# DRF AUTHENTICATION  TASK

→You are building an authentication platform for a book management system using Django REST Framework. The platform allows users to perform CRUD operations on a Book model, which has the following fields:

● title: CharField, the title of the book

● description: TextField, a description of the book

● author: ForeignKey, the author of the book

● price: DecimalField, the price of the book

The authentication platform should support token authentication, where users can obtain a token by providing their email and password. The token should be sent with subsequent requests to authenticate the user.

Implement appropriate field types and validations for each field.

Your task is to create the necessary models, implement token authentication using

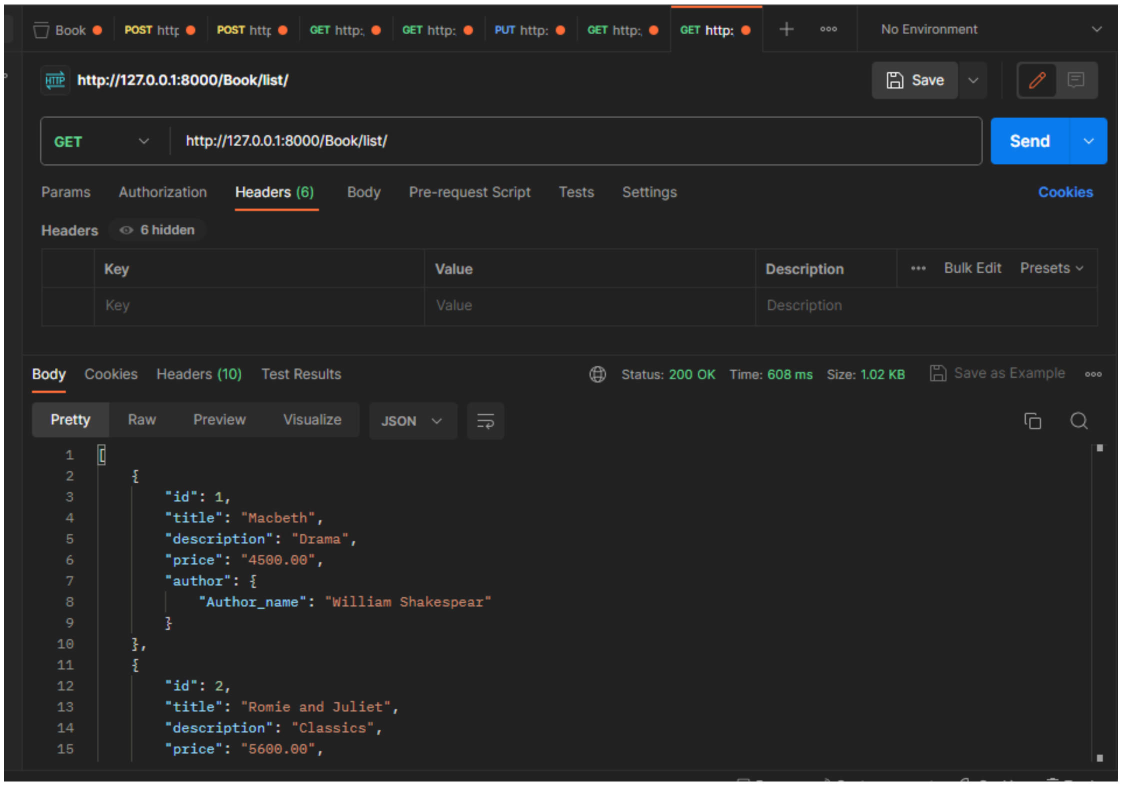
Django REST Framework, and provide CRUD operations for the Book model.

Please note that the Book detail and list APIs can be accessed by the public without

authentication.

Remember to include appropriate error handling, such as handling unauthorized access attempts and validating user permissions for CRUD operations on the books.

