Internship Project - HTML To Do List

Domain: Web Development - Front End

Intern: Saisha Suresh Bore
Date: 08-08-2024

ABSTRACT

This report details the creation of a basic To-Do list application using HTML, CSS, JavaScript developed as part of an internship centred on front-end web development. The primary goal of the project is to construct a simple yet functional web application that allows users to add, manage, and remove tasks in a list format.

HTML, or Hypertext Markup Language, is the standard markup language used to create web pages. It provides the structure of a web page, enabling developers to arrange content and create a user interface. By developing this To-Do list, the project aims to reinforce the principles of HTML, demonstrating how a simple web-based tool can be constructed using basic tags and attributes.

In addition to serving as a practical application of HTML skills, this project lays the groundwork for more advanced topics in web development, such as incorporating CSS for styling and JavaScript for interactivity. The report documents the entire development process, from the initial design and coding to testing and final deployment, offering a comprehensive overview of the project and its outcomes.

Objective

The primary objective of this project is to design and develop a basic To-Do list application using HTML, with the aim of demonstrating the fundamental ability to structure a simple yet effective web application. This project serves as an introduction to front-end web development, focusing specifically on the use of HTML to create a well-organized and functional user interface.

The To-Do list application is intended to showcase the core principles of HTML, such as the correct usage of tags, attributes, and elements to build a coherent and user-friendly interface. By focusing on HTML alone, the project emphasizes the importance of semantic markup, which not only ensures that the application is accessible and maintainable but also lays the foundation for integrating other technologies like CSS and JavaScript in future enhancements.

Additionally, this project aims to provide a hands-on experience that solidifies the learner's understanding of HTML. It challenges the developer to think critically about how to structure content on a webpage, how to manage user input, and how to display dynamic information, such as a list of tasks that users can interact with.

Moreover, the project is designed to cultivate essential skills necessary for tackling more complex web development tasks in the future. These skills include understanding how to design a user interface that is both functional and intuitive, writing clean and organized HTML code, and planning for scalability and future enhancements.

By the end of this project, the developer should not only have a functional To-Do list application but also a deeper appreciation for the role HTML plays in web development. This foundational knowledge will be invaluable as they progress to more advanced topics, such as styling the application with CSS or adding interactivity with JavaScript.

Introduction

The To-Do list is a widely recognized tool used for task management. In this project, a simple To-Do list application is developed using only HTML. While a full-fledged To-Do list typically requires JavaScript and CSS for interactivity and styling, this project focuses on the foundational structure, making it an excellent exercise for understanding HTML's role in web development.

In this project, an interactive website has been created so that the user can enter the details of the task that has to be done and the person who have given the particular task and the date and time under which they have to complete the task. In the website, we can add as many tasks that we want and have to complete the task.

Methodology

The development of the To-Do list application was approached systematically to ensure that each phase of the project was thoroughly planned, executed, and tested. The methodology consisted of three main stages: Planning, Development, and Testing.

Planning

The planning phase is crucial for the successful execution of any project. For this To-Do list application, the planning stage involved a detailed understanding of the project requirements and the overall structure of the web application. The following key aspects were considered during planning:

1. **Defining the Features:** The primary features of the To-Do list were identified, including the ability for users to add new tasks, view existing tasks, and remove tasks. While the project focuses on HTML, planning also considered how these features would be implemented logically within the constraints of HTML. In the beginning of the website, the logo of 1stop has also been set up using html code.

2. **Structuring the HTML Document:** A plan was made for the HTML structure, which included deciding on the tags and elements that would be used to create the user interface. This included selecting appropriate HTML tags such as <input> for entering tasks, <button> for adding tasks, and <ul> for displaying the list of tasks. The hierarchy and nesting of elements were carefully planned to ensure that the code would be clean, organized, and easy to maintain. In the project, Bootstarp also has been used. Several icons have been put up in the project.

3. **User Interface Design:** Although styling was not the focus, the planning phase included a basic design layout to ensure that the To-Do list was visually clear and user-friendly. The layout was kept simple, with a focus on functionality, ensuring that users could easily understand how to interact with the application.

Development

The development phase involved translating the plans into actual HTML code. This phase was carried out with meticulous attention to detail, ensuring that the HTML code accurately reflected the planned structure. The key activities in the development phase included:

1. **Creating the HTML Skeleton:** The initial step in development was to create the basic structure of the HTML document. This included defining the <!DOCTYPE html> declaration, setting up the <head> section with necessary metadata, and starting the <body> section where the main content would reside.

2. **Building the Input Field and Button:** The next step was to add an input field where users could enter their tasks. This was done using the <input> tag with a placeholder attribute to guide the user. A button was then added using the <button> tag, which users could click to add their tasks to the list. These elements were carefully placed to ensure ease of use.

3. **Constructing the Task List:** The task list was built using a <div> tag where each task would be represented. Although interactivity was not implemented in this HTML only project, the structure was designed to allow for future enhancements, such as adding JavaScript functionality to dynamically update the list.

4. **Commenting and Documentation:** As the code was written, comments were added to explain the purpose of different sections and elements. This documentation is essential for maintaining the code and for any future developers who might work on the project.

Testing

The final phase of the project was testing, where the HTML code was executed in a web browser to ensure that the application worked as intended. The testing phase involved the following steps:

1. **Correction of mistakes:** There were many mistakes corrected and rectified which made the concept clearer and more understandable to me and has given a deeper knowledge in this field.

