MOST ASKED

REACT

Coding Interview Questions

Question - 1

Implement a Pagination Component

To implement a pagination component in React, the following steps can be followed:

Detailed Steps:

1. **Initial Setup**:
   - **State Management**: Set up `useState` to track:
     - List of items fetched from the API.
     - Current page.
     - Number of items per page.
   - **Effect Hook for API Calls**: Use `useEffect` to fetch data from an API when the component mounts.
2. **Pagination Logic**:
   - **Calculate Pagination**: Use `currentPage` and `itemsPerPage` to calculate the range of items to display.
   - **Total Pages Calculation**: Calculate the total number of pages based on the total number of items (`Math.ceil(items.length / itemsPerPage)`).
3. **Page Navigation**:
   - **Next and Previous**: Add buttons for navigating between pages.
   - **Handle Page Clicks**: Update the current page when the user clicks a page number or uses the next/previous buttons.
4. **Render Items**:
   - **Display Items**: Only display the items for the current page using `slice()`.

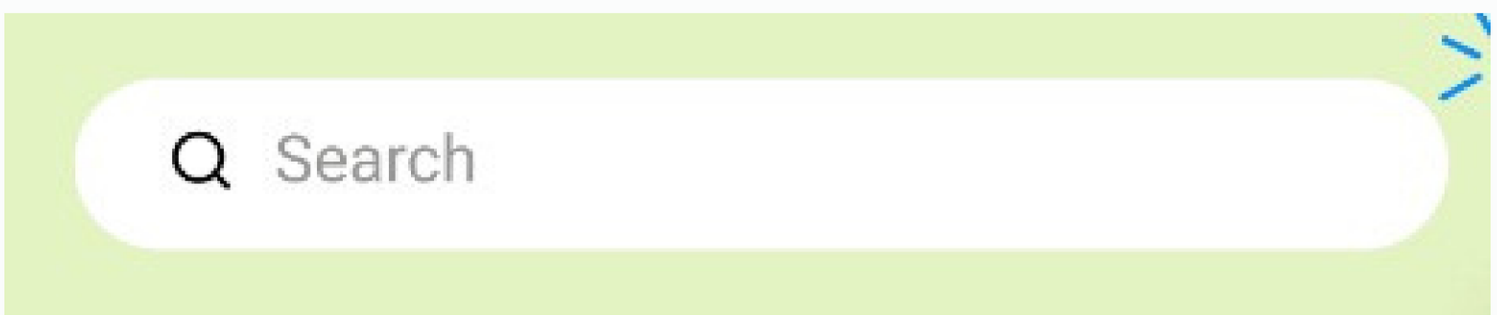
Question - 2

Design a Search Filter

To implement a search filter, we can follow a simple approach that filters the displayed items based on the user's input in real-time.

Detailed Steps:

1. **State Management**:
   - Track the user's search input (`query`).
   - Store the original list of items.
2. **Filter Logic**:
   - Use the `filter()` method to return only items that match the query (case-insensitive).
   - This happens every time the user types something into the search input.
3. **Real-time Updates**:
   - Use `onChange` in the search input to update the `query` state and filter the list dynamically.
4. **Rendering Filtered Items**:
   - Render only the filtered items.



Question - 3

Build a Real-time Chat Application

To build a real-time chat application using React, you'll typically use WebSockets to enable live communication between clients.

Detailed Steps:

1. **WebSocket Setup**:
   - Use **Socket.io** or the WebSocket API to establish a connection between the client and server.
2. **State Management**:
   - Track `messages` (the list of messages) and `input` (the user's input).
3. **Send/Receive Messages**:
   - **Send**: Emit messages from the client to the server using `socket.emit()`.
   - **Receive**: Listen for incoming messages from the server using `socket.on()` and update the state.
4. **Rendering Messages**:
   - Render the list of messages in real-time as they are sent/received.