→public api

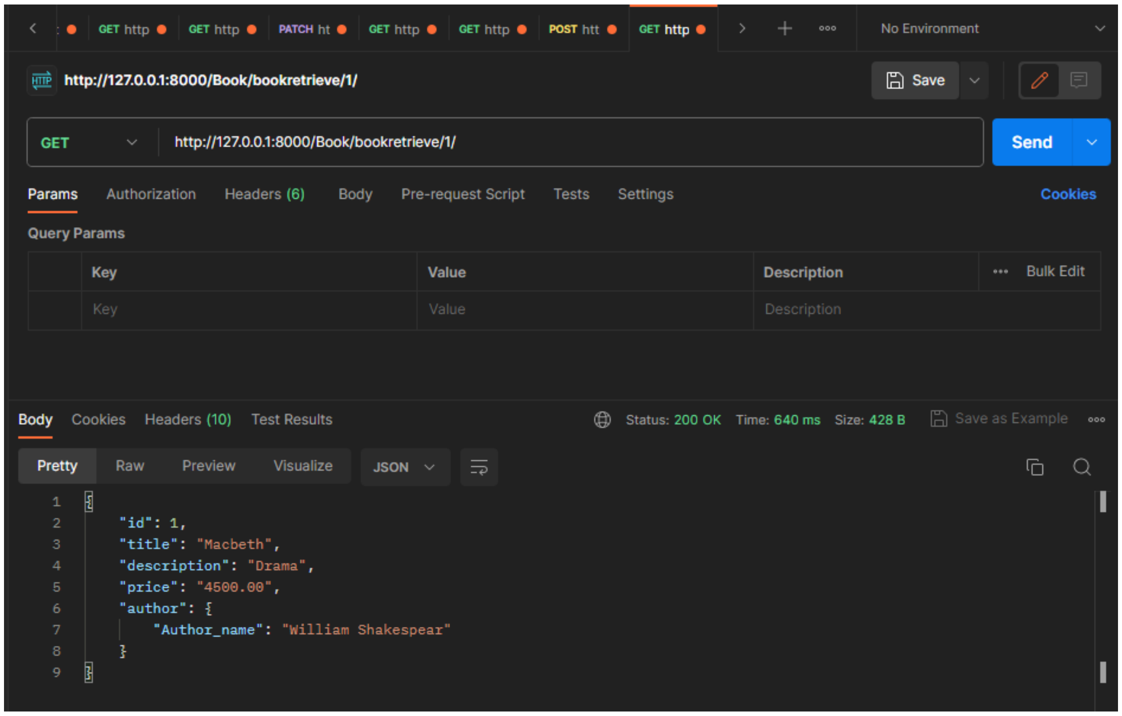

http://127.0.0.1:8000/Book/list/

GET http://127.0.0.1:8000/Book/list/

Status: 200 OK Time: 608 ms Size: 1.02 KB

```
[
    {
        "id": 1,
        "title": "Macbeth",
        "description": "Drama",
        "price": "4500.00",
        "author": {
            "Author_name": "William Shakespear"
        }
    },
    {
        "id": 2,
        "title": "Romie and Juliet",
        "description": "Classics",
        "price": "5600.00",
```

