Library Management System API Documentation

▪ Project Description :-

The Library Management System API simplifies the management of library resources by providing a CRUD operations api on books, user, and Borrowing book details. This API enables you to create, read, update, and delete records within your library's database, making it easier to track and maintain your collection.

▪ Install dependency :- from requirement.txt

1. Djnago
2. Django-rest framework

▪ Endpoints :-

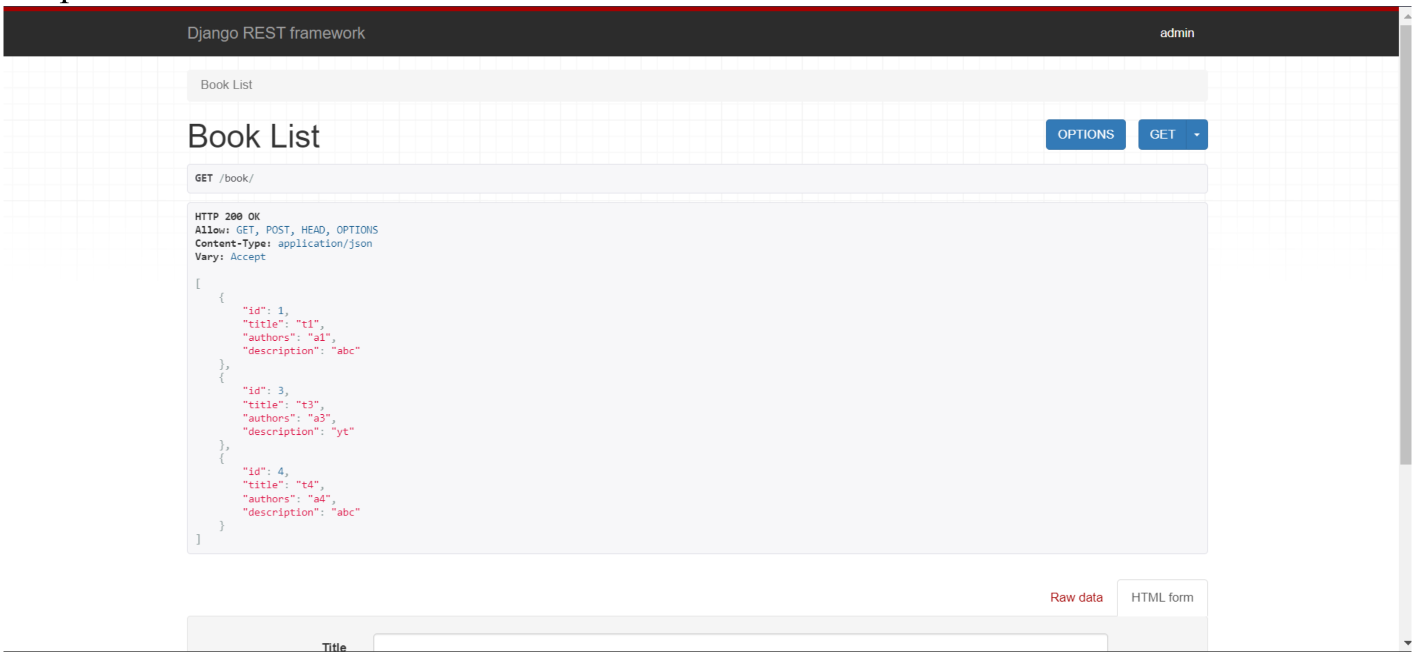
Books

GET/POST /api/books/

Add new book in library

Retrieve a list of all books in the library..

Response:



Book List

```
GET /book/
```

```
HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "title": "t1",
        "authors": "a1",
        "description": "abc"
    },
    {
        "id": 3,
        "title": "t3",
        "authors": "a3",
        "description": "yt"
    },
    {
        "id": 4,
        "title": "t4",
        "authors": "a4",
        "description": "abc"
    }
]
```

PUT /api/books/

Update/delete a book in the library.