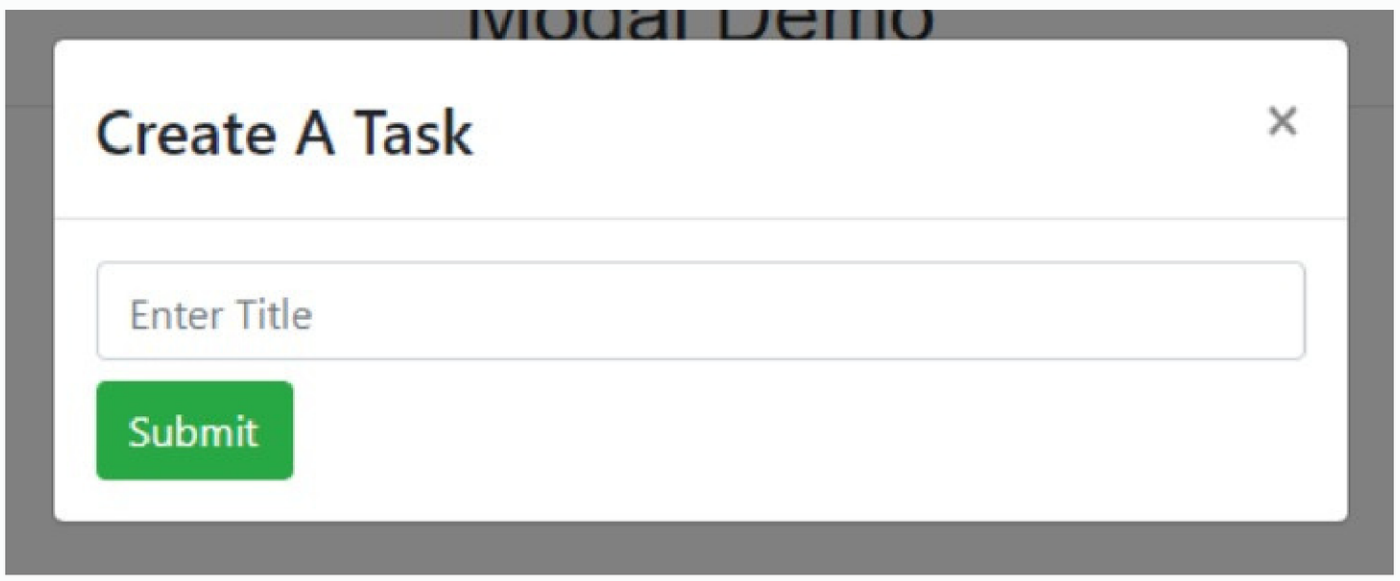
Question - 4

Create a Modal Component

A reusable modal component in React can be triggered by buttons or links and should be able to display different types of content.

Detailed Steps:

1. **State Management**:
   - Track the visibility of the modal (`isModalOpen`).
2. **Reusable Content**:
   - Use `children` props to pass dynamic content into the modal.
3. **Trigger the Modal**:
   - Add buttons or links to toggle the modal visibility.
4. **Close Modal**:
   - Add a close button inside the modal that updates the modal's state to hide it.



Question - 5

Implement an Image Gallery with Lazy Loading

Lazy loading helps improve performance by loading images only when they are about to enter the viewport (as the user scrolls).

Detailed Steps:

1. **State Management**:
   - Track which images are visible and their loading status.
2. **Intersection Observer**:
   - Use the `IntersectionObserver` API to detect when an image enters the viewport.
3. **Lazy Loading**:
   - Load full-resolution images only when they are about to be viewed.
4. **Rendering**:
   - Display a placeholder until the image is fully loaded.