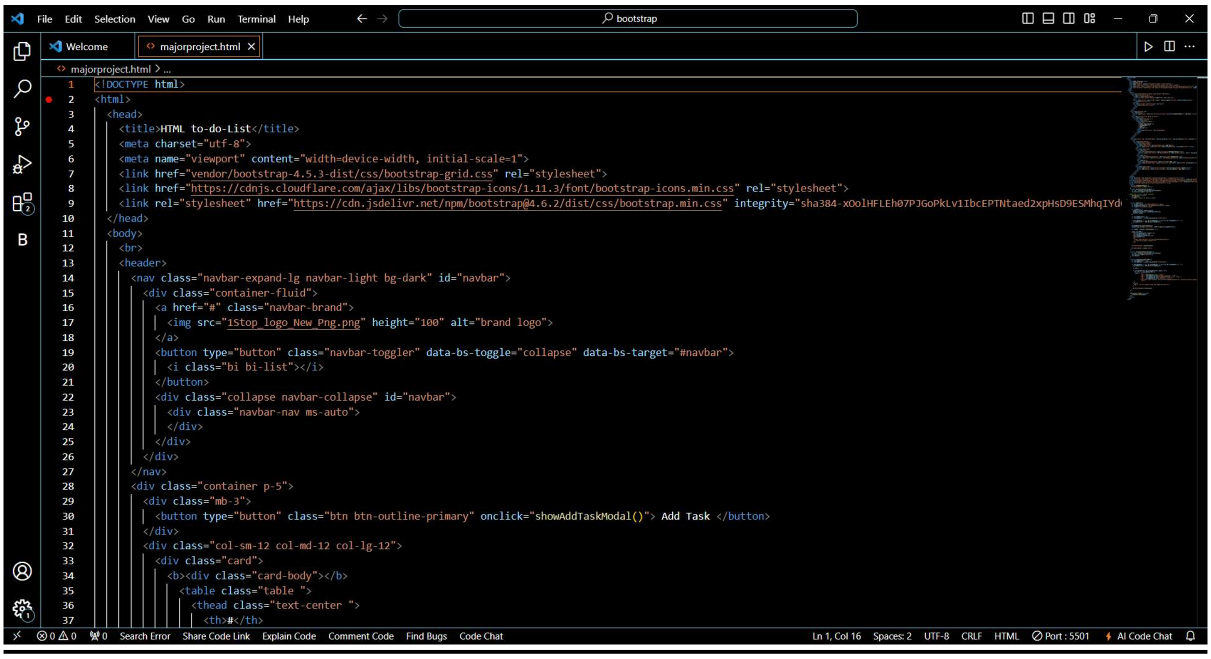
2. **Initial Testing:** The HTML file was opened in various web browsers to check if the structure displayed correctly and consistently. Special attention was given to ensure that the input field, button, and task list were properly aligned and functioning as expected.

3. **User Interaction Testing:** The application was tested by simulating user interactions, such as entering a task in the input field and clicking the "Add Task" button. Although no backend or JavaScript was involved in this project, it was crucial to ensure that the input field and button behaved as intended, displaying the user input correctly.

4. **Cross-Browser Compatibility:** The To-Do list was tested across different web browsers (e.g., Chrome, Firefox, Edge) to ensure that the HTML structure was consistent and displayed correctly across various environments. This step was vital to ensure that the application was accessible to all users, regardless of their browser choice.

5. **Final Review and Refinement:** After initial testing, any issues or inconsistencies were addressed, and the HTML code was refined. This step involved minor adjustments to ensure that the structure was not only functional but also optimized for future development, such as adding CSS for styling or JavaScript for interactivity.

CODE

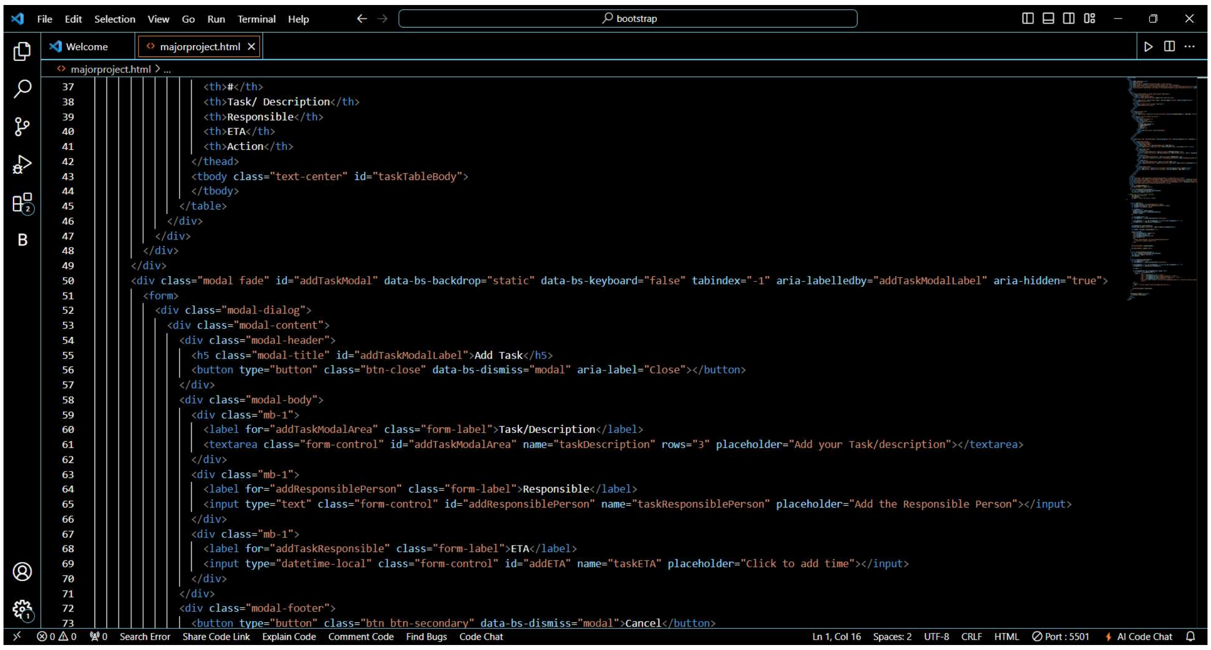


```html
<!DOCTYPE html>
<html>
  <head>
    <title>HTML to-do-List</title>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <link href="vendor/bootstrap-4.5.3-dist/css/bootstrap-grid.css" rel="stylesheet">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-icons/1.11.3/font/bootstrap-icons.min.css" rel="stylesheet">
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@4.6.2/dist/css/bootstrap.min.css" integrity="sha384-xOolHFLEh07PJGoPkLv1IbcEPTNtaed2xpHsD9ESMhqIYd
  </head>
  <body>
    <br>
    <header>
      <nav class="navbar-expand-lg navbar-light bg-dark" id="navbar">
        <div class="container-fluid">
          <a href="#" class="navbar-brand">
            <img src="1Stop_logo_New_Png.png" height="100" alt="brand logo">
          </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">
            </div>
          </div>
        </div>
      </nav>
      <div class="container p-5">
        <div class="mb-3">
          <button type="button" class="btn btn-outline-primary" onclick="showAddTaskModal()"> Add Task </button>
        </div>
        <div class="col-sm-12 col-md-12 col-lg-12">
          <div class="card">
            <b><div class="card-body"></b>
              <table class="table ">
                <thead class="text-center ">
                  <th>#</th>
```

