

Assignment No. 04

Abhishek Singh (57)

Q. Develop a RESTful API for a Library Management System using Django REST Framework (DRF). The system should manage books and authors with basic CRUD functionality.

Ans. 1. First Install Dependencies

1. pip install django
2. pip install djangorestframework

Steps to Build the API

1. Create a Django project: LibraryAPI
2. Create an app: library
3. Add the app to INSTALLED_APPS in settings.py


1. Model (library/models.py)

This defines a Book model with title and isbn fields.

```
LibraryAPI > library > models.py > ...
1  from django.db import models
2
3  class Book(models.Model):
4      title = models.CharField(max_length=200)
5      isbn = models.CharField(max_length=13, unique=True)
6
7      def __str__(self):
8          return self.title
9
```

2. Serializer (library/serializers.py)


Converts the Book model into JSON format.

LibraryAPI > library >  serializers.py > ...

```
1  from rest_framework import serializers
2  from .models import Book
3
4  class BookSerializer(serializers.ModelSerializer):
5      class Meta:
6          model = Book
7          fields = '__all__'
8
```

3. Views (library/views.py)


Handles API requests to get, create, update, and delete books.

LibraryAPI > library >  views.py > ...

```
1  from rest_framework import generics
2  from .models import Book
3  from .serializers import BookSerializer
4  from django.http import HttpResponse
5
6  def home(request):
7      return HttpResponse('<h1 style="color:blue;">Welcome to the 🌳 '
8      | | | | | | | | | | '<hr><br>Library Management System!</h1>')
9
10 class BookListCreateView(generics.ListCreateAPIView):
11     queryset = Book.objects.all()
12     serializer_class = BookSerializer
13
14 class BookDetailView(generics.RetrieveUpdateDestroyAPIView):
15     queryset = Book.objects.all()
16     serializer_class = BookSerializer
17
```


4. URL Configuration (library/urls.py)

Defines API endpoints for books.

```
LibraryAPI > library >  urls.py > ...  
1  from django.urls import path  
2  from .views import BookListCreateView, BookDetailView  
3  
4  urlpatterns = [  
5      path('books/', BookListCreateView.as_view(), name='book-list-create'),  
6      path('books/<int:pk>', BookDetailView.as_view(), name='book-detail'),  
7  ]  
8
```

5. Project-Level URLs (LibraryAPI/urls.py)

Maps the API routes and includes Django's admin panel.

```
LibraryAPI > LibraryAPI >  urls.py > ...  
1  from django.contrib import admin  
2  from django.urls import path, include  
3  from library.views import home  
4  
5  urlpatterns = [  
6      path('admin/', admin.site.urls),  
7      path('api/', include('library.urls')),  
8      path('', home),  
9  ]  
10
```

Note: 1. Create a superuser

py manage.py createsuperuser

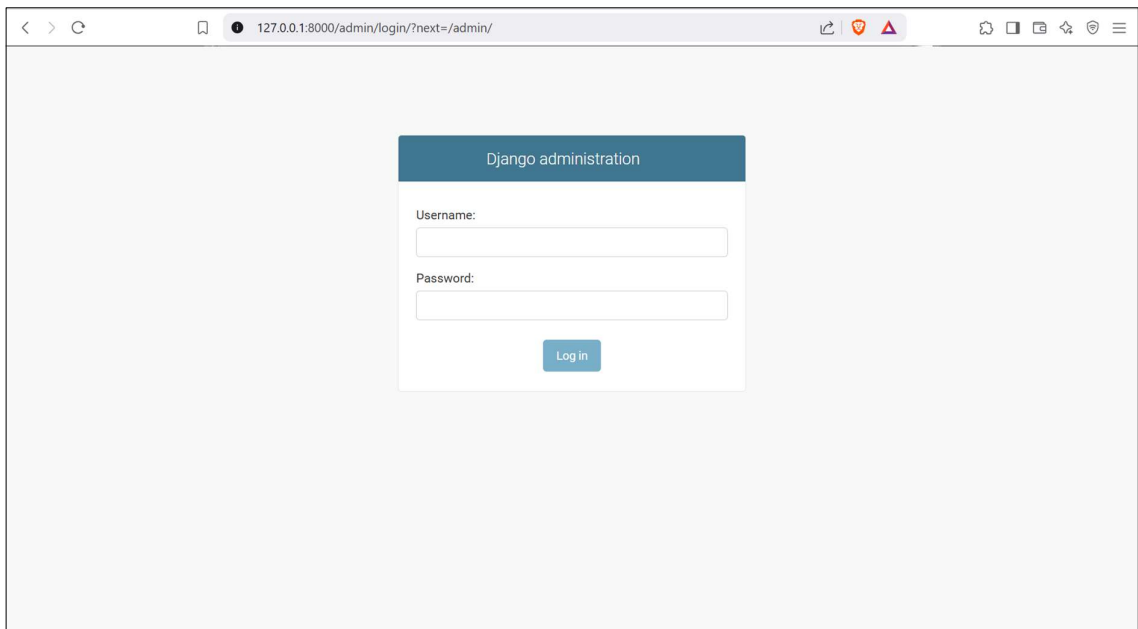
(put your username=abhishek,email=xyz@gmail.com>Password: **1334**)

