Hands-on Lab - React Todo_List Application

Estimated Time Needed: 1 hour 30 mins

In this lab, you will be building a TODO list application using React components.You will discover how to create, view, delete, complete, modify and save to-do lists.

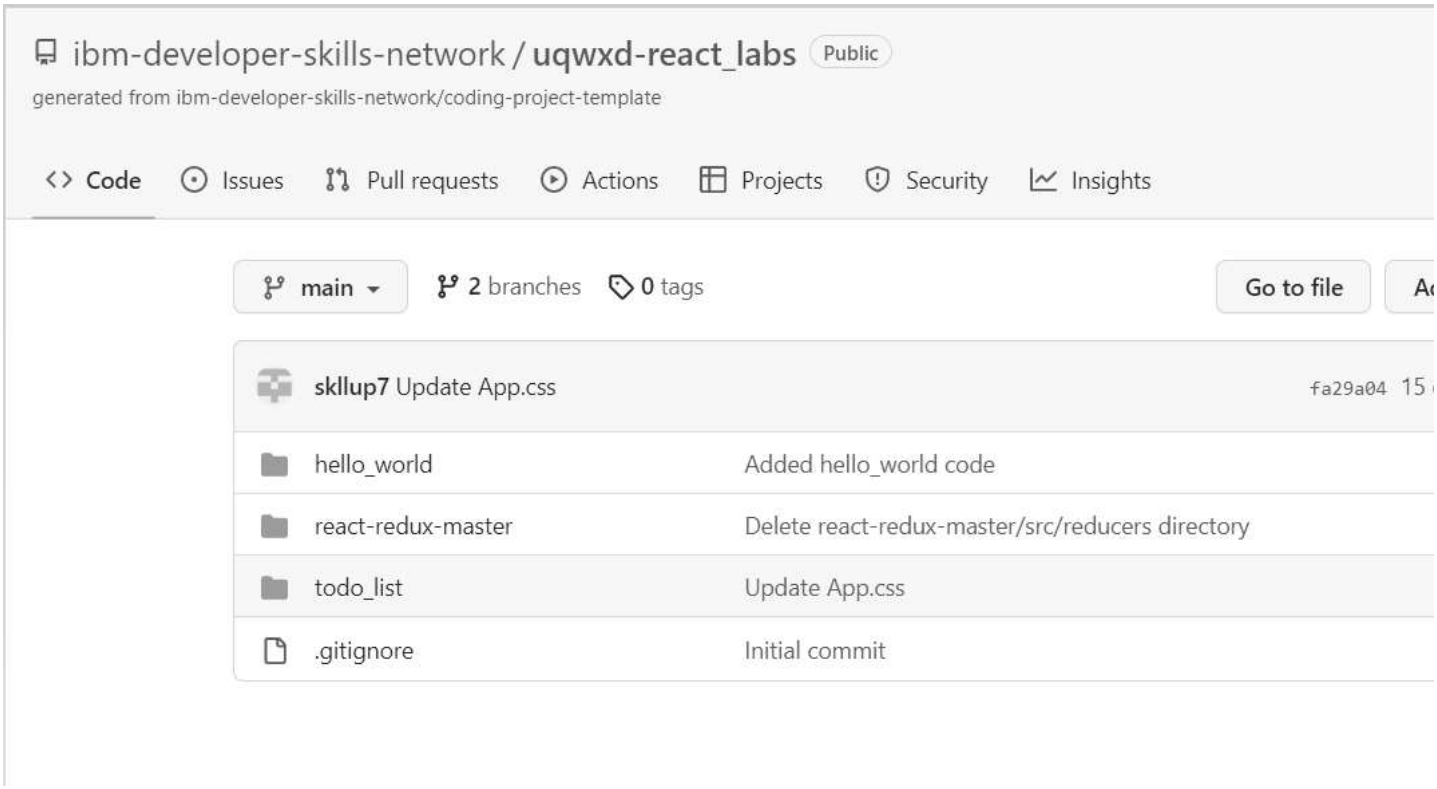
Objective:

After completing this lab, you will be able to:

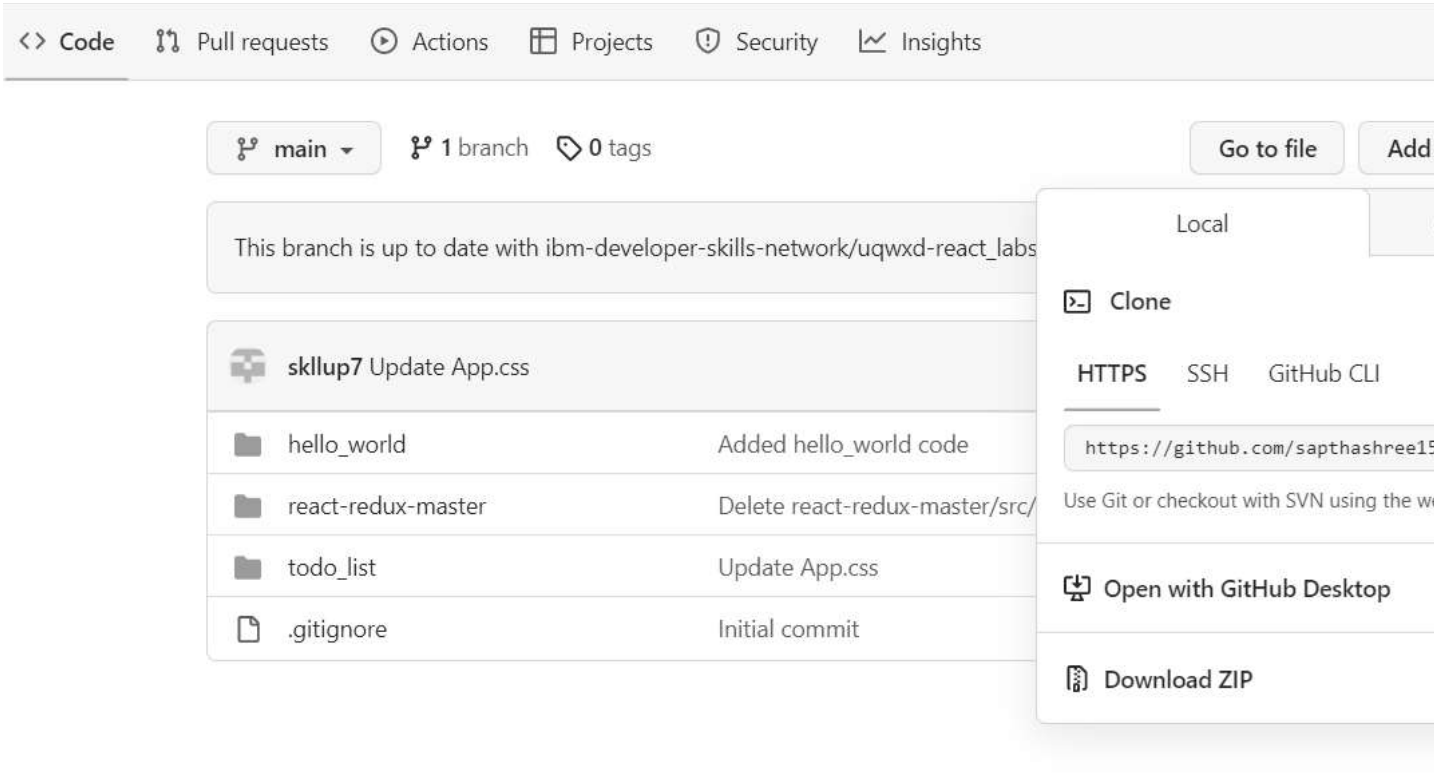
1. Use event handler for creating add todo task.
2. Delete the completed task from todo list using filter method.
3. Add the Toggle function and checkbox to check completed or not completed task.
4. Edit an added Todo task and submit it using map function.
5. Use the UseEffect hook to save new todos into localstorage.

Set-up : Fork and Clone the repository

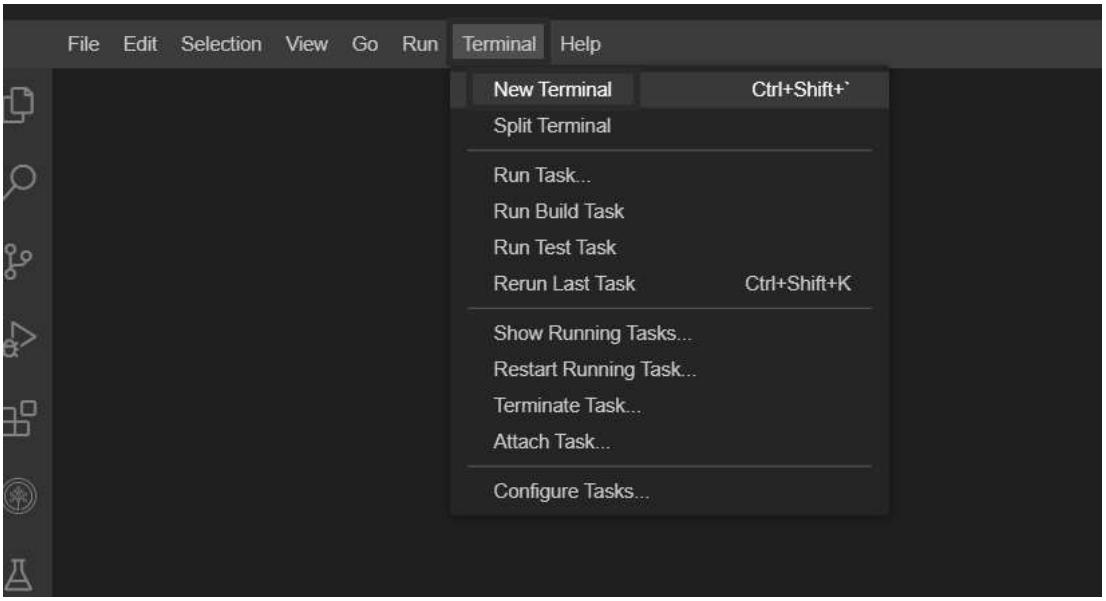
1. Go to the git repository https://github.com/ibm-developer-skills-network/uqwxd-react_labs.git that contains all the labs folder. You will be able to view todo-list for this lab which contains the starter code.
2. Create a fork of the repository into your own. You will need to have a github account of your own to do so.



3. Go to your repository and copy the clone url.



4. In the lab environment, open a terminal window by choosing Terminal > New Terminal from the menu.



5. Change to your project folder, if you are not in the project folder already.

```
1
```

```
cd /home/project
```

Copied! Executed!

6. Clone your forked Git repository into the lab environment, if it doesn't already exist. Please replace your GitHub username in the space provided, before running the command.

```
1
```

```
git clone https://github.com/<your Github username>/uqwxd-react_labs.git
```

Copied!

7. Change to the directory uqwxd-react_labs/todo_list/ to start working on the lab.

```
1
```

```
cd uqwxd-react_labs/todo_list/
```

Copied! Executed!

