Lesson-End Project

Creating a React Application Using React Hooks

Project agenda: To create a Todo List application using hooks and styling

Description: The application enables users to input tasks, triggering event handlers that dynamically update the state. This results in the addition of new tasks to the interface, showcasing the practical implementation of React Hooks for efficient task management.

Tools required: Node terminal, React app, and Visual Studio Code

Prerequisites: Knowledge of creating a React app and an understanding of the folder structure

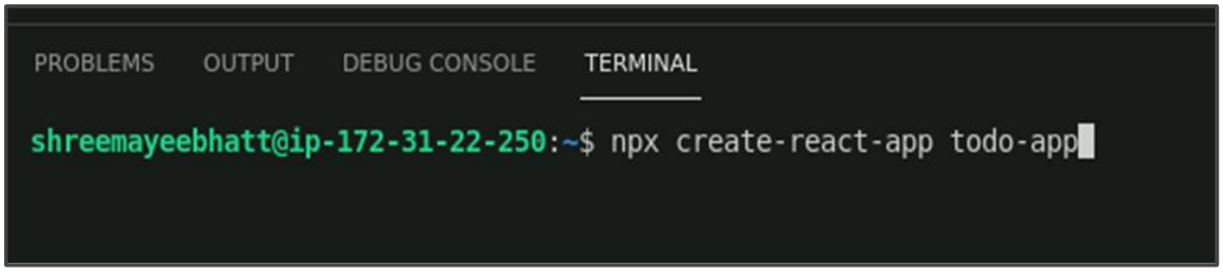
Expected deliverables: Functional React Todo List application with dynamic state updates using React Hooks

Steps to be followed:

1. Create a new React app
2. Configure the files src/App.js and src/App.css
3. Run the app and verify the functionality

Step 1: Create a new React app

1.1. Create a new React project using the **create-react-app** command in your terminal:
npx create-react-app todo-app



This will create a new React app in a directory named **todo-app**.

1.2. Navigate into the newly created directory by running the **cd todo-app** command

Step 2: Configure the files src/App.js and src/App.css

2.1 Open the React project in the Visual Studio code and navigate through the project directory of **todo-app** to open the **src/App.js** file. Replace the existing code in **App.js** with the following code:

```
import React, { useState } from 'react';
import './App.css';
function App() {
 const [tasks, setTasks] = useState([]);
 const [newTask, setNewTask] = useState('');
 const handleAddTask = () => {
  if (newTask.trim() !== '') {
   setTasks([...tasks, { task: newTask, completed: false }]);
   setNewTask('');
  }
 };
 const handleRemoveTask = (index) => {
  setTasks(tasks.filter((task, i) => i !== index));
 };
 const handleToggleCompleted = (index) => {
  const newTasks = [...tasks];
  newTasks[index].completed = !newTasks[index].completed;
 setTasks(newTasks);
 };
 return (
  <div className="App">
   <h1>Todo List</h1>
   <input
    type="text"
    value={newTask}
    onChange={(e) => setNewTask(e.target.value)}
   />

   <button onClick={handleAddTask}>Add Task</button>
   <ul>
```

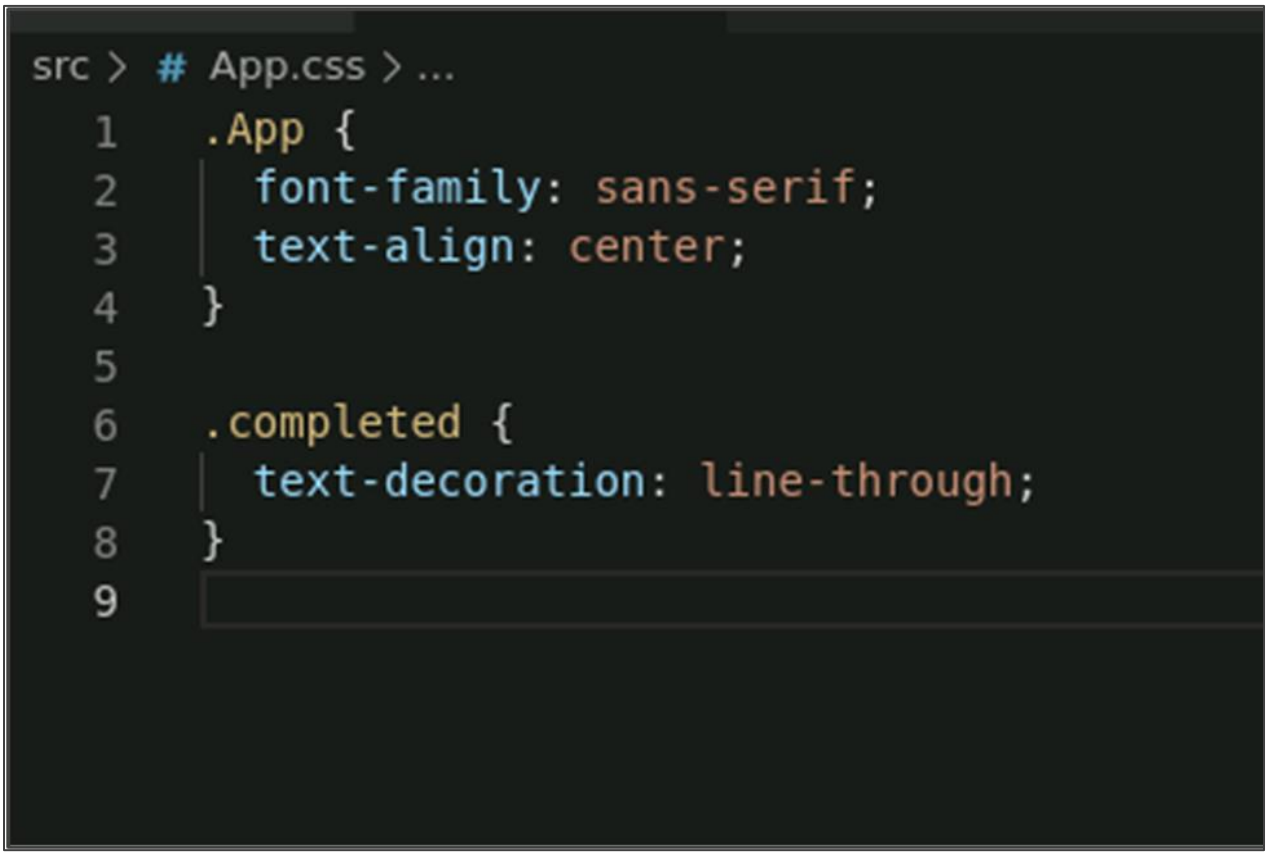
```
      {tasks.map((task, index) => (

        <li key={index}>
          <input
            type="checkbox"
            checked={task.completed}
            onChange={() => handleToggleCompleted(index)}
          />
          <span className={task.completed ? 'completed' : ''}>{task.task}</span>
          <button onClick={() => handleRemoveTask(index)}>Remove</button>
        </li>
      ))}
    </ul>
  </div>
 );
}

export default App;
```