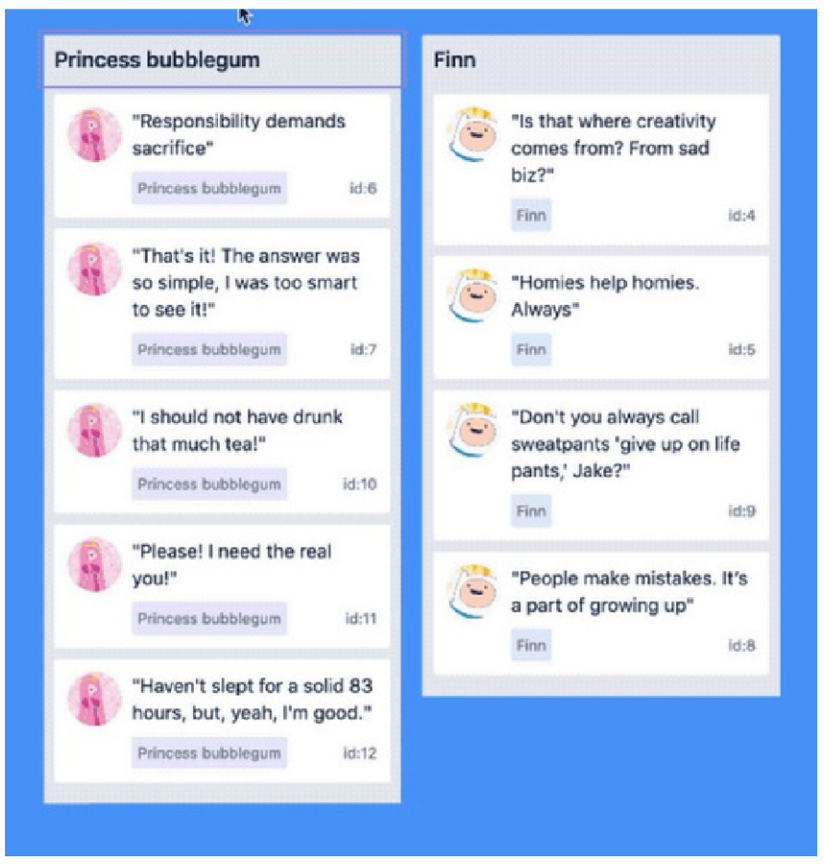
Question - 6

Develop a Drag-and-Drop Interface

You can create a drag-and-drop interface in React using the HTML5 Drag and Drop API or a library like React DnD or react-beautiful-dnd.

Detailed Steps:

1. **State Management:**
   - Track the list of items and their order.
2. **Drag Events:**
   - Implement `onDragStart`, `onDragOver`, and `onDrop` handlers to handle dragging and reordering.
3. **Dynamic Rendering**:
   - Update the order of the items and re-render them when the drag-and-drop operation is completed.



Question - 7

Build a Product List with Sorting and Filtering

To implement a product list with sorting and filtering, we'll need to manage the product data and provide options for sorting by different criteria (like price) and applying filters (like categories).

Detailed Steps:

1. **State Management**:
    - Track the list of products, the current sort criteria, filters (category, price range, availability).
2. **Filter and Sorting Logic**:
    - Filter products by category, price range, and availability.
    - Sort products by price or rating.
3. **Rendering**:
    - Render the filtered and sorted products.

Search: 6 records...

Role: All

Name	Title	Status	Age	Role
Jane Cooper	Regional Paradigm Technician	Active	27	Admin
Cody Fisher	Product Directives Officer	Active	43	Owner
Esther Howard	Forward Response Developer	Active	32	Member
Jenny Wilson	Central Security Manager	Active	29	Member
Kristin Watson	Lean Implementation Liaison	Active	36	Admin
Cameron Williamson	Internal Applications Engineer	Active	24	Member

Toggle SortBy

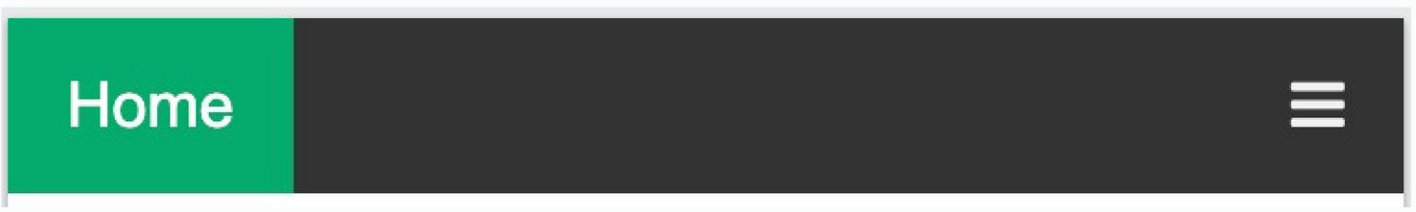
Question - 8

Create a Responsive Navbar

To create a responsive navigation bar, you can use media queries and implement a hamburger menu for mobile views.

Detailed Steps:

1. **State Management**:
   - Use `useState` to track whether the mobile menu is open or closed.
2. **CSS Media Queries**:
   - Use media queries to hide/show the hamburger menu on smaller screens.
3. **Smooth Transitions**:
   - Add transitions for smooth opening/closing of the menu.