8. List the contents of this directory to see the artifacts for this lab.

```
1
```

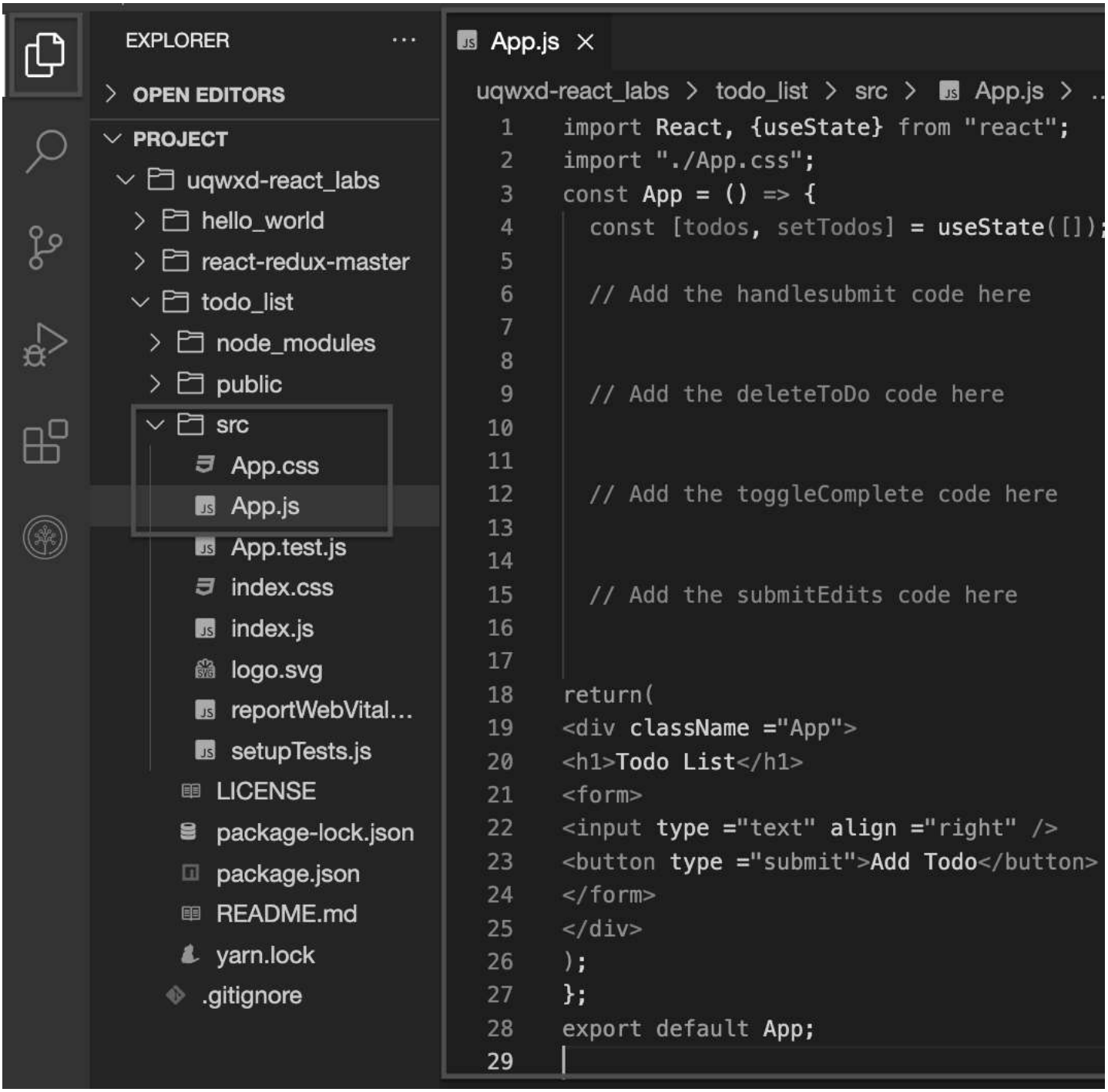
```
ls
```

Copied! Executed!

Installing and running the server for React Application

1. Click on the button below or in the file Explorer go to App.js file under the src folder and view it. It would appear like this.

Open **App.js** in IDE



2. When you initialize the variable for the input field using the useState Hook, you define a getter, to get the value of the state and a setter, to set the value of the state.

In the code in `App.js`, **todos** is the state, which maintains list of all todo tasks, and **setTodos** is the function that sets the value of the state.

```
1
```

```
const [todos, setTodos] = useState([]);
```

Copied!

3. In the terminal, ensure you are in the **uqwxd-react_labs/todo_list** directory and run the following command to install all the packages that are required for running the server.

```
1
```

```
npm install --save -s react react-dom react-scripts web-vitals
```

Copied! Executed!

This will install all the required packages as defined in packages.json.

▶ Click here for instructions to do this on your Windows system.

