**COMSATS University Islamabad (CUI), Lahore Campus**

**Department of Electrical & Computer Engineering**

# CSC336 – Web Technologies

Lab Manual v 1.0.1

**Lab Resource Person**

Muhammad Babar Ali

**Theory Resource Person**

Muhammad Babar Ali

FALL 2024

| | | | |
|---|---|---|---|
| **Name:** | | **Registration No:** | |
| **Program:** | | **Batch:** | |
| **Semester:** | | | |

# Laboratory 08

# To manipulate HTML DOM in React.js



## 1 Objective

The purpose of the lab is to introduce students to the *referencing and forwarding references* in React.js. Students will learn basics to create application programming interface (API) to expose certain functionality on custom components in React. They will become familiar with built-in React Hooks. Students will learn state management by using **useState** – the State Hook. They will extend their knowledge by building an API using **useRef –** the Ref Hook.

## 2 Introduction to React Hooks

A special mechanism which facilitates the components to access or use different features is called Hook. React.js also offers following types of built-in Hooks, commonly referred as React Hooks:

- State Hooks
- Ref Hooks
- Context Hooks
- Effect Hooks
- Performance Hooks
- Resource Hooks
- Library Hooks

In addition to the above categories of built-in React Hooks, an application developer can write custom Hooks too. However, the following laboratory will elaborate on usage of a State Hook and Ref Hooks.

## 3 Exposing DOM node to parent

To expose some functionality of the DOM node in a child component to its parent component, the react offer a solution with the help of forwardRef function.

## 4 Lab Activity

### 4.1 Task 1

Use React to develop an application namely “todo-app” as shown in Figure 4.1.

