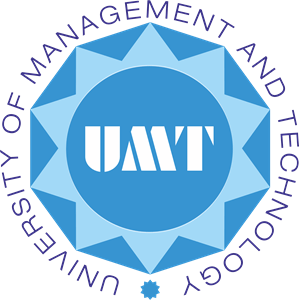
## Advance Web Technology

**React-app**



**Project Supervisor: Sir Safiullah**

**Submitted By:**

Maryam Shehzadi 20006105016

BS-IT BATCH 6

## Session

## (2020-2024)

## University of Management and Technology

## Sialkot, Pakistan

**Question: 01**

**React To-Do List Challenge: Boost Your Productivity!**

**Objective:**

**Build a single-page to-do list application using React.js. This application should offer**

**features to manage your daily tasks effectively.**

**Functionality:**

**Task Management:**

* **Users can enter new tasks using an input field.**
* **New tasks are added to the to-do list on submission.**
* **Existing tasks can be marked as completed by clicking a checkbox beside**
* **them.**
* **Users should be able to delete unwanted tasks from the list.**

**Persistence:**

* **Utilize local storage to save the current state of the to-do list (including**
* **tasks and their completion status) even after the browser is refreshed.**
* **When the application loads, retrieve the saved data from local storage and**
* **populate the list accordingly.**

**Technical Specifications:**

* **Use React.js to build the user interface and manage component state.**
* **Employ JavaScript for handling user interactions and logic.**
* **Consider using CSS for basic styling of the application.**
* **Bonus Features (Optional):**
* **Implement a &quot;clear completed&quot; button to remove all completed tasks from the list**
* **at once.**

**Allow users to edit existing tasks by double-clicking or providing an edit**

**functionality.**

* **Categorize tasks by priority levels (high, medium, low).**
* **Enhance user experience with animations or visual feedback for actions like**
* **adding, deleting, or marking tasks complete.**

**Evaluation Criteria:**

* **Functionality: (70%) Does the application fulfill all the core requirements**
* **(adding, marking complete, deleting tasks) and persist data using local storage?**
* **Code Quality: (20%) Is the code well-structured, readable, and commented?**
* **User Interface: (10%) Does the application have a clean and user-friendly**
* **interface? (Bonus points for extra features)**

**Hints:**

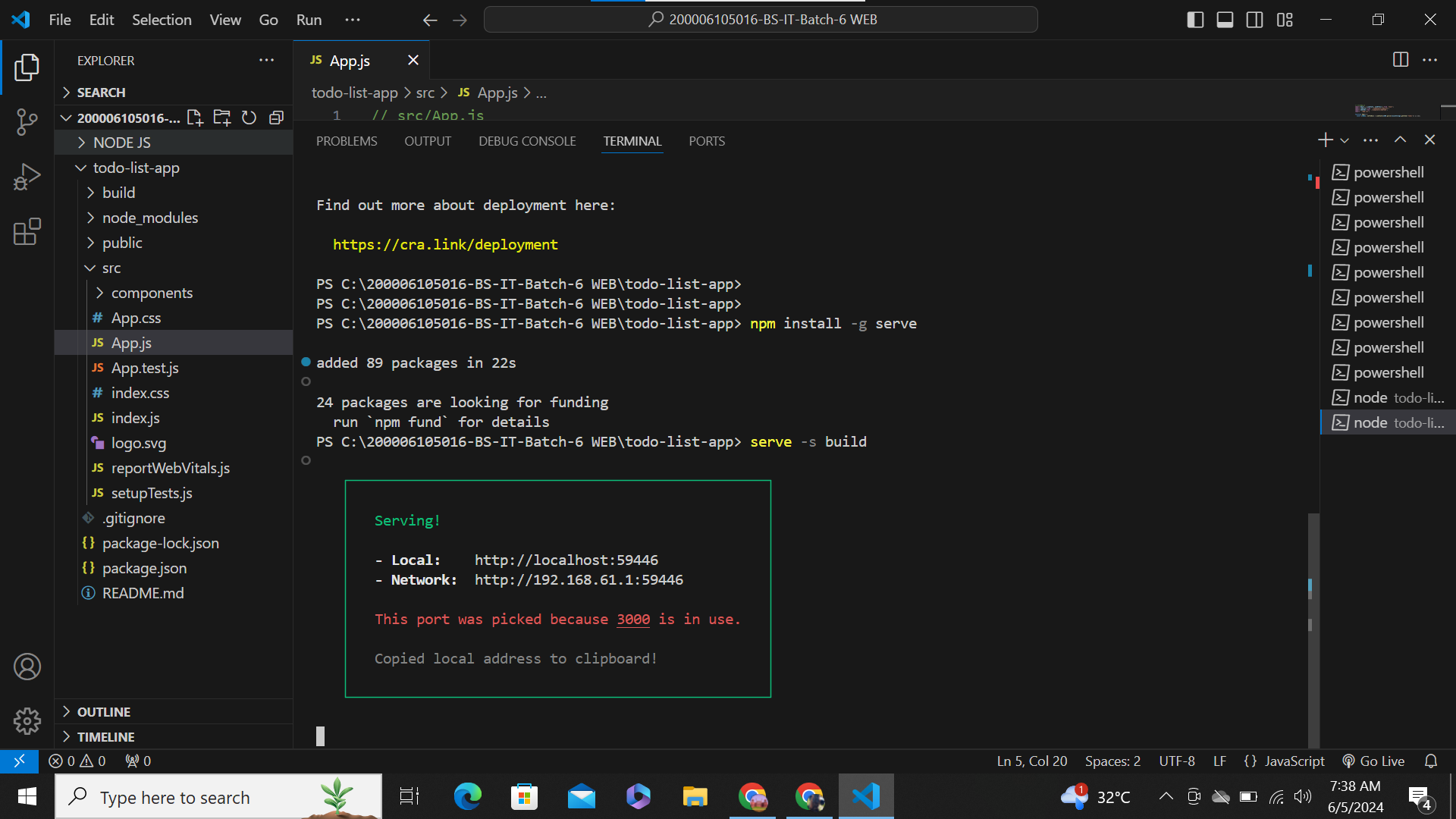
* **Take advantage of React&#39;s state management features like useState to track the**
* **list of tasks and their completion status.**
* **Explore using local storage APIs to save and retrieve data.**
* **Consider using libraries like classnames for conditional styling based on task**
* **completion.**