This HTML document provides the structure for a task management webpage, utilizing Bootstrap for a responsive and visually appealing design. The <head> section includes links to Bootstrap's CSS for styling and grid layout, along with Bootstrap Icons for additional visual elements.

The <body> features a navigation bar with a dark background and a brand logo, which collapses into a toggle button on smaller screens, ensuring a responsive design. This navbar is designed for branding and potential future menu items.

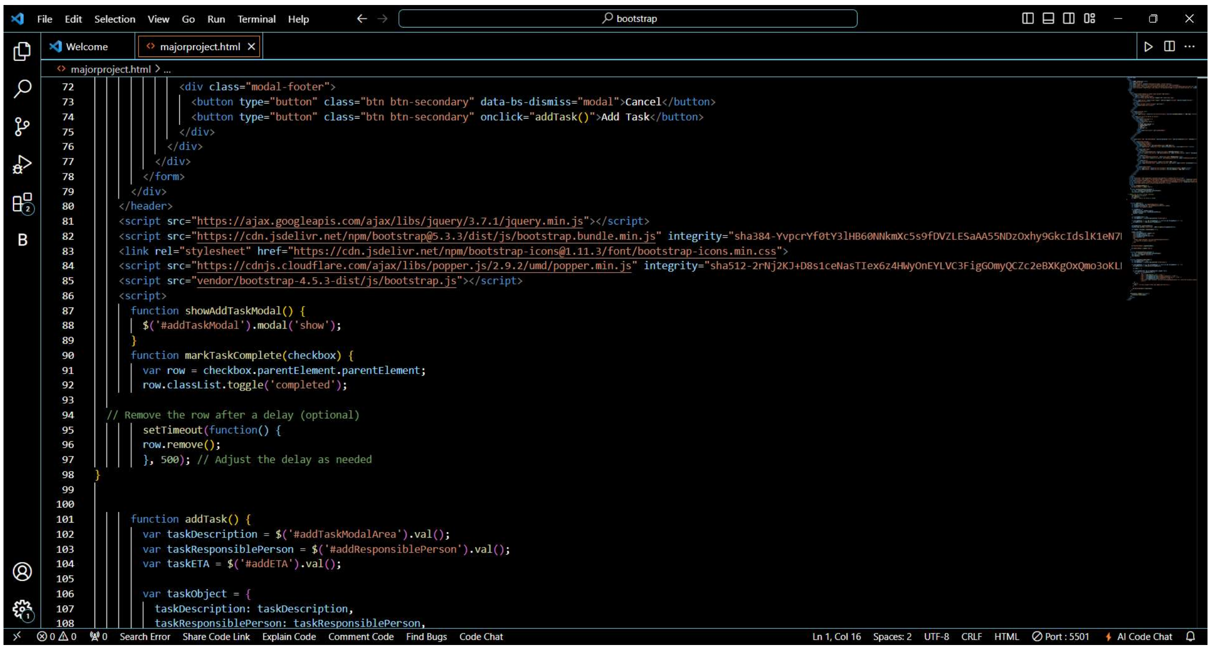
The main content area, enclosed in a Bootstrap container with padding, includes a button labeled "Add Task" that activates a JavaScript function to open a task addition modal. Below the button, a Bootstrap card holds a table intended for task display. This layout ensures that the page is user-friendly and adaptable to various devices, integrating HTML, CSS, and JavaScript effectively to manage tasks and enhance the user experience.



```html
            <th>#</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 class="modal fade" id="addTaskModal" data-bs-backdrop="static" data-bs-keyboard="false" tabindex="-1" aria-labelledby="addTaskModalLabel" aria-hidden="true">
  <form>
    <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="addTaskModalArea" class="form-label">Task/Description</label>
            <textarea class="form-control" id="addTaskModalArea" name="taskDescription" rows="3" placeholder="Add your Task/description"></textarea>
          </div>
          <div class="mb-1">
            <label for="addResponsiblePerson" class="form-label">Responsible</label>
            <input type="text" class="form-control" id="addResponsiblePerson" name="taskResponsiblePerson" placeholder="Add the Responsible Person"></input>
          </div>
          <div class="mb-1">
            <label for="addTaskResponsible" class="form-label">ETA</label>
            <input type="datetime-local" class="form-control" id="addETA" name="taskETA" placeholder="Click to add time"></input>
          </div>
        </div>
        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cancel</button>
```

