INTERNSHIP REPORT

Title: To Do List

**Internship Details:**

Name of the Intern: Pawan B

Internship Duration: 8 weeks (may- July)

Internship Position: Web development front end.

**Academic Institution Details:**

Name of the Institution: JSS Academy of Technical Education, Bangalore Department/Program: CSE(AIML).

Date of Submission: 10-Sep-2023

**Abstract**

In today's fast-paced world, managing daily tasks and responsibilities can become overwhelming. The "Efficient Task Management with a Todo List Application" abstract explores the development and implementation of a digital task management tool, commonly known as a "Todo List." This application aims to help individuals organize their tasks, enhance productivity, and reduce stress.

The Todo List application provides a user-friendly interface that allows users to create, edit, prioritize, and categorize tasks effortlessly. It incorporates features such as due dates, reminders, and task completion tracking. Additionally, users can access their tasks from various devices, ensuring seamless integration into their daily routines.

Ultimately, the "Efficient Task Management with a Todo List Application" abstract underscores the importance of task management tools in modern life and the potential for such applications to empower individuals to take control of their daily tasks and achieve their goals more efficiently.

The To-Do List web application that combines Bootstrap for styling and JavaScript for interactivity. This application offers a user-friendly interface, featuring a responsive navigation bar and a neatly organized task list displayed in a table format. Users can easily add, edit, and mark tasks as completed through the use of modal dialogs for data entry. Furthermore, Local Storage is leveraged to maintain task data persistently. The code provides a visually appealing design with a background image, enhancing the overall user experience. While functional, there's room for improvement, particularly in refining the navigation bar's layout and incorporating error handling and input validation for data accuracy. Advanced features, such as task categorization or reminders, could also be integrated to extend the application's capabilities. In essence, this HTML code offers a solid starting point for a basic To-Do List application, with the potential for further enhancement and customization to meet specific requirements.

The HTML Todo List Application presented here demonstrates the power of combining HTML, Bootstrap, and JavaScript to create a functional task management system. This application provides users with a seamless and intuitive interface to add, edit, and track tasks efficiently.

Key Features:

1. User-Friendly Interface: The application starts with a responsive navigation bar that includes a brand logo and collapsible menu for easy navigation.

2. Task Management: Users can easily add new tasks by clicking the "Add Task" button, which triggers a modal form for task input. Tasks can be described with text, assigned to responsible individuals, and given an estimated time of completion (ETA) through a date-time input.

3. Data Persistence: Task data is stored locally using JavaScript's local Storage feature, ensuring that tasks are retained even if the page is refreshed or closed.

4. Task List Display: Tasks are displayed in a clean and organized table format, making it easy for users to view task details, including their descriptions, responsible parties, and ETAs.

5. Interactive Actions: Users can mark tasks as done, which removes them from the list. Additionally, tasks can be edited, and changes are reflected instantly in the task list.

6. Styling and Icons: Bootstrap and Font Awesome libraries are utilized for styling, providing an aesthetically pleasing and consistent user experience. Icons are used for buttons, enhancing the application's visual appeal.

7. Background Image: The application features a customizable background image, adding a personalized touch to the user interface.

8. Responsive Design: The application adapts to different screen sizes, ensuring a consistent experience across devices.

This HTML Todo List Application serves as a practical example of how modern web technologies can be leveraged to create a user-friendly and efficient task management tool. It can be further extended and customized to meet specific needs, making it a versatile solution for individuals and teams seeking an organized approach to task management.

**OBJECTIVE**

The objectives of a Todo List, whether it's a physical list or a digital application, are to help individuals or teams:

1. Organize Tasks: The primary objective is to provide a structured and organized way to list tasks, responsibilities, and goals. This organization ensures that nothing important is overlooked or forgotten.

2. Prioritize Work: A Todo List allows users to prioritize tasks based on their importance and deadlines. This helps individuals focus on high-priority tasks first, ensuring they are completed on time.

3. Increase Productivity: By providing a clear outline of what needs to be done, a Todo List can boost productivity. It prevents procrastination and reduces the time spent deciding what to work on next.

4. Time Management: Users can allocate specific time slots to tasks, helping with time management. This ensures that there is enough time for essential tasks and prevents overloading schedules.

5. Reduce Stress: Knowing that tasks are organized and manageable can reduce stress and anxiety. Users can see progress and gain a sense of accomplishment as they check off completed tasks.

6. Set Goals and Objectives: Todo Lists can be used to set both short-term and long-term goals. Breaking down larger goals into smaller, manageable tasks makes them more achievable.

7. Improve Accountability: In team settings, Todo Lists help assign responsibilities and hold team members accountable for their tasks. This improves collaboration and project management.

8. Enhance Communication: Shared Todo Lists or task management applications facilitate communication within teams. Team members can see each other's tasks, deadlines, and progress, promoting collaboration.

9. Track Progress: Users can track their progress and see how many tasks they've completed over time. This feedback can motivate them to stay on track.

10. Flexibility: Todo Lists can adapt to various needs, whether for personal use, work-related tasks, project management, or even as a shopping list. They are flexible tools that cater to different requirements.

11. Remember Important Details: Todo Lists help users remember important details associated with tasks, such as due dates, notes, and task descriptions.

12. Time Efficiency: By minimizing the time spent on figuring out what to do next, individuals can use their time more efficiently, increasing overall productivity.

