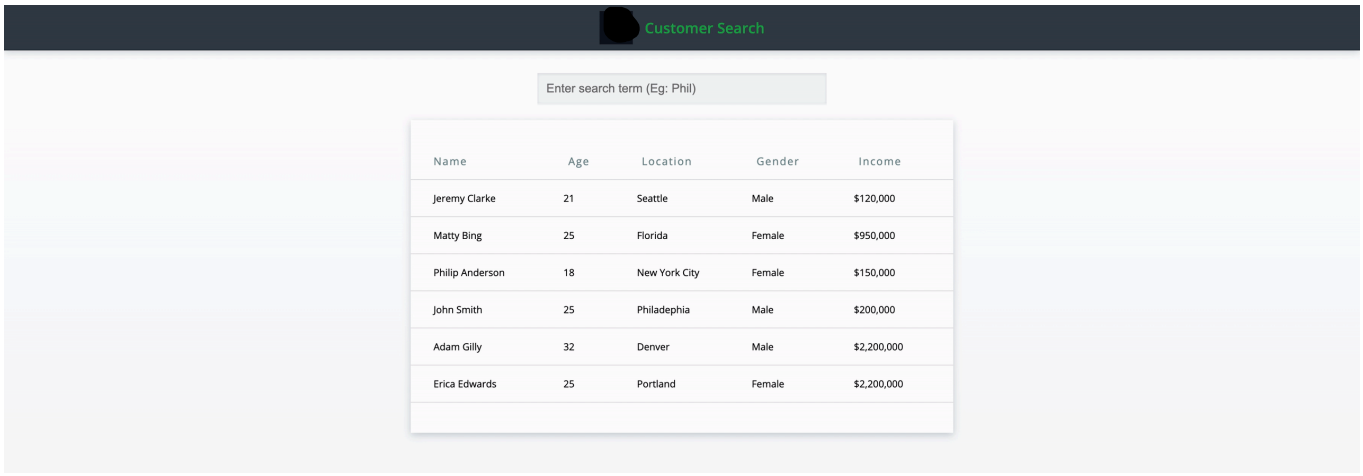
Question - 1

React: Customer Search

Complete the component as shown to pass all the test cases. Certain core React functionalities are already implemented.

Hide animation



The list of available customers and their details is imported as `List` in the SearchCustomer component. Use the list to render them as shown.

The data in List file is in this form.

```
{
   name: "Jeremy Clarke",
   age: 21,
   location: "Seattle",
   gender: "Male",
   income: "$120,000"
}
```

The application has 2 components:

- `SearchCustomer` - contains and search box and CustomerList component.
- `CustomerList` - displays the details of the customer based on the search term in a table.

The component must have the following functionalities:

- Initially, the input field must be empty. Whenever the input is empty, all the customers passed in the input to the component must be rendered in the list.
- The type of input in the input box should be `text`.
- As soon as the search term is typed in the input, search for customer records with any field that starts with the search term. For example, if the search term is "Phil", then records with the name "Philip Anderson" and the location "Philadelphia" should be displayed in the results.
- The filtered list should preserve the order customers are given in the input to the component.
- If the search term returns no customers, do not render the table. Instead, display the message "No Results Found!".
- The search results should be case-sensitive.

The following data-testid attributes are required in the component for the tests to pass:

Attribute	Component

search-input	Search Input Box
searched-customers	List of searched customers
no-results	div when no customer matches the search term

Note:

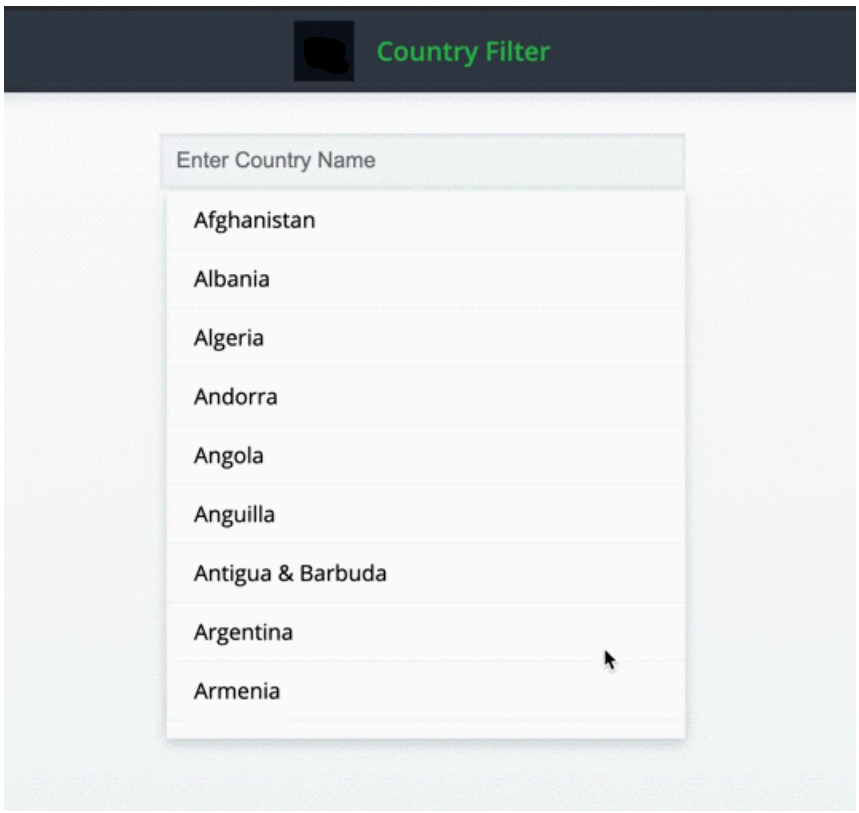
- Components have data-testid attributes for test cases and certain classes and ids for rendering purposes. They should not be changed.
- The files that should be modified by the candidate are `src/components/CustomerList.js` and `src/components/SearchCustomer.js`. They are open by default in the system editor.
- Avoid making changes to other files in the project structure.

Question - 2

React: Country Filter

Given a partially completed React application with the HTML template built and ready, implement a filter that searches and displays matching countries in a list.

Hide animation



The application has 2 components:

1. The `Search` view, which has the input box where the user can type to filter countries.
2. The `CountryList` view, which renders the list of countries that are filtered.

The entire list of countries to display is stored in a variable named `response` inside the file `src/response.js`.

The app should implement the following functionalities:

- The initial view should render an empty input box with all countries appearing in the same order as in the `response` variable.
- When a character is typed, each country containing the set of typed characters should be filtered and displayed.
- The filtering should be case insensitive.
- When there is no character typed in the search box, it should show all countries.

Note:

- Get the list of countries to be displayed from the `response.js` file.
- The order in which each country appears should follow the same order in the `response.js` file, even after filtering.

The following data-testid attributes are required in the component for the tests to pass:

- The country filter field <input> tag should have the data-testid attribute 'filterInput'.
- The countries wrapper <ul> tag should have the data-testid attribute 'countryList'.

