MERN Stack Training

Weekly Tasks

Week 6:

Exercise 1: Click Counter

Objective:
Create a React component that displays a counter and a button to increment the counter.

Steps:

1. Create a new component called `ClickCounter`.
2. Initialize a state variable named `count` with an initial value of 0.
3. Render a button and a display area for the counter.
4. Implement an event handler function that increments the `count` state when the button is clicked.
5. Connect the event handler to the button's `onClick` event.

Exercise 2: Toggle Text

Objective:
Create a React component with a button that toggles between two different pieces of text when clicked.

Steps:

1. Create a new component called `ToggleText`.
2. Initialize a state variable named `isTextOne` with an initial value of `true`.
3. Render a button and a display area for the text.
4. Implement an event handler that toggles the value of `isTextOne`.
5. Based on the state, display one of the two pieces of text.

Exercise 3: List Item Deletion

Objective:
Create a list of items with a button next to each for deletion.

Steps:

1. Create a new component named `DeletableList`.
2. Initialize a state variable `items` with an array of sample items.
3. Render the list of items along with delete buttons next to each item.
4. Implement a function that deletes an item when its corresponding delete button is clicked.
5. Pass the function as an event handler for each delete button.

Exercise 4: Color Changer

Objective:
Create a button that changes the background color of the component when clicked.

Steps:

1. Create a new component named `ColorChanger`.
2. Initialize a state variable `backgroundColor` with an initial color value.
3. Render a button.
4. Implement an event handler that changes the `backgroundColor` state when the button is clicked.
5. Set the component's background color based on the `backgroundColor` state.

Exercise 5: Form Submission

Objective:
Create a simple form with a text input and a submit button that logs the input value to the console when submitted.

Steps:

1. Create a component named SimpleForm.
2. Initialize a state variable to keep track of the input text.
3. Implement a function that updates the state when the input changes.
4. Implement a function that logs the input value when the form is submitted.
5. Create a form with an input field and a submit button. Attach the above functions to the corresponding form events.

Exercise 6: Mouse Over Highlighter

Objective:
Create a React component that changes color when the mouse hovers over it.

Steps:

1. Create a new component called Highlighter.
2. Implement an event handler that changes the component's style when the mouse hovers over it.
3. Attach the event handler to the component's onMouseOver and onMouseOut events.

Exercise 7: Dynamic Input Field

Objective:
Create a React component that dynamically displays the value of an input field.

Steps:

1. Create a component called DynamicInput.

2. Implement an event handler that updates a state variable with the current input value.
3. Attach this handler to the input's onChange event.

Exercise 8: Double Click to Remove

Objective:
Create a list item that can be removed by double-clicking on it.

Steps:

1. Create a component called DoubleClickRemove.
2. Render a list of items.
3. Implement an event handler that removes an item when double-clicked.
4. Attach this event handler to the list items' onDoubleClick event.

Exercise 9: Right-Click Menu

Objective:
Create a React component that displays a context menu when right-clicked.

Steps:

1. Create a component called ContextMenu.
2. Implement an event handler that displays a context menu when right-clicked.
3. Attach this event handler to the component's onContextMenu event.

Exercise 10: Keyboard Events

Objective:
Create a React component that responds to specific keyboard events.

Steps:

1. Create a component called `KeyboardListener`.
2. Implement an event handler for keyboard events.
3. Attach this event handler to the component's `onKeyDown` event.

Topic: State: A Component's Memory

Exercise 11: Local Time Display

Objective:
Create a React component that displays the current local time and updates every second.

Steps:

1. Create a new component called `LocalTime`.
2. Initialize a state variable with the current time.
3. Implement a function to update this state every second.

Exercise 12: Counter with Reset

Objective:
Enhance the Click Counter example to include a reset button that resets the count.

Steps:

1. Add a reset button to the `ClickCounter` component.
2. Implement an event handler to reset the `count` state variable.

3. Attach this event handler to the reset button.

Exercise 13: Text Length Indicator

Objective:
Create a React component with a text input that displays the length of the text entered.

Steps:

1. Create a component called `TextLengthIndicator`.
2. Initialize a state variable to keep track of text length.
3. Render an input box and a text length display area.
4. Update the text length state variable whenever the input changes.

Exercise 14: Password Strength Indicator

Objective:
Create a React component with a password input that indicates the strength of the password.

Steps:

1. Create a component called `PasswordStrength`.
2. Initialize a state variable for the password.
3. Render an input box and a password strength indicator.
4. Implement a function that sets the password strength based on certain criteria.

Exercise 15: Auto-complete Dropdown

Objective:
Create a React component with a text input that shows an auto-complete dropdown based on the input.

Steps:

1. Create a component called AutoComplete.
2. Initialize a state variable for the current text and possible completions.
3. Render an input box and a list of possible completions.
4. Update the possible completions whenever the input changes.

Mini Project: "Dynamic Task Manager"

Objective:

Create a task manager using React where users can add, delete, and update tasks. The project will focus on state management and responding to user events like clicks and form submissions.

Components to Create:

1. **TaskList (Parent Component)**
2. **Task (Child Component)**
3. **AddTaskForm (Child Component)**
4. **UpdateTaskForm (Child Component)**

Project Workflow:

1. **Initialize the project:**

- Create a new React project using `create-react-app`.

2. **Create Components:**
   - Create the TaskList, Task, AddTaskForm, and UpdateTaskForm components.
3. **State Management:**
   - In the `TaskList` component, initialize a state variable `tasks` to keep track of the tasks.
   - Also, maintain a state variable `selectedTask` for updating tasks.
4. **Adding a Task:**
   - Use the `AddTaskForm` component to add a new task.
   - Implement an `onSubmit` event handler in the form to update the `tasks` state in the `TaskList` component.
5. **Displaying Tasks:**
   - In the `TaskList` component, map over the `tasks` state and render each `Task` component.
6. **Deleting a Task:**
   - Implement an `onClick` event handler in each `Task` component that removes the task from the `tasks` state in `TaskList`.
7. **Updating a Task:**
   - Implement an `onClick` event in each `Task` component that sets `selectedTask` in the `TaskList` state.
   - Render the `UpdateTaskForm` when a task is selected. It should pre-fill with the selected task's details.
8. **Conditional Rendering:**
   - Render the `UpdateTaskForm` only if a task is selected.

9. **Dynamic Updates:**
    - Implement an `onSubmit` event handler in the `UpdateTaskForm` to update the `tasks` and reset `selectedTask` in the `TaskList` component.

Bonus Features:

1. **Task Complete Toggle:**
    - Add a checkbox in each `Task` component to mark it as complete or incomplete.
2. **Task Prioritization:**
    - Add a dropdown in the `AddTaskForm` and `UpdateTaskForm` to set the priority of each task (High, Medium, Low).
3. **Search Bar:**
    - Implement a search bar to filter tasks by their names.