STEELEYE ASSIGNMENT

Front End

Name – Sejal Kumari

Lovely Professional University

College Reg. No. : 12015961

Hosted Output Link: https://steeleye-assignment-sejal12015961.netlify.app/

Q.1. Explain what the simple List component does.

Ans :

This code defines a simple list component in React.

The list component is composed of two sub-components: **`SingleListItem`** and **`List`.**

- The **`SingleListItem`** component is a memoized functional component that represents a single item in the list.

It takes four props as input:

- **`index`:** A number that represents the index of the item in the list.
- **`isSelected`:** A boolean that indicates whether the item is currently selected or not.
- **`onClickHandler`:** A function that is called when the item is clicked. It takes the index of the clicked item as an argument.
- **`text`:** A string that represents the text content of the item.

The **`SingleListItem`** component renders an **`li`** element with a style that depends on whether the item is selected or not. If the item is selected (**`isSelected`** prop is **`true`**), the background color is set to green, otherwise, it's set to red. The **`onClick`** event is attached to the **`li`** element, which calls the **`onClickHandler`** prop with the **`index`** of the clicked item when the item is clicked.

- The **`List`** component is also a functional component that represents a list of items. It takes one prop as input:
  - **`items`:** An array of objects, where each object represents an item in the list. Each item object must have a `text` property, which is a required string.

The **`List`** component uses the **`useState`** hook to manage the state of the currently selected item. It initializes the **`selectedIndex`** state using destructuring method to **`[setSelectedIndex, selectedIndex]`**. The **`useEffect`** hook is used to reset the **`selectedIndex`** state to **`null`** whenever the **`items`** prop changes.

The **`List`** component defines a **`handleClick`** function that sets the **`selectedIndex`** state to the index of the clicked item. This function is passed as a prop to the **`SingleListItem`** component using an arrow function to avoid immediate invocation of the function during rendering.

Inside the **`return`** statement, the **`List`** component maps over the **`items`** prop and renders a **`SingleListItem`** component for each item. The **`onClickHandler`**, **`text`**, **`index`**, and **`isSelected`** props are passed to the **`SingleListItem`** component accordingly. The **`memo`** function is used to memoize the **`List`** component, which helps optimize performance by preventing unnecessary re-renders when the **`items`** prop hasn't changed.

The **`propTypes`** and **`defaultProps`** are defined for both **`WrappedSingleListItem`** and **`WrappedListComponent`** to specify the expected prop types and default prop values. Note that **`items`** prop in **`WrappedListComponent`** is set to **`null`** as the default value.

Q.2. What problems / warnings are there with code?

Ans:

- setSelectedIndex is used as a function in the onClick event handler of the WrappedSingleListItem component, but it is actually a state setter function returned by the useState hook in the WrappedListComponent. The correct way to use setSelectedIndex as an event handler would be to wrap it in a callback function, like this:
  onClick={() => onClickHandler(index)}
- In the WrappedListComponent component, the items prop is defined with PropTypes.array(PropTypes.shapeOf({...})), which is not a valid syntax for specifying the shape of an object in PropTypes. The correct syntax would be *PropTypes.arrayOf(PropTypes.shape({...})), like this:*
  items: PropTypes.arrayOf(PropTypes.shape({
  text: PropTypes.string.isRequired,
  }))
- The isSelected prop in WrappedSingleListItem is used as a boolean value to determine the background color, but it is passed as isSelected={selectedIndex} from WrappedListComponent, which is a number (the selected item index). This could lead to unexpected behavior. We can update the prop type of isSelected in WrappedSingleListItem to PropTypes.number instead of PropTypes.bool, and use a comparison to check if the index prop matches the selectedIndex state, like this:
  style={{ backgroundColor: index === isSelected ? 'green' : 'red'}}
- The selectedIndex state in WrappedListComponent is not initialized with a default value. In order to fix this, we can update the useState hook like this:
  const [selectedIndex, setSelectedIndex] = useState(null);
- In the WrappedListComponent, the index prop is not passed correctly to the onClickHandler prop of SingleListItem. It should be passed as an argument in a callback function, like this:
  onClickHandler={() => handleClick(index)}

- The memo HOC is used for both SingleListItem and ListComponent components, but it may not be necessary for SingleListItem, as it does not have any internal state or prop changes. We can remove memo from SingleListItem for performance optimizations.
- The propTypes and defaultProps for WrappedListComponent are defined with incorrect casing. The correct naming should be propTypes and defaultProps (camelCase), like this:

```
WrappedListComponent.propTypes = {
  items: PropTypes.arrayOf(PropTypes.shape({
    text: PropTypes.string.isRequired,
  })),
};
WrappedListComponent.defaultProps = {
items: null,
};
```

- Also PropTypes is a separate library that needs to be installed and imported separately in order to use it for prop type checking.
- Also created an array of objects named items