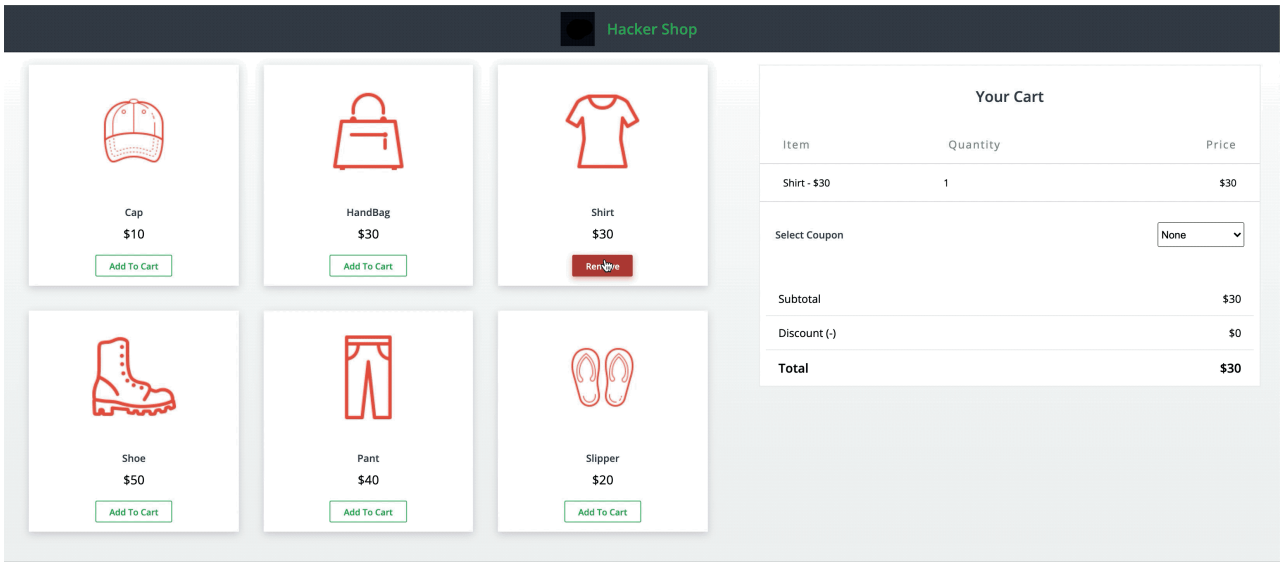
Please note that the component has these data-testid attributes for test cases. They should not be changed.

Question - 3

React: HackerShop Checkout

Create a basic shopping application, as shown below. Some core functionalities have already been implemented, but the application is not complete. Application requirements are given below, and the finished application must pass all of the unit tests.

Hide animation



The app has two separate views/components: the Product Listing component and the Cart component. The list of products to be displayed is already provided in the app.

The app should implement the following functionalities:

- Clicking on each 'Add To Cart' button should add the item to the shopping cart. When an item is added to the cart:
  - The 'Add To Cart' button should be removed from view, and the 'Remove From Cart' button should be displayed.
  - An entry should be added to the table in the Cart component.
- Clicking on each 'Remove' button should remove the item from the cart and display 'Add to Cart' for the product item.
- The Cart component should have the following functionalities:
  - Display all the items in the cart in a table.
  - Display the cart's subtotal, discount value, and total price.
  - The cart has a 'Select Coupon' input. On selecting a coupon from this input, an appropriate discount is applied and the total price is calculated and displayed. (Subtotal - Discount = Total Price)
- Items should be displayed in the Cart component in the order they are added to the cart.
- The list of products and the cart object are passed as props to the Product Listing component and the Cart component respectively.

Each product object contains the following properties:

- name: Name of the product. [STRING]
- price: Price of the product. [NUMBER]

- id: Unique ID of the product. (Auto Generated) [NUMBER]
- image: The image URL of the product. [STRING]
- cartQuantity: The quantity of the item in the cart. The default value should be 0. [NUMBER]

Each item in the cart, CartItem, has the following properties:

- id: The ID of the product added to the cart. [NUMBER]
- item: The heading property of the product. [STRING]
- quantity: The quantity of the item in the cart. [NUMBER]
- price: The total price of the item in the cart. (quantity x product.price) [NUMBER]

The following data-testid/class attributes are required in the component for the tests to pass:

- Each product item in the Product Listing component should have the class 'product-item'.
- Each 'Add to Cart' button should have the data-testid attribute 'btn-item-add'.
- Each 'Remove' button should have the data-testid attribute 'btn-item-remove'.
- The table rows <tr> in the Cart component, corresponding to items in the cart, should have the data-testid attribute of 'cart-item-0', 'cart-item-1', and so on.
- The table data <td>, containing the name of the item in the cart, should have the data-testid attribute 'cart-item-name'.
- The table data <td>, containing the quantity of the item in the cart, should have the data-testid attribute 'cart-item-quantity'.
- The table data <td>, containing the price of the item in the cart, should have the data-testid attribute 'cart-item-price'.
- The 'Select Coupon' input should have the data-testid attribute 'cart-coupon'
- The cart's 'Subtotal' value container should have the data-testid 'cart-subtotal'.
- The cart's 'Discount' value container should have the data-testid 'cart-discount'.
- The cart's 'Total Price' value container should have the data-testid 'cart-total'.