13. Customization: Users can customize their Todo Lists to fit their specific preferences, such as categorizing tasks, color-coding, or using different task management methodologies like GTD (Getting Things Done) or Kanban.

The primary objective of the provided HTML code is to create a functional and user-friendly To-Do List web application. This application is designed to address the fundamental need for organizing tasks, making them more manageable, and enhancing productivity. The code leverages several key components and techniques to achieve this objective.

Firstly, the code aims to deliver a visually pleasing and responsive user interface. It employs the Bootstrap framework for consistent and attractive styling. The responsive design ensures that the application adapts seamlessly to various screen sizes and devices, making it accessible and usable for a broad audience. This objective of visual appeal and responsiveness is not just about aesthetics but also about providing a comfortable and engaging user experience.

Secondly, the code focuses on ease of task management. Users can quickly add new tasks by clicking the "Add Task" button, which triggers a modal dialog. This intuitive design ensures that users can effortlessly enter task details, including descriptions, responsible persons, and estimated completion times (ETA). The objective here is to streamline the task creation process, reducing friction and encouraging users to utilize the application for efficient task tracking.

Thirdly, the code incorporates functionality for task persistence. By utilizing Local Storage, it ensures that task data remains available even after the browser is closed or refreshed. This objective aligns with the core purpose of a To-Do List application, which is to serve as a reliable repository for tasks over time. Users can confidently rely on this application to store and retrieve their task lists without data loss, contributing to a more dependable task management experience.

Furthermore, the code seeks to enhance user interaction through modal dialogs. These dialogs not only facilitate task input but also enable editing and updating of existing tasks. By allowing users to edit task details and ETA, the code empowers them to maintain an accurate and up-to-date task list. This objective of interactivity promotes user engagement and encourages the continued use of the application as a dynamic tool for task management.

Another aspect of the code's objective is to provide clear and organized task presentation. Tasks are displayed in a table format with relevant columns for description, responsible persons, ETA, and actions. This structured presentation ensures that users can easily scan and comprehend their task list. Each task's details are clearly outlined, contributing to efficient task tracking and prioritization.

Lastly, the code aims to offer potential for future enhancement and customization. While it provides essential task management features, there is room for improvement and expansion to meet specific requirements. This objective acknowledges the evolving nature of task management needs and positions the code as a starting point for further development. Features such as task categorization, due date reminders, or user accounts could be incorporated to enhance functionality based on user preferences and requirements.

**Introduction**

The definition is a simple one. **It’s a list of tasks you need to complete or things that you want to do.**

Most typically, they’re organised in order of priority. Traditionally, they’re written on a piece of paper or post it notes and act as a memory aid. As technology has evolved we have been able to create a todo lists with excel spreadsheets, word documents, [email](https://en.wikipedia.org/wiki/Email) lists, todo list apps, Microsoft to do and google to do list to name a few. You can[use a to do list](https://checkify.com/blog/how-to-use-a-to-do-list/) in your home and personal life, or in the workplace.

Having a list of everything you need to do written down in one place means you shouldn’t forget anything important. By prioritising the tasks in the list you plan the order in which you’re going to do them and can quickly see what needs your immediate attention and what tasks you can leave until a little later.

A "To-Do List," often referred to simply as a "Todo List," is a widely used time management and productivity tool. It is a list of tasks or activities that an individual or a team needs to complete within a specified period. Todo lists are a simple yet effective way to organize, prioritize, and track tasks, ensuring that important activities are not forgotten or overlooked.

Key components and features of a Todo List include:

1. Task Description: Each item on the list should have a clear and concise description of the task or activity to be performed. This helps users understand what needs to be done.

2. Priority: Tasks can be categorized or marked by priority to indicate their importance or urgency. This allows users to focus on high-priority tasks first.

3. Due Dates: Assigning due dates or deadlines to tasks helps individuals and teams manage their time effectively and ensures that tasks are completed on time.

4. Checklists: For more complex tasks, checklists can be created within the Todo List item. Users can mark off sub-tasks as they are completed, providing a sense of accomplishment and progress.

5. Notes and Details: Users can add additional information, notes, or details related to a task to provide context or instructions for themselves or team members.

6. Task Status: Tasks can be marked as "completed," "in progress," or "not started" to provide a quick overview of the current status of each task.

7. Crossing Off or Deleting: Once a task is completed, users can either cross it off the list or delete it to declutter their Todo List.

Benefits of Using Todo Lists:

Organization: Todo lists help individuals and teams stay organized by providing a central place to list and track tasks.

Productivity: They improve productivity by allowing users to prioritize tasks and focus on what needs to be done next.

Time Management: Due dates and priorities assist in managing time effectively, ensuring that important tasks are addressed promptly.

Reduced Stress: Keeping track of tasks reduces stress and anxiety associated with forgetting important commitments.

Motivation: Crossing off completed tasks provides a sense of achievement and motivation to tackle more tasks.

Collaboration: Todo lists can be used collaboratively, allowing teams to work together on projects and track progress.

Todo lists are versatile tools used in various aspects of life, from personal chores and shopping lists to work-related projects and team collaboration. They come in many forms, including physical paper lists, digital apps, and integrated task management tools. The choice of format depends on individual preferences and the specific needs of the task at hand.

In summary, Todo Lists are invaluable tools for organizing tasks, managing time, and enhancing productivity, making them an essential part of both personal and professional life.