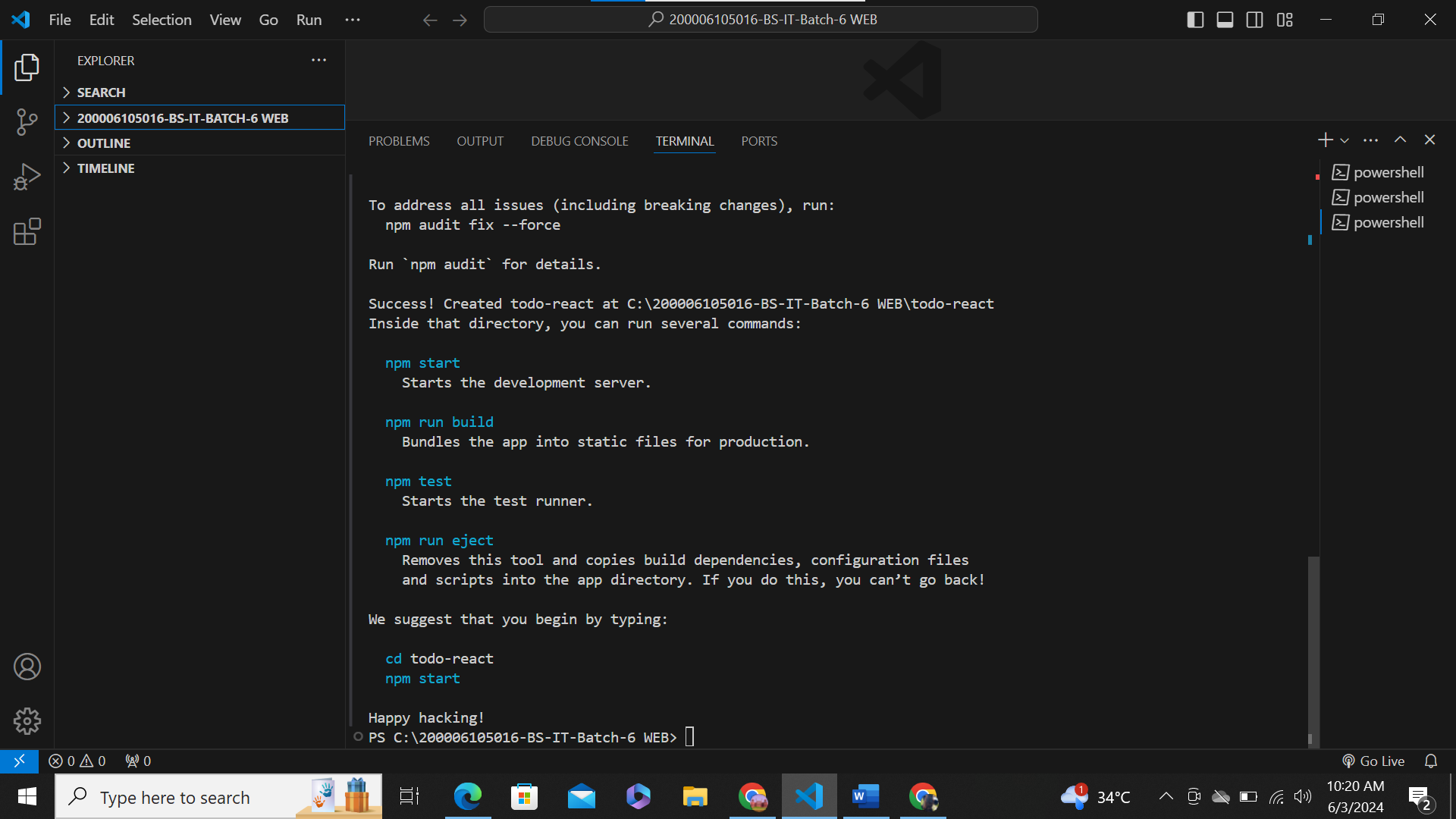
**Ans:**

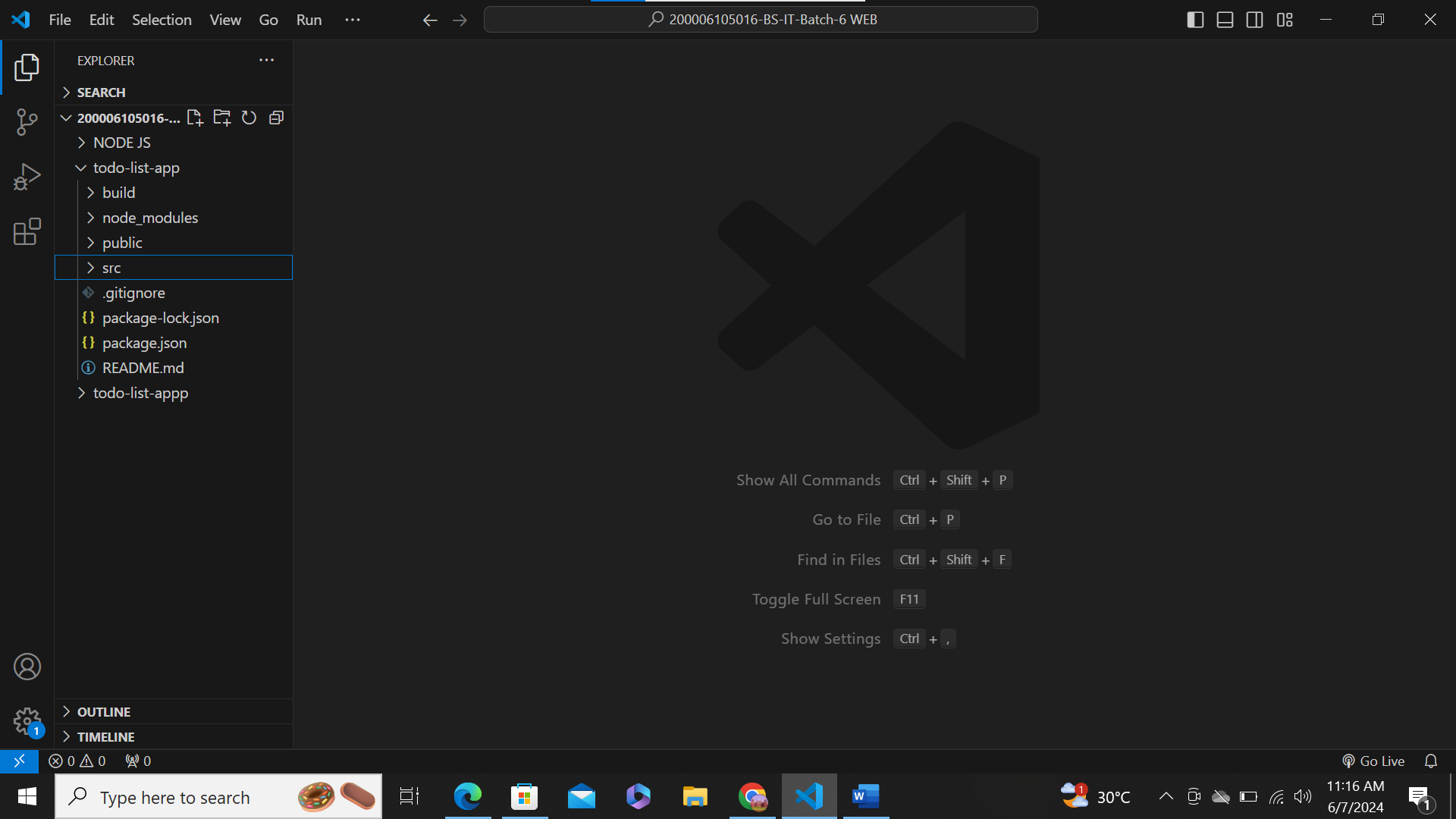
**Set Up Your Development Environment:**

* Ensure you have Node.js and npm installed.
* Create a new React application using Create React App

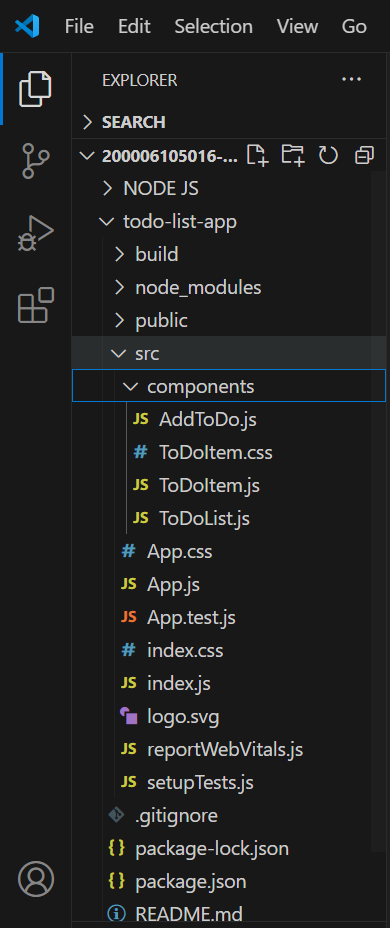
****

****

****

****

React Folder Name

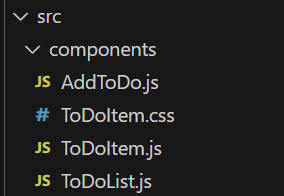
****

**Create the To-Do List Component:**

* In the ***src*** directory, create a new file called TodoList.js.

