**Theory**

**Pagination in Django REST Framework (DRF)**

Pagination is a technique used in APIs to break down large datasets into smaller, more manageable chunks or pages. It is essential for handling large amounts of data efficiently and ensuring that clients can request and load data progressively, rather than all at once, which can lead to performance issues.

In Django REST Framework (DRF), pagination can be applied to any list-based view (such as a list of doctor profiles) so that only a subset of results is returned at a time. DRF provides built-in pagination classes to make this process straightforward.

**Key Concepts:**

1. **Benefits of Pagination:**
   * **Improved Performance**: Instead of loading all data at once, the data is split into pages, reducing memory usage and speeding up response times.
   * **User Experience**: Users can navigate through large datasets in chunks rather than waiting for a large response to load.
   * **Reduces Server Load**: Pagination helps by ensuring the server does not need to process or return too many results at once.
2. **Common Pagination Styles**:
   * **Page Number Pagination**: The client requests a specific page of results, and the server returns a corresponding subset of the data.
   * **Limit/Offset Pagination**: The client specifies how many results to return and from which point in the dataset (e.g., limit=10&offset=20).
   * **Cursor Pagination**: This method uses a cursor (often a unique identifier) to paginate through results, avoiding issues with changing data during pagination.
3. **DRF Built-in Pagination Classes**:
   * **PageNumberPagination**: This is the most common pagination style, where you specify which page to fetch using a page query parameter.
   * **LimitOffsetPagination**: It returns results based on a specified limit and an offset.
   * **CursorPagination**: This style ensures consistent results when data changes while paginating, useful in APIs with frequently changing data.