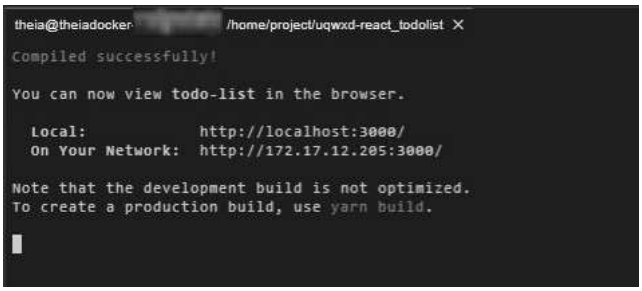
4. Start the server using the below command in the terminal.

```
1
```

```
npm start
```

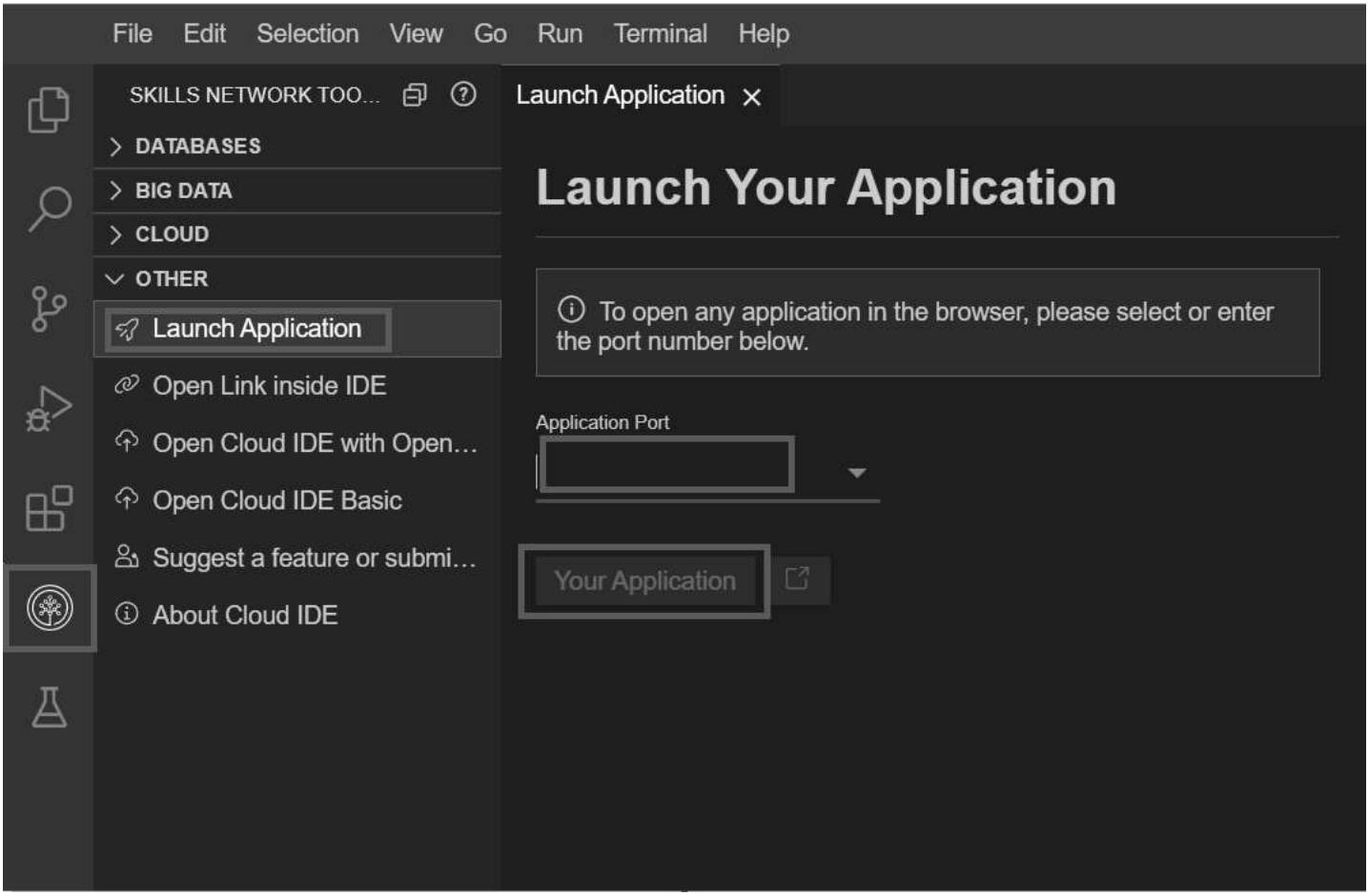
Copied! Executed!

You will see this output indicating that the server is running.



5. To verify that the server is running, click the button below or click on the Skills Network button on the left to open the Skills Network Toolbox. Then click **Other**. Choose Launch Application and enter the port number 3000 on which the server is running and click the launch icon.

To-do Application



The todoapp UI will appear on the browser as seen in the image below.

You cannot add any task to it yet. We will be adding the functionality to do this, in the next part of the lab.

6. To stop the server, go to the terminal in the lab environment and press `ctrl + c`.

Exercise 1: Create and Implement code to add task.

1. In `App.js` file there is a placeholder where you need to add the `handleSubmit` function. Now, give the application the power to add a new task for the to-do list app. You will add a `handleSubmit` function that can will add the task to the list.

The handleSubmit handler will prevent the default action that would normally be taken on the form and add a new Task using the latest value that is in the input field. The user input is validated to ensure the input is non-empty and doesn't have preceeding or succeeding spaces.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
```

```
  function handleSubmit(e) {
    e.preventDefault();

    let todo = document.getElementById('todoAdd').value
    const newTodo = {
      id: new Date().getTime(),
      text: todo.trim(),
      completed: false,
    };
    if (newTodo.text.length > 0 ) {
        setTodos([...todos].concat(newTodo));
    } else {

        alert("Enter Valid Task");
    }
    document.getElementById('todoAdd').value = ""
  }
```

Copied!

2. On click of the `Add Todo` button, the form is submitted and the task is added to the todo list. The code uses the map to iterate through the todo list, and renders each task as a list item. Using useState, this component registers a state with value `todo` and a function for updating it `setTodo`.

Paste the below code inside the return function of App, replacing the existing content.