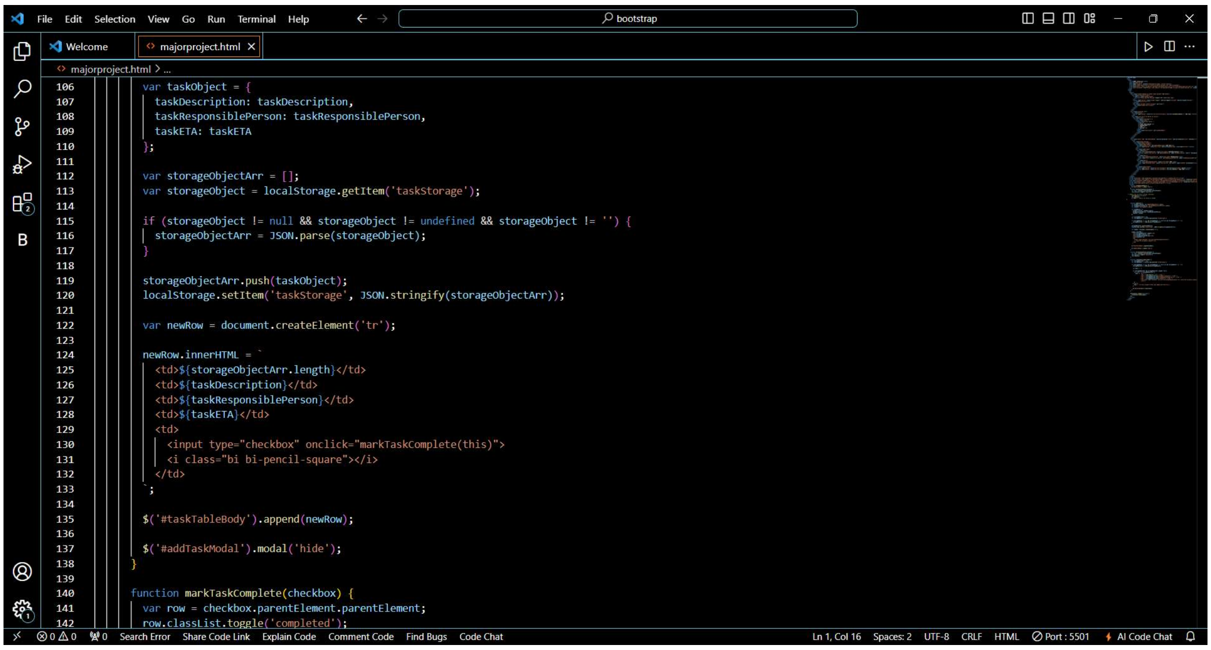
This HTML snippet enhances the task management webpage with a detailed table and a modal form for task entry. The table, styled with Bootstrap, is designed to display tasks with columns for an ID, task description, responsible person, estimated time of arrival (ETA), and action buttons. The <thead> element defines column headers, while the <tbody> section is prepared for dynamic task entries.

The modal, triggered by the "Add Task" button, provides a form for users to input task details. It includes fields for task description, the responsible person, and ETA, with a submit button to add the task and a cancel button to close the modal. Bootstrap’s modal component ensures that the task addition interface is user-friendly and seamlessly integrated into the webpage. This setup combines HTML, CSS, and Bootstrap to create a functional and interactive task management experience.