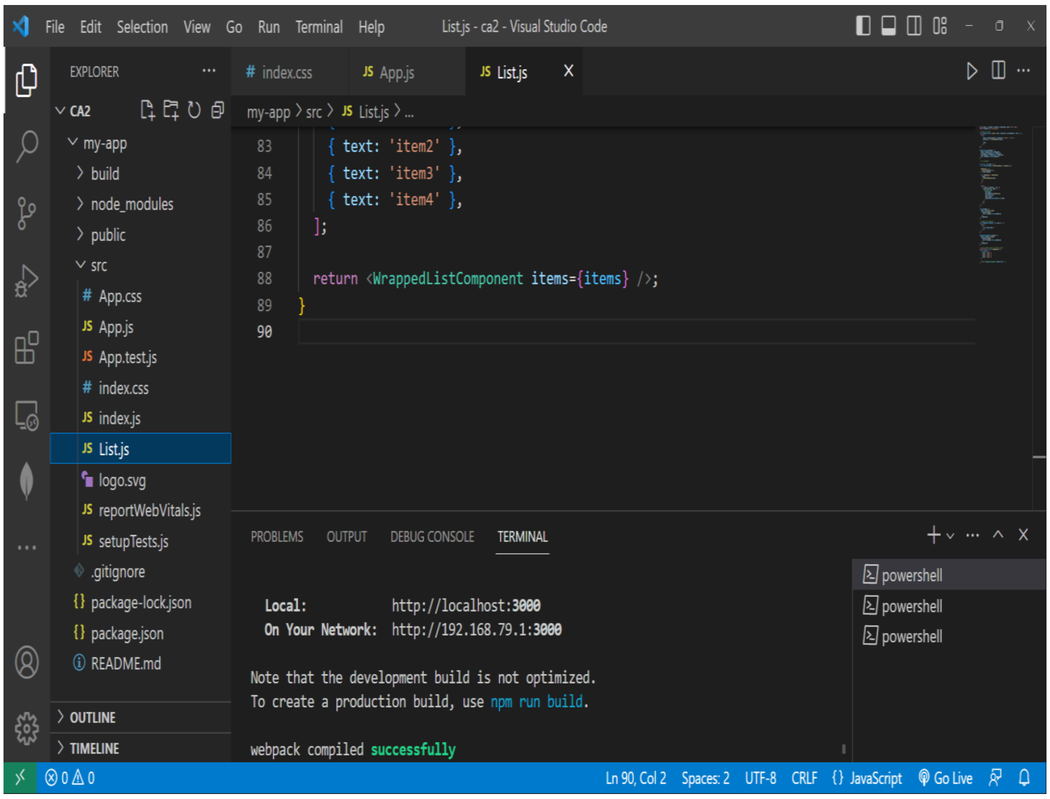
Q.3. Please fix, optimize, and/or modify the component as much as you think is necessary.

Ans :

Optimized Code
File name -> List.js then imported this file in App.js

```
import React, { useState, useEffect, useCallback, memo } from 'react';
import PropTypes from 'prop-types';

// Single List Item
const SingleListItem = memo(({ index, isSelected, onClickHandler, text }) => {
  return (
    <li
```

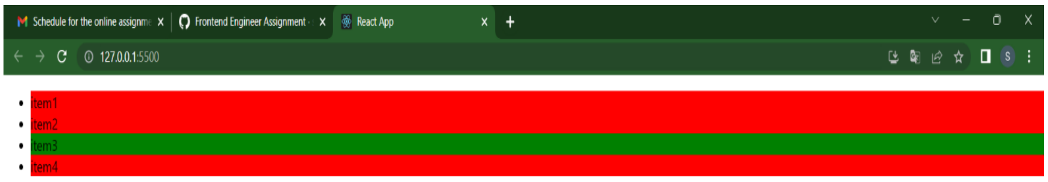
```
      style={{ backgroundColor: isSelected ? 'green' : 'red' }}
      onClick={() => onClickHandler(index)}
    >
      {text}
    </li>
  );
});

SingleListItem.propTypes = {
  index: PropTypes.number.isRequired,
  isSelected: PropTypes.bool.isRequired,
  onClickHandler: PropTypes.func.isRequired,
  text: PropTypes.string.isRequired,
};

// List Component

const List = ({ items }) => {
  const [selectedIndex, setSelectedIndex] = useState(null);

  useEffect(() => {
    setSelectedIndex(null);
  }, [items.length]);

  const handleClick = useCallback(
    (index) => {
      setSelectedIndex(index);
    },
    []
  );

  return (
    <ul style={{ textAlign: 'left' }}>
      {items.map((item, index) => (
        <SingleListItem
          key={index}
          onClickHandler={handleClick}
          text={item.text}
          index={index}
          isSelected={selectedIndex === index}
        />
      ))}
    </ul>
  );
};

List.propTypes = {
  items: PropTypes.arrayOf(
    PropTypes.shape({
      text: PropTypes.string.isRequired,
    })
  ).isRequired,
};

// Wrapped List Component
const WrappedListComponent = ({ items }) => {
  return (
    <div>
      <List items={items} />
    </div>
  );
};

WrappedListComponent.propTypes = {
  items: PropTypes.arrayOf(
    PropTypes.shape({
      text: PropTypes.string.isRequired,
    })
  ).isRequired,
};

// Export named function for default export
export default function WrappedList() {
  const items = [
    { text: 'item1' },
    { text: 'item2' },
    { text: 'item3' },
    { text: 'item4' },
  ];

  return <WrappedListComponent items={items} />;
}
```



```
    { text: 'item2' },
    { text: 'item3' },
    { text: 'item4' },
  ];

  return <WrappedListComponent items={items} />;
}
```

```
  Local:            http://localhost:3000
  On Your Network:  http://192.168.79.1:3000

Note that the development build is not optimized.
To create a production build, use npm run build.

webpack compiled successfully
```

Output



- item1
- item2
- item3
- item4