Parameters:

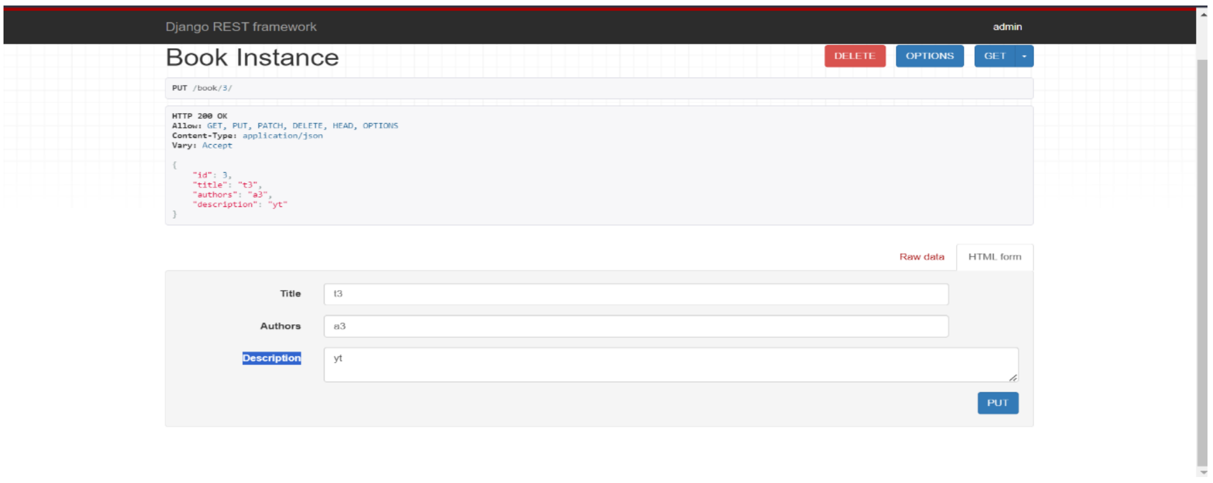
title (required): Title of the book.

author (required): Author of the book.

Description: Book Description

Response:

201 Created: New book details in JSON format. {"id":3,"title":"t3","authors":"a3","description":"yt"}



Book Instance

```
PUT /book/3/
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 3,
    "title": "t3",
    "authors": "a3",
    "description": "yt"
}
```

Title	t3
Authors	a3
Description	yt

<u>**User**</u>

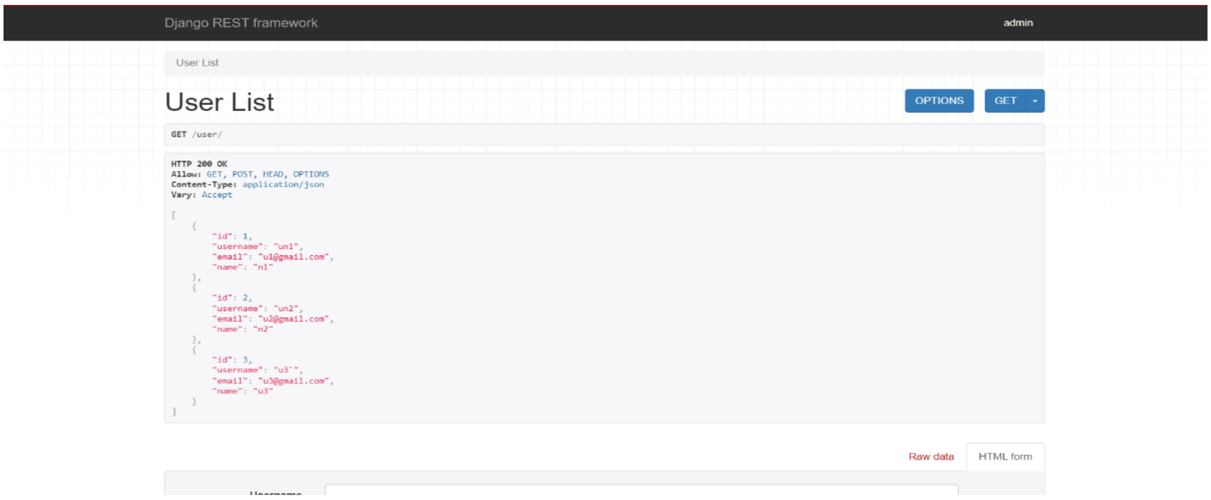
GET/POST /api/members/

Retrieve and create a list of all library user.