**Functionality:**

* **Add Task:** Adds a new task to the list.
* **Toggle Task Completion:** Toggles the completion status of a task.
* **Delete Task:** Deletes a task from the list**.**

****

* **AddtoDo.js File:**

// src/components/AddToDo.js

import React, { useState } from 'react';

function AddToDo({ addTask }) {

    const [value, setValue] = useState('');

    const handleSubmit = (e) => {

        e.preventDefault();

        if (!value.trim()) return;  // Prevent adding empty tasks

        addTask(value);

        setValue('');  // Clear input field after submission

    };

    return (

        <form onSubmit={handleSubmit} className="task-form">

            <input

                type="text"

                className="task-input"

                value={value}

                onChange={(e) => setValue(e.target.value)}

                placeholder="Enter a new task"

            />

            <button type="submit" className="task-button">Add Task</button>

        </form>

    );

}

export default AddToDo;

* **ToDoItem.css File:**

/\* ToDoItem.css \*/

.edit-button, .delete-button {

    margin-left: 10px;

    padding: 5px 10px;

    border: none;

    border-radius: 4px;

    cursor: pointer;

    outline: none;

  }

  .edit-button {

    background-color: #4CAF50; /\* Green \*/

    color: white;

  }

  .delete-button {

    background-color: #f44336; /\* Red \*/

    color: white;

  }

  .task-input {

    margin-right: 10px;

    padding: 5px;

    border: 2px solid #ddd;

    border-radius: 4px;

  }

* **ToDoItem.js File:**

import React, { useState } from 'react';

function ToDoItem({ item, toggleTask, deleteTask, editTask }) {

  const [isEditing, setIsEditing] = useState(false);

  const [editText, setEditText] = useState(item.task);

  const handleEditChange = (e) => {

    setEditText(e.target.value);

  };

  const handleToggleEdit = () => {

    if (isEditing) {

      editTask(item.id, editText); // Call the editTask function passed from parent

      setIsEditing(false); // Exit editing mode

    } else {

      setIsEditing(true); // Enter editing mode

      setEditText(item.task); // Initialize the text input with the current task

    }

  };

  return (

    <div className="task-item" style={{ marginBottom: "10px" }}>

      <input

        type="checkbox"

        checked={item.completed}

        onChange={() => toggleTask(item.id)}

        className="checkbox"

      />

      {isEditing ? (

        <input

          type="text"

          value={editText}

          onChange={handleEditChange}

          className="task-input"

          style={{ marginLeft: '10px' }}

        />

      ) : (

        <span style={{ textDecoration: item.completed ? 'line-through' : 'none', marginLeft: '10px' }}>

          {item.task}

        </span>

      )}

      {isEditing ? (

        <button onClick={handleToggleEdit} style={{ marginLeft: '10px' }}>

          Save

        </button>

      ) : (

        <>

          <button onClick={handleToggleEdit} style={{ marginLeft: '10px', color: 'Blue' , backgroundColor: 'light-pink'}}>

            Edit

          </button>

          <button onClick={() => deleteTask(item.id)} style={{ marginLeft: '10px' , color: 'red' }}>

            Delete

          </button>

        </>

      )}

    </div>

  );

}

export default ToDoItem;

* **ToDoList.js File:**

import React, { useState } from 'react';

import ToDoItem from './ToDoItem';

function ToDoList() {

  const [items, setItems] = useState([

    { id: 1, task: 'First Task', completed: false },

    { id: 2, task: 'Second Task', completed: false },

    { id: 3, task: 'Third Task', completed: false },

  ]);

  const toggleTask = (id) => {

    setItems(prevItems =>

      prevItems.map(item =>

        item.id === id ? { ...item, completed: !item.completed } : item

      )

    );

  };

  const deleteTask = (id) => {

    setItems(prevItems => prevItems.filter(item => item.id !== id));

  };

  return (

    <div>

      {items.map(item => (

        <ToDoItem key={item.id} item={item} toggleTask={toggleTask} deleteTask={deleteTask} />

      ))}

    </div>

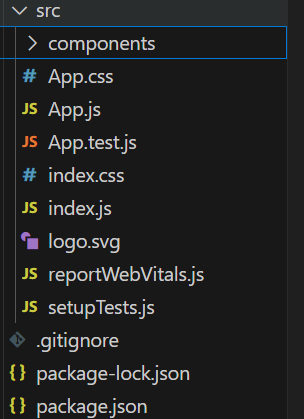
  );

}

export default ToDoList;

**src App.js**

**Main Component:** App.js serves as the main component of the React application, defining its overall structure and layout.

****

* **App.js File:**

// src/App.js

import React, { useState, useEffect } from 'react';

import ToDoList from './components/ToDoList';

import AddToDo from './components/AddToDo';

import './App.css';

function App() {

    const [toDos, setToDos] = useState([]);

    useEffect(() => {

        const storedToDos = JSON.parse(localStorage.getItem('todos'));

        if (storedToDos) {

            setToDos(storedToDos);

        }

    }, []);

    useEffect(() => {

        localStorage.setItem('todos', JSON.stringify(toDos));

    }, [toDos]);

    const addTask = (task) => {

        const newTask = { id: Date.now(), task, completed: false };

        setToDos([...toDos, newTask]);

    };

    const clearList = () => {

        setToDos([]);

    };

    return (

        <div className="app">

            <h1>To-Do List</h1>

            <AddToDo addTask={addTask} />

            <ToDoList items={toDos} />

            <button onClick={clearList} className="clear-list-button">Clear List</button>

        </div>

    );

}

export default App;

* **App.css File:**

/\* src/App.css \*/

body {

  font-family: 'Arial', sans-serif;

  background-color: #f7f7f7;

  margin: 0;

  padding: 20px;

  display: flex;

  justify-content: center;

  align-items: center;

  height: 100vh;

}

.app {

  width: 100%;

  max-width: 360px;

  background: #fff;

  padding: 20px;

  border-radius: 8px;

  box-shadow: 0 4px 8px rgba(0,0,0,0.1);

}

h1 {

  color: #333;

  text-align: center;

}

/\* src/App.css \*/

.clear-list-button {

  margin-top: 20px;

  padding: 10px 20px;

  background-color: #d6589f; /\* Tomato red, change as needed \*/

  color: white;

  border: none;

  border-radius: 4px;

  cursor: pointer;

  width: 100%;

}

.clear-list-button:hover {

  background-color: #e55345; /\* Darker shade for hover effect \*/

}

.task-input, .task-button {

  width: 100%;

  padding: 10px;

  margin: 5px 0;

  box-sizing: border-box;

}

.task-button {

  background-color: #2980b9;

  color: white;

  border: none;

}

.task-item {

  display: flex;

  align-items: center;

  justify-content: space-between; /\* This will push the delete button to the right \*/

}

.delete-button {

  color: rgb(234, 23, 62); /\* Text color \*/

  background-color: rgb(196, 94, 94); /\* Background color \*/

  border: none; /\* Optional: removes the border \*/

  padding: 5px 10px; /\* Padding around the text \*/

  cursor: pointer; /\* Changes the cursor to a pointer on hover \*/

}

.task-text {

  flex-grow: 1; /\* Allows the task text to take up any extra space, pushing the button right \*/

}

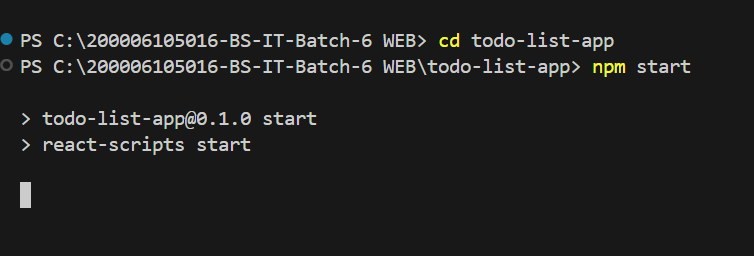
.task-form {

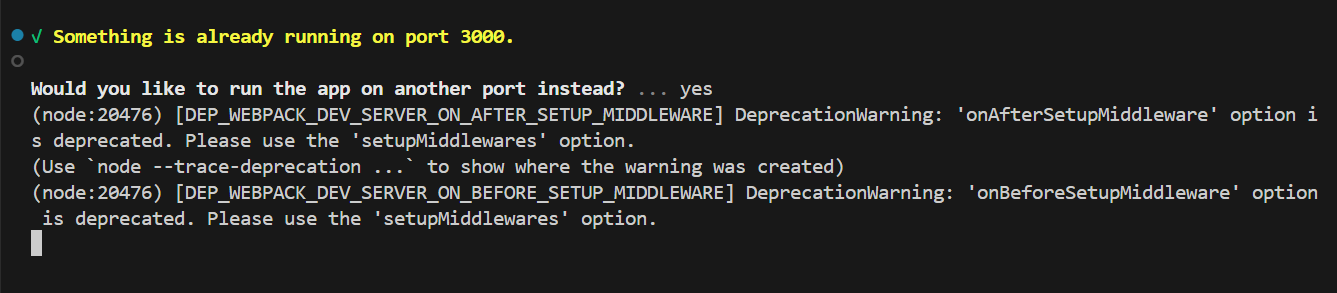
  margin-bottom: 20px;