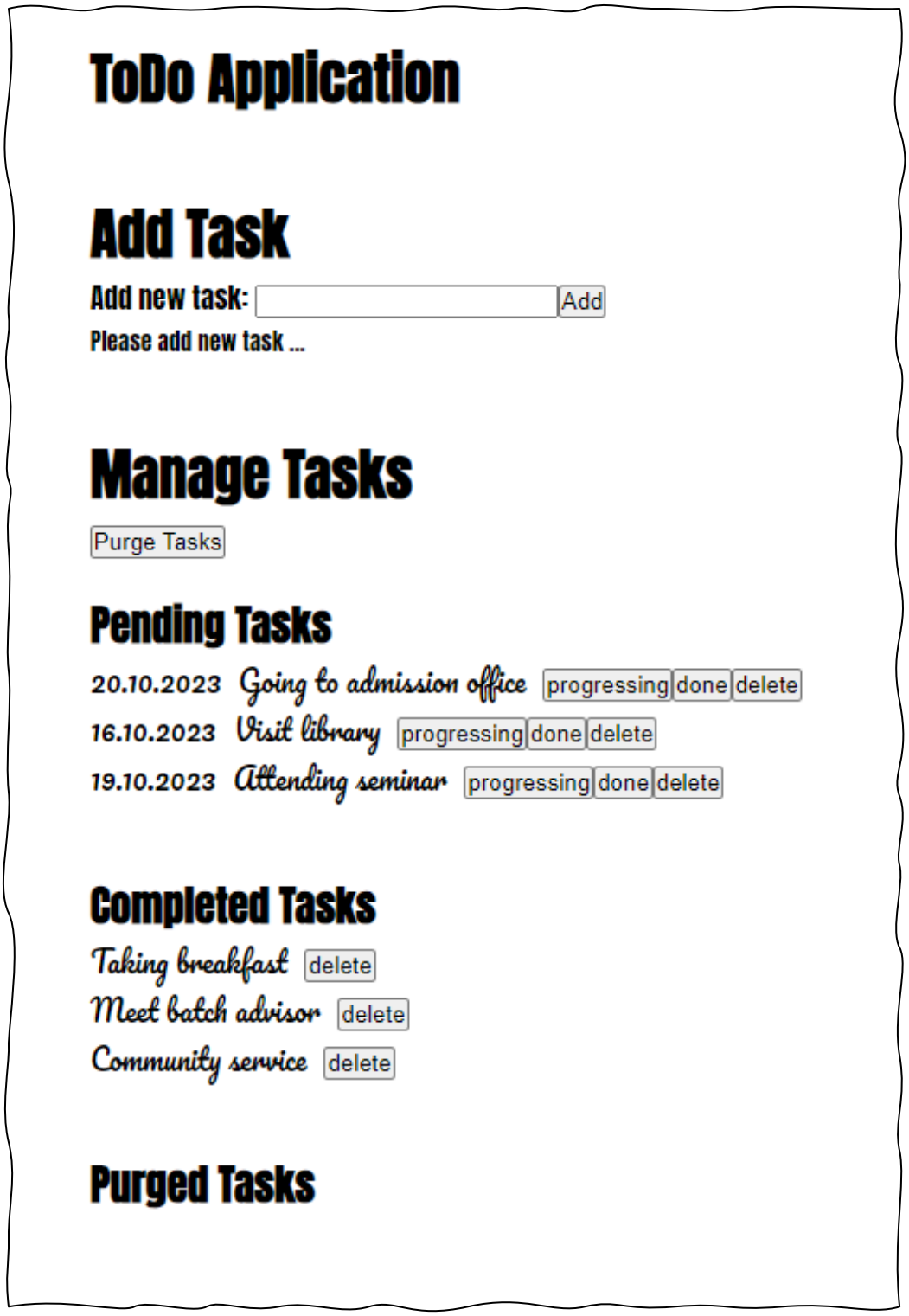


Figure 4.1: UI for ToDo React Application

Proceed through the following tasks.

## Part 01: Events and States Lifting

The initial markup will be provided by the lab instructor for the application shown in Figure 4.1. You are required to perform the following modifications for the **useState** React Hook:

- Lift all the events to the App component.
- Lift the states up to App component.

## Part 02: Update Parent and Child Components

In this task, you are going to enable the purge tasks button in the <ManageTasks> component so that when the user clicks on it, it will delete all the in-progress tasks in the Pending Tasks list. To perform this task, we will also use another React Hook called **useRef**. Proceed through the following steps to achieve the desired result.

### Step 01: Updating ManageTasks.jsx

**a)**

Import useRef in ManageTasks.jsx as shown below:

```
import { useRef } from “react”;
```

**b)**

Get reference to the ref variable (we are naming it as purgeInProgress) using useRef Hooks, by adding the following line in code for the <ManageTasks> component (i.e. In ManageTasks(props) functions: Before *const [showContent, setShowContent] = useState(true);*)

```
const purgeInProgress = useRef();
```

**c)**

Next, add the following handle function in the ManageTasks functional component.

```
  function onPurge() {
    console.log("[ManageTasks] Purge tasks ...");
    console.log("[ManageTasks] " +
purgeInProgress.current);
```

```
    purgeInProgress.current();
  }
```

**d)**

Next, replace the following line for "Purge Tasks" button,

```
<button>Purge Tasks</button>
```

with the following line to add an onClick event listener.

```
<button onClick={onPurge}>Purge Tasks</button>
```

**e)**

Pass the ref attribute to <PendingTasks> component from within <ManageTasks> component and set its value to ref JS variable in ManageTasks() function. This attribute value pair is written below.

```
ref={purgeInProgress}
```

**Step 02: Updating PendingTasks.jsx**

To receive ref in the <PendingTasks> component, the PendingTasks (.e. PendingTasks function) must be passed as an argument to the **forwardRef** function. However, now PendingTasks() must have *ref* as the 2nd argument and *props* as the 1st argument. To do this, proceed through the instructions from a to d, below.

**a)**

Import **forwardRef** using following line in PendingTasks.jsx

```
import { forwardRef } from "react";
```

**b)**

To receive ref in the <PendingTasks> component replace the following line

```
function PendingTasks(props) {
```

with

```
const PendingTasks = forwardRef(function (props, ref) {
```

**c)**

In the <PendingTasks> component, add the following handler function and the instruction

```
const purge = function () {
    console.log("Purge tasks ...");

    InProgress.forEach((id) => {
      onDelete(id);
    });
  };

  ref.current = purge;
```

below the following line in the PendingTasks.jsx

```
const [InProgress, setInProgress] = useState([]);
```

**d)**

The following line (closing brace of the PendingTask()) in PendingTasks.jsx

```
}
```

which is written before

```
export default PendingTasks;
```

replace it with

```
});
```

**Part 03: Verifying the Functionality**

In this task, you are going to verify the operation of "Purge Tasks" button by proceeding through the instructions below.

**a)**

Refresh the browser at http://localhost:3000/. Make sure that react app development server must be running.