```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
```

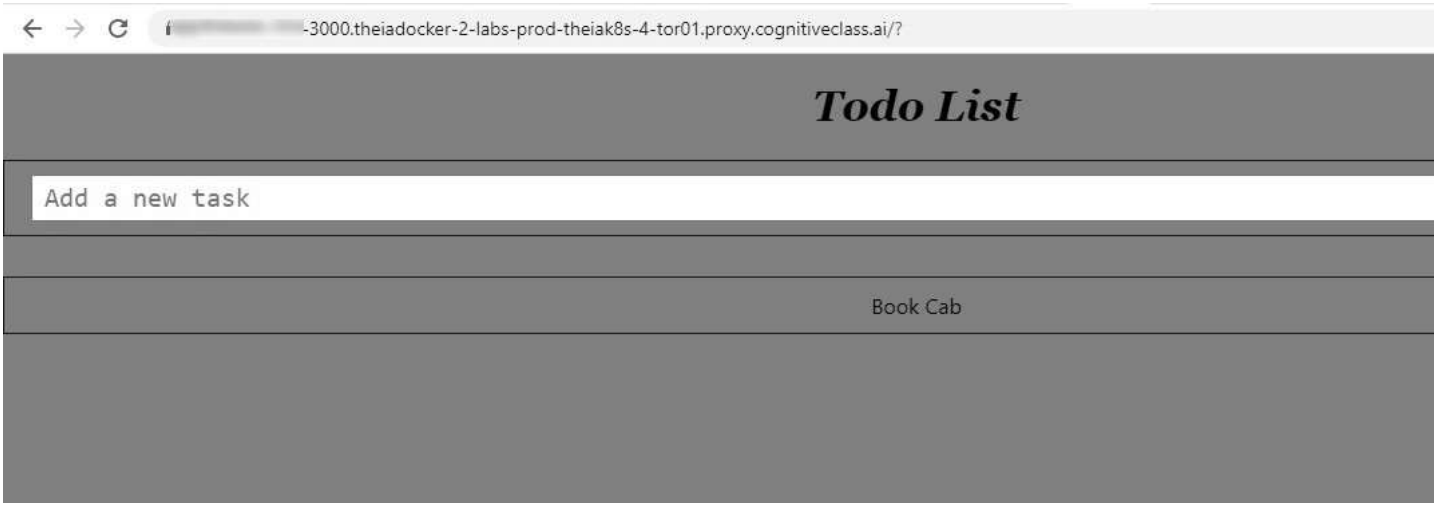
```
<div id="todo-list">
    <h1>Todo List</h1>
        <form onSubmit={handleSubmit}>
            <input
              type="text"
              id = 'todoAdd'
```

```
            />
            <button type="submit">Add Todo</button>
        </form>
        {todos.map((todo) =>
            <div className="todo" key={todo.id}>
                <div className="todo-text">{todo.text}</div>
            {/* insert delete button below this line */}
            </div>)}
</div>
```

Copied!

▶ Click here to view the complete code.

3.Start the server and enter the valid task inside the input box where you can see “Add a new task “. You will see the ouput as below:



Exercise 2: Delete a completed task from the list.

1. Delete task can be handled in many ways. Now you will write the code using the filter method that filters out the task to be deleted based on the id, and returns the rest of the tasks. Placeholder is added in the App.js file where you need to add the delete to do function.

```
1
2
3
4
```

```
function deleteTodo(id) {
    let updatedTodos = [...todos].filter((todo) => todo.id !== id);
    setTodos(updatedTodos);
  }
```

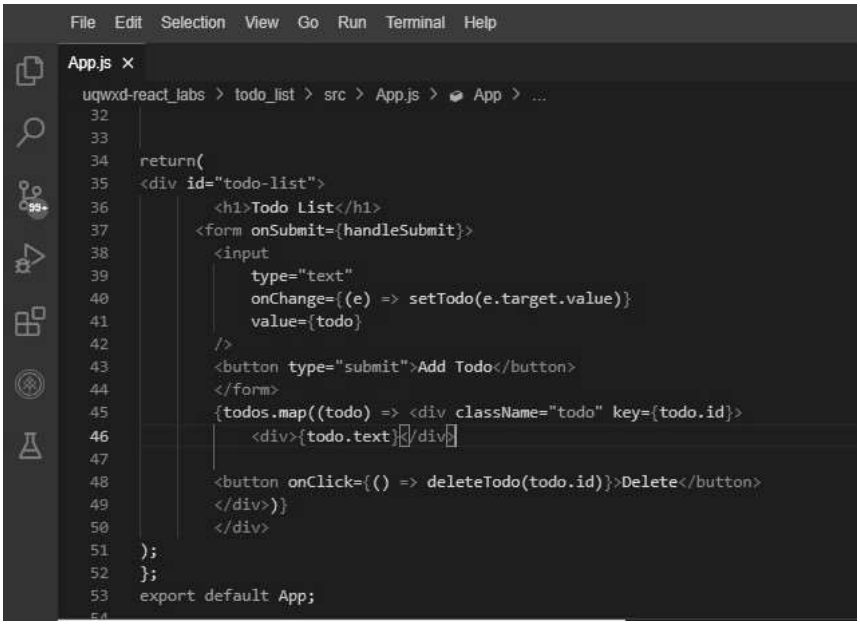
Copied!

2. Add a button to delete the task, which will call the method `deleteTodo` passing the id of the todo item, on click. A placeholder has been provided to add the button.

```
1
```

```
<button onClick={() => deleteTodo(todo.id)}>Delete</button>
```

Copied!



3. Start the server, add the task to-do list, and then try to delete it by pressing the delete button. The task must be removed from the list.

Exercise 3: Adding Checkbox and Toggle function.

1. You will now add a checkbox to mark task completion. Create a new function, **toggleComplete** function that uses the map method to iterate through the task and mark them complete inside App.js.

