publisher object with given ID	Publisher	
/api/publisher/name/{na	GET	Get publisher object with given	Publisher	

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
me} _		name		
_ /api/publisher/city/{city} _	GET	Get all publishers in a city	{Publisher}	
_ /api/publisher/update/state/{publisherId} _	GET	Get all publishers with given state name	{Publisher}	
_ /api/publisher/update/city/{publisherId} _	PUT	Update the publisher's name with given id		
_ /api/publisher/update/name/{publisherId} _	PUT	Update the publisher's city with given id		
_ /api/publisher/state/{state} _	PUT	Update the publisher's state with given state		
Untitled				
/api/book/post	POST	Add new Book object in DB	{ "code": "POSTSUCCESS",	{ "code": ADDFAILS",

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
			`"message": "Book added successfully" }`	`"message":Book already exist" }`
/api/books/	GET	Get all book object	{Books}	
/api/book/{isbn}	GET	Get Book Object	Book	
/api/book/title/{title}	GET	Get Book Object with title	Book	
/api/book/category/{category}	GET	Get Books of given category	{Books}	
/api/book/publisher/{publisherId}	GET	Get Books published by a publisher	{Books}	
/api/book/update/title/{isbn}	PUT	Update title of Book object with given ISBN		
Update title of Book object with given ISBN _	PUT	Update description of Book object with given ISBN		
/api/book/update/category/{isbn}	PUT	Update category of Book object with given ISBN		
/api/book/update/edition/{isbn}	PUT	Update edition of Book object with given ISBN		

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
/api/book/update/publisher/{isbn}	PUT	Update publisher Object of Book object with given ISBN		
AUTH				
New Endpoint				
New Endpoint				
Code				
/api/state/post	POST	Add new state object	{ "code": "POSTSUCCESS", "message": "State added successfully" }	{ "code": ADDFAILS", "message":State already exist" }
/api/states	GET	Get all states	{State}	
/api/state/update/name/{stateId}	GET	Get state with given state ID	State	
/api/state/{stateId}	PUT	Update state's name with given state ID		
/api/reviewer/post	POST	Add new reviewer object	{ "code": "POSTSUCCESS", "message": "Reviewer added successfully" }	{ "code": ADDFAILS", "message":Reviewer already exist" }

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
_ /api/reviewer/employedby/{reviewerId} _	GET	Get reviewer with given id	Reviewer	
_ /api/reviewer/name/{reviewerId} _	PUT	Update first name with given Id		
_ /api/reviewer/{reviewerId} _	PUT	Update employedby with given Id		
_ /api/user/post _	POST	Add new user object	{ "code": "POSTSUCCESS", "message": "User added successfully" }	{ "code": ADDFAILS", "message":User already exist" }
_ /api/user/{userId} _	GET	Get user with given id	User	
_ /api/user/update/phonenumber/{userId} _	PUT	Update first name with given Id		
_ /api/user/up	PUT	Update last name with		

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
date/lastname/{userId}		given Id		
/api/user/update/firstname/{userId}	PUT	Update phone number with given Id		
/api/bookreview/post	POST	Add new book review object	{ "code": "POSTSUCCESS", "message": "Book Reviewer added successfully" }	{ "code": ADDFAILS", "message":Book Reviewer already exist" }
/api/bookreview/{isbn}	GET	Get book review with given id	Book Review	
/api/bookreview/update/rating/{isbn}	PUT	Update rating with given Id		
/api/bookreview/update/comments/{isbn}	PUT	Update comments with given Id		
/api/inventory/post	POST	Add new inventory object	{ "code": "POSTSUCCESS", "message": "Inventory added successfully" }	{ "code": ADDFAILS", "message":Inventory already exist" }

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
/api/inventory/update/purchased/{inventoryId}	GET	Get Inventory with given id	Inventory	
/api/inventory/{inventoryId}	PUT	Update purchased with given Id		
/api/bookcondition/post	POST	Add new book condition object	{ "code": "POSTSUCCESS", "message": "Book Condition added successfully" }	{ "code": ADDFAILS", "message":Book condition already exist" }
/api/bookcondition/update/price/{ranks}	GET	Get book condition with given ranks	Book condition	
/api/bookcondition/update/fullDescription/{ranks}	PUT	Update description with given ranks		
/api/bookcondition/update/descriptio	PUT	Update full description with		

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
n/{ranks}		given ranks		
/api/bookcondition/{ranks}	PUT	Update price with given ranks		
/api/category/post	POST	Add new category object	String: category added successfully	
/api/category/update/description/{catId}	GET	Get category with given id	Category	
/api/category/{catId}	PUT	Update description with given Id	Category	
/api/shoppingcart/post	POST	Add new shopping cart object	String: Shopping cart added successfully	
/api/ shoppingcart /{userId}	GET	Get cart with given id	Shopping cart	
/api/ shoppingcart /update/isbn/{userId}	PUT	Update isbn with given Id	Shopping cart	

Endpoint	HTTP Verb/Method	Description	SuccessResponse	Errorresponse
/api/purchaselog/post	POST	Add new purchaselog object	String: Purchase Log added successfully	
/api/purchaselog/{userId}	GET	Get purchase log for the given userid	Purchase Log	
/api/purchaselog/update/inventoryid/{userId}	PUT	Update inventory Id of purchase log given user id	Purchase log	
/api/permrole/post	POST	Add new permrole object	{ "code": "POSTSUCCESS", "message": "Perm Role added successfully" }	{ "code": ADDFAILS", "message":Perm Role already exist" }
/api/permrole/update/permeole/{rolenumber}	GET	Get permrole object for the given rollnumber	Permrole object	
/api/permrole/{rolenumber}	PUT	Update perm role of a given role number		