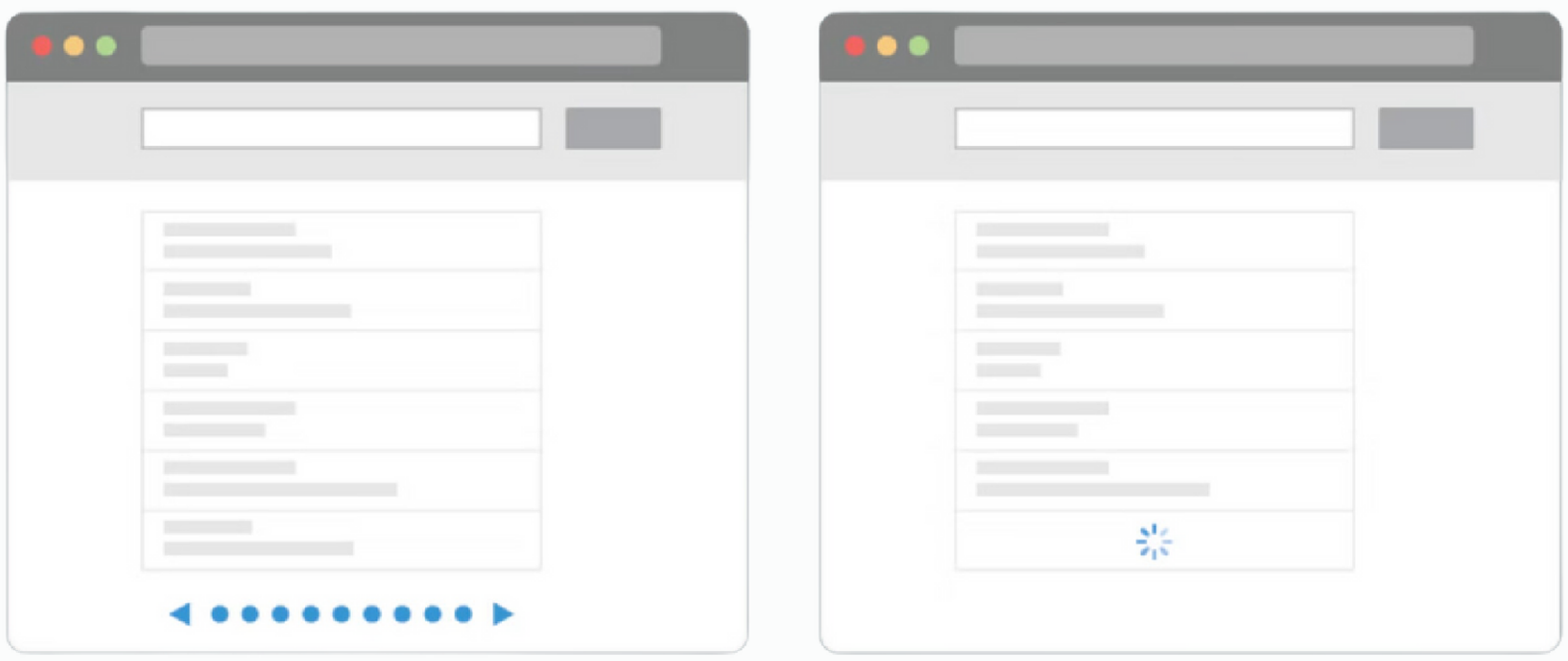
Question - 9

Implement an Infinite Scrolling List

Infinite scrolling loads more items as the user scrolls, improving performance by only loading what's necessary. This is commonly done by fetching new data when the user reaches the bottom of the list.

Detailed Steps:

1. **State Management**:
    - Track the list of items and whether the component is loading more items.
2. **Intersection Observer**:
    - Use the `IntersectionObserver` API to detect when the user reaches the bottom of the list.
3. **Load More Data**:
    - Fetch more items from the server when the user reaches the bottom.



Question - 10

Build a Real-time Notifications Component

A real-time notifications component dynamically displays alerts or messages. This is often done using WebSockets or a service like Socket.io to push real-time notifications.

Detailed Steps:

1. **WebSocket Setup**:
   - Use **Socket.io** to receive notifications from the server.
2. **State Management**:
   - Track the list of notifications.
3. **Dismiss/Interaction Logic**:
   - Allow users to dismiss or interact with notifications.