Response:

200 OK: List of user in JSON format.

Request:
[{"id":1,"username":"un1","email":"u1@gmail.com","name":"n1"},{"id":2,"username":"un2","email":"u2@gmail.com","name":"n2"},{"id":3,"username":"u3`","email":"u3@gmail.com","name":"u3"}]



```
GET /user/

HTTP 200 OK
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

[
    {
        "id": 1,
        "username": "un1",
        "email": "u1@gmail.com",
        "name": "n1"
    },
    {
        "id": 2,
        "username": "un2",
        "email": "u2@gmail.com",
        "name": "n2"
    },
    {
        "id": 3,
        "username": "u3`",
        "email": "u3@gmail.com",
        "name": "u3"
    }
]
```

PUT/DELETE /api/user/

Update and delete a new user.

Parameters:

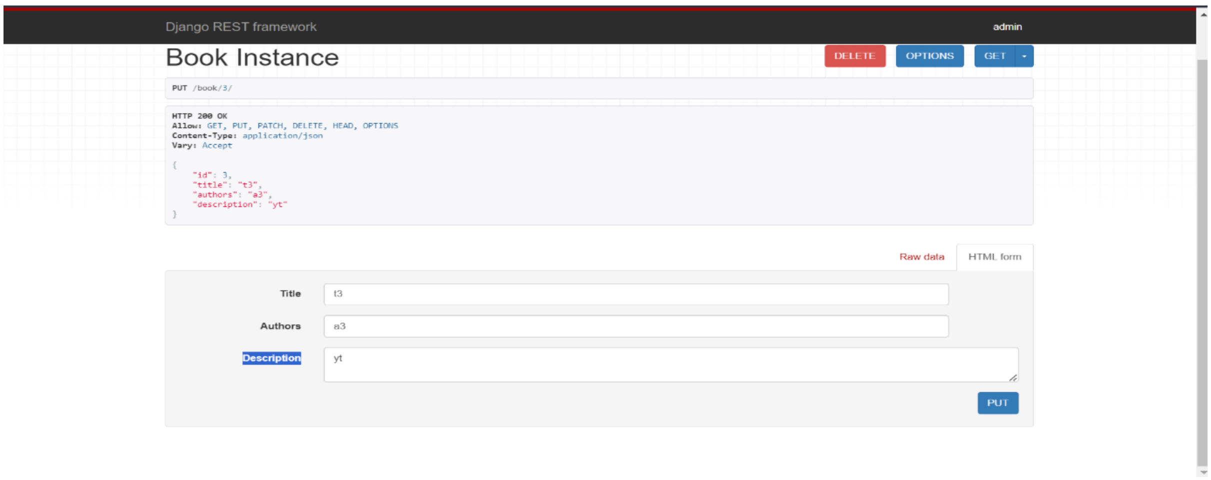
username (required):

 (required): Author of the book.

Description: Book Description

Response:

201 Created: New book details in JSON format. {"id":3,"title":"t3","authors":"a3","description":"yt"}



```
PUT /book/3/

HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 3,
    "title": "t3",
    "authors": "a3",
    "description": "yt"
}
```