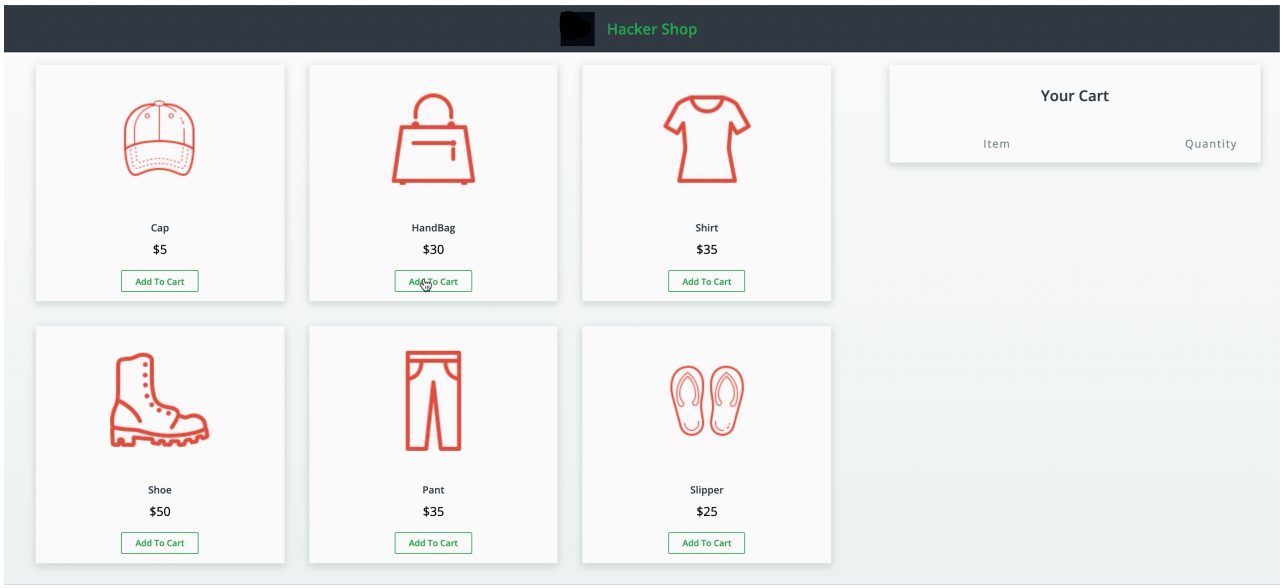
Please note that the component has the above data-testid attributes for test cases and certain classes and ids for rendering purposes. It is advised not to change them.

Question - 4

React: HackerShop Shopping Cart

Create a basic shopping application, as shown below. Some core functionalities have already been implemented, but the application is not complete. Application requirements are given below, and the finished application must pass all of the unit tests.

Hide animation



The app has two separate views/components: the Product Listing component and the Cart component. The list of products to be displayed is already provided in the app.

The app should implement the following functionalities:

- Clicking on each 'Add To Cart' button should add the item to the shopping cart. The listing in the Product Listing component should be updated to show the 'Increase/Decrease Quantity' button and the quantity of the item in the cart.
- Clicking on each 'Increase Quantity' button should increase the quantity of the item in the cart.
- Clicking on each 'Decrease Quantity' button should do the following:
  - When the cart quantity of the item is 1: This should remove the item from the cart, hide the 'Increase/Decrease Quantity' button, and show the 'Add to Cart' button.
  - When the cart quantity is greater than 1: The quantity of the item in the cart should be decreased.
- On every quantity update operation, the text for the quantity of the item should be updated both in the Product Listing component as well as in the corresponding entry in the Cart component.
- Items should be displayed in the Cart component in the order they are added to the cart.
- The list of products and the cart object are passed as props to the Product Listing component and Cart component respectively.