**Methodology**

HTML

HTML stands for Hyper Text Mark-up Language. It is used to design web pages using mark-up language. HTML is the combination of Hypertext and Mark-up language. Hypertext defines the link between the web pages. Mark-up language is used to define the text document within tag which defines the structure of web pages. HTML5 is the fifth and current version of HTML. It has improved the mark-up available for documents and has introduced application programming interfaces (API) and Document Object Model (DOM).

1. HTML Structure:

The HTML structure is defined using standard HTML tags. It includes elements for the header, navigation bar, container for the Todo List, and modal dialogs for adding and editing tasks.

2. CSS Styling:

CSS is used to style the web page. It includes Bootstrap classes for responsive design, icons from Font Awesome and Bootstrap Icons, and custom styles for the background image.

3. Navigation Bar:

The navigation bar is created using Bootstrap's `navbar` component. It contains a logo, a collapsible menu button, and menu items for "Home" and "Contact Us."

4. Task List Table:

The main content of the page is a table that displays tasks. It uses Bootstrap's `table` class for styling. The table includes columns for serial number (SL.no), task/description, responsible person, ETA (Estimated Time of Arrival), and actions.

5. Add Task Modal:

Clicking the "Add Task" button opens a modal dialog. This modal contains input fields for task description, responsible person, and ETA. Users can input task details and click the "Add Task" button to add a new task.

6. Update Task Modal:

Clicking the edit icon in the task list opens an "Edit Task" modal. This modal allows users to edit an existing task's description, responsible person, and ETA. It also includes a hidden input field to track the index of the task being edited.

7. JavaScript Functionality:

JavaScript is used to add interactivity to the web page. Key functions include:

- `showAddTaskModal()`: Shows the "Add Task" modal when the button is clicked.

- `addTask()`: Handles the addition of new tasks to local storage and updates the task list.

- `createHtmlfromStorage()`: Renders tasks from local storage to the task list table.

- `markAsDone(index)`: Allows users to mark tasks as done and removes them from the list.

- `editTask(index)`: Loads task details into the "Edit Task" modal for editing.

- `updateTask()`: Updates an existing task with the edited details and refreshes the task list.

8. External Libraries:

The web page includes external libraries like Bootstrap, jQuery, and Popper.js, which enhance the user interface and functionality.

9. Local Storage:

The application stores task data in the browser's local storage, allowing tasks to persist across sessions.

10. Background Image:

A custom background image is added to the body of the page using CSS.

Overall, this methodology outlines how the HTML, CSS, and JavaScript work together to create a functional Todo List web application with features for adding, editing, and marking tasks as done. It also incorporates responsive design elements for a user-friendly experience on various devices.

**CODE - Include the screenshots of RESULTS AND OUTPUT**

Todo list

<!DOCTYPE html>

<html>

    <head>

        <meta charset="utf-8">

        <meta name="viewport" content="width=device-width, initial-scale=1">

        <title> HTML Todo List </title>

        <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-4bw+/aepP/YC94hEpVNVgiZdgIC5+VKNBQNGCHeKRQN+PtmoHDEXuppvnDJzQIu9" crossorigin="anonymous">

        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.2/css/all.min.css" integrity="sha512-z3gLpd7yknf1YoNbCzqRKc4qyor8gaKU1qmn+CShxbuBusANI9QpRohGBreCFkKxLhei6S9CQXFEbbKuqLg0DA==" crossorigin="anonymous" referrerpolicy="no-referrer" />

        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.10.5/font/bootstrap-icons.css">

        <style>

            body {

                background-image: url("img.jpeg");

                background-size: cover;

                background-repeat: no-repeat;

                background-attachment: fixed;

            }

        </style>

    </head>

    <body>

        <nav class="navbar navbar-expand-lg navbar-light bg-light" id="navbar">

            <div class="container-fluid">

                <a href="#" class="navbar-brand">

                    <img src="jss logo.png" class="img-fluid" alt="logo"width="160" >

                </a>

                <button type="button" class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbar">

                    <i class="bi bi-list"></i>

                </button>

                <div class="collapse navbar-collapse" id="navbar">

                    <div class="navbar-nav ms-auto">

                         <i class="bi bi-house-heart">Home</i>

                        <br>

                        <hr>

                         <i class="bi bi-person-lines-fill">Contact Us</i>

                    </div>

                </div>

            </div>

        </nav>

        <div class="container p-5">

            <div class="mb-3">

                <button type="submit" class="btn btn-outline-primary" onclick="showAddTaskModal()">Add Task</button>

            </div>

            <div class="d-flex justify-content-center">

               <div class="col-sm-12 col-md-12 col-lg-12">

                    <div class="card">

                        <div class="card-body">

                            <table class="table">

                                    <thead class="text-center">

                                        <th>SL.no</th>

                                        <th>Task/Description</th>

                                        <th>Responsible</th>

                                        <th>ETA</th>

                                        <th>Action</th>

                                    </thead>

                                    <tbody class="text-center" id = "taskTableBody">

                                    </tbody>

                             </table>

                        </div>

                     </div>

                </div>

            </div>

        </div>

        <div class="modal fade" id="addTaskModal" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="addTaskModalLabel" aria-hidden="true">