→Book retrieve

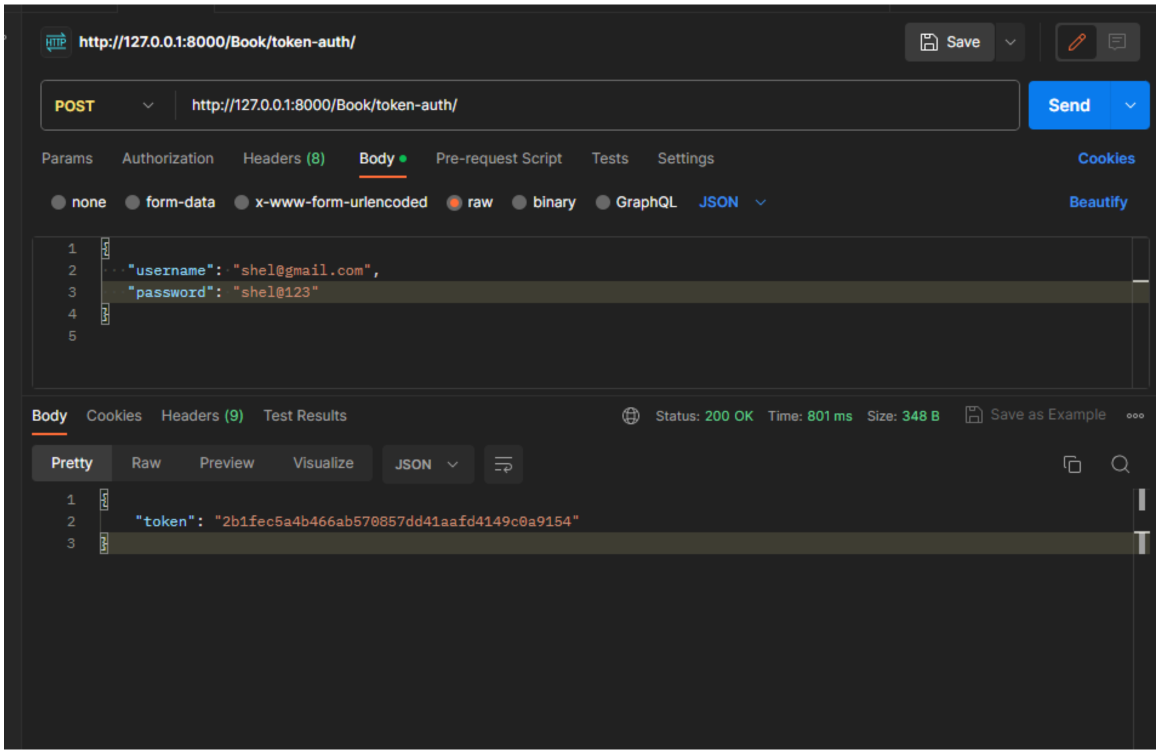

http://127.0.0.1:8000/Book/bookretrieve/1/

GET http://127.0.0.1:8000/Book/bookretrieve/1/

Status: 200 OK Time: 640 ms Size: 428 B

```
{
    "id": 1,
    "title": "Macbeth",
    "description": "Drama",
    "price": "4500.00",
    "author": {
        "Author_name": "William Shakespear"
    }
}
```

## →Token generation



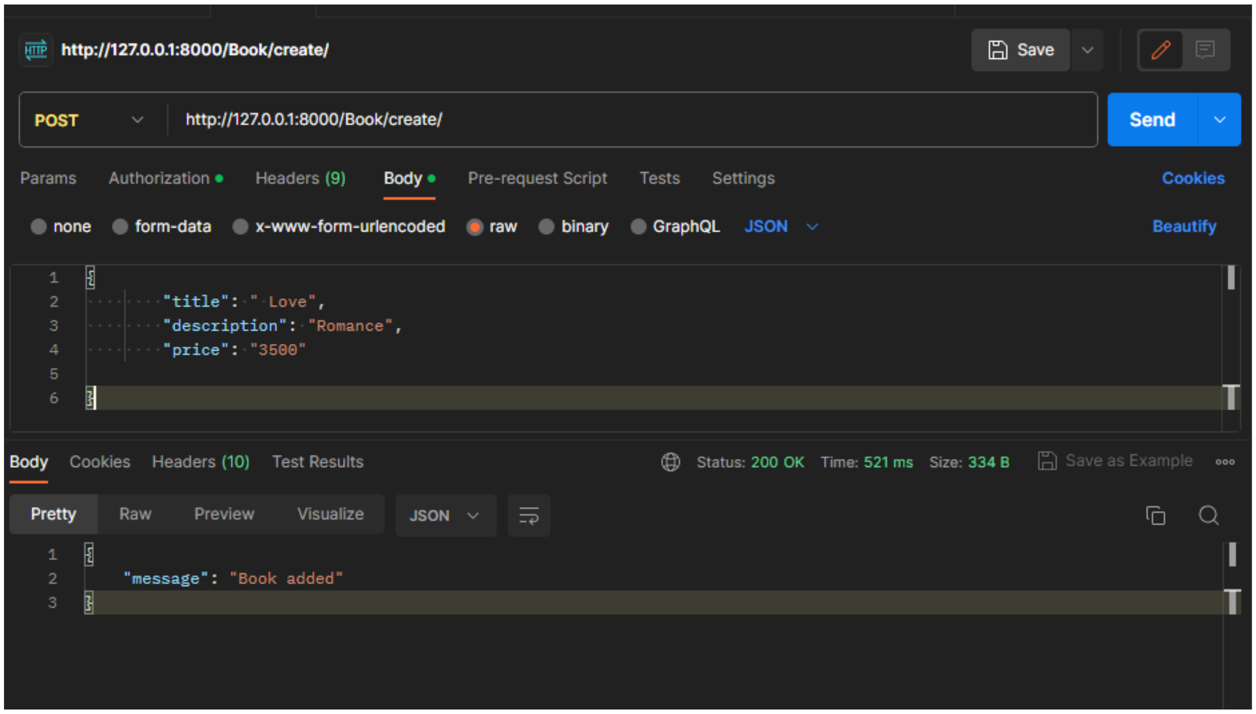
POST http://127.0.0.1:8000/Book/token-auth/

```
{
  "username": "shel@gmail.com",
  "password": "shel@123"
}
```

Status: 200 OK Time: 801 ms Size: 348 B

```
{
    "token": "2b1fec5a4b466ab570857dd41aafd4149c0a9154"
}
```