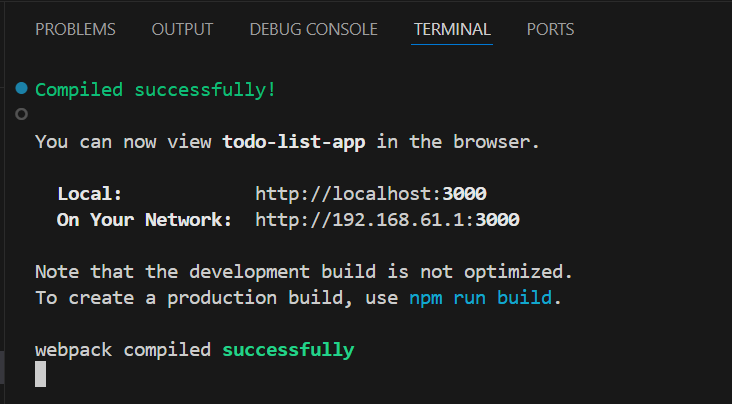
}

**Change Directory:** The primary purpose of cd is to navigate through the directory structure of the file system. You can move from one directory to another using this command.

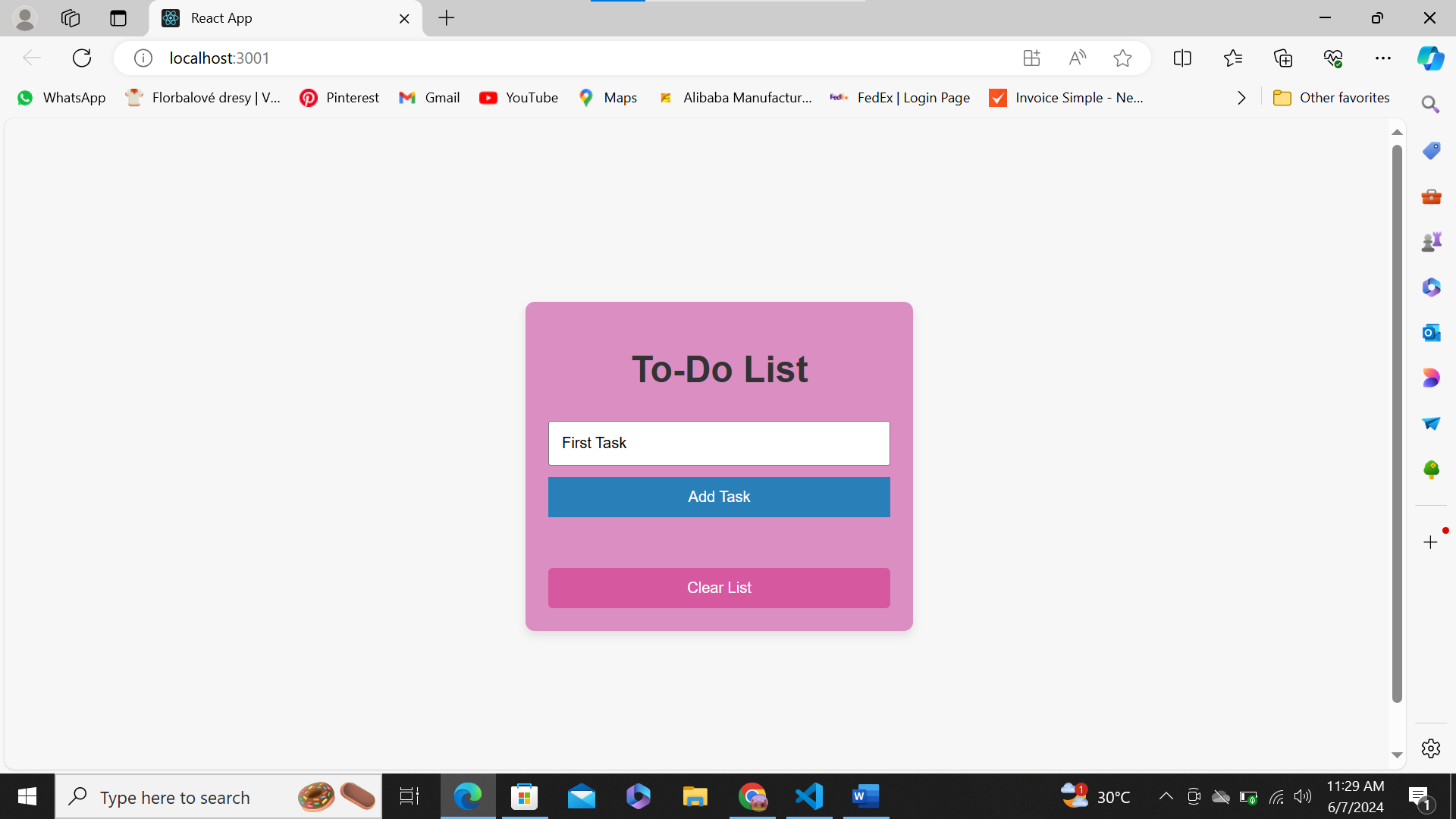
* **npm start** is a command used in Node.js/npm projects to start the application. When you run **npm** start in a Node.js project that has been set up to use this command, npm will look for a script called **"start"** in the "**scripts"** section of the **package.Json** file and execute it.