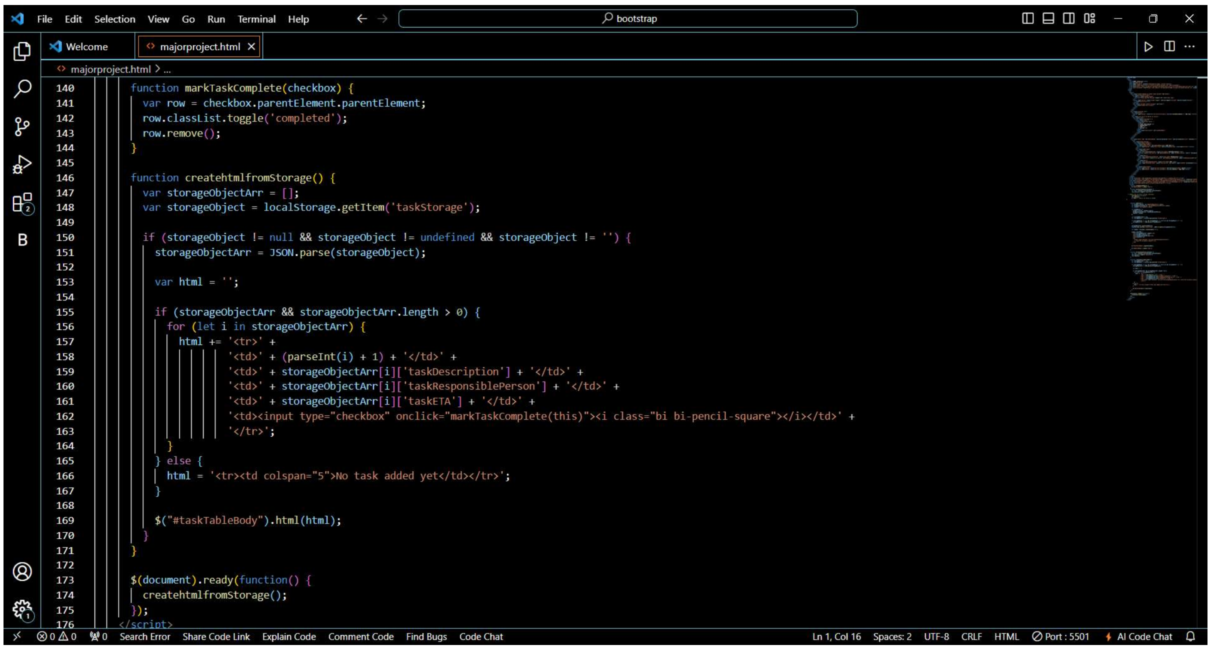
```
              <div class="modal-footer">
                <button type="button" class="btn btn-secondary" data-bs-dismiss="modal">Cancel</button>
                <button type="button" class="btn btn-secondary" onclick="addTask()">Add Task</button>
              </div>
            </div>
          </div>
        </form>
      </div>
    </header>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.7.1/jquery.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js" integrity="sha384-YvpcrYf0tY3lHB60NNkmXc5s9fDVZLESaAA55NDzOxhy9GkcIdslK1eN7
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-icons@1.11.3/font/bootstrap-icons.min.css">
    <script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.9.2/umd/popper.min.js" integrity="sha512-2rNj2KJ+D8s1ceNasTIex6z4HWyOnEYLVC3FigGOmyQCZc2eBXKgOxQmo3oKL
    <script src="vendor/bootstrap-4.5.3-dist/js/bootstrap.js"></script>
    <script>
      function showAddTaskModal() {
        $('#addTaskModal').modal('show');
      }
      function markTaskComplete(checkbox) {
        var row = checkbox.parentElement.parentElement;
        row.classList.toggle('completed');

  // Remove the row after a delay (optional)
        setTimeout(function() {
        row.remove();
        }, 500); // Adjust the delay as needed
}


      function addTask() {
        var taskDescription = $('#addTaskModalArea').val();
        var taskResponsiblePerson = $('#addResponsiblePerson').val();
        var taskETA = $('#addETA').val();

        var taskObject = {
          taskDescription: taskDescription,
          taskResponsiblePerson: taskResponsiblePerson,
```

This HTML snippet completes the task management webpage by incorporating additional JavaScript and Bootstrap functionalities. It includes a modal footer with "Cancel" and "Add Task" buttons, where the latter triggers the addTask() function. Bootstrap and jQuery libraries are linked for modal and script support. The JavaScript code defines two functions: showAddTaskModal() to display the task addition modal and markTaskComplete() to handle task completion. The markTaskComplete() function toggles a "completed" class on the task row and optionally removes it after a brief delay, enhancing user interaction with the task list.



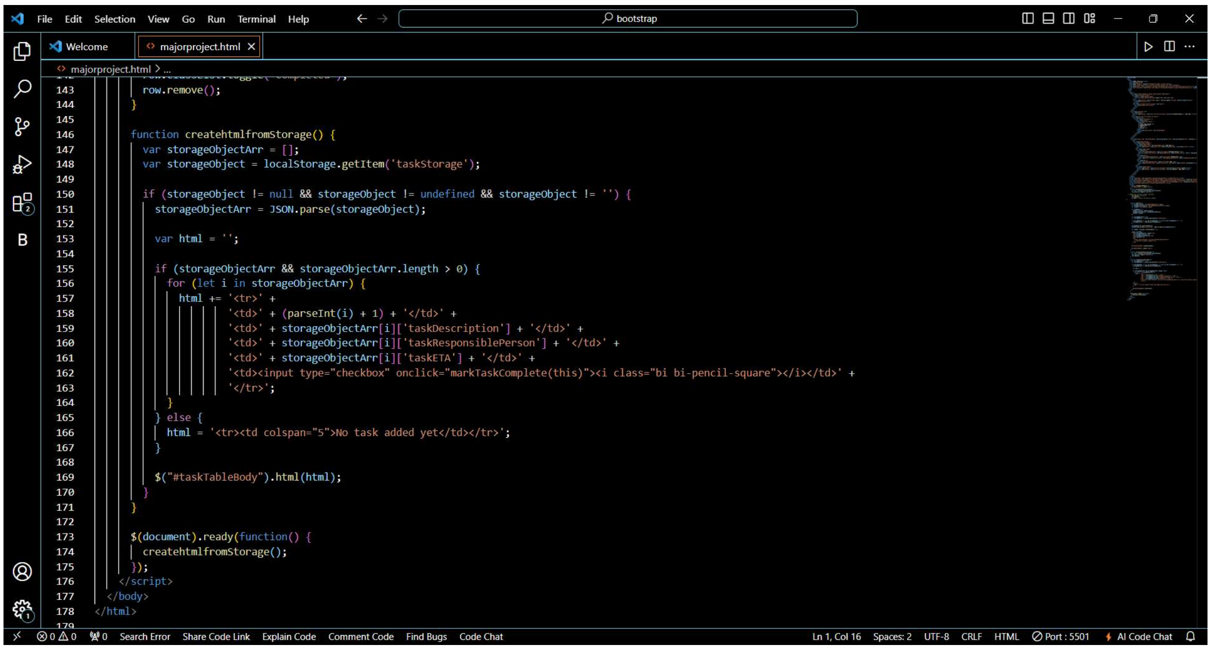
```
  var taskObject = {
    taskDescription: taskDescription,
    taskResponsiblePerson: taskResponsiblePerson,
    taskETA: taskETA
  };

  var storageObjectArr = [];
  var storageObject = localStorage.getItem('taskStorage');

  if (storageObject != null && storageObject != undefined && storageObject != '') {
    storageObjectArr = JSON.parse(storageObject);
  }

  storageObjectArr.push(taskObject);
  localStorage.setItem('taskStorage', JSON.stringify(storageObjectArr));

  var newRow = document.createElement('tr');

  newRow.innerHTML = `
    <td>${storageObjectArr.length}</td>
    <td>${taskDescription}</td>
    <td>${taskResponsiblePerson}</td>
    <td>${taskETA}</td>
    <td>
      <input type="checkbox" onclick="markTaskComplete(this)">
      <i class="bi bi-pencil-square"></i>
    </td>
  `;

  $('#taskTableBody').append(newRow);

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

function markTaskComplete(checkbox) {
  var row = checkbox.parentElement.parentElement;
  row.classList.toggle('completed');
```