src > JS App.js > ...

```
import React, { useState } from 'react';
import './App.css';

function App() {
  const [tasks, setTasks] = useState([]);
  const [newTask, setNewTask] = useState('');

  const handleAddTask = () => {
    if (newTask.trim() !== '') {
      setTasks([...tasks, { task: newTask, completed: false }]);
      setNewTask('');
    }
  };

  const handleRemoveTask = (index) => {
    setTasks(tasks.filter((task, i) => i !== index));
  };

  const handleToggleCompleted = (index) => {
    const newTasks = [...tasks];
    newTasks[index].completed = !newTasks[index].completed;
    setTasks(newTasks);
  };

  return (
    <div className="App">
      <h1>Todo List</h1>
      <input
        type="text"
        value={newTask}
        onChange={(e) => setNewTask(e.target.value)}
      />
      <button onClick={handleAddTask}>Add Task</button>
      <ul>
        {tasks.map((task, index) => (
          <li key={index}>
            <input
              type="checkbox"
              checked={task.completed}
              onChange={() => handleToggleCompleted(index)}
            />
            <span className={task.completed ? 'completed' : ''}>{task.task}</span>
```

2.2 Similarly, open the **src/App.css** file and replace the existing code with the provided code:

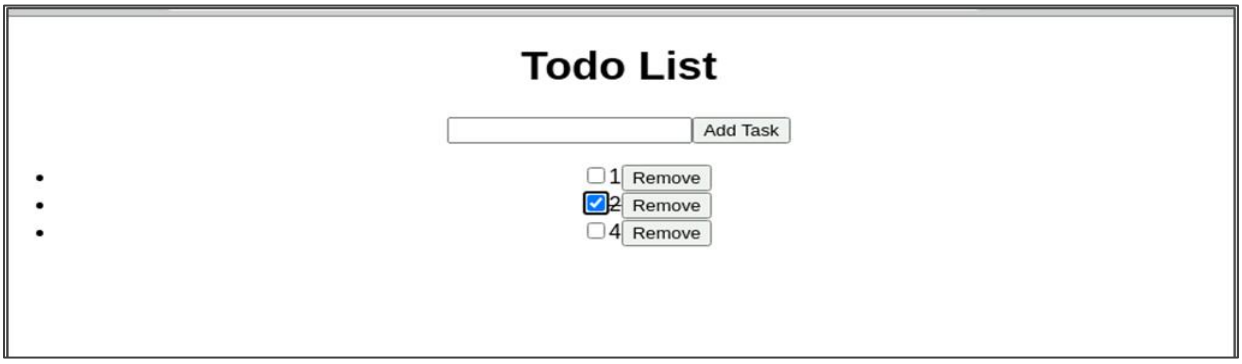
```
.App {

  font-family: sans-serif;
  text-align: center;
}

.completed {
  text-decoration: line-through;
}
```



Step 3: Run the app and verify the functionality

3.1 Go to the terminal and execute the command **npm start** within the **todo-app** project directory to run the application

3.2 Once the server starts successfully, open **http://localhost:3000** in your browser to view the app



In the browser, you will see the heading **Todo List** along with an input field and an **Add Task** button. Enter a task in the input field and click on **Add Task** to add it to the list. Each task will be displayed as a list item with a checkbox. Click the checkbox to toggle the task's completion status, which will be indicated by a line-through text style. Finally, click on the **Remove** button to remove the corresponding task from the list.

With this, you have successfully created a Todo List application using hooks and styling.