****

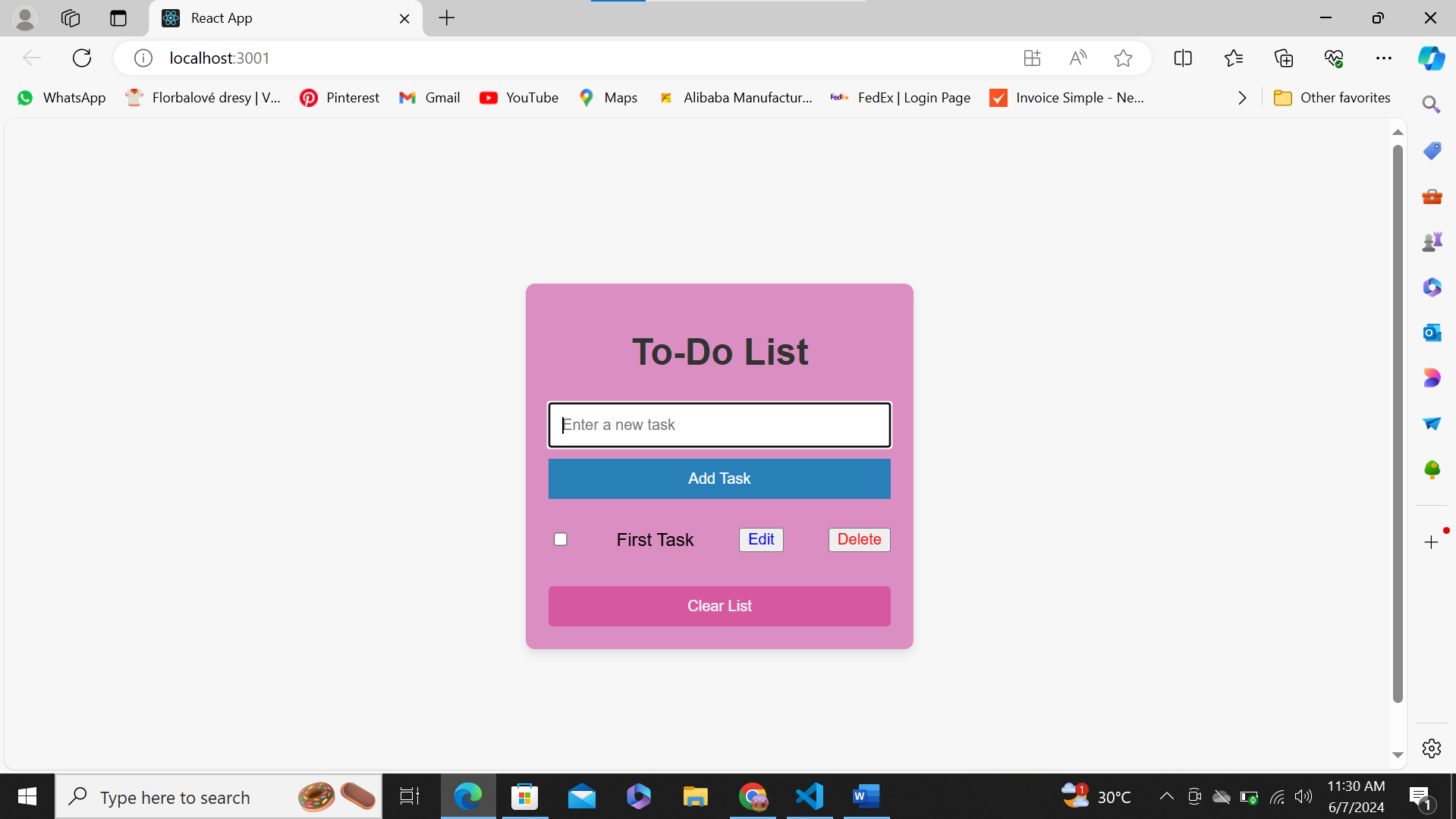
****

****

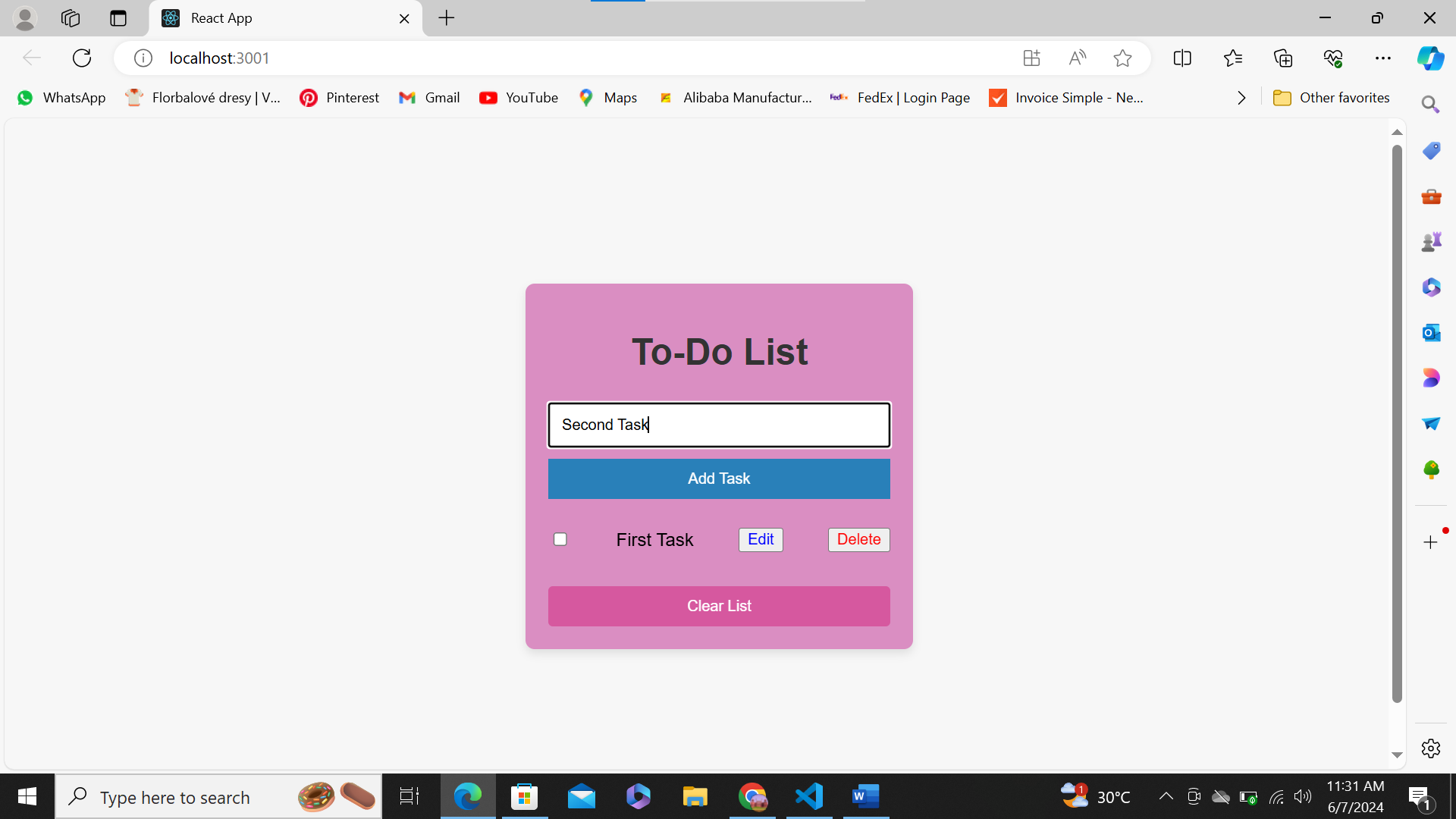
**Todo List Output:**

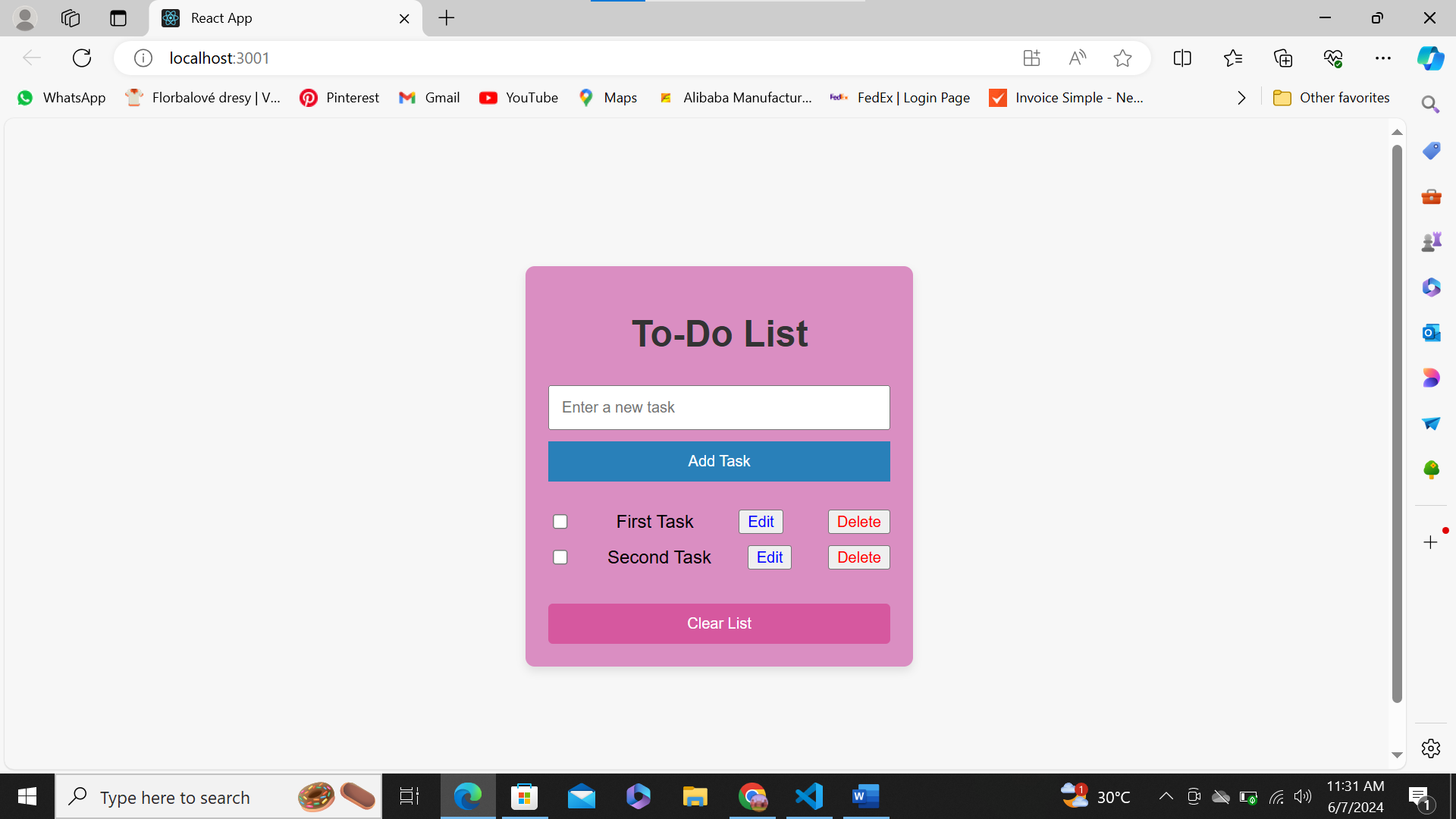
****

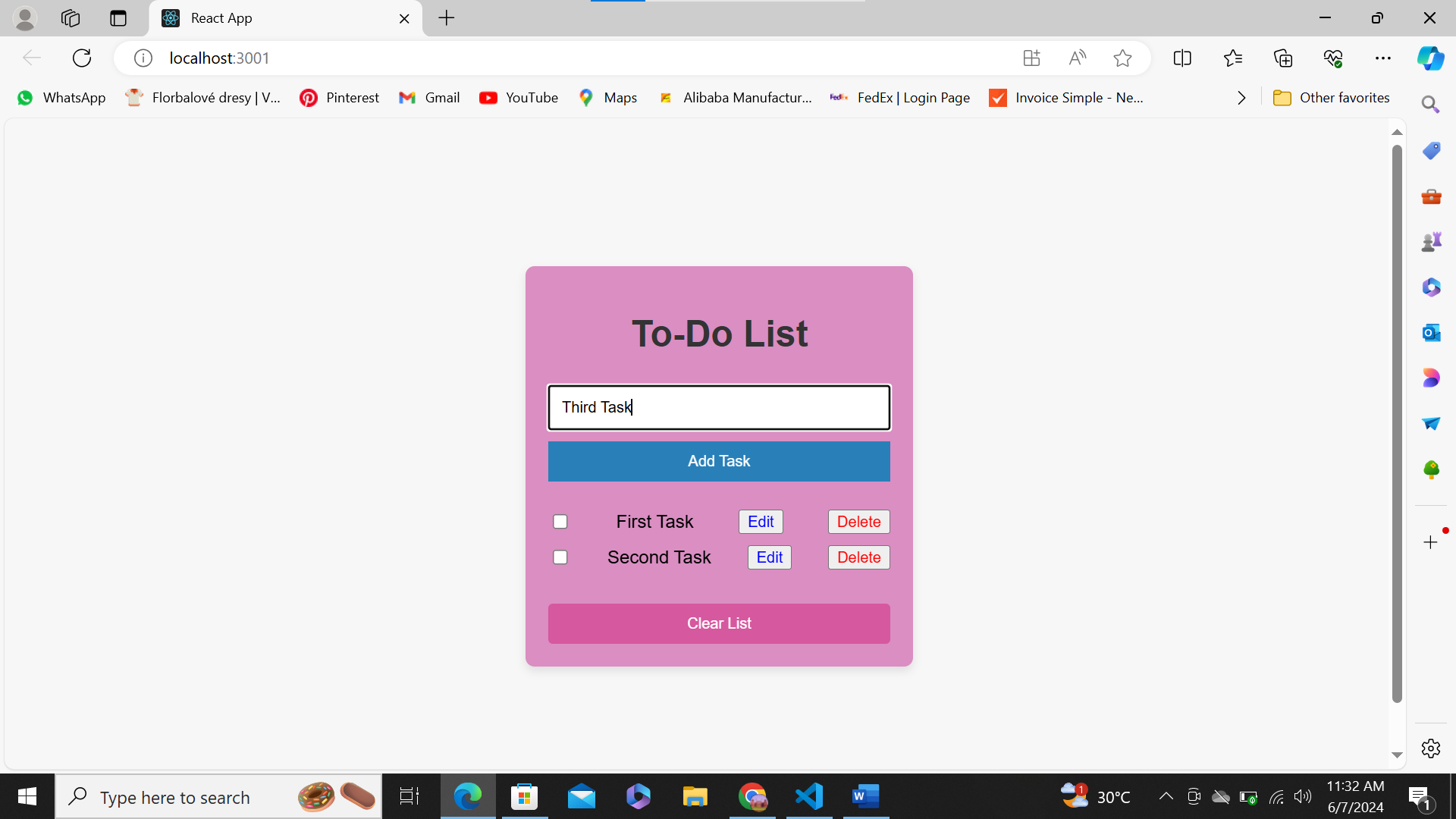