```
1
2
3
4
5
6
7
8
9
```

```
function toggleComplete(id) {
    let updatedTodos = [...todos].map((todo) => {
      if (todo.id === id) {
        todo.completed = !todo.completed;
      }
      return todo;
    });
    setTodos(updatedTodos);
  }
```

Copied!

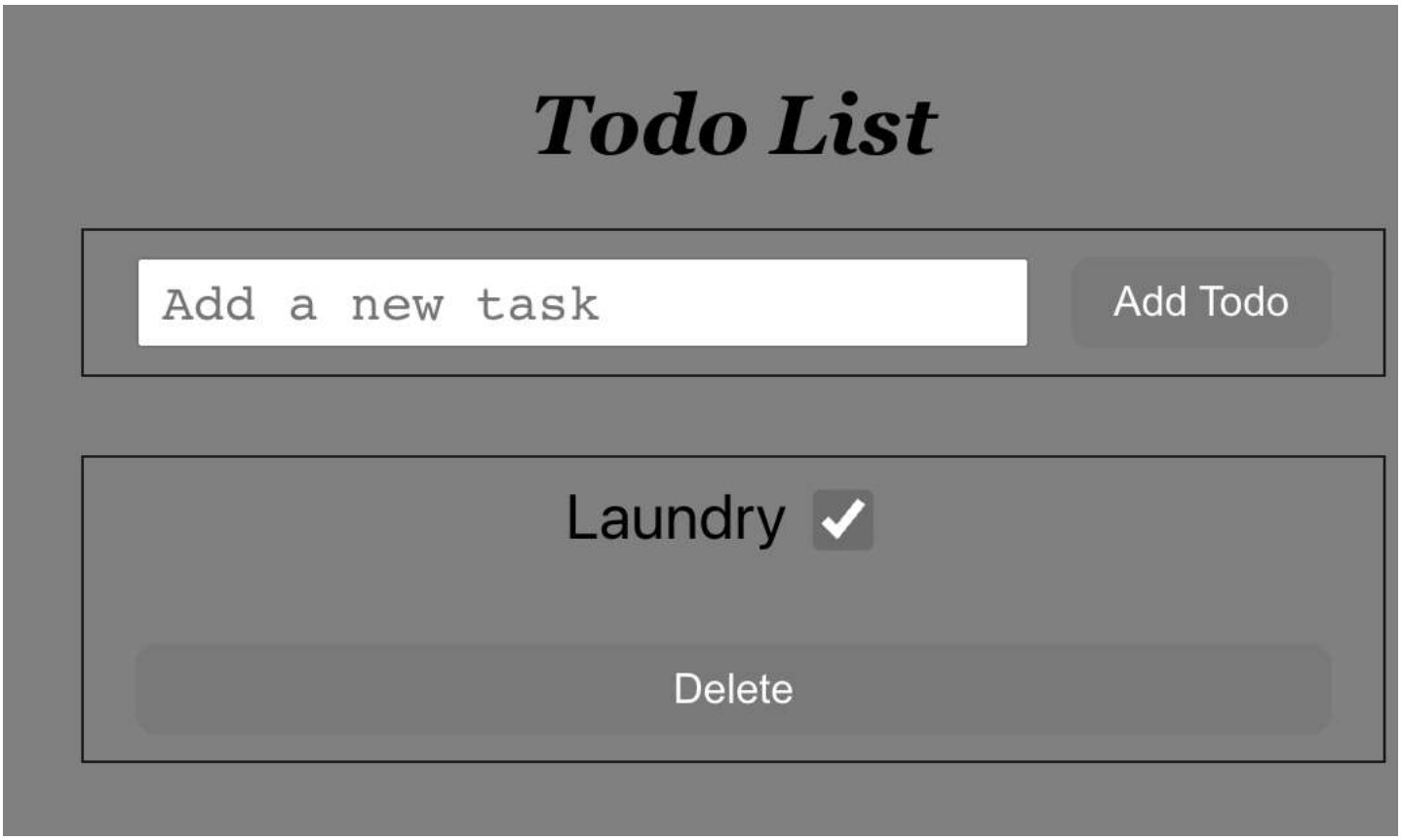
2. Add the following code to add the checkbox component, next to `{todo.text}`, inside the same `div` tag.

```
1
```

```
<input type="checkbox" id="completed" checked={todo.completed} onChange={() => toggleComplete(todo.id)}/>
```

Copied!

3. Start the server, add the task to-do list, and then use the checkbox if the task is completed.



Exercise 4: Edit a added Todo task and submit it.

1. In the previous exercise, you added the ability to toggle the checkbox for completed task in todos. Now add the functionality to edit the task.
2. You need to another state to implement the editing functionality.

```
1
```

```
const [todoEditing, setTodoEditing] = useState(null);
```

Copied!

3. Now add the submitEdit function in the App.js which will help you to submit the task edit of todo list using the map function.

