ChatGPT

📚 Django Library Management System – Complete Flow (Notes)

These notes walk through a complete Django + DRF flow from setting up routes to building views, serializers, models, and handling requests and responses in a Library Management System.

---

1. Project and App Setup

a. Create Project and App

```
django-admin startproject library_project
cd library_project
python manage.py startapp books
```

b. Register App in `settings.py`

```
INSTALLED_APPS = [
    ...
    'rest_framework',
    'books',
]
```

🛣️ 2. Routing

a. Project-level `library_project/urls.py`

```
from django.contrib import admin
from django.urls import path, include

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

---

3. Models

a. `books/models.py`

```
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    total_copies = models.IntegerField()
    borrowed_copies = models.IntegerField(default=0)

    def available_copies(self):
        return self.total_copies - self.borrowed_copies

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

---

4. Migrations

```
python manage.py makemigrations
python manage.py migrate
```

📦 5. Serializers

a. `books/serializers.py`

```
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
    available_copies = serializers.IntegerField(read_only=True)

    class Meta:
        model = Book
        fields = '__all__'
```

---

6. Views

a. `books/views.py`

```
from rest_framework.views import APIView
from rest_framework.response import Response
from rest_framework import status
from .models import Book
from .serializers import BookSerializer

class BookListCreate(APIView):
    def get(self, request):
        books = Book.objects.all()
        serializer = BookSerializer(books, many=True)
        return Response(serializer.data)

    def post(self, request):
        serializer = BookSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=status.HTTP_201_CREATED)
        return Response(serializer.errors, status=status.HTTP_400_BAD_REQUEST)

class BookDetail(APIView):
    def get_object(self, pk):
        try:
            return Book.objects.get(id=pk)
        except Book.DoesNotExist:
            return None

    def get(self, request, pk):
        book = self.get_object(pk)
        if not book:
            return Response({'error': 'Book not found'}, status=404)
        serializer = BookSerializer(book)
        return Response(serializer.data)

    def put(self, request, pk):
        book = self.get_object(pk)
        if not book:
            return Response({'error': 'Book not found'}, status=404)
        serializer = BookSerializer(book, data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data)
        return Response(serializer.errors, status=400)
```

```
    def delete(self, request, pk):
        book = self.get_object(pk)
        if not book:
            return Response({'error': 'Book not found'}, status=404)
        book.delete()
        return Response(status=204)
```

---

7. URL Patterns

a. `books/urls.py`

```
from django.urls import path
from .views import BookListCreate, BookDetail

urlpatterns = [
    path('', BookListCreate.as_view(), name='book-list-create'),
    path('<int:pk>/', BookDetail.as_view(), name='book-detail'),
]
```

---

8. Sample Request/Response

a. Create Book Request

POST `/api/books/`

```
{
    "title": "Harry Potter",
    "author": "J.K. Rowling",
    "total_copies": 10,
    "borrowed_copies": 2
}
```

b. Sample Response

```
{
    "id": 1,
    "title": "Harry Potter",
    "author": "J.K. Rowling",
    "total_copies": 10,
```

```
    "borrowed_copies": 2,
    "available_copies": 8
}
```

---

9. Authentication (JWT with DRF)

a. Install JWT Auth

```
pip install djangorestframework-simplejwt
```

b. In `settings.py`

```
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': (
        'rest_framework_simplejwt.authentication.JWTAuthentication',
    )
}
```

---

10. RBAC – Role-Based Access Control

Use `is_staff` or create a custom role field in your User model. Use permission classes:

```
from rest_framework.permissions import IsAuthenticated, BasePermission

class IsAdmin(BasePermission):
    def has_permission(self, request, view):
        return request.user and request.user.is_staff

class BookCreateOnlyAdmin(APIView):
    permission_classes = [IsAuthenticated, IsAdmin]
```

---

11. Example – Borrow Book

```
class BorrowBook(APIView):
    def post(self, request, pk):
        book = Book.objects.get(id=pk)
        if book.available_copies() <= 0:
```

```
            return Response({'error': 'No copies available'}, status=400)
        book.borrowed_copies += 1
        book.save()
        return Response({'message': 'Book borrowed'})
```

Final Flow Summary

```
Client -> URL -> View -> Serializer -> Model -> DB -> Response
```