Add First Task

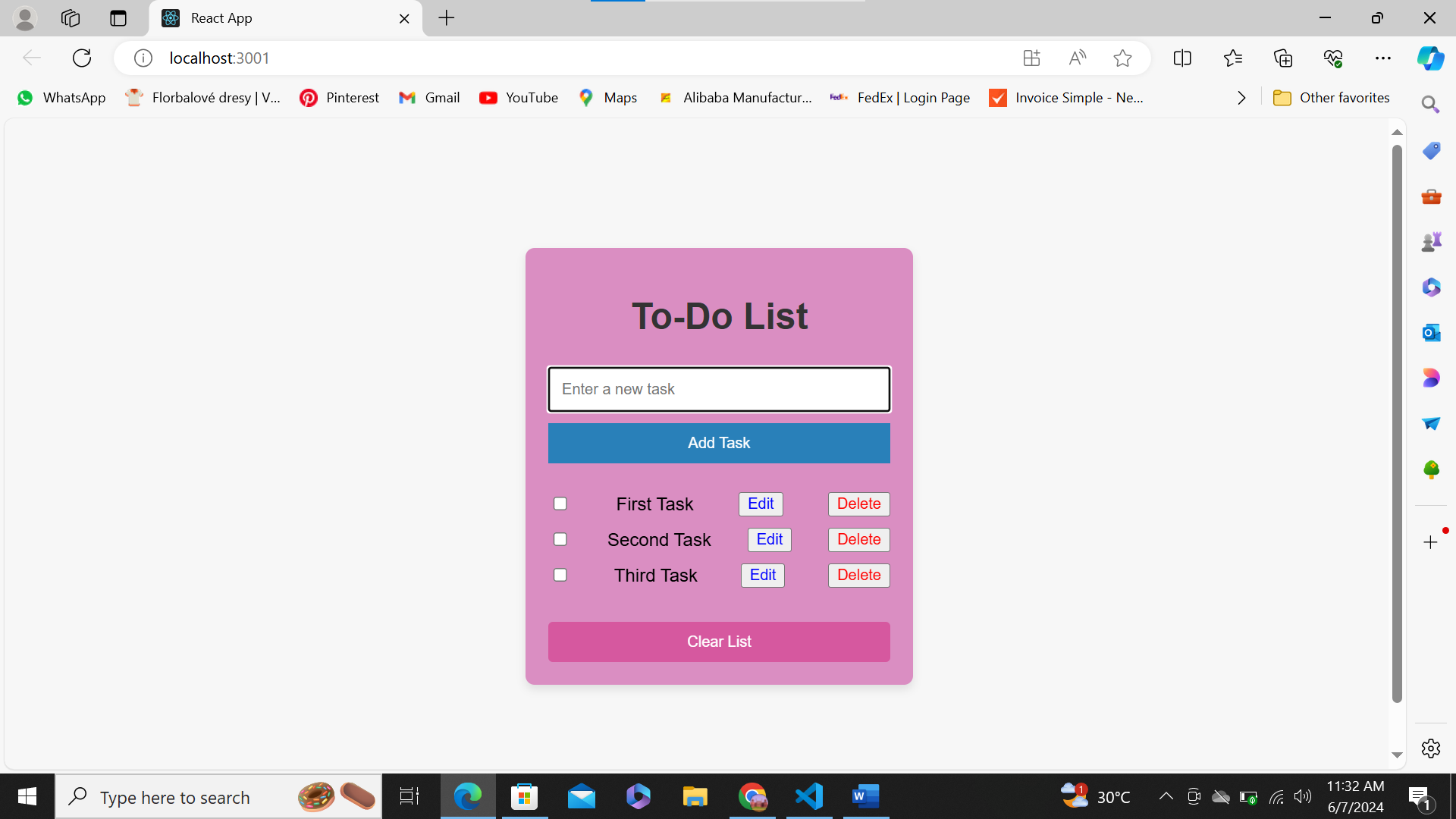
****

Task Added. Adding "Delete”, “Checkbox" and "Edit" buttons to a to-do list provides users with more control and flexibility over their tasks.