```
1
2
3
4
5
6
7
8
9
10
```

```
function submitEdits(newtodo) {
    const updatedTodos = [...todos].map((todo) => {
      if (todo.id === newtodo.id) {
        todo.text = document.getElementById(newtodo.id).value;
        }
        return todo;
      });
      setTodos(updatedTodos);
      setTodoEditing(null);
    }
```

Copied!

4. You should allow edits only in the edit mode. Provide a button which will enable the edit mode for each task. When edit mode in enabled, switch the task label to a textbox where one can edit the to-do task and submit it back to the list. In addition to that, when the edit mode is enabled, change the `Edit` button to `Submit Edit` button, which will invoke `submitEdit` function when cliked. The code below has been provided for this purpose. It is achieved using ternary operator.

Replace the code inside the `return` block with the following code.

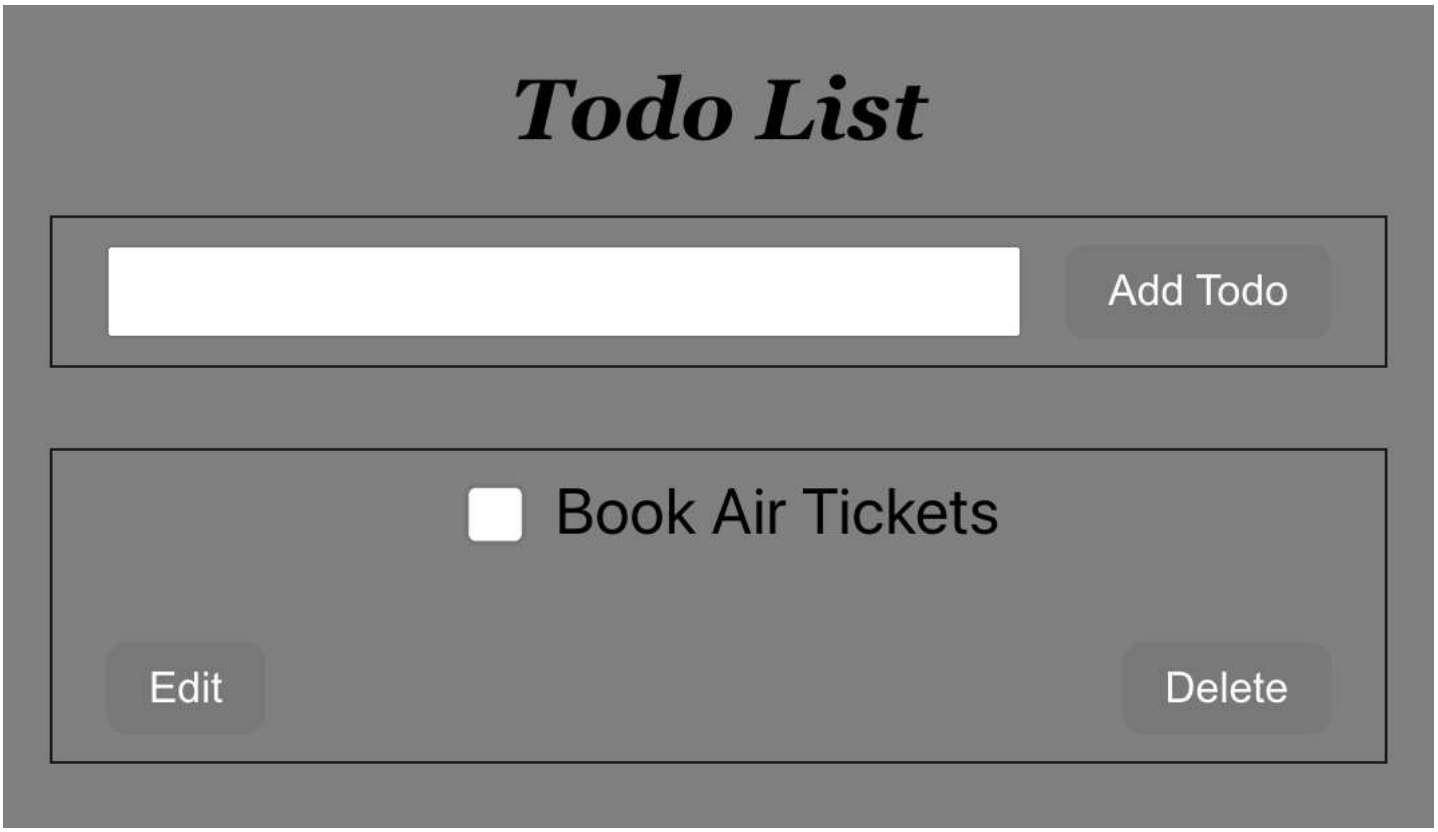
```
1
2
3
4
5
6
7
8
9
10
11
12
```

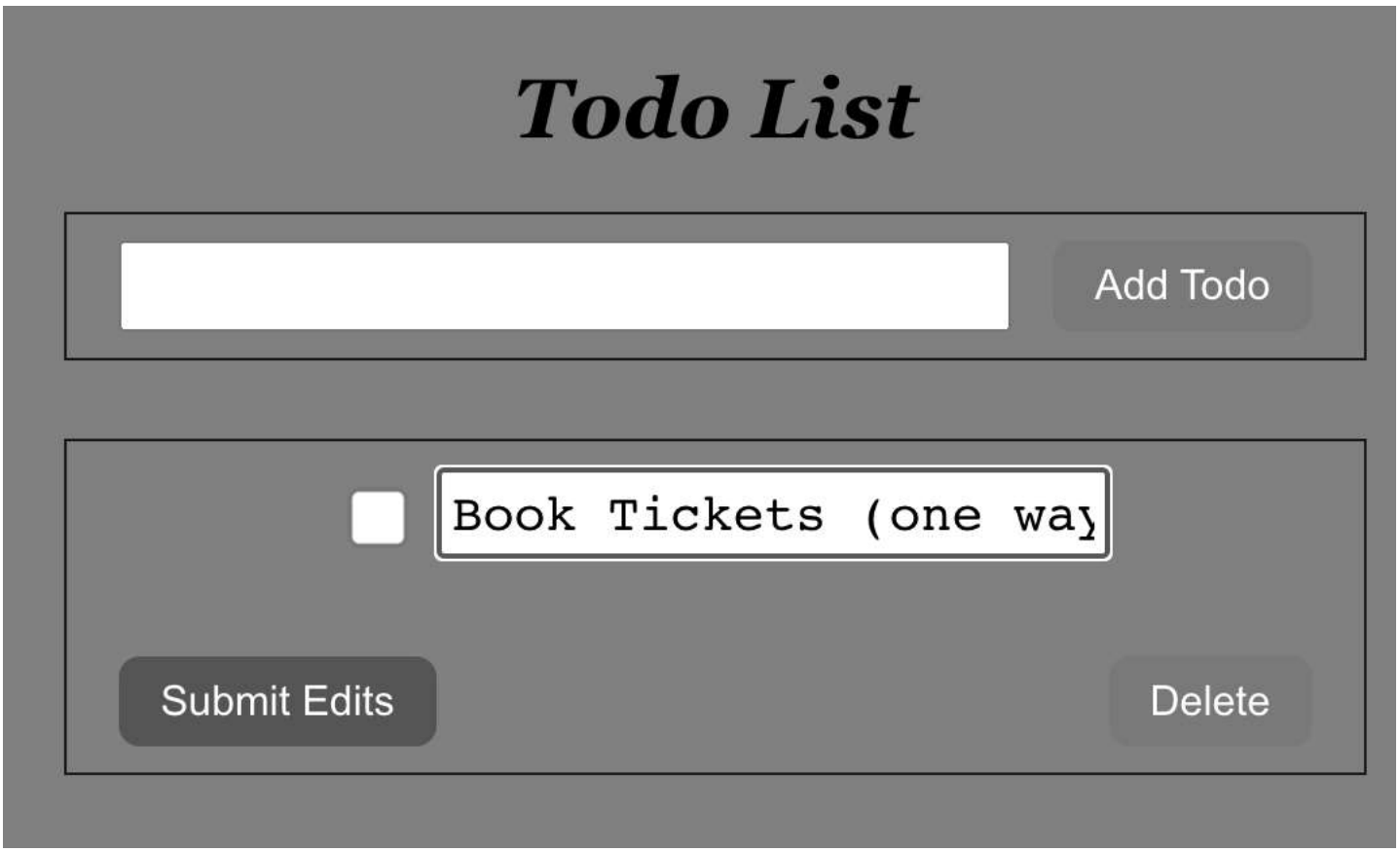
```
        <div id="todo-list">
          <h1>Todo List</h1>
          <form onSubmit={handleSubmit}>
            <input
              type="text"
              id = 'todoAdd'
            />
            <button type="submit">Add Todo</button>
          </form>
        {todos.map((todo) => (