2. Run server

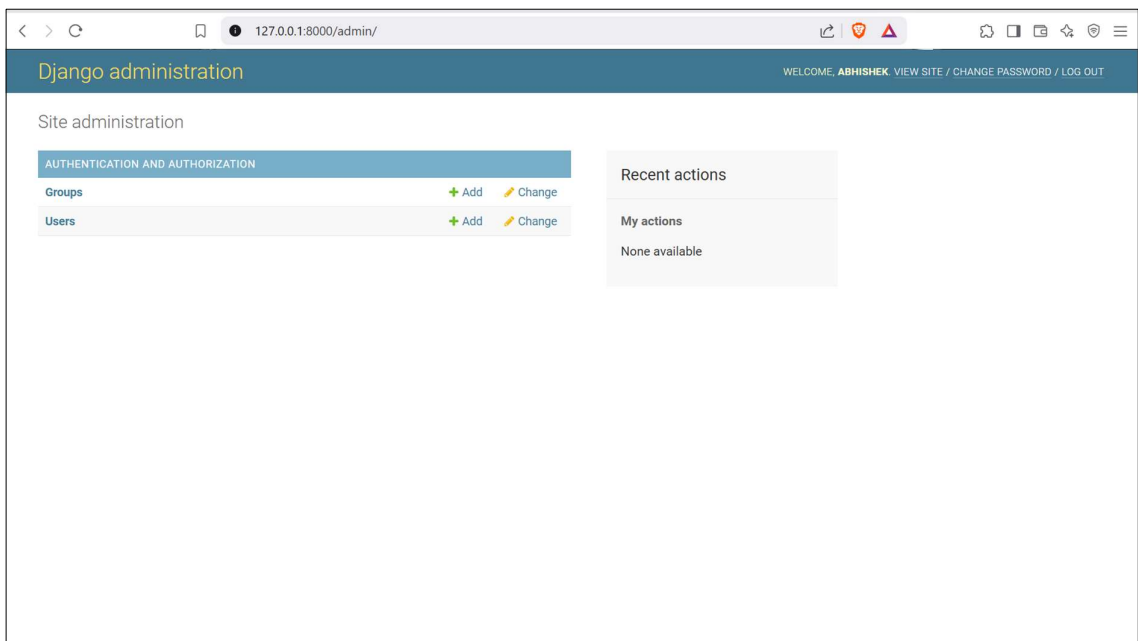
py manage.py runserver

Output:

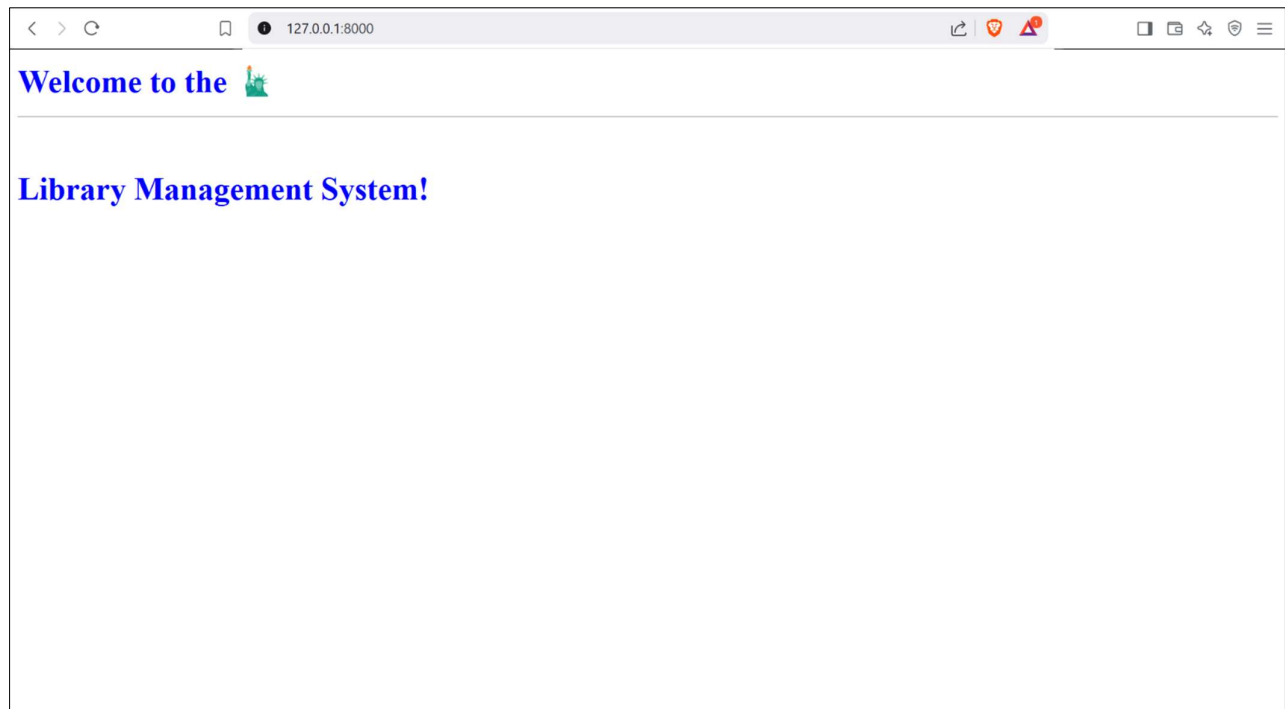
Admin site:



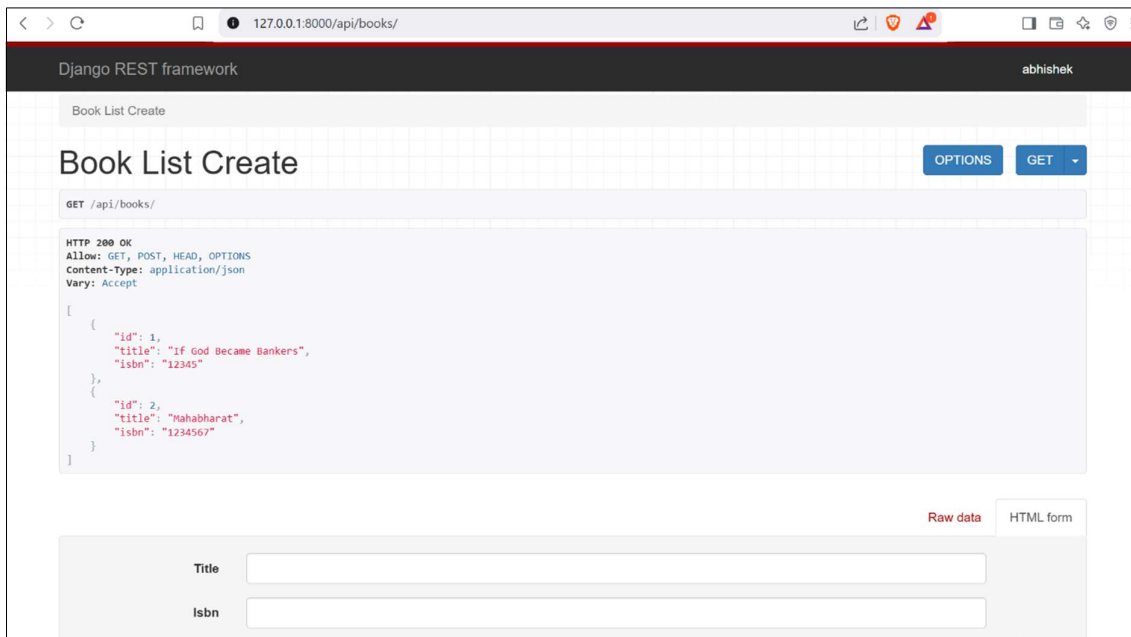
Enter your data when you run superuser cmd



Home Pg.



Book List (Perform CRUD)



Book Details (Perform CRUD Operation)

Django REST framework

abhishek

Book List Create / Book Detail

Book Detail

DELETEOPTIONSGET

GET /api/books/2/

HTTP 200 OK

Allow: GET, PUT, PATCH, DELETE, HEAD, OPTIONS

Content-Type: application/json

Vary: Accept

```
{
  "id": 2,
  "title": "Mahabharat",
  "isbn": "1234567"
}
```

Raw dataHTML form

Title

Mahabharat

Isbn

1234567

PUT