Django REST Framework (DRF) Exercise: Building a Bookstore API

In this exercise, students will build a **Bookstore API** using **Django REST Framework (DRF)** to apply the following concepts:

- **CRUD operations** using **serializers**, **class-based views**, and **viewsets**.
- Implementing **JWT authentication** and **permissions**.
- Creating **relationships** and **hyperlinked APIs**.
- Using **routers** for simplified URL management.

Step-by-Step Guide

Step 1: Project Setup and App Creation

1. **Create a Django project** called `bookstore_api` and a new app called `books`:

   ```
   bash
   Copy code
   django-admin startproject bookstore_api
   cd bookstore_api
   python manage.py startapp books
   ```

2. **Install Django REST Framework** and **djangorestframework-simplejwt**:

   ```
   bash
   Copy code
   pip install djangorestframework djangorestframework-simplejwt
   ```

3. **Add DRF to `INSTALLED_APPS`** in `bookstore_api/settings.py`:

   ```
   python
   Copy code
   INSTALLED_APPS = [
       'rest_framework',
       'books',
       'rest_framework_simplejwt',  # JWT support
   ]
   ```

Step 2: Create Models

In `books/models.py`, define the models for **Book** and **Author**:

```
python
Copy code
from django.db import models

class Author(models.Model):
    name = models.CharField(max_length=100)
    biography = models.TextField(blank=True)

    def __str__(self):
        return self.name

class Book(models.Model):
    title = models.CharField(max_length=150)
    description = models.TextField(blank=True)
    published_date = models.DateField()
    author = models.ForeignKey(Author, related_name='books',
on_delete=models.CASCADE)

    def __str__(self):
        return self.title
```

Run Migrations:

```
bash
Copy code
python manage.py makemigrations
python manage.py migrate
```

Step 3: Create Serializers

In `books/serializers.py`, create serializers for `Author` and `Book`:

```
python
Copy code
from rest_framework import serializers
from .models import Author, Book

class AuthorSerializer(serializers.ModelSerializer):
    class Meta:
        model = Author
        fields = ['id', 'name', 'biography', 'books']

class BookSerializer(serializers.ModelSerializer):
    author = serializers.HyperlinkedRelatedField(
        view_name='author-detail',
        read_only=True
    )

    class Meta:
        model = Book
        fields = ['id', 'title', 'description', 'published_date', 'author']
```

Step 4: Create Class-Based Views

Step 5: Set Up URLs Using Routers

In `books/urls.py`, use **routers** to manage the API URLs:

```
python
Copy code
from django.urls import path, include
from rest_framework.routers import DefaultRouter
from .views import AuthorListCreateView, AuthorDetailView,
BookListCreateView, BookDetailView

router = DefaultRouter()
urlpatterns = [
    path('authors/', AuthorListCreateView.as_view(), name='author-list'),
    path('authors/<int:pk>/', AuthorDetailView.as_view(), name='author-
detail'),
    path('books/', BookListCreateView.as_view(), name='book-list'),
    path('books/<int:pk>/', BookDetailView.as_view(), name='book-detail'),
]
```

In `bookstore_api/urls.py`:

```
python
Copy code
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
    path('api/', include('books.urls')),
]
```

Step 6: Implement JWT Authentication

1. **Configure JWT** in `bookstore_api/settings.py`:

   ```
   python
   Copy code
   REST_FRAMEWORK = {
       'DEFAULT_AUTHENTICATION_CLASSES': (
           'rest_framework_simplejwt.authentication.JWTAuthentication',
       ),
       'DEFAULT_PERMISSION_CLASSES': (
           'rest_framework.permissions.IsAuthenticated',
       ),
   }
   ```

2. **Set up JWT URLs** in `bookstore_api/urls.py`:

   ```
   python
   Copy code
   from rest_framework_simplejwt.views import TokenObtainPairView,
   TokenRefreshView

   urlpatterns += [
       path('api/token/', TokenObtainPairView.as_view(),
   name='token_obtain_pair'),
       path('api/token/refresh/', TokenRefreshView.as_view(),
   name='token_refresh'),
   ]
   ```

Step 7: Configure Permissions

1. **Use custom permissions** in `books/views.py`:

```
python
Copy code
from rest_framework.permissions import IsAuthenticatedOrReadOnly

class BookListCreateView(generics.ListCreateAPIView):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
    permission_classes = [IsAuthenticatedOrReadOnly]
```

Step 8: Testing the API

Use **Postman** or **curl** to test the API endpoints.

1. **Retrieve JWT Token**:

   ```
   bash
   Copy code
   curl -X POST http://127.0.0.1:8000/api/token/ -d
   "username=admin&password=admin123"
   ```

2. **Access Protected Endpoint**:

   ```
   bash
   Copy code
   curl -H "Authorization: Bearer <your_token>"
   http://127.0.0.1:8000/api/books/
   ```

Bonus Challenges

1. **Hyperlinked APIs**:
   - Use hyperlinked serializers to show relationships between authors and their books.
2. **Search and Filtering**:
   - Add search and filtering options using Django's **filter backends**.

What Students Will Learn

By completing this exercise, students will learn:

- How to implement CRUD operations using DRF.
- How to use **serializers** and **class-based views**.
- How to secure APIs using **JWT authentication** and **permissions**.
- How to use **routers** for API routing.
- How to build relationships and hyperlinked APIs.

This exercise will give students hands-on experience in building a real-world API using Django REST Framework.