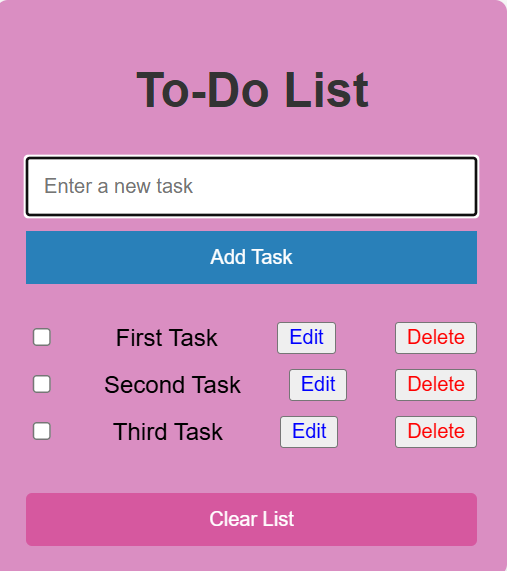
****

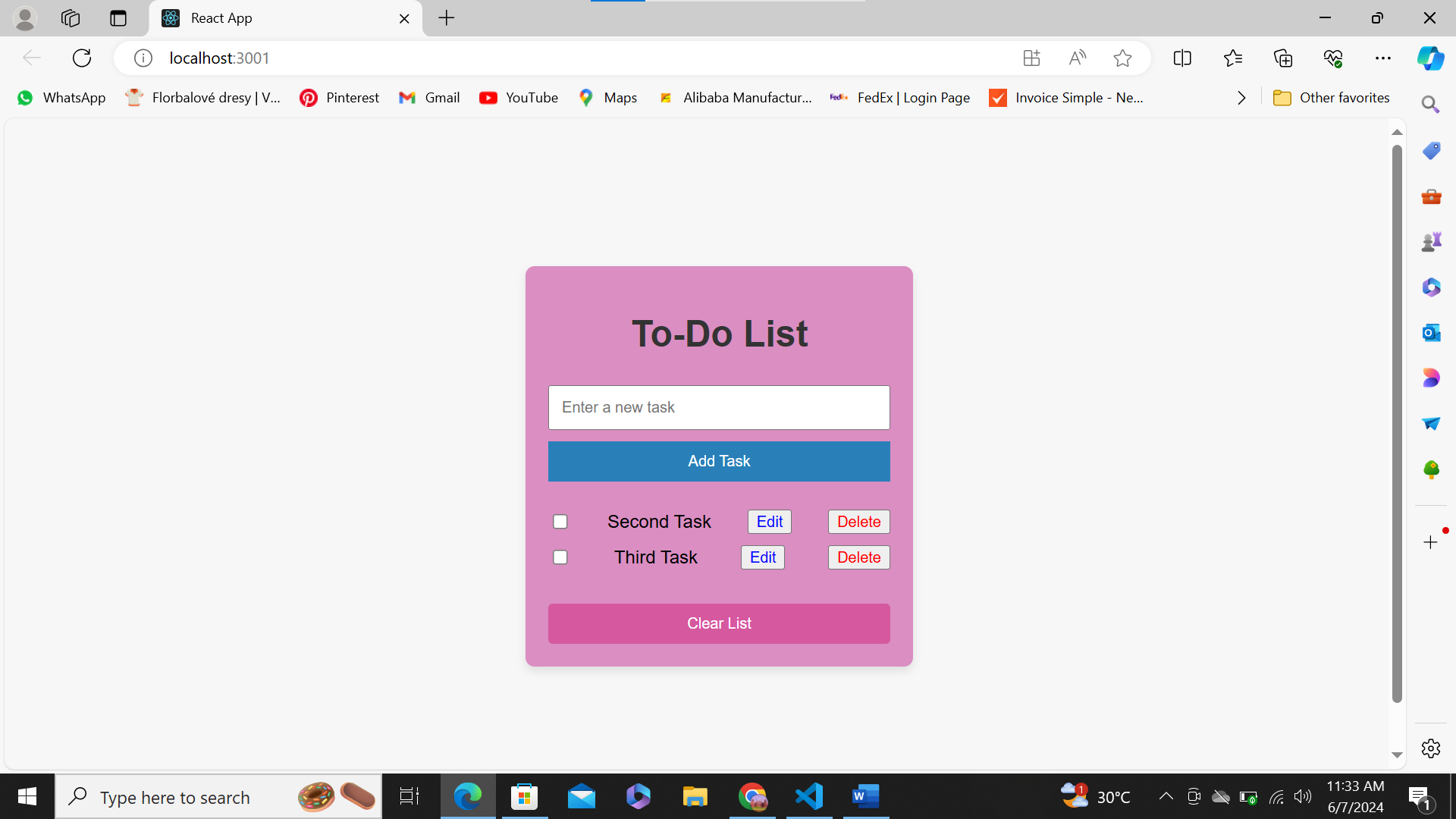
****

****

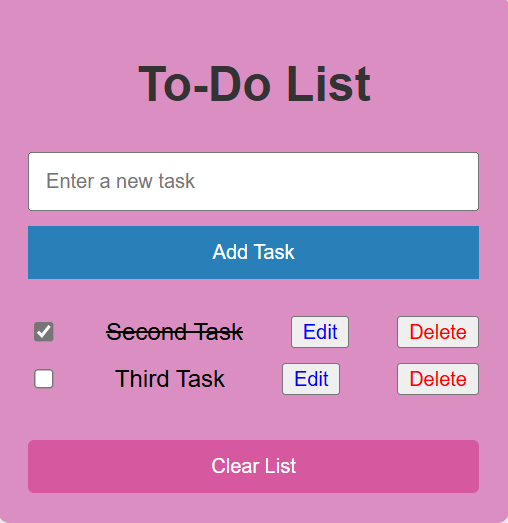
****

Add Three Task

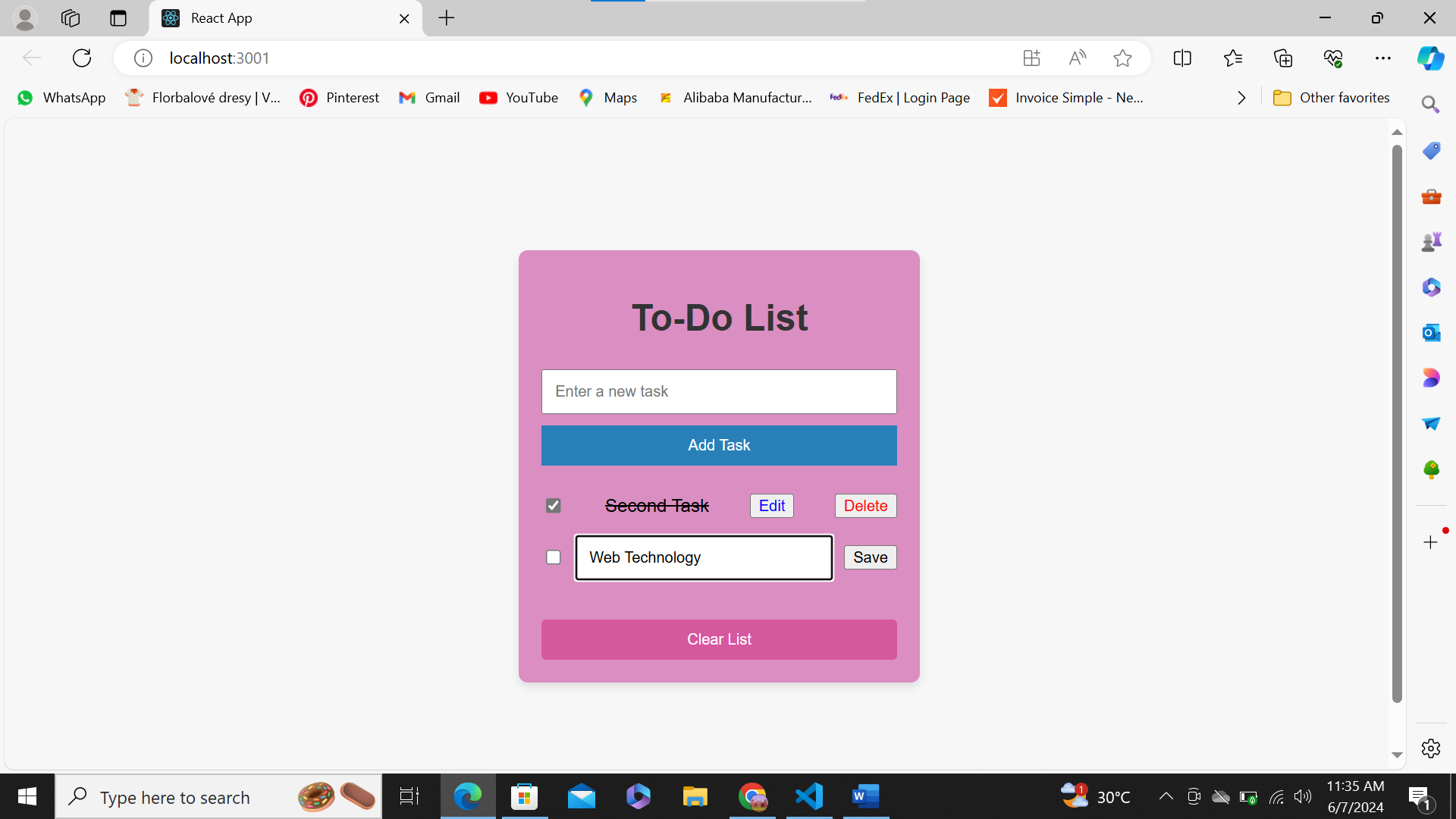