This JavaScript code enhances the task management application by implementing functionality for adding and completing tasks. The showAddTaskModal() function displays the modal for task entry using Bootstrap's modal methods. The addTask() function retrieves values from the modal’s input fields, creates a task object, and stores it in the browser's local storage. It then dynamically adds a new row to the task table with the task details and interactive elements, including a checkbox for marking tasks as complete and an icon for editing. The markTaskComplete() function toggles a "completed" class on the row and removes it from the table. The code effectively manages task addition and completion, providing a smooth user experience and persistent task storage.


File Edit Selection View Go Run Terminal Help
bootstrap
Welcome
majorproject.html
majorproject.html > ...
Search Error Share Code Link Explain Code Comment Code Find Bugs Code Chat
Ln 1, Col 16 Spaces: 2 UTF-8 CRLF HTML Port : 5501 AI Code Chat

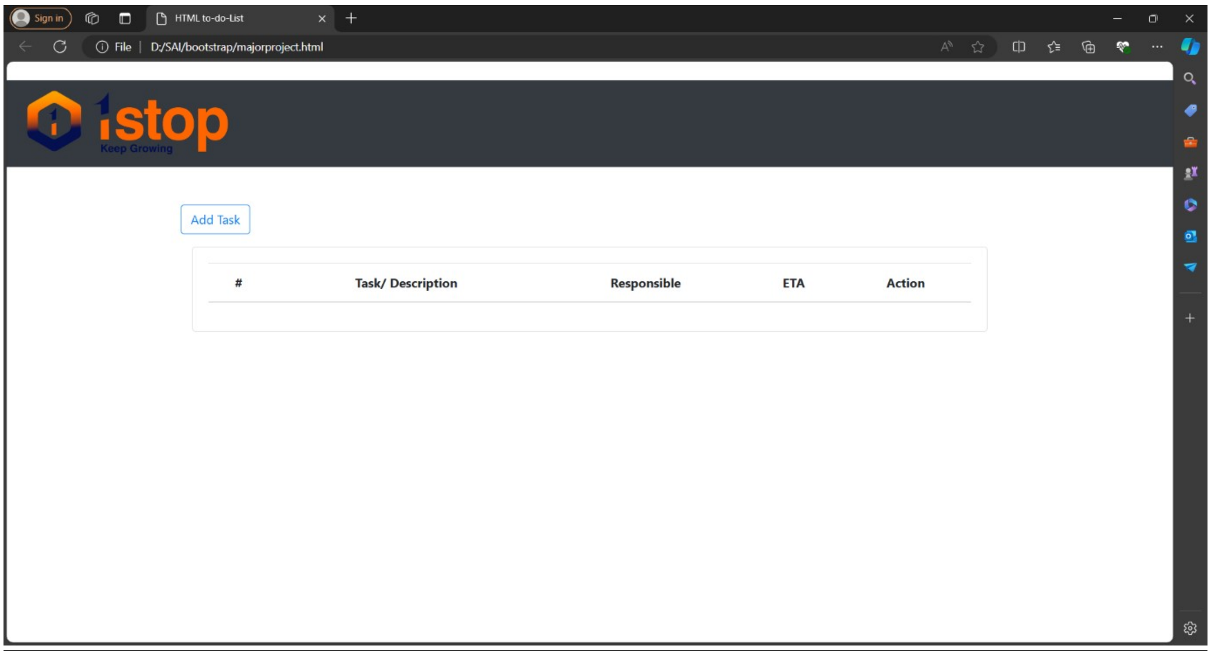
```
    function markTaskComplete(checkbox) {
      var row = checkbox.parentElement.parentElement;
      row.classList.toggle('completed');
      row.remove();
    }

    function createhtmlfromStorage() {
      var storageObjectArr = [];
      var storageObject = localStorage.getItem('taskStorage');

      if (storageObject != null && storageObject != undefined && storageObject != '') {
        storageObjectArr = JSON.parse(storageObject);

        var html = '';

        if (storageObjectArr && storageObjectArr.length > 0) {
          for (let i in storageObjectArr) {
            html += '<tr>' +
                    '<td>' + (parseInt(i) + 1) + '</td>' +
                    '<td>' + storageObjectArr[i]['taskDescription'] + '</td>' +
                    '<td>' + storageObjectArr[i]['taskResponsiblePerson'] + '</td>' +
                    '<td>' + storageObjectArr[i]['taskETA'] + '</td>' +
                    '<td><input type="checkbox" onclick="markTaskComplete(this)"><i class="bi bi-pencil-square"></i></td>' +
                    '</tr>';
          }
        } else {
          html = '<tr><td colspan="5">No task added yet</td></tr>';
        }

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

    $(document).ready(function() {
      createhtmlfromStorage();
    });
  </script>
```

