What are some React best practices not covered in this week's videos? List at least 3 and describe why they are important.

Error Handling: In order to avoid confusion when an error happens, it's important to handle errors in a clear and organized way. This involves returning specific HTTP response codes that indicate the type of error that occurred. By doing so, the people responsible for maintaining the API will have enough information to understand the issue that occurred. It's crucial to prevent errors from causing system failures, so instead of leaving them unhandled, it's best to handle them properly. This means that the API consumer or user will need to implement error handling on their end.

HTTP Status Codes:

- **400 Bad Request –** This means that client-side input fails validation.
- **401 Unauthorized –** This means the user isn't not authorized to access a resource. It usually returns when the user isn't authenticated.
- **403 Forbidden –** This means the user is authenticated, but it's not allowed to access a resource.
- **404 Not Found –** This indicates that a resource is not found.
- **500 Internal server error –** This is a generic server error. It probably shouldn't be thrown explicitly.
- **502 Bad Gateway –** This indicates an invalid response from an upstream server.
- **503 Service Unavailable –** This indicates that something unexpected happened on server side (It can be anything like server overload, some parts of the system failed, etc.).

Filtering & Pagination: The databases supporting a REST API can become extremely large. Sometimes, there's so much data that it's not feasible to retrieve it all at once. It would either be too slow or overwhelm our systems. Hence, we need methods to filter the data.

Additionally, we need a way to divide the data into smaller chunks, or pages, so that we only retrieve a few results at a time. This prevents us from tying up resources for an extended period of time while trying to fetch all the requested data.

Filtering and pagination are both techniques that improve performance by reducing the strain on server resources. As the database accumulates more data over time, these features become increasingly important to ensure efficient operations.

URL: https://stackoverflow.blog/2020/03/02/best-practices-for-rest-api-design/

What are some other features the REST entails that we didn't cover this week?

Hypermedia as the Engine of Application State (HATEOAS): HATEOAS is a principle of REST that allows the API to provide links or references to related resources within the response. Clients can navigate and discover available actions and resources dynamically by following these links, reducing the need for prior knowledge of the API's structure.

Stateless Communication: REST APIs do not store any client context on the server. Each request from the client must contain all the necessary information for the server to understand and process it. This enables better scalability and reliability as servers can handle requests independently.

URL: https://www.knowledgehut.com/blog/programming/rest-api#features-of-rest-api