****

****

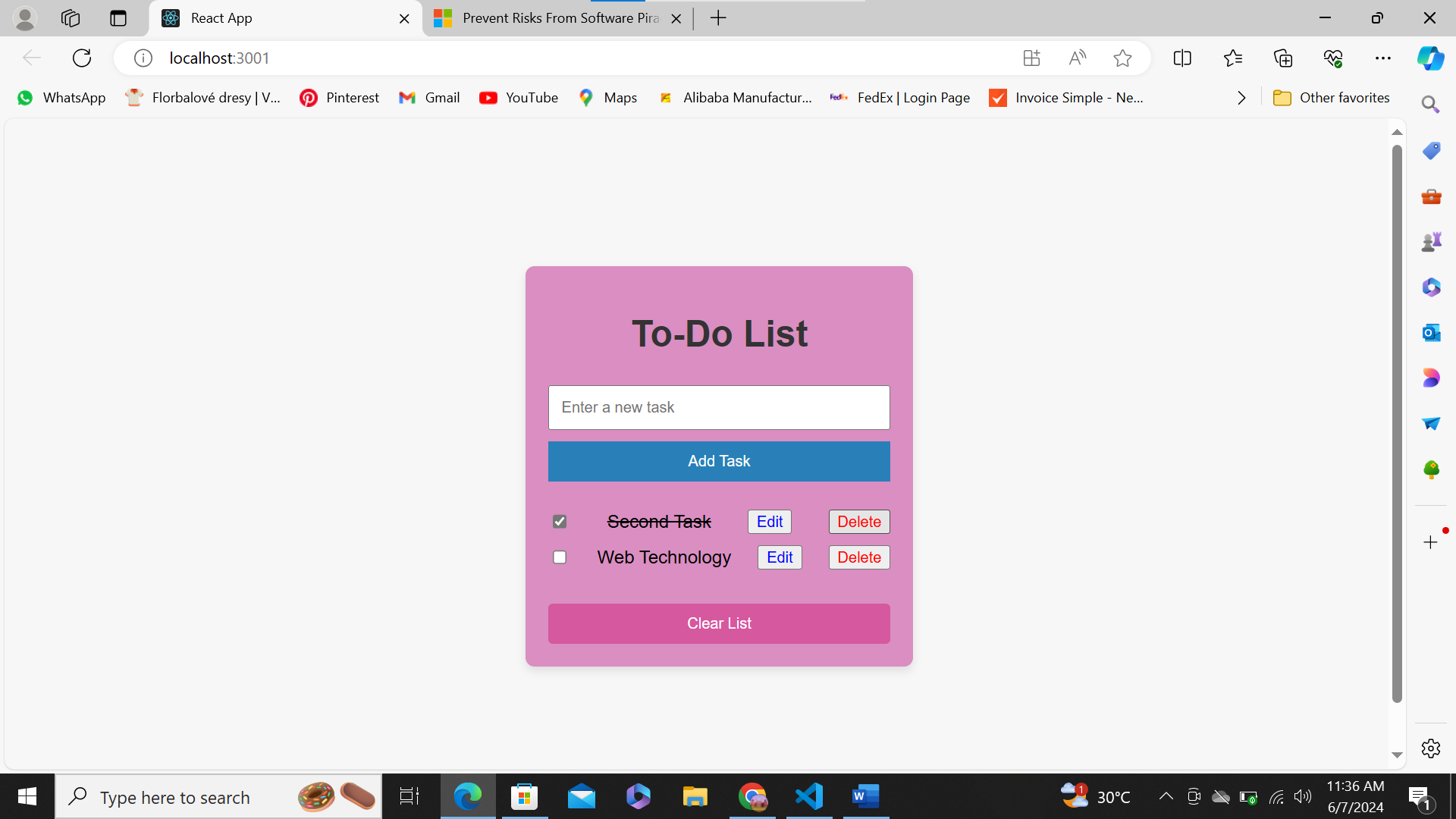
First Task deleted.

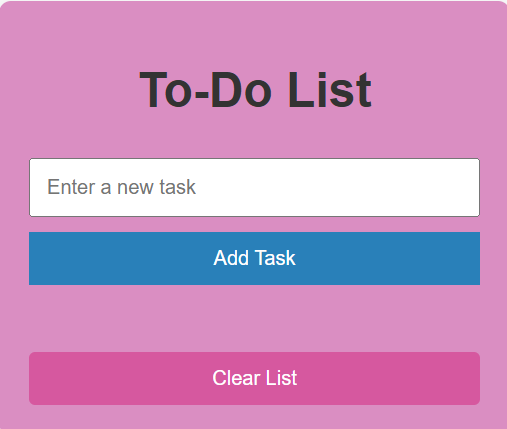
****

Checkboxes allow users to mark tasks as complete

****

Edit the Third Task and save.

****

****

Clear List of All Task