            <form id="taskInputForm">

                <div class="modal-dialog">

                    <div class="modal-content">

                        <div class="modal-header">

                            <h5 class="modal-title" id="addTaskmodalLabel">Add task</h5>

                            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="close"></button>

                        </div>

                        <div class="modal-body">

                            <div class="mb-1">

                                <label for ="addTaskTextArea" class="form-label"> Task/Description </label>

                                <textarea class="form-control" id="addTaskTextArea" name="taskDescription" rows="3" placeholder="Add your Task/Description"></textarea>

                            </div>

                            <div class="mb-1">

                                <label for ="addTaskresponsibleperson" class="form-label"> Responsible </label>

                                <input type="text"  class="form-control" id="addresponsibleperson" name="taskresponsibleperson" placeholder="Add the responsible person's name">

                            </div>

                            <div class="mb-1">

                                <label for="addTaskresponsibility" class="form-label"> ETA </label>

                                <input type="datetime-local"  class="form-control" id="addETA" name="taskETA" placeholder="click to Add time ">

                            </div>

                        </div>

                        <div class="modal-footer">

                            <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">cancel</button>

                            <button type="button" class="btn btn-primary" onclick="addTask()" >Add Task</button>

                        </div>

                    </div>

                </div>

            </form>

        </div>

        <div class="modal fade" id="updateTaskModal" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="updateTaskModalLabel" aria-hidden="true">

            <form id="taskupdateForm">

                <div class="modal-dialog">

                    <div class="modal-content">

                        <div class="modal-header">

                            <h5 class="modal-title" id="editTaskmodalLabel">Edit task</h5>

                            <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="close"></button>

                        </div>

                        <div class="modal-body">

                        <div class="mb-1">

                            <label for="editTaskTextArea" class="form-label"> Task/Description </label>

                            <textarea class="form-control" id="editTaskTextArea" name="taskDescription" rows="3" placeholder="Add your Task/Description"></textarea>

                        </div>

                        <div class="mb-1">

                            <label for="editresponsibleperson" class="form-label"> Responsible </label>

                            <input type="text" class="form-control" id="editresponsibleperson" name="taskresponsibleperson" placeholder="Add the responsible person's name">

                        </div>

                        <div class="mb-1">

                            <label for="editETA" class="form-label"> ETA </label>

                            <input type="datetime-local" class="form-control" id="editETA" name="taskETA" placeholder="click to Add time">

                        </div>

                        <input type="hidden" id="editIndex" name="taskIndex">

                        </div>

                        <div class="modal-footer">

                            <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">cancel</button>

                            <button type="button" class="btn btn-primary" onclick="updateTask()" >Add Task</button>

                        </div>

                    </div>

                </div>

            </form>

        </div>

        <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.0/jquery.min.js"></script>

        <script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.js" integrity="sha384-I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8ODewa9r" crossorigin="anonymous"></script>

        <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.1/dist/js/bootstrap.min.js" crossorigin="anonymous"></script>

        <script>

            createHtmlfromStorage();

            function showAddTaskModal() {

                $("#addTaskModal").modal('show');

            }

            function addTask() {

                console.log('Add Task clicked');

                $("#addTaskModal").modal('hide');

                var dataArr = $("#taskInputForm").serializeArray();

                var taskObject = {};

                for (var i in dataArr) {

                    var name = dataArr[i]['name'];

                    var value = dataArr[i]['value'];

                    taskObject[name] = value;

                }

                var StorageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];

                StorageObjectArr.push(taskObject);

                localStorage.setItem('taskStorage', JSON.stringify(StorageObjectArr));

                createHtmlfromStorage();

            }

            function createHtmlfromStorage() {

                var StorageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];

                var html = '';

                if (StorageObjectArr.length > 0) {

                    for (var i = 0; i < StorageObjectArr.length; i++) {

                        var date = new Date(StorageObjectArr[i]['taskETA']);

                        html += '<tr>' +

                            '<td>' + (i + 1) + '</td>' +

                            '<td>' + StorageObjectArr[i]['taskDescription'] + '</td>' +

                            '<td>' + StorageObjectArr[i]['taskresponsibleperson'] + '</td>' +

                            '<td>' + date.toDateString() + '</td>' +

                            '<td><i class="bi bi-check-circle-fill" onclick="markAsDone(' + i + ')"></i><i class="bi bi-pencil-square" onclick="editTask('+ i +')"></i></td></tr>';

                    }

                } else {

                    html = '<tr><td colspan="5" >No Tasks Added yet</td></tr>';

                }

                $("#taskTableBody").html(html);

            }

            function markAsDone(index) {

                var StorageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];

                if (StorageObjectArr.length > index) {

                    StorageObjectArr.splice(index, 1);

                    localStorage.setItem('taskStorage', JSON.stringify(StorageObjectArr));

                    createHtmlfromStorage();

                }

            }

            function editTask(index)

             {

                var StorageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];

                if (StorageObjectArr.length > index) {

                    $("#editresponsibleperson").val(StorageObjectArr[index]['taskresponsibleperson']);

                    $("#editTaskTextArea").val(StorageObjectArr[index]['taskDescription']);

                    $("#editETA").val(StorageObjectArr[index]['taskETA']);

                    $("#editIndex").val(index);

                    $('#updateTaskModal').modal('show');

                }

            }

            function updateTask() {

                $('#updateTaskModal').modal('hide');

                var dataArr = $("#taskupdateForm").serializeArray();

                var taskObject = {};

                for (var i in dataArr) {

                    var name = dataArr[i]['name'];

                    var value = dataArr[i]['value'];

                    taskObject[name] = value;