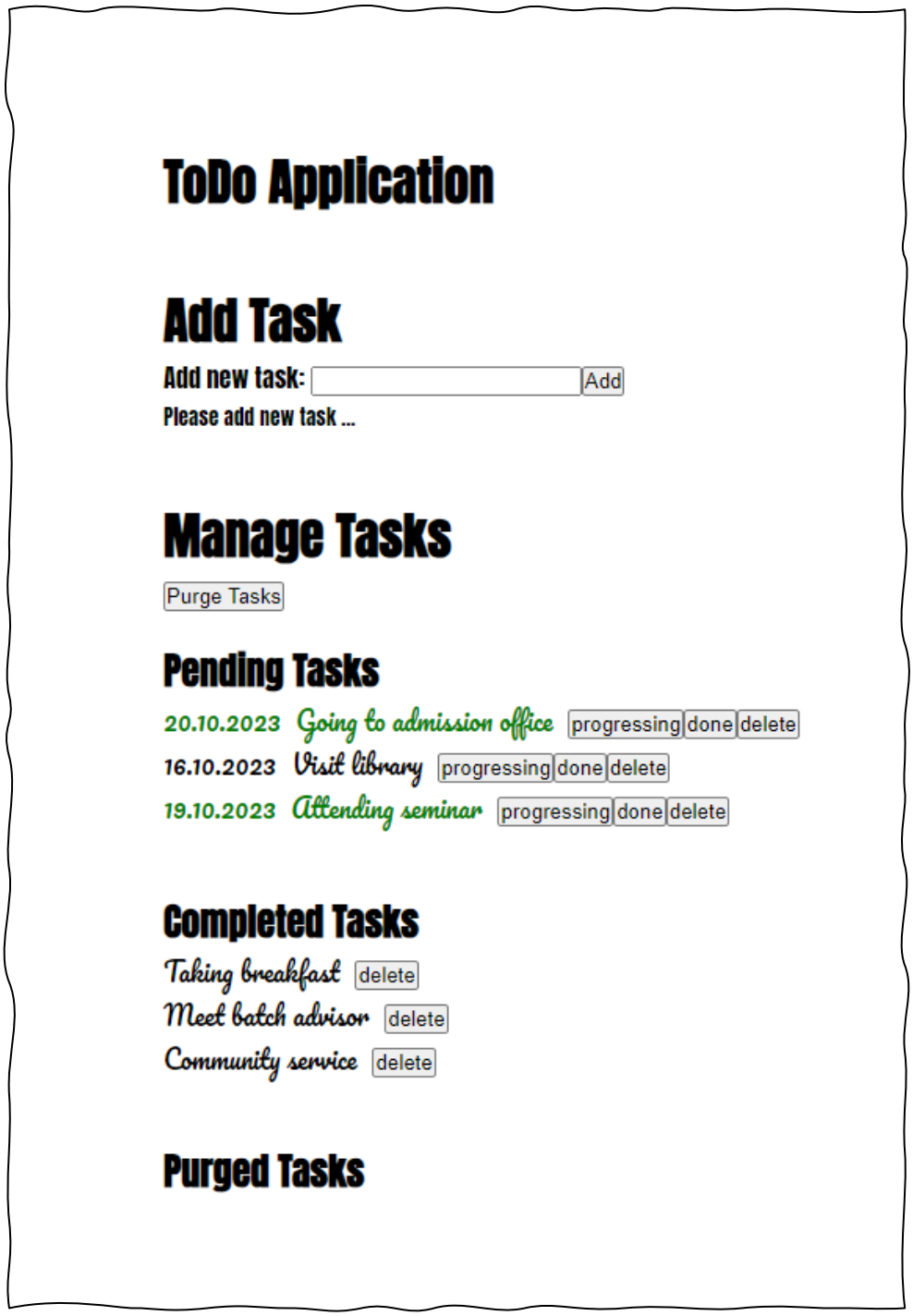


Figure 4.2: UI with two in-progress tasks

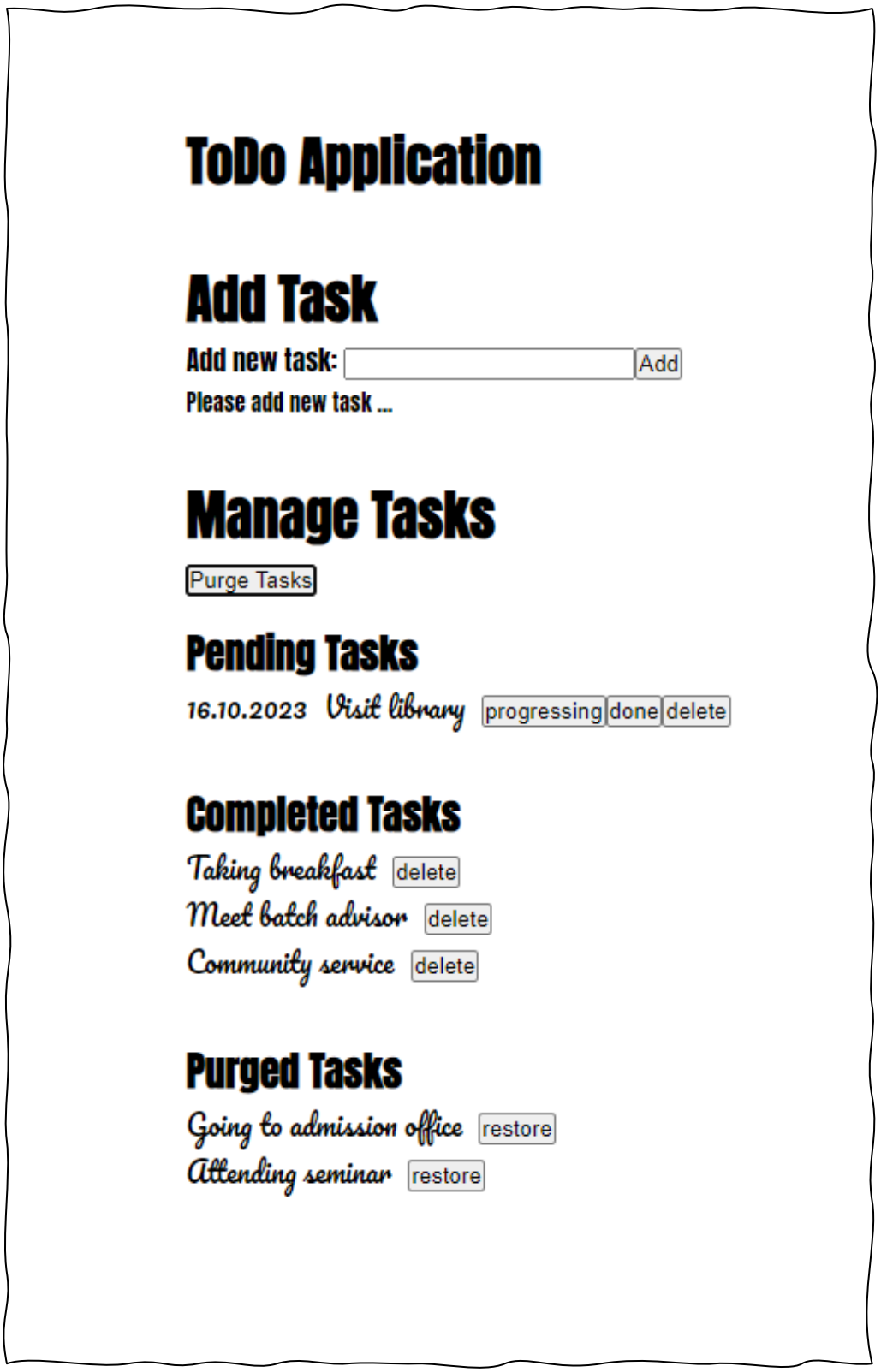


Figure 4.3: UI after clicking "Purge Tasks" button

**b)**

Mark any of the two out of three pending tasks as progressing by clicking on the "progressing" button once. The text color of the title of these progressing tasks must have been changed to **green** color, as shown in Figure 4.2.

**c)**

Click on "Purge Tasks" button under the "Manage Tasks" heading in UI.

**d)**

The two in-progress tasks (tasks in **green** color) from step b, must have now been moved to Purge Tasks list at the bottom of the web page as shown in Figure 4.3.

## 4.2 Task 2

Create the shopping web application as shown in Figure 4.4. You can use vanilla JavaScript application by clicking here.

