**Library Management App**

Library Management API helps you automate library functions in a secure way. This API provides a good exposure to its functionalities Even for a layman and it makes it very handy to deal with Users ,Librarians and Books' data in back-end. 🚀

**Library Management API**

Operations that can be performed:

* **Create users**
* **Create Librarians**
* **Get Book details from DB**
* **Inserting New Books in DB**
* **Update users' credentials( Refreshing of Tokens )**
* **Delete users**
* **Issuing book**

### **POST**

**Post request for signing up of new users**

URL: <http://127.0.0.1:8000/users>

this request is used for creating a new user in database

This endpoint lets you create a new user in database. As you provide Name , Password and Role fields as BODY() param. This POST request will insert a new user row in database schema.

**Parameters:**

**Request Body ( Required ): Example**

{

"Name": "Joelay field",

"Password": "any",

"Role": "Librarian"

}

**Responses**:

Code: Description

200 Successful Response

"string"

422 Validation Error

{

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

]

}

400 Invalid request message framing

404 Not Found

### **POST**

**Post request for user login**

URL: <http://127.0.0.1:8000/user/login>

This endpoint is to get refreshed tokens for registered users.

This endpoint will let you generate new refreshed jwt tokens for already registered users. incase if your token gets expired just use this request and get a new one.

**Parameters:**

Body Parameter:

Example Value:

{

"Name": "sitaram ",

"Password": "any"

}

**Responses**:

Code: Description

200 Successful Response

"string"

422 Validation Error

{

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

]

}

400 Invalid request message framing

404 Not Found

**DELETE:**

**DELETE request for user deletion**

URL: <http://127.0.0.1:8000/user/{user_id}/del>

**You can perform delete action for users.**

**Parameters:**

{Path Param} user\_id : String( ID of user)

**Responses**:

Code: Description

200 Successful Response

"string"

422 Validation Error

{

"detail": [

{

"loc": [

"string"

],

"msg": "string",

"type": "string"

}

]

}

400 Invalid request message framing

404 Not Found

**GET**

**GET request To see all books from database**

URL: <http://127.0.0.1:8000/book/all>

This endpoint lets you see all the books that are available in database. It exposes unique ID of books that can be used while issuing any book.

**Parameters:**

None:

**Responses**:

Code: Description

200 Successful Response

[

{

"Publication": "Gita Press",

"id": "fcc2aa6b-24a6-4324-a7b9-eaf2191eac2e",

"No\_of\_copies": 511,

"Subject": "Religious",

"Title": "sitaram",

"Author": "any"

}

]

|  |  |  |  |
| --- | --- | --- | --- |
|  | 400 Invalid request message framing  404 Resources Not available  **POST**  **POST request To Add Up book in database**  URL: <http://127.0.0.1:8000/book/add>  Librarians can use this API end point to add up books in database. Note that Other users are not authorized to perform this action even if they are authenticated.  **Requests:**  **{Query param}: user\_id: UUID(Id of user)**  **{Query param}: book\_id: UUID(Id of book)**  **{Query param}: return\_date: date( date when user will be returning book to library)**  **Responses:**  Code: Description  200 Successful Response  [  {  "Publication": "Gita Press",  "id": "fcc2aa6b-24a6-4324-a7b9-eaf2191eac2e",  "No\_of\_copies": 511,  "Subject": "Religious",  "Title": "sitaram",  "Author": "any"  }  ]   |  |  | | --- | --- | |  | 400 Invalid request message framing  404 Resources Not available  **POST**  **POST request To Issue book from database**  URL: http://127.0.0.1:8000/book/issue  Librarians can use this API end point to add up books in database. Note that Other users are not authorized to perform this action even if they are authenticated.  **Requests:**  **{Query param}: user\_id: UUID(Id of user)**  **{Query param}: book\_id: UUID(Id of book)**  **{Query param}: return\_date(YYYY/DD/MM format when user will be returning book to library)**  **Responses:**  Code: Description  200 Successful Response  {  "msg": "book has been issued for you"  }  422 Validation Error  {  "detail": [  {  "loc": [  "string"  ],  "msg": "string",  "type": "string"  }  ]  }  400 Invalid request message framing  404 Resources Not available  **POST**  **POST request To Return book to library database**  URL: http://127.0.0.1:8000/book/return  To return any book of library , use this API end point.  **Requests:**  **{Query param}: book\_id: UUID(Id of book)**  **Responses:**  Code: Description  200 Successful Response  {  "msg":"Your book has been returned. Thanks for using our Library:D"  }  422 Validation Error  {  "detail": [  {  "loc": [  "string"  ],  "msg": "string",  "type": "string"  }  ]  }  400 Invalid request message framing  404 Resources Not available  **DELETE**  **DELETE request To delete book from library database**  URL: http://127.0.0.1:8000/book/del  This endpoint lets you delete any book from database. As librarian provides book\_id as QUERY parameter the API shall delete it. Note that it's an irreversible process hence can't be undone.  **Requests:**  **{Query param}: book\_id: UUID(Id of book)**  **Responses:**  Code: Description  200 Successful Response  {  "Response":"successfully deleted"  }  422 Validation Error  {  "detail": [  {  "loc": [  "string"  ],  "msg": "string",  "type": "string"  }  ]  }  400 Invalid request message framing  404 Resources Not available |   **GET**  **GET request To see Issue\_table book from library database**  URL: http://127.0.0.1:8000/book/issue\_table  Any user entitled as librarian can use this endpoint to see records of users who has issued books from library. just provide skip and limit values as query parameters to filter out number of records that you want to see. and API GET request will handle the rest for you. but make sure you authenticate yourself as librarian first in API otherwise you won't be authorized to use it..  **Request:**  **{Query param}: skip: INT(Id of first record)**  **{Query param}: limit: INT(Id of last record)**  **Responses:**  Code: Description  200 Successful Response  {  “key”: “value”  }  422 Validation Error  {  "detail": [  {  "loc": [  "string"  ],  "msg": "string",  "type": "string"  }  ]  }  400 Invalid request message framing  404 Resources Not available  **PATCH**  **PATCH request To UPDATE book details in library database**  URL: http://127.0.0.1:8000/book/edit  This endpoint will let you update data of books available in db.  **Request:**  **{body Param example}:**  {  "Title": "sitaram",  "Author": "any",  "Publication": "Gita Press",  "Subject": "Religious",  "No\_of\_copies": "2"  }  **Responses:**  Code: Description  200 Successful Response  {  “key”: “value”  }  422 Validation Error  {  "detail": [  {  "loc": [  "string"  ],  "msg": "string",  "type": "string"  }  ]  }  400 Invalid request message framing  404 Resources Not available |
|  |  |
|  |  |