## →Book creation(AUTHARIZED)

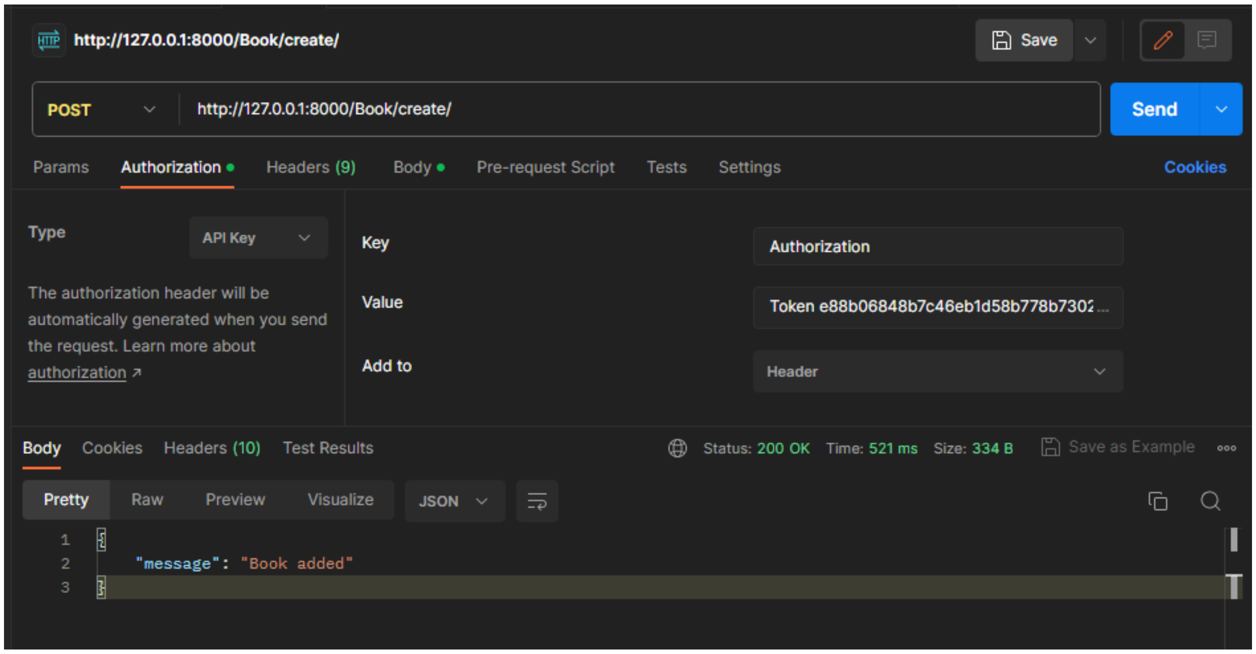


POST http://127.0.0.1:8000/Book/create/

```
{
        "title": " Love",
        "description": "Romance",
        "price": "3500"

}
```

Status: 200 OK Time: 521 ms Size: 334 B

```
{
    "message": "Book added"
}
```


http://127.0.0.1:8000/Book/create/

POST http://127.0.0.1:8000/Book/create/

Authorization

Type: API Key

The authorization header will be automatically generated when you send the request. Learn more about authorization