                }

                var StorageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];

                var index = parseInt(taskObject['taskIndex']);

                if (StorageObjectArr.length > index) {

                    StorageObjectArr[index]['taskresponsibleperson'] = taskObject['taskresponsibleperson'];

                    StorageObjectArr[index]['taskDescription'] = taskObject['taskDescription'];

                    StorageObjectArr[index]['taskETA'] = taskObject['taskETA'];

                    localStorage.setItem('taskStorage', JSON.stringify(StorageObjectArr));

                    createHtmlfromStorage();

                }

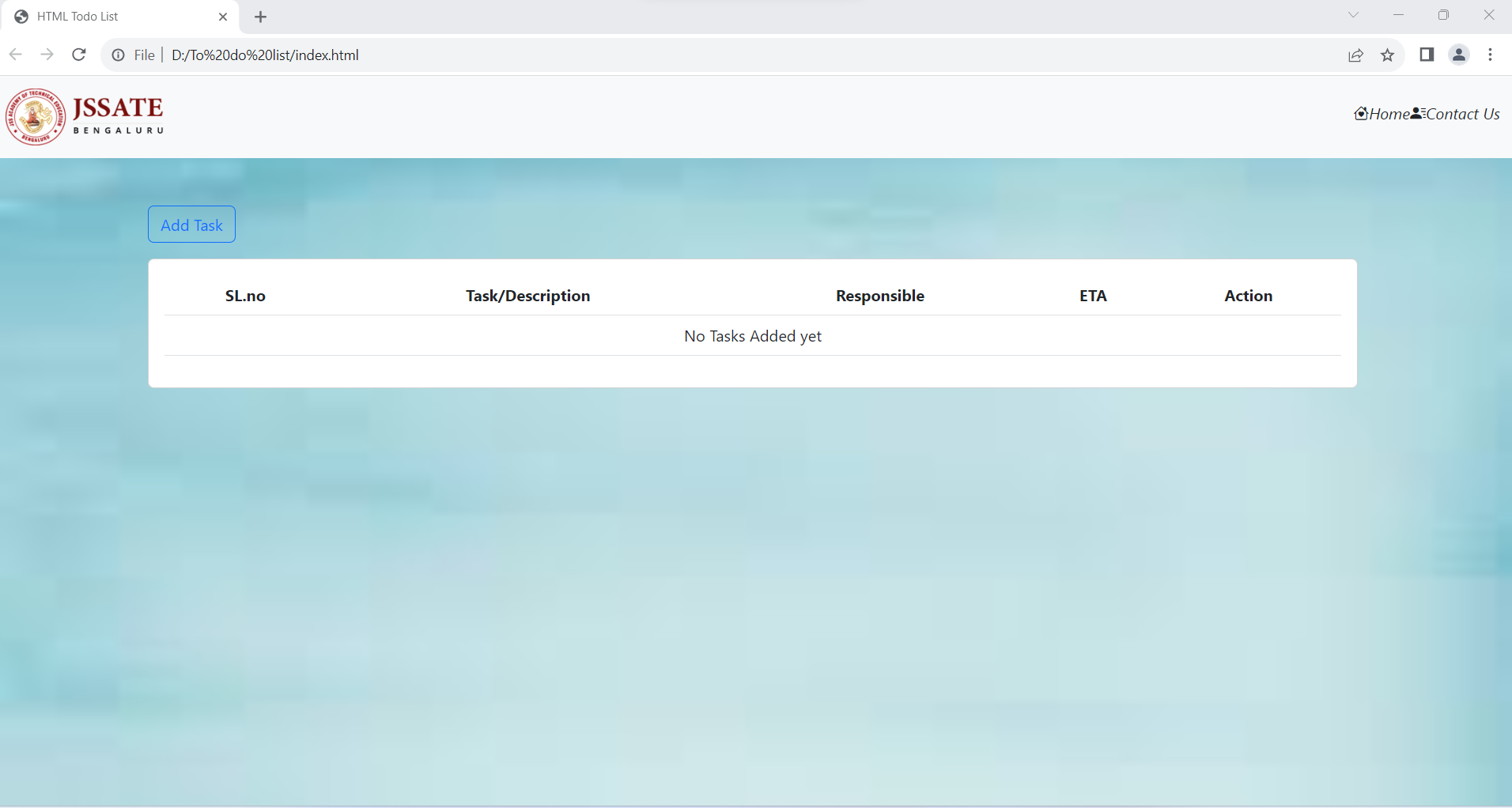
            }

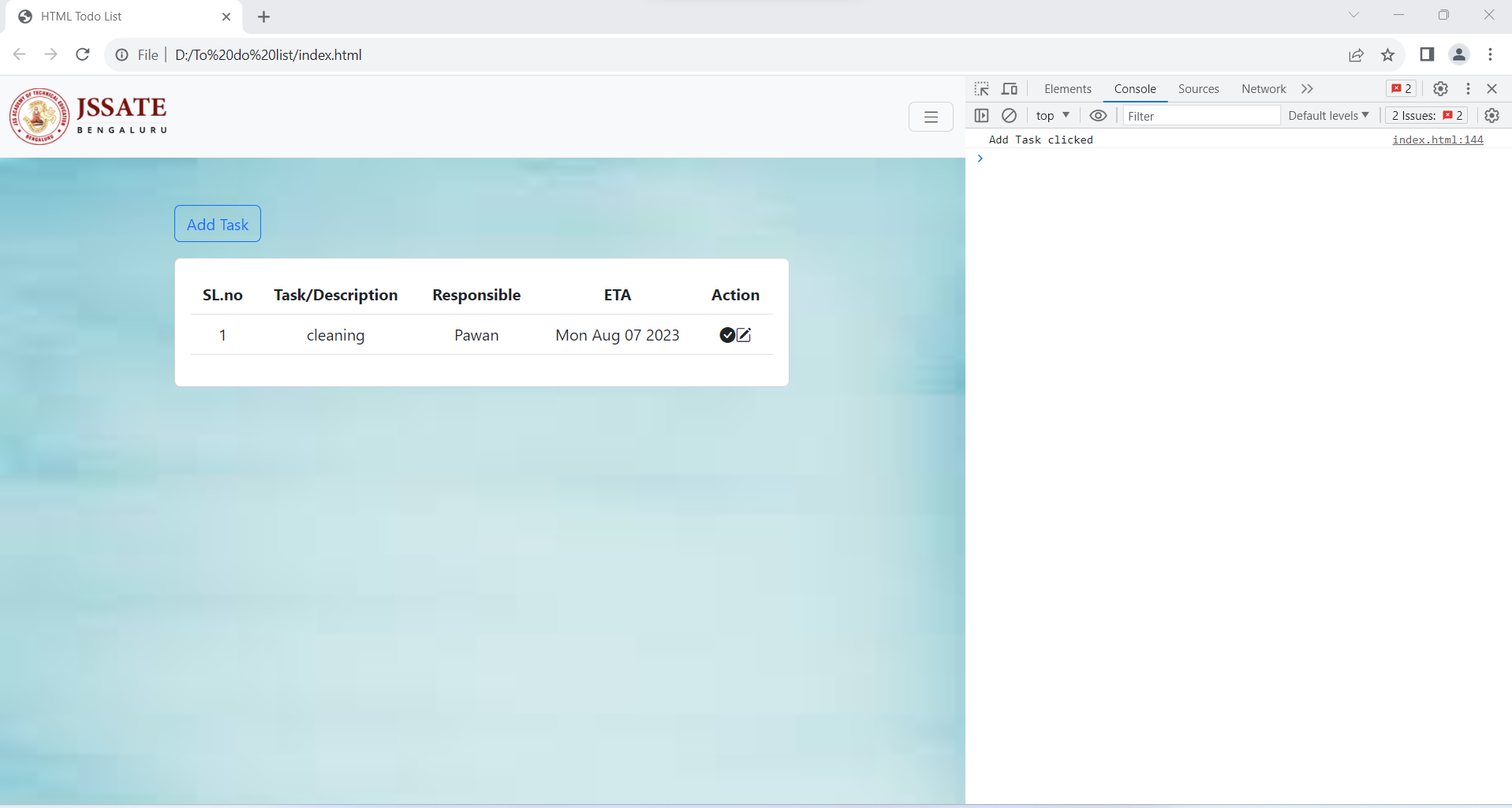
        </script>

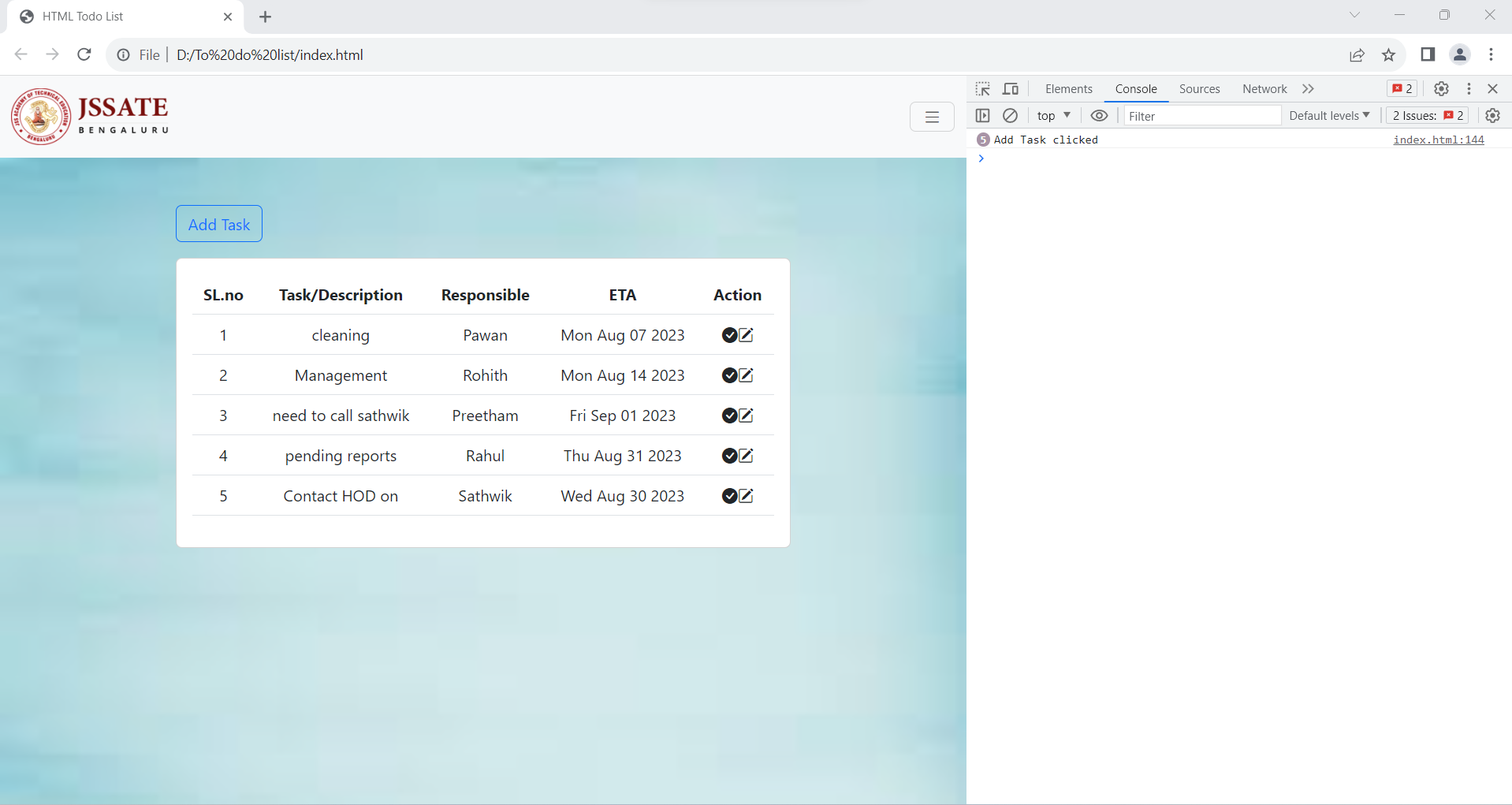
    </body>

</html>

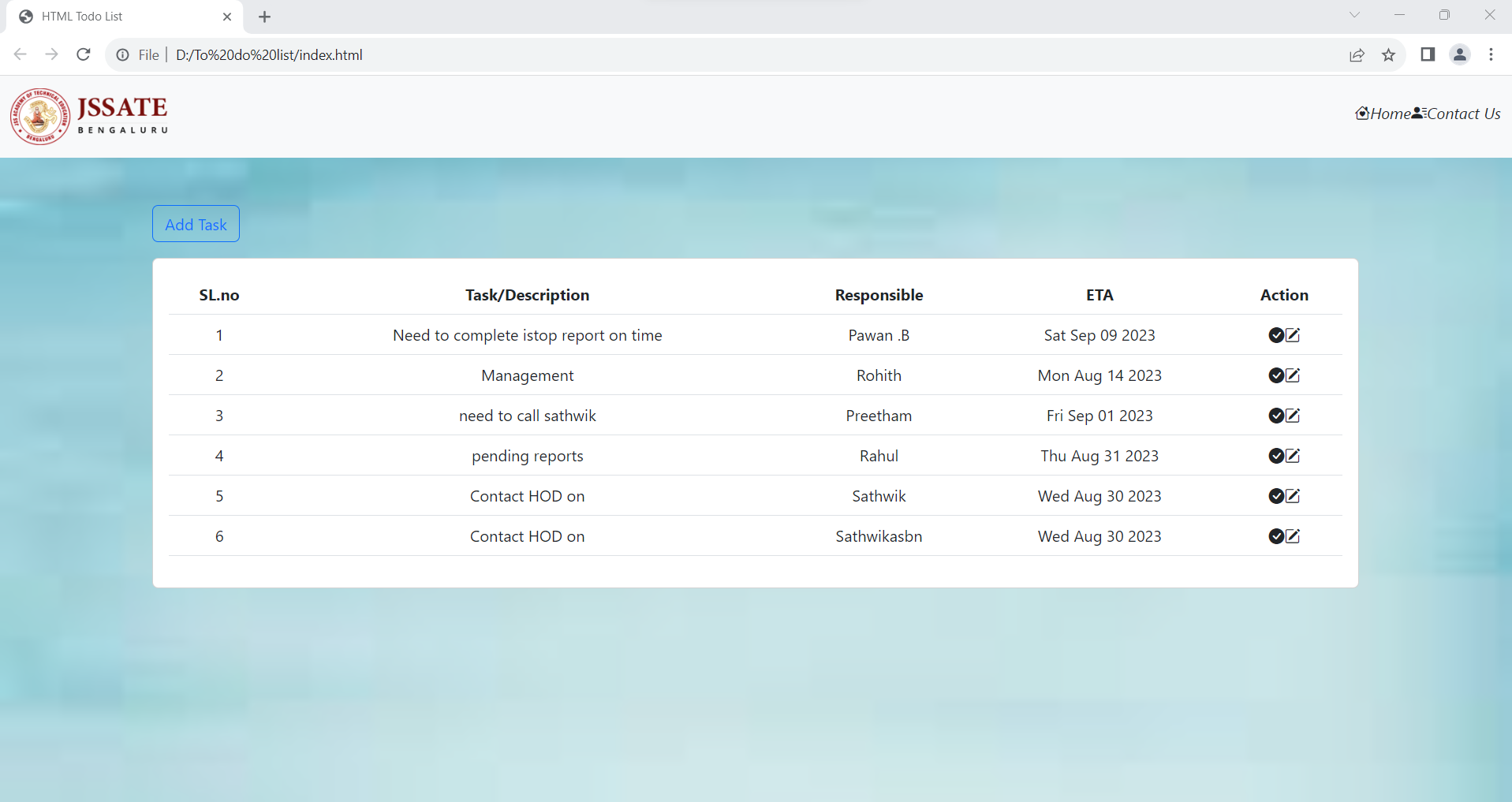
**Result and Output**



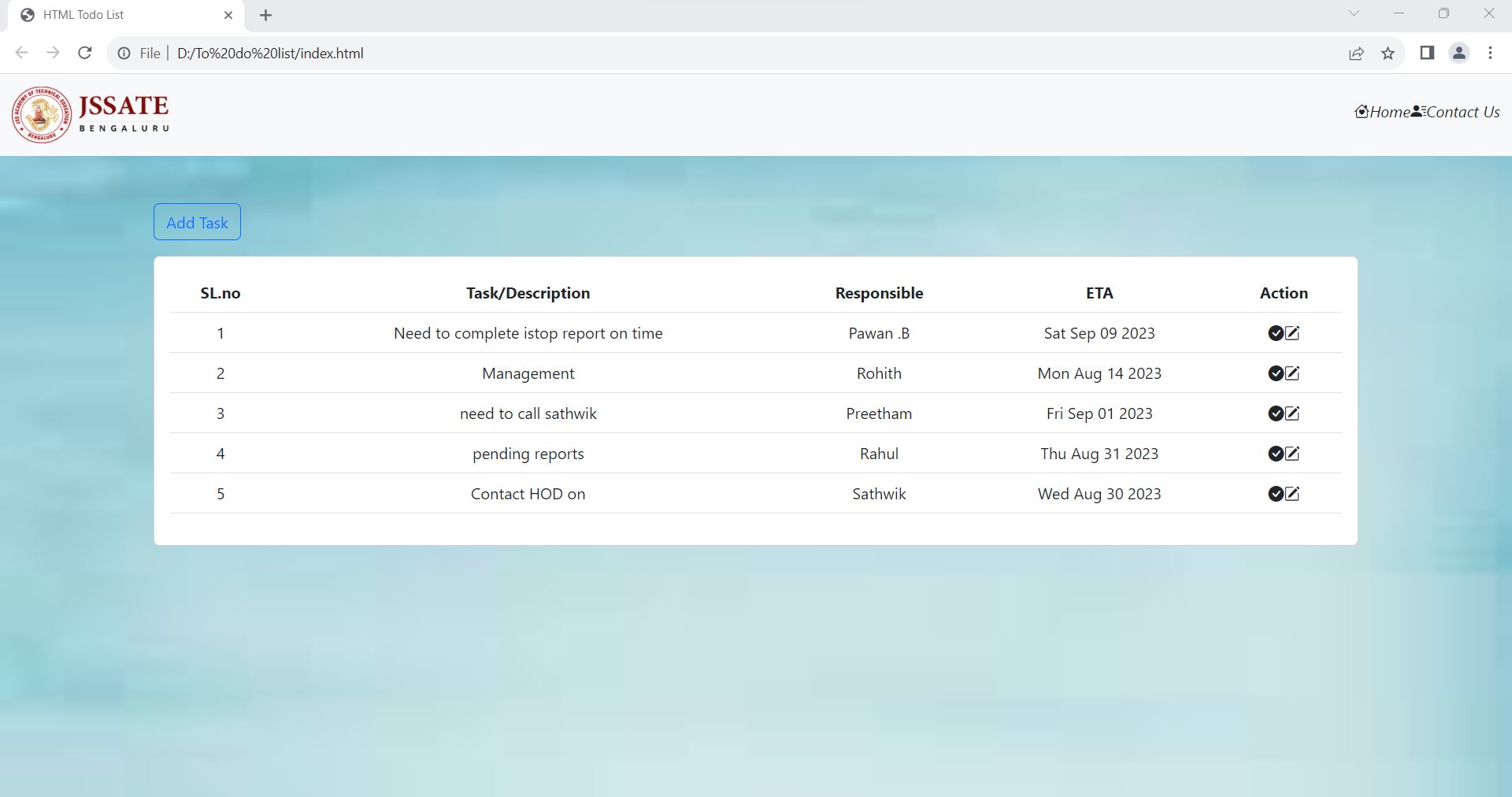




After edit



Removing the list



**Conclusion**

In this HTML code, a basic To-Do List web application is constructed, incorporating Bootstrap for styling and JavaScript for interactivity. The page features a responsive navigation bar, a task list displayed in a table, and the capability to add, edit, and mark tasks as done. Additionally, the code employs modal dialogs for task input and editing, and it utilizes Local Storage for rudimentary data persistence.

The web page presents a visually pleasing interface, enhanced by background styling. Users can conveniently initiate the task addition process through a "Add Task" button. The task data is collected via a modal dialog that solicits task descriptions, responsible persons, and estimated completion times. The task list is displayed neatly in a table format, with each task's details clearly outlined. Users have the flexibility to edit or mark tasks as done, and these actions are promptly reflected in the task list.

Despite its functional aspects, there are opportunities for refinement. The navigation bar could be adjusted for improved user experience by relocating the icons associated with "Home" and "Contact Us" to align them properly with their respective links. Additionally, the code might benefit from further enhancements, such as robust error handling and input validation, to ensure data accuracy. Furthermore, advanced features like task categorization or due date reminders could be incorporated to augment functionality.

In summary, this HTML code serves as an effective foundation for a basic To-Do List application. While it successfully full fills its core objectives, there remains room for refinement and expansion to create a more comprehensive and user-friendly task management experience.