Pagination

Pagination is the process of dividing large data sets into smaller, manageable chunks or “pages”.

For example:

- If you have 1000 books in your DB, you don’t want to send all 1000 at once.
- Instead, you send 10–20 per page, and let the client request more pages as needed.

Why Do We Need Pagination?

Benefit	Explanation
Reduces server load	You don’t query and serialize 1000+ rows every time.
Speeds up response	Less data → faster network transfer.
Improves user experience	Frontend shows data page-by-page (infinite scroll, next/prev).
Saves memory	Backend and frontend both use less RAM.

Types of Pagination in DRF

There are three types : PageNumberPagination, LimitOffsetPagination and CursorPagination

Type	Class Name	Use Case
Page Number Pagination	`PageNumberPagination`	Common. Simple numbered pages.
Limit-Offset Pagination	`LimitOffsetPagination`	Specify how many (`limit`) and where to start (`offset`). Good for slicing.
Cursor Pagination	`CursorPagination`	Most secure and performant. Uses encrypted cursor instead of page number.

How to Setup Pagination in DRF

1. Global Pagination (for all views)

In your settings.py, under REST_FRAMEWORK

```
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS': 'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 5,  # default page size
}
```

2. Individual Pagination (per ViewSet)

```
from rest_framework.pagination import PageNumberPagination

class CustomPagination(PageNumberPagination):
    page_size = 5
    page_size_query_param = 'page_size'
    max_page_size = 100

# Apply in ViewSet
class BookViewSet(viewsets.ModelViewSet):
    ...
    pagination_class = CustomPagination
```

Example : Library Management System - Book Catalog

This example used PageNumberPagination and individual pagination type.

Feature	DRF Concept Used
Store book data	Django `Model`
Serialize book data	`ModelSerializer`
Expose API endpoints	`ViewSet` with Router
Paginate book list	`PageNumberPagination`

model.py

```
# library/models.py
from django.db import models

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    isbn = models.CharField(max_length=13, unique=True)
    publication_date = models.DateField()
    genre = models.CharField(max_length=50)

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

Serializer

```
# library/serializers.py
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
    class Meta:
```

```
        model = Book
        fields = '__all__'
```

Pagination

```
# library/pagination.py
from rest_framework.pagination import PageNumberPagination

class BookPagination(PageNumberPagination):
    page_size = 5  # show 5 books per page
    page_size_query_param = 'page_size'
    max_page_size = 100
```

ViewSet

```
# library/views.py
from rest_framework import viewsets
from .models import Book
from .serializers import BookSerializer
from .pagination import BookPagination

class BookViewSet(viewsets.ModelViewSet):
    queryset = Book.objects.all().order_by('title')
    serializer_class = BookSerializer
    pagination_class = BookPagination
```

Router and Url

```
# library/urls.py
from rest_framework.routers import DefaultRouter
from .views import BookViewSet
from django.urls import path, include

router = DefaultRouter()
router.register(r'books', BookViewSet)

urlpatterns = [
    path('', include(router.urls)),
]

project/urls
# project/urls.py
from django.contrib import admin
from django.urls import path, include

urlpatterns = [
    path('admin/', admin.site.urls),
```

```
    path('api/', include('library.urls')),
]
```

install filtering

```
pip install django-filter
```

Update settings.py

```
INSTALLED_APPS = [
    ...
    'django_filters',
]

REST_FRAMEWORK = {
    'DEFAULT_FILTER_BACKENDS': [
        'django_filters.rest_framework.DjangoFilterBackend',
        'rest_framework.filters.SearchFilter',
        'rest_framework.filters.OrderingFilter',
    ]
}
```

Update BookViewSet

```
# library/views.py
from rest_framework import viewsets, filters
from django_filters.rest_framework import DjangoFilterBackend
from .models import Book
from .serializers import BookSerializer
from .pagination import BookPagination

class BookViewSet(viewsets.ModelViewSet):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
    pagination_class = BookPagination

    # Add filtering, searching, ordering backends
    filter_backends = [DjangoFilterBackend, filters.SearchFilter,
filters.OrderingFilter]

    # Fields to filter on
    filterset_fields = ['genre', 'author']

    # Fields to search in
    search_fields = ['title', 'author', 'isbn']
```

```
    # Fields to allow ordering
    ordering_fields = ['publication_date', 'title']
```