**✅ PART 1: REST API Practical Coding Questions**

**🔹 Basic Level**

1. Build a simple REST API in Django/DRF to perform CRUD operations on a model (e.g., Book, Student, Product).
2. Create an API endpoint that returns a list of all objects in JSON.
3. Create a GET API to retrieve an object by its ID.
4. Write POST API to create a new object using serializers.
5. Write PUT/PATCH API to update object partially/fully.
6. Write DELETE API to remove an object.

**🔹 Intermediate Level**

1. Write a serializer that converts Django model to JSON and vice versa.
2. Write a custom serializer field for password hashing.
3. Implement filtering in an API using query parameters (?name=John).
4. Implement search in DRF using SearchFilter.
5. Implement ordering (e.g., ?ordering=name).
6. Use ModelViewSet and Router to auto-generate all CRUD endpoints.

**🔹 Authentication & Permissions**

1. Implement Token Authentication in DRF.
2. Protect a view so only authenticated users can access it.
3. Restrict a view to only admin users using IsAdminUser.
4. Create a custom permission (e.g., only allow owner to edit their own object).

**🔹 Pagination**

1. Add pagination to an API list view using PageNumberPagination.
2. Customize page size in pagination.

**🔹 Relationships**

1. Create API for models with ForeignKey and display nested data using depth in serializer.
2. Write a serializer for a model with ManyToMany relationship.
3. Create nested serializers for models like Course and Enrollments.

**🔹 Custom API Logic**

1. Create a custom action in ViewSet (@action(detail=True)) like “mark-as-read”.
2. Create a non-model-based API view using APIView to return custom response.
3. Handle validation errors in serializer manually.
4. Raise custom ValidationError in serializer.

**✅ PART 2: DRF + Real-World Task Questions**

These are **project-based** or **task-style** questions asked in real interviews.

**🔸 Mini Tasks**

1. Create an API that returns all users with more than 5 blog posts.
2. Return only active records (is\_active=True) in the response.
3. Create login API and return token using DRF.
4. Create register API that hashes password before saving.
5. Create an API to upload an image file and store it.

**🔸 Business Use Cases**

1. Design an API where user can enroll in a course (ManyToMany through model).
2. Create API for submitting customer complaints (POST + GET with filters).
3. Create APIs for leave application system – apply, approve, list, reject.
4. Write an API where a user can upvote/downvote a post.
5. Build a commenting system where comments can have replies (recursive serializer).

**✅ PART 3: Bonus DRF Internals & Advanced**

These may not be coding, but often come with practical follow-ups.

1. Difference between APIView, GenericAPIView, and ViewSet.
2. When to use Serializer vs ModelSerializer?
3. What is throttling in DRF? Implement it.
4. What is the difference between permission, authentication, and throttling?
5. What is the role of .save() in serializers? How to override it?

**⚡️Pro Coding Test Tip:**

Most companies ask:

* Basic CRUD
* Custom serializer
* Auth-protected view
* Filtering or pagination
* One real-world use case API (example: register, comment, enroll)