Each product object contains the following properties:

- name: Name of the product. [STRING]
- price: Price of the product. [NUMBER]
- id: Unique ID of the product. (Auto Generated) [NUMBER]
- image: The image URL of the product. [STRING]
- cartQuantity: The quantity of the item in the cart. The default value should be 0. [NUMBER]

Each item in the cart, CartItem, has the following properties:

- id: The ID of the product added to the cart. [NUMBER]
- item: The name of the product added to the cart. [STRING]
- quantity: The quantity of the item in the cart [NUMBER]

The following data-testid attributes are required in the component for the tests to pass:

- Each product item in the Product Listing component should have the data-testid attribute of 'product-item-0', 'product-item-1', and so on.
- Each 'Add to Cart' button should have the data-testid attribute 'btn-item-add'.
- Each 'Increase Quantity' button should have the data-testid attribute 'btn-quantity-add'.
- Each 'Decrease Quantity' button should have the data-testid attribute 'btn-quantity-subtract'.
- Each input to display the cart quantity in the Product Listing component should have the data-testid attribute 'cart-quantity'.
- The table rows <tr> in the Cart component, corresponding to items in the cart, should have the data-testid attribute as 'cart-item-0', 'cart-item-1', and so on.
- The table data <td>, containing the name of the item in the cart, should have the data-testid attribute 'cart-item-name'.
- The table data <td>, containing the quantity of the item in the cart, should have the data-testid attribute 'cart-item-quantity'.

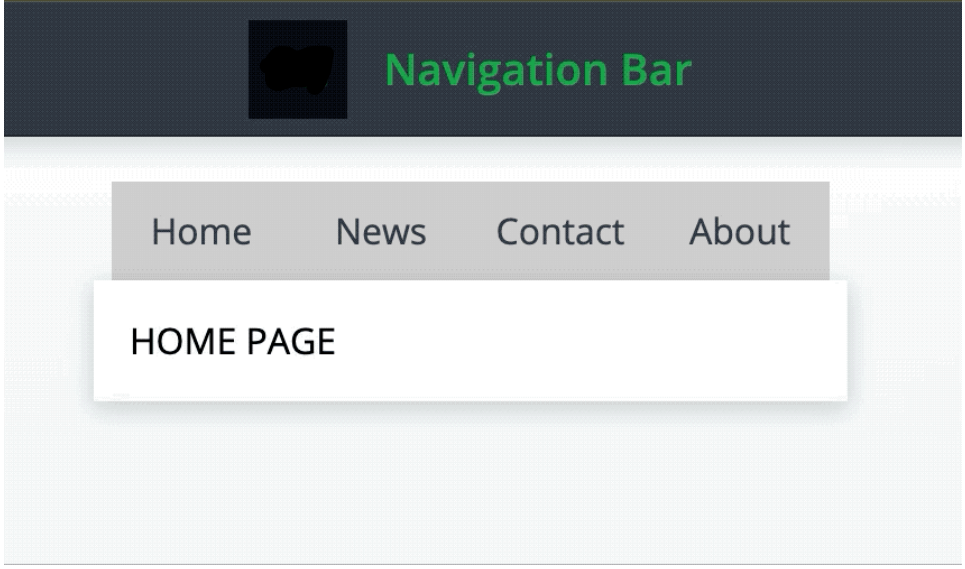
Please note that the component has the above data-testid attributes for test cases and certain classes and ids for rendering purposes. It is advised not to change them.

Question - 5

React: Navigation Bar Component

Create a navigation bar component as shown below.

Hide animation



The component has the following functionalities:

- There are 4 tabs: Home, News, Contact, and About.
- The default selected tab is the 'Home' tab.
- Clicking on a tab renders the relevant content.
    - Clicking on the Home tab renders content 'HOME PAGE'.
    - Clicking on the News tab renders content 'NEWS PAGE'.
    - Clicking on the Contact tab renders content 'CONTACT PAGE'.
    - Clicking on the About tab renders content 'ABOUT PAGE'.
- Since the default selected tab is the 'Home' tab, the default displayed content is 'HOME PAGE'.

Please note that components have some classes and ids for rendering purposes. It also has the following *data-testid* attributes for test cases.

- The parents of 4 tabs have data-testid attribute 'navigation-tabs'.
- The content sections have data-testid attribute 'tab-content'.

It is not advised to change any of these classes, ids, or data-testid attributes.