SUNSTONE

“TO-DO-LISTS”

Task - 3

TO-DO-LISTS

To-do lists offer a way to increase productivity, stopping you from forgetting things, helps prioritise tasks, manage tasks effectively, use time wisely and improve time management as well as workflow.To-Do List project is an application specially built to keep track of errands or tasks that need to be done. This application will be like a task keeper where the user would be able to enter the tasks that they need to do. Once they are done with their tasks they can also remove them from the list.

LMS Username	Name	Batch
au910020104001	AATHAVAJAYANTHR.S	CC2
au910020104303	DHARSHINI A	CC2
au910020104308	PRAVEEN M	CC2

Task-3

FRONT END:

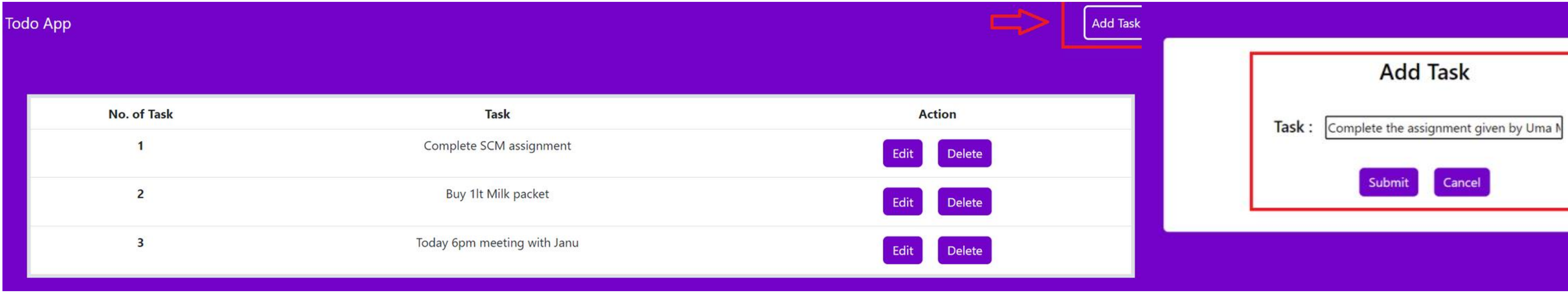
```java
package com.aathava.raect.controller;

import java.util.List;


@RestController
@RequestMapping("/api/works")
@CrossOrigin("http://localhost:3000")
public class WorkController
{
    @Autowired
    private WorkService ser;

    @PostMapping
    public Work addWork(@RequestBody Work work)
    {
        return ser.addWork(work);
    }

    @GetMapping
    public List<Work> getWork()
    {
        return ser.getWork();
    }

    @GetMapping("{id}")
    public Work getWorkById(@PathVariable("id") Long id)
    {
        return ser.getWorkById(id);
    }


    @PutMapping("{id}")
    public Work updateWork(@PathVariable("id") Long id ,@RequestBody Work work)
    {
        return ser.updateWork(id,work);
```

Writable

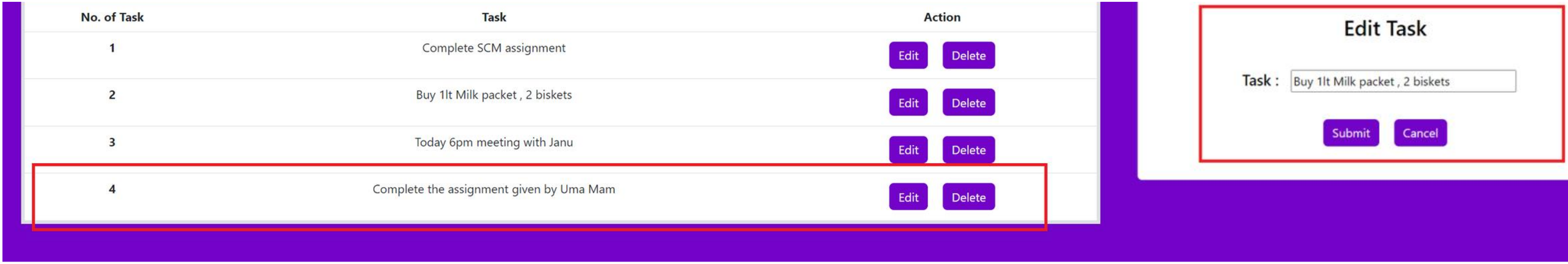
```jsx
import AddTask from './Controller/AddTask';
import EditTask from './Controller/EditTask';

function App() {
  return (
    <div className="App">



      <Router>
        <Navbar />

        <Routes>

          <Route exact path="/" element={<Home />}></Route>
          <Route exact path="/addtask" element={<AddTask />}></Route>
          <Route exact path="/edittask/:id" element={<EditTask />}></Route>

        </Routes>
      </Router>


    </div>
  );
}
```

Ln 1, Col 1 Spaces: 2 UTF-8 CRLF {} Java

ADD TASK



EDIT TASK



Your paragraph text

EVALUATION METRIC

- **Collaboration Features :**
  Team Interaction: Assess tools for collaboration and communication.
- **User Interface (UI) Design:**
  - Intuitiveness: How easy is it for users to navigate and use the interface?
  - Clarity: Is the layout and design clear, aiding efficient task management?
- **Platform Compatibility:**
  - Cross-device functionality: Evaluate if the to-do list works seamlessly across different devices (e.g., desktop, mobile)..
- **Integration Capabilities:**
  - Compatibility: Check if the to-do list integrates with other commonly used tools or platforms.
  - API Support: Assess the availability of APIs for customization or integration with other systems.
- **Consistency:**
  Ensure a consistent design and interaction pattern throughout the application.
  Responsiveness:* Check how quickly the interface responds to user interactions, ensuring a smooth and efficient user experience.

STEPWISE DESCRPITION

HTML Structure:

- Create an HTML file - Set up the basic structure with `<html>`<head> and <body> tags.
- Inside the `<body>, create a container for the to-do list, e.g., ` `.

CSS Styling:

- Link a CSS file (e.g., `styles.css`) to your HTML.
- Style the container, list items, input field, and buttons using CSS properties like `margin`, `padding`, `border`, etc.

To-Do List Form:

- Inside the container (`<div id="app">`), add an input field for the task and a button to add tasks.

JavaScript Interactivity:

- Use JavaScript to select the input field and button.
- Add an event listener to the button to execute a function when clicked.

Handling Tasks:

- In the JavaScript file, create an array to store tasks.
- When the button is clicked, retrieve the task from the input field, add it to the array, and update the display.

Displaying Tasks:

- Create a function to update the display with the tasks from the array.
- You can use a loop to iterate through the tasks and dynamically create HTML elements to represent each task.

Marking Tasks as Completed:

- Add a mechanism (e.g., a checkbox) to mark tasks as completed.
- Adjust the display function to show completed tasks differently.

Deleting Tasks:

- Implement a way to delete tasks
- Modify the display function to include the delete option.

Task Summary

In summary, Creating a to-do list front end involves designing an intuitive user interface with features like task input, task display, and task management (edit, delete). Use HTML for structure, CSS for styling, and JavaScript for dynamic interactions. Consider frameworks like React or Vue for efficient development. Prioritize responsiveness for a seamless experience across devices. Test thoroughly to ensure smooth functionality.

Submission
Github

[Github Link]

Thank you!