📚 1. Introduction to Django REST Framework (DRF)

What is DRF?

- Django REST framework (DRF) is a powerful and flexible toolkit for building Web APIs.
- It provides features like serialization, authentication, and browsable API.

👉 Why Use DRF?

- Easy integration with Django.
- Built-in serialization and authentication.
- Supports RESTful APIs with minimal code.

🔥 2. Key Concepts in DRF

✅ a) Serializers

- Converts complex data types like Django QuerySets into Python data types that can be easily rendered into JSON/XML.

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

class Book(models.Model):
    title = models.CharField(max_length=100)
    author = models.CharField(max_length=100)
    published_date = models.DateField()

# serializers.py
from rest_framework import serializers
from .models import Book

class BookSerializer(serializers.ModelSerializer):
```

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

🎯 **Interview Question:**

> What is the difference between `serializers.Serializer` and `serializers.ModelSerializer`?

✅ b) Views

- Defines the logic to handle incoming HTTP requests.

Function-Based View (FBV)

```
from rest_framework.decorators import api_view
from rest_framework.response import Response

@api_view(['GET'])
def hello_world(request):
    return Response({"message": "Hello, World!"})
```

Class-Based View (CBV)

```
from rest_framework.views import APIView
from rest_framework.response import Response

class HelloWorldAPIView(APIView):
    def get(self, request):
        return Response({"message": "Hello, World!"})
```

🎯 **Interview Question:**

> What is the difference between Function-Based Views and Class-Based Views in DRF?

✅ c) ViewSets & Routers

- ViewSets reduce boilerplate code by combining CRUD operations.

```
# views.py
from rest_framework import viewsets
from .models import Book
from .serializers import BookSerializer

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

python
CopyEdit

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

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

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

🎯 **Interview Question:**

What is the role of `DefaultRouter` in DRF?

🔒 3. Authentication and Permissions

✅ a) Basic Authentication

```
# settings.py
INSTALLED_APPS += ['rest_framework']

REST_FRAMEWORK = {
```

```
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.BasicAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES': [
        'rest_framework.permissions.IsAuthenticated',
    ],
}
```

🎯 **Interview Question:**

How do you customize authentication in DRF?

✅ b) Token Authentication

```
# Install DRF Token Auth
pip install djangorestframework


# settings.py
INSTALLED_APPS += ['rest_framework.authtoken']

REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.TokenAuthentication',
    ],
}

# Generate Token for a User
from rest_framework.authtoken.models import Token
from django.contrib.auth.models import User

user = User.objects.get(username='sanket')
token = Token.objects.create(user=user)
print(token.key)
```

🎯 **Interview Question:**

How does Token Authentication differ from Basic Authentication in DRF?

🚀 4. CRUD Operations with DRF

✅ a) Create Operation (POST)

```
# views.py
class BookCreateAPIView(APIView):
    def post(self, request):
        serializer = BookSerializer(data=request.data)
        if serializer.is_valid():
            serializer.save()
            return Response(serializer.data, status=201)
        return Response(serializer.errors, status=400)
```

🎯 Interactive Task:

- Create an endpoint to create a `Student` model with `name` and `age`.

✅ b) Retrieve Operation (GET)

```
class BookRetrieveAPIView(APIView):
    def get(self, request, pk):
        book = Book.objects.get(pk=pk)
        serializer = BookSerializer(book)
        return Response(serializer.data)
```

🎯 Interview Question:

What is the difference between `APIView` and `GenericAPIView`?

🔄 5. Pagination and Filtering

✅ a) Pagination

```
# settings.py
REST_FRAMEWORK = {
    'DEFAULT_PAGINATION_CLASS':
'rest_framework.pagination.PageNumberPagination',
    'PAGE_SIZE': 10,
}
```

✅ b) Filtering

```
# views.py
from django_filters.rest_framework import DjangoFilterBackend

class BookViewSet(viewsets.ModelViewSet):
    queryset = Book.objects.all()
    serializer_class = BookSerializer
    filter_backends = [DjangoFilterBackend]
    filterset_fields = ['author', 'published_date']
```

🎯 Interview Question:

> How do you implement filtering and searching in DRF?

💡 6. Advanced Concepts

✅ a) Throttling

```
# settings.py
REST_FRAMEWORK = {
    'DEFAULT_THROTTLE_CLASSES': [
        'rest_framework.throttling.AnonRateThrottle',
    ],
    'DEFAULT_THROTTLE_RATES': {
        'anon': '5/day',
    }
}
```

✅ b) Custom Permissions

```
from rest_framework.permissions import BasePermission

class IsAdminOrReadOnly(BasePermission):
    def has_permission(self, request, view):
        if request.method in ['GET']:
            return True
        return request.user and request.user.is_staff
```

🎯 Interview Question:

How can you define a custom permission in DRF?

🎮 7. Interactive Quiz

1. **Which of the following is not a valid authentication class in DRF?**
    - a) BasicAuthentication
    - b) JWTAuthentication
    - c) TokenAuthentication
    - d) OAuth2Authentication
2. **Which method is used to fetch a single object in DRF?**
    - a) get_object()
    - b) retrieve()
    - c) fetch()
    - d) get()
3. **How do you generate a token for a user in DRF?**

- a) Token.create()
- b) Token.objects.create(user=user)
- c) Token.objects.get(user=user)
- d) GenerateToken(user=user)

🔗 8. Additional Interview Tips

- Be clear about the difference between `APIView`, `ViewSet`, and `ModelViewSet`.
- Know how to implement custom middleware and serializers.
- Practice with JWT and OAuth2 authentication.
- Implement a simple CRUD app with permissions and throttling.

🎯 Bonus Task:

- Build a REST API with CRUD operations for a `Student` model and implement token-based authentication.

Happy Coding! 🎉🚀