          <div key={todo.id} className="todo">
            <div className="todo-text">
              {/* Add checkbox for toggle complete */}
              <input
                type="checkbox"
                id="completed"
                checked={todo.completed}
                onChange={() => toggleComplete(todo.id)}
              />
              {/* if it is edit mode, display input box, else display text */}
              {todo.id === todoEditing ?
                (<input
                  type="text"
                  id = {todo.id}
                  defaultValue={todo.text}
                />) :
                (<div>{todo.text}</div>)
              }
            </div>
            <div className="todo-actions">
              {/* if it is edit mode, allow submit edit, else allow edit */}
              {todo.id === todoEditing ?
              (
                <button onClick={() => submitEdits(todo)}>Submit Edits</button>
              ) :
              (
                <button onClick={() => setTodoEditing(todo.id)}>Edit</button>
              )}

              <button onClick={() => deleteTodo(todo.id)}>Delete</button>
             </div>
          </div>
        ))}
        </div>
```

Copied!

▶ Click here to view the complete code.

5. Start the server, add the task to-do list and edit it using the edit button.





Exercise 5: Adding the useEffect hook.

You will add the useEffect hook to your application. This useEffect hook will be responsible to save new todos into localstorage.

1. Import the useEffect from the react package. Replace the existing import statement with following code.

```
1
```

```
import React, {useState,useEffect} from "react";
```

Copied!

2. Now add the code which will allow you to store the tasks in the local storage as a JSON, just below the states definition.

```
1
```

```
2
3
4
5
6
7
8
9
10
11
12
13
14
```

```
useEffect(() => {
    const json = localStorage.getItem("todos");
    const loadedTodos = JSON.parse(json);
    if (loadedTodos) {
      setTodos(loadedTodos);
    }
  }, []);

useEffect(() => {
    if(todos.length > 0) {
        const json = JSON.stringify(todos);
        localStorage.setItem("todos", json);
    }
  }, [todos]);
```

Copied!

The wholistic view

1. After putting everything together, the final code for To-Do list is as follows.

▶ Click here to view the complete code

2. (Optional) Please commit and push all the changes to your forked Github repo.

   Note: Please refer to this lab, if you need any help in committing and pushing changes to your repo.

Congratulations! You have completed the lab for creating To-Do list Application using React.

Summary

In this lab, you used React to build a to-do list application that allows you to view the list of tasks, add tasks, edit tasks, and delete tasks.

Author(s)

Sapthashree K S

Other Contributor(s)

Lavanya T S

Changelog

Date	Version	Changed by	Change Description
2022-10-28	1.0	Sapthashree K S	Initial version created based videos
2023-01-31	1.1	Sapthashree K S	Instructions and screenshots updated
2023-09-20	1.2	Lavanya T S	Instructions and screenshots updated
2023-10-04	1.3	Sapthashree K S	Minor code correction