Key: Authorization

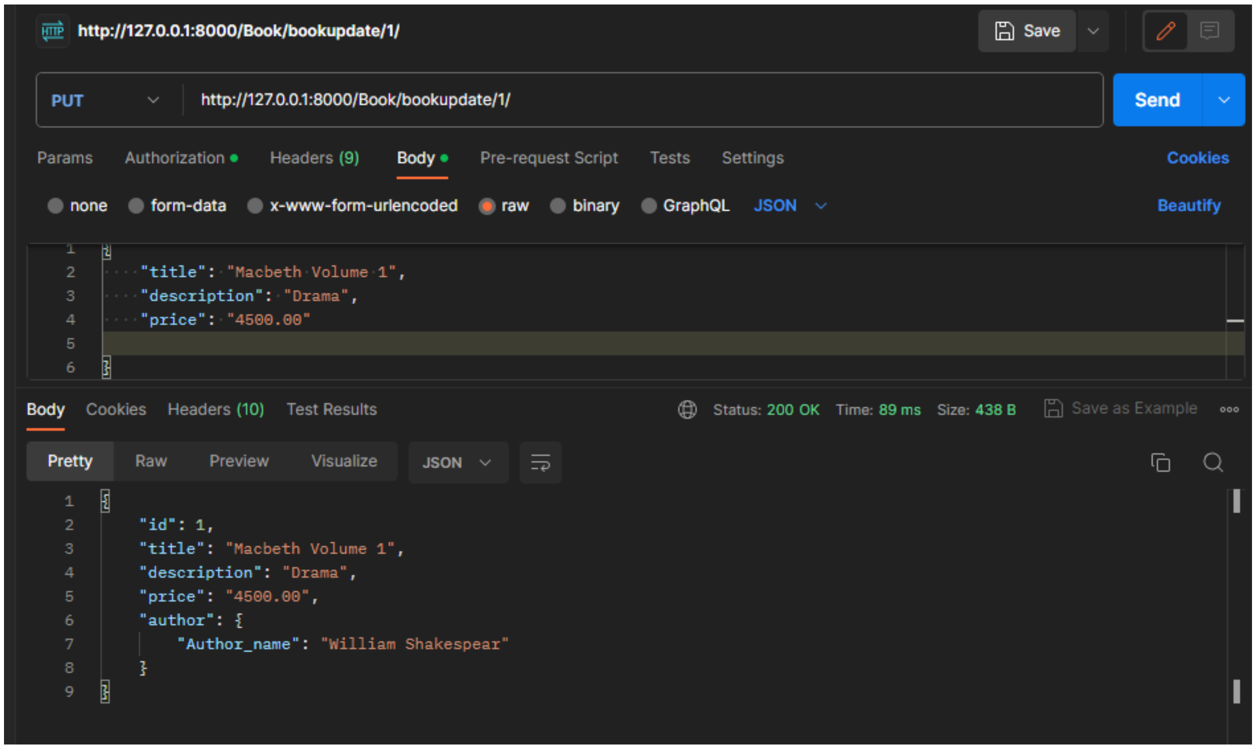
Value: Token e88b06848b7c46eb1d58b778b7302...

Add to: Header

Status: 200 OK Time: 521 ms Size: 334 B

```
{
    "message": "Book added"
}
```

→Book Updation

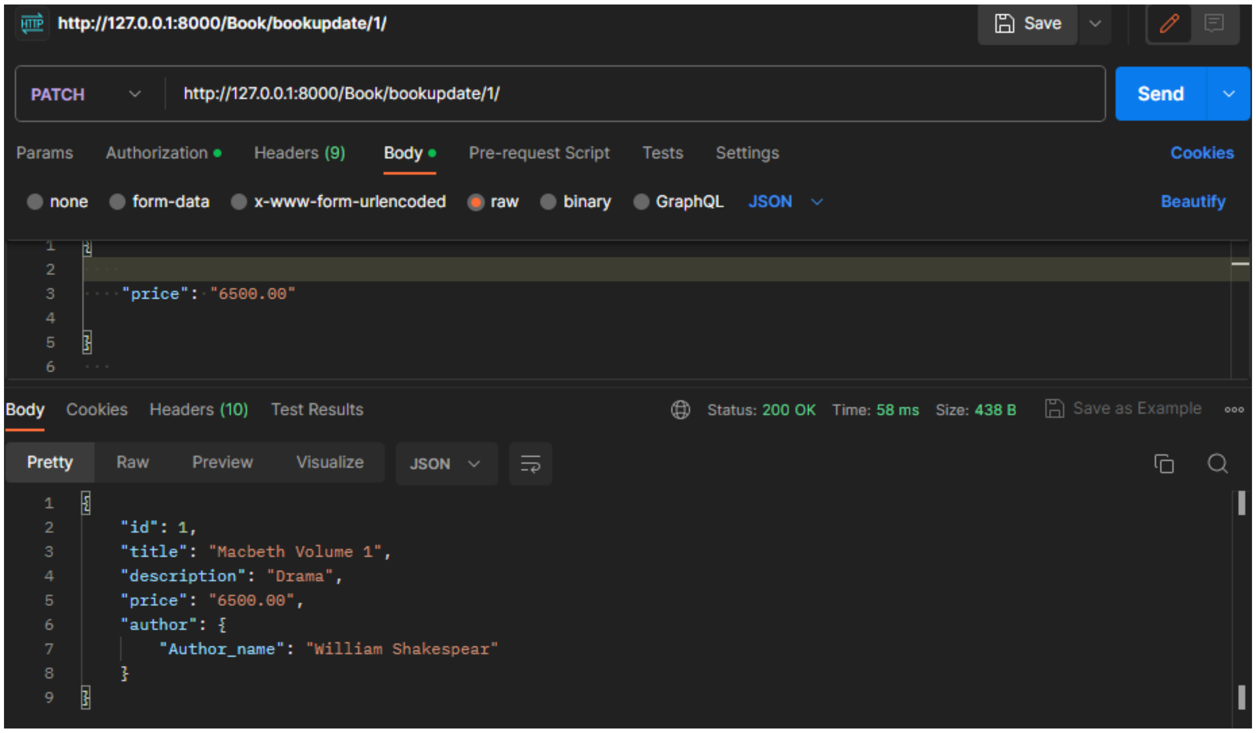

http://127.0.0.1:8000/Book/bookupdate/1/

PUT http://127.0.0.1:8000/Book/bookupdate/1/

```
{
    "title": "Macbeth Volume 1",
    "description": "Drama",
    "price": "4500.00"

}
```