Title	t3
Authors	a3
Description	yt

BorrowingBook

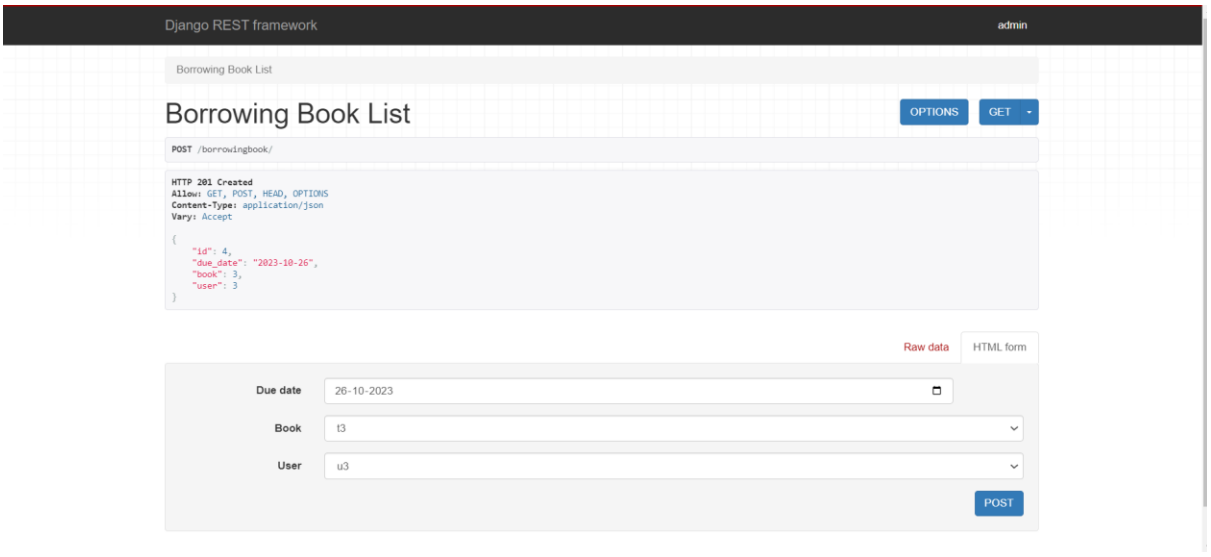
GET /POST/api/BorrowingBook/

Retrieve a list of all library borrowing book details.

Response:

200 OK: List of borrowing book details.in JSON format.

Request:
{"id":1,"due_date":"2023-10-30","book":1,"user":1}



Borrowing Book List

```
POST /borrowingbook/
```

```
HTTP 201 Created
Allow: GET, POST, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 4,
    "due_date": "2023-10-26",
    "book": 3,
    "user": 3
}
```

Due date: 26-10-2023

Book: t3

User: u3

PUT/DELETE /api/BorroewingBook/

Update and delete a new user.

Parameters:

Book id(required):

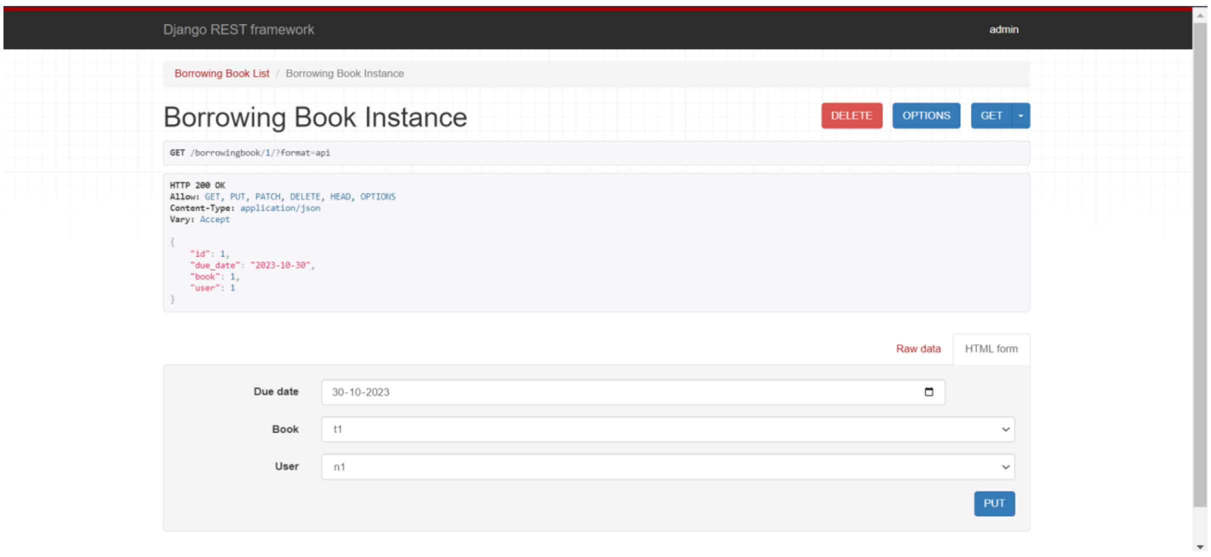
User if (required): Author of the book.

Due date ; return date of the book

Response:

201 Created: New book details in JSON format.

[{"id":1,"due_date":"2023-10-30","book":1,"user":1},{"id":3,"due_date":"2023-11-01","book":3,"user":3},{"id":4,"due_date":"2023-10-26","book":3,"user":3}]



Borrowing Book Instance

```
GET /borrowingbook/1/?format=api
```

```
HTTP 200 OK
Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS
Content-Type: application/json
Vary: Accept

{
    "id": 1,
    "due_date": "2023-10-30",
    "book": 1,
    "user": 1
}
```

Due date	30-10-2023
Book	t1
User	n1

- **<u>Things had been learned.:-</u>**

1. Working of API,
2. Serializers
3. Data conversion from python to Json
4. Viewset and generics diff.
5. Django rest Framework
6. django Rest Routerwork
7. HTTP methods
8. Pep 8