Pagination in APIs:

Pagination is needed when API responses contain a large number of items, such as records or results. It allows clients to retrieve data in smaller, manageable chunks.

How It Works:

- The API typically supports parameters like `page` and `per_page` or `limit` to control pagination.
- Clients specify the page number and the number of items per page in their requests.
- The API responds with a subset of the total data, and clients can navigate through pages to access all data.

Example: In a blog API, a client can request articles using `page=2` and `per_page=10` to get the second page of 10 articles.

Filtering in APIs: Filtering is essential when clients want to narrow down the data they receive from an API based on specific criteria.

How It Works:

- APIs provide filter parameters, such as `filter` or `query`, that accept criteria like keywords, dates, or values.
- Clients include filter parameters in their requests.
- The API processes the filters and returns only the data that matches the specified criteria.

Example: In an e-commerce API, a client can use a `filter` parameter to retrieve products with specific attributes like `category=electronics` or `price<100`.

Rate Limiting in APIs: Rate limiting is necessary to prevent abuse, protect server resources, and ensure fair usage of the API by all clients.

How It Works:

- The API sets limits on the number of requests a client can make within a specific time window (e.g., requests per minute or per hour).
- Clients must include an API key or token in their requests.
- The API tracks the client's usage and enforces rate limits.
- When a client exceeds its rate limit, the API responds with a "429 Too Many Requests" status code.

Rate Limiting Algorithms:

- Fixed Window: Limits are enforced based on a fixed time window, and unused requests from one window do not carry over to the next.
- Sliding Window: Limits are based on a moving time window, allowing a more continuous rate of requests.
- Token Bucket: Clients receive tokens at a fixed rate, and they spend tokens for each request. When tokens run out, clients must wait.