Status: 200 OK Time: 89 ms Size: 438 B

```
{
    "id": 1,
    "title": "Macbeth Volume 1",
    "description": "Drama",
    "price": "4500.00",
    "author": {
        "Author_name": "William Shakespear"
    }
}
```


http://127.0.0.1:8000/Book/bookupdate/1/
Save
PATCH
Send
Params Authorization Headers (9) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL JSON Beautify
Body Cookies Headers (10) Test Results
Save as Example
Pretty Raw Preview Visualize JSON

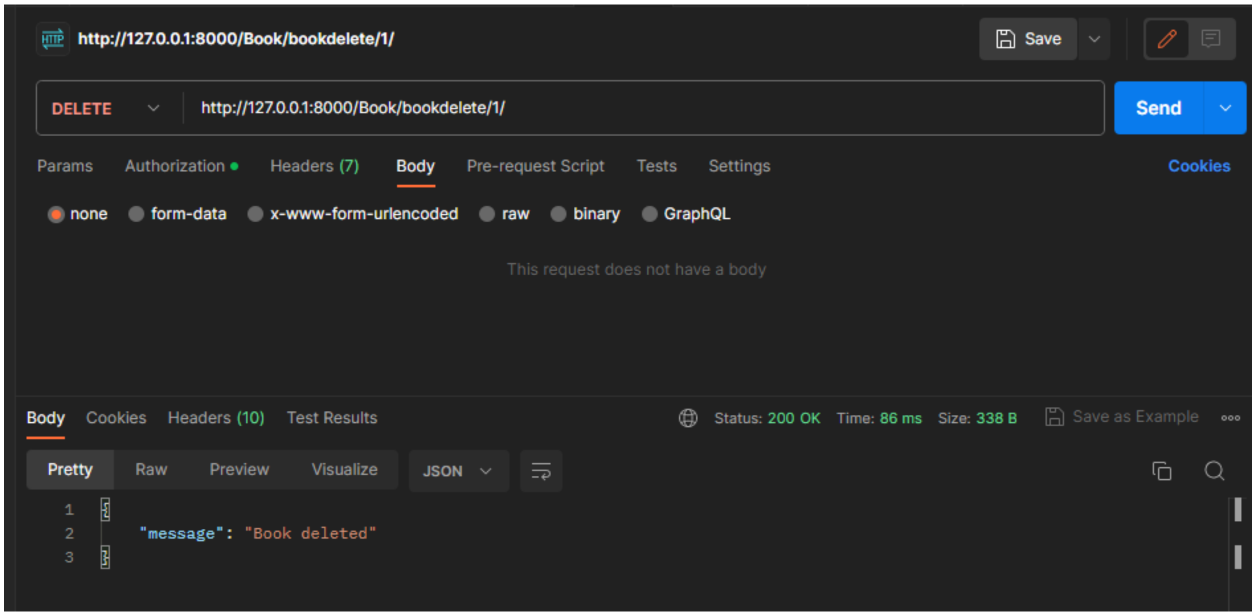
PATCH http://127.0.0.1:8000/Book/bookupdate/1/

```
{

    "price": "6500.00"

}
```

Status: 200 OK Time: 58 ms Size: 438 B

```
{
    "id": 1,
    "title": "Macbeth Volume 1",
    "description": "Drama",
    "price": "6500.00",
    "author": {
        "Author_name": "William Shakespear"
    }
}
```

→Book deletion


http://127.0.0.1:8000/Book/bookdelete/1/
Save
DELETE
Send
Params Authorization Headers (7) Body Pre-request Script Tests Settings Cookies
none form-data x-www-form-urlencoded raw binary GraphQL
Body Cookies Headers (10) Test Results
Save as Example
Pretty Raw Preview Visualize JSON