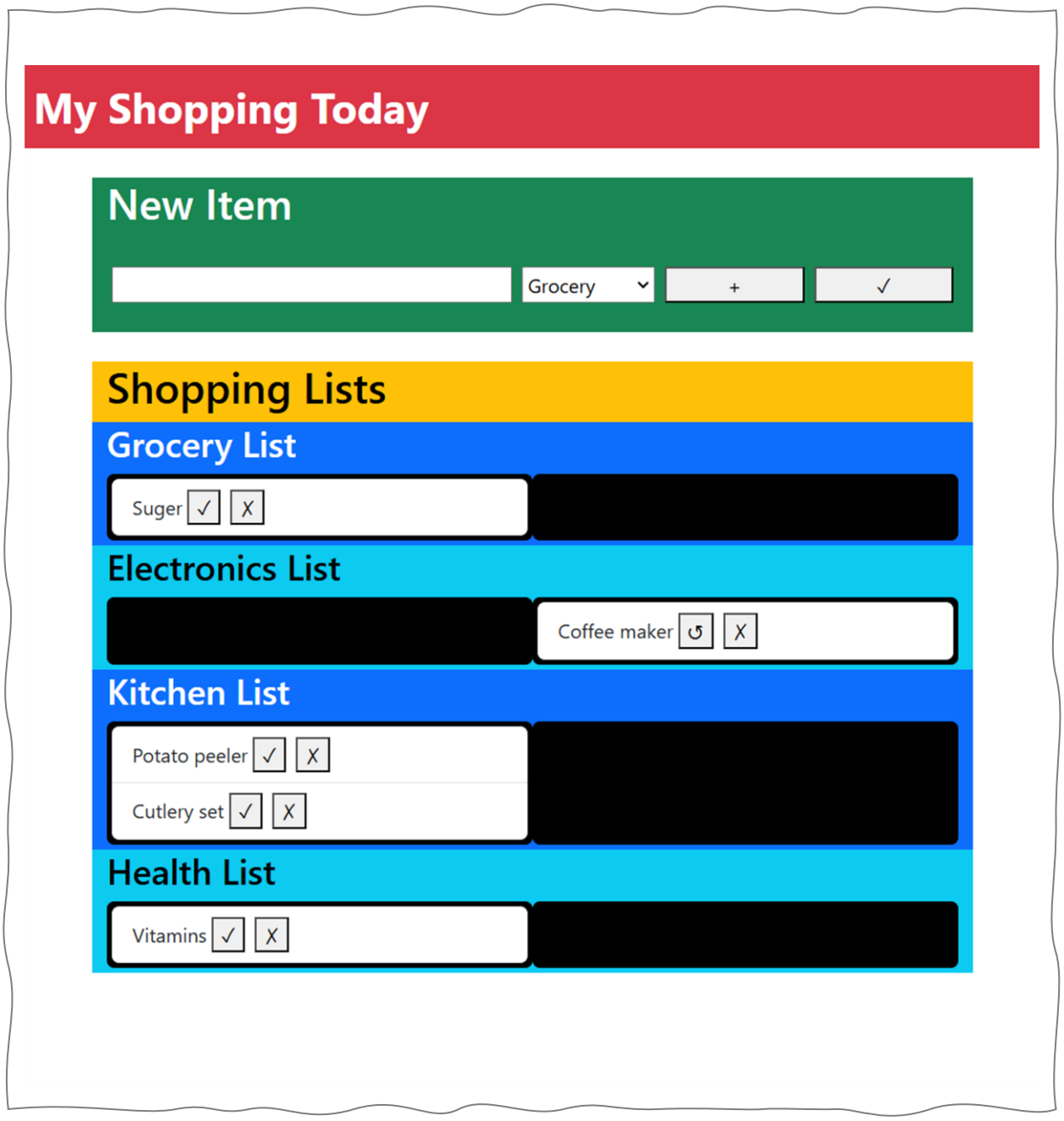


Figure 4.4: Shopping list app

# 5 Home Activity

## 5.1 Work 1

Use React DnD library to introduce drag-n-drop feature in the react application created in Task 2 (Figure 4.4).

# 6 External Links

| | |
|---|---|
| React Hooks | https://react.dev/reference/react/hooks |
| useState Hook | https://react.dev/reference/react/useState |
| useRef Hook | https://react.dev/reference/react/useRef |
| forwardRef API | https://react.dev/reference/react/forwardRef |
| React Drag n drop | https://react-dnd.github.io/react-dnd/about |

# 7 Rubric for Lab Assessment

| **The student performance for the assigned task during the lab session was:** | | | |
|---|---|---|---|
| Excellent | The student completed assigned tasks without any help from the instructor and showed the results appropriately. | 4 | |
| Good | The student completed assigned tasks with minimal help from the instructor and showed the results appropriately. | 3 | |
| Average | The student could not complete all assigned tasks and showed partial results. | 2 | |
| Worst | The student did not complete assigned tasks. | 1 | |

**Instructor Signature:**_____ **Date:** _____