This JavaScript code manages task display and initialization from browser storage. The createhtmlfromStorage() function retrieves tasks from the browser's local storage and dynamically generates HTML to populate the task table. It parses stored task data and constructs table rows with task details, including task description, responsible person, ETA, and interactive elements like checkboxes for marking tasks complete and an icon for editing. If no tasks are found in storage, a message is displayed indicating that no tasks have been added yet. This function is called when the document is ready, ensuring that the task list is correctly populated with previously stored tasks each time the page is loaded. The code enhances user experience by maintaining and displaying task data consistently across page reloads.


File Edit Selection View Go Run Terminal Help
bootstrap
Welcome
majorproject.html
majorproject.html > ...
0 0 0 Search Error Share Code Link Explain Code Comment Code Find Bugs Code Chat
Ln 1, Col 16 Spaces: 2 UTF-8 CRLF HTML Port : 5501 AI Code Chat

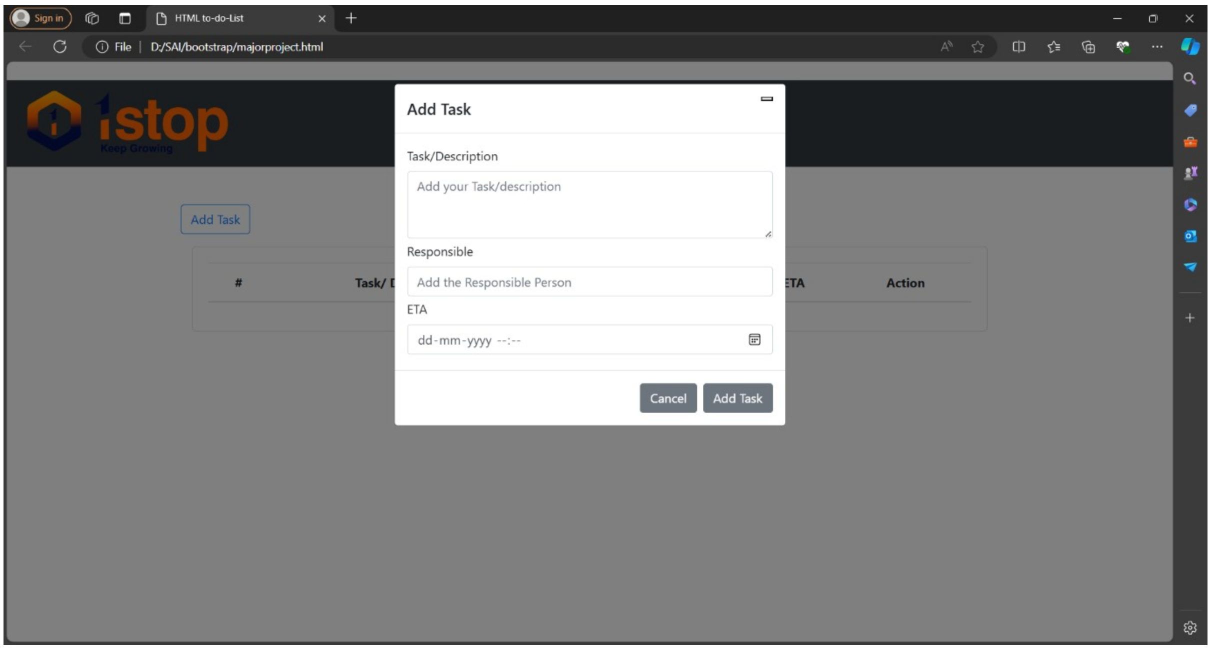
```
        row.remove();
      }

      function createhtmlfromStorage() {
        var storageObjectArr = [];
        var storageObject = localStorage.getItem('taskStorage');

        if (storageObject != null && storageObject != undefined && storageObject != '') {
          storageObjectArr = JSON.parse(storageObject);

          var html = '';

          if (storageObjectArr && storageObjectArr.length > 0) {
            for (let i in storageObjectArr) {
              html += '<tr>' +
                      '<td>' + (parseInt(i) + 1) + '</td>' +
                      '<td>' + storageObjectArr[i]['taskDescription'] + '</td>' +
                      '<td>' + storageObjectArr[i]['taskResponsiblePerson'] + '</td>' +
                      '<td>' + storageObjectArr[i]['taskETA'] + '</td>' +
                      '<td><input type="checkbox" onclick="markTaskComplete(this)"><i class="bi bi-pencil-square"></i></td>' +
                      '</tr>';
            }
          } else {
            html = '<tr><td colspan="5">No task added yet</td></tr>';
          }

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

      $(document).ready(function() {
        createhtmlfromStorage();
      });
    </script>
  </body>
</html>
```

OUTPUT



#	Task/ Description	Responsible	ETA	Action

This is the front page of the website in which no data has been uploaded. When we click Add Task button, it will enable us to add tasks in the following table