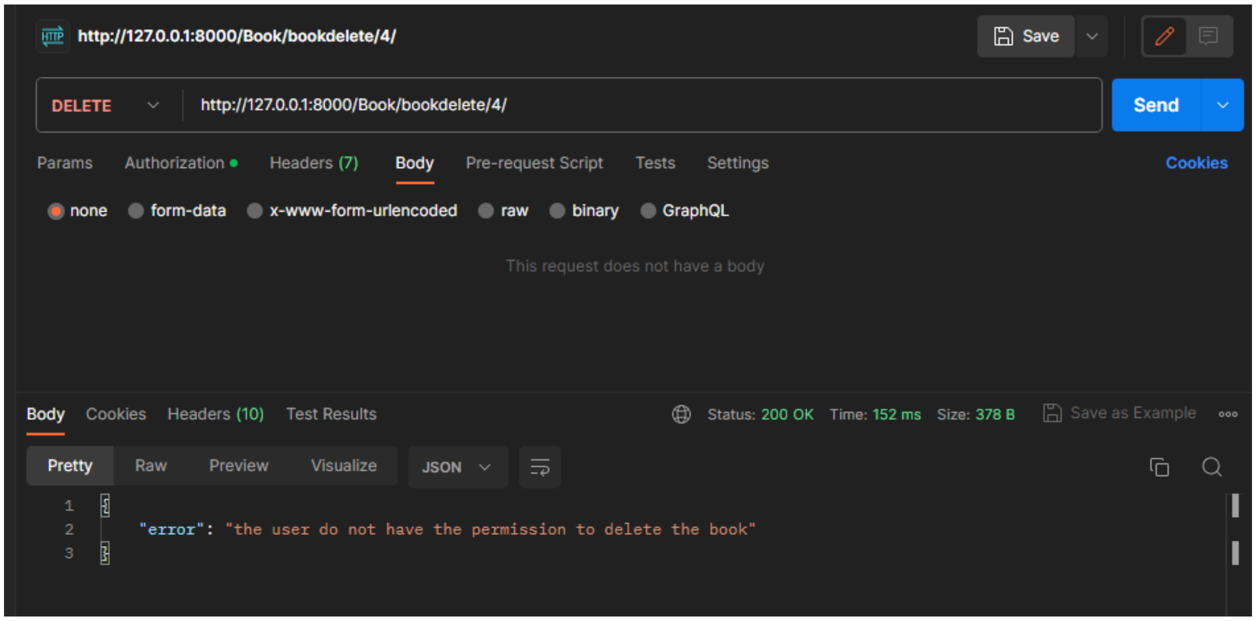
DELETE http://127.0.0.1:8000/Book/bookdelete/1/

This request does not have a body

Status: 200 OK Time: 86 ms Size: 338 B

```
{
    "message": "Book deleted"
}
```

→Deletion (unauthorized)


http://127.0.0.1:8000/Book/bookdelete/4/

DELETE http://127.0.0.1:8000/Book/bookdelete/4/

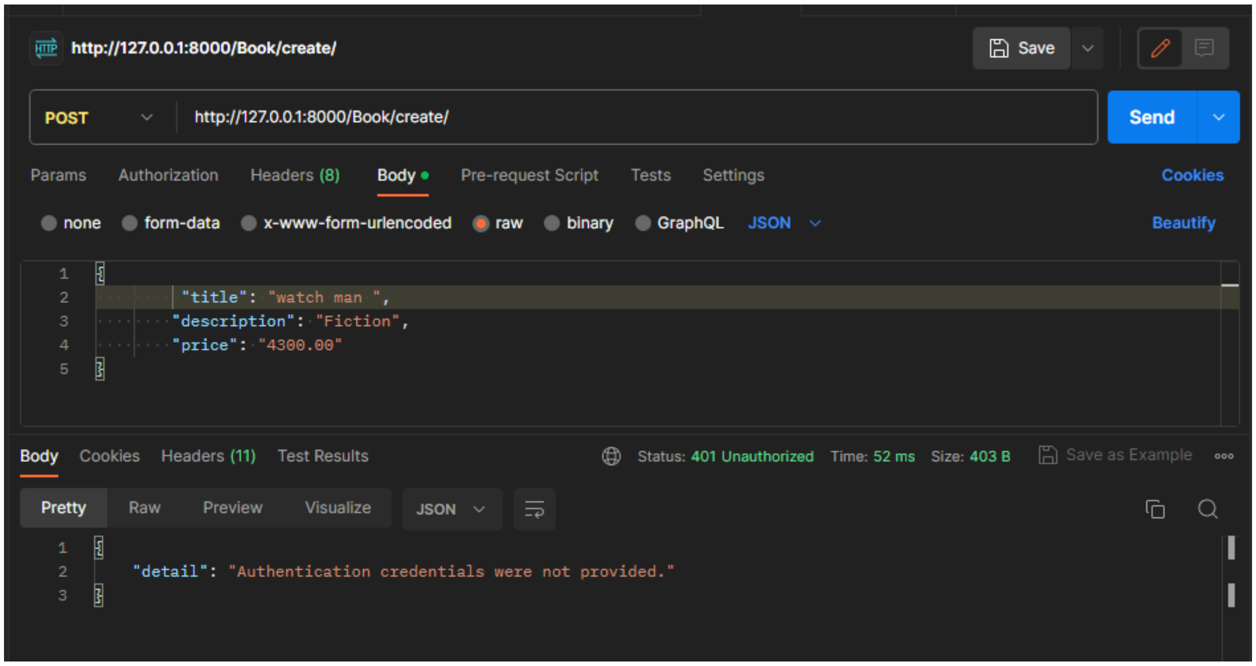
Body: none

This request does not have a body

Status: 200 OK Time: 152 ms Size: 378 B

```
{
    "error": "the user do not have the permission to delete the book"
}
```

→Creation (unauthorized)


http://127.0.0.1:8000/Book/create/

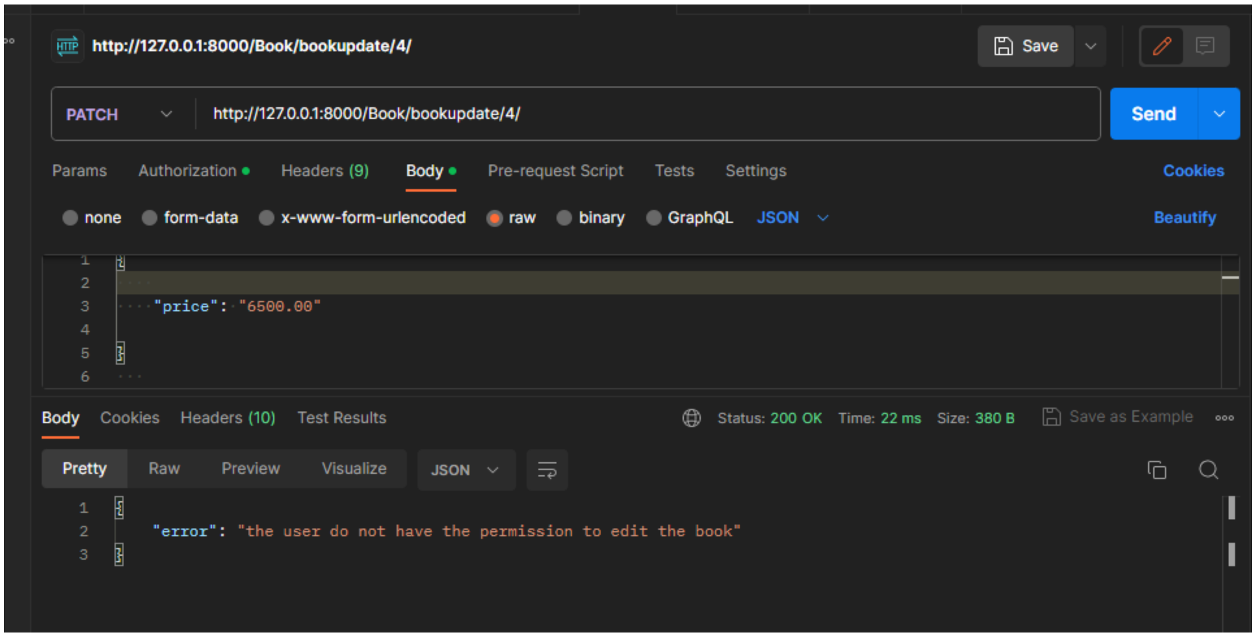
POST http://127.0.0.1:8000/Book/create/

```
{
    "title": "watch man ",
    "description": "Fiction",
    "price": "4300.00"
}
```

Status: 401 Unauthorized Time: 52 ms Size: 403 B

```
{
    "detail": "Authentication credentials were not provided."
}
```

→Book Update (unauthorized)



http://127.0.0.1:8000/Book/bookupdate/4/

PATCH http://127.0.0.1:8000/Book/bookupdate/4/

none form-data x-www-form-urlencoded [✓ raw] binary GraphQL JSON

```
{

    "price": "6500.00"

}
```

Status: 200 OK Time: 22 ms Size: 380 B

JSON

```
{
    "error": "the user do not have the permission to edit the book"
}
```

PROJECT: https://github.com/minta19/DRF_Authentication-12-07-23.git