Add Task

Task/Description

Add your Task/description

Responsible

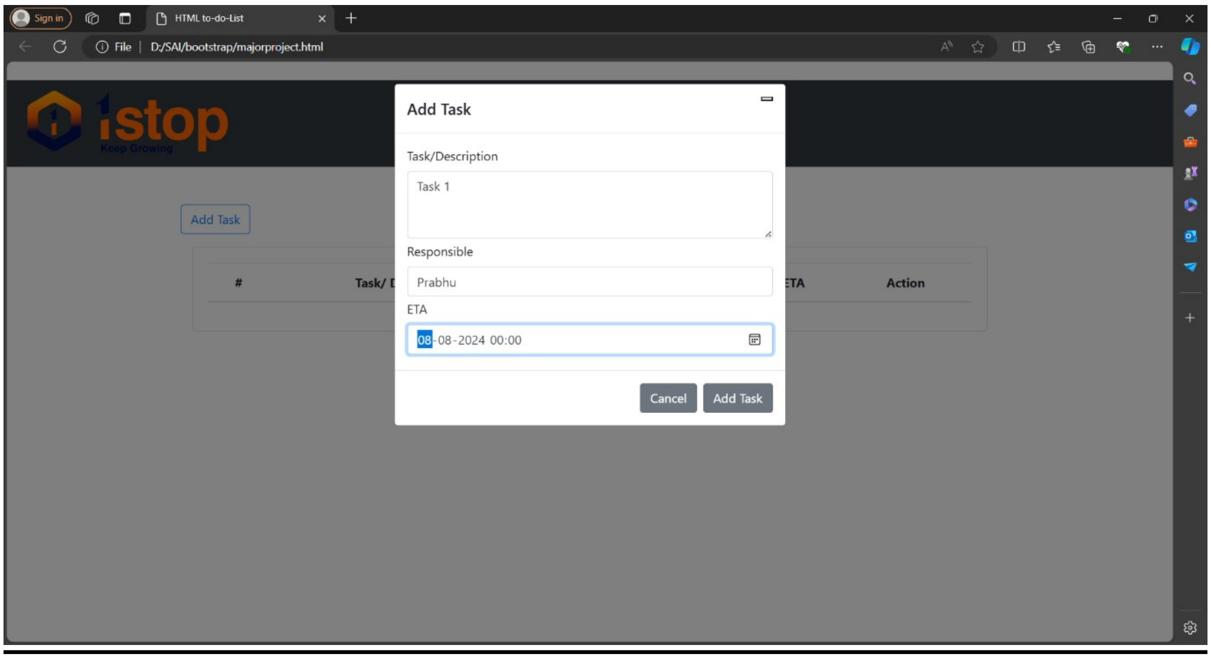
Add the Responsible Person

ETA

dd-mm-yyyy --:--

Cancel Add Task

When the add task button is clicked, we are able to add the task description , who is resposible for it and the ETA.



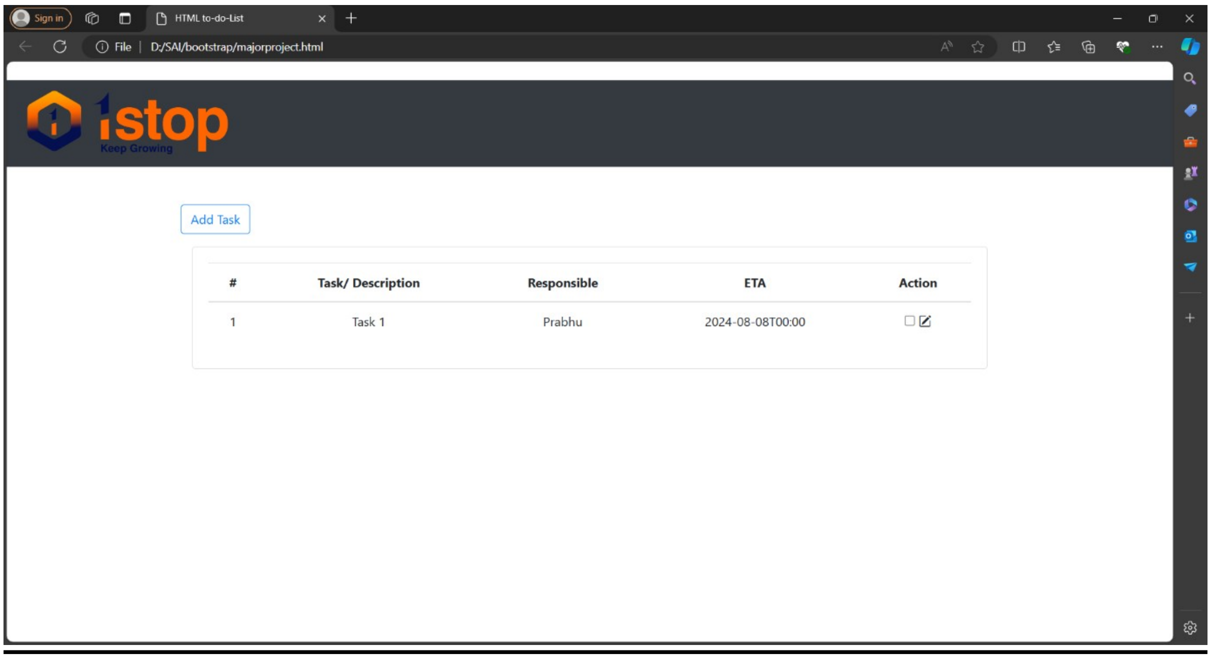
Add Task

Task/Description: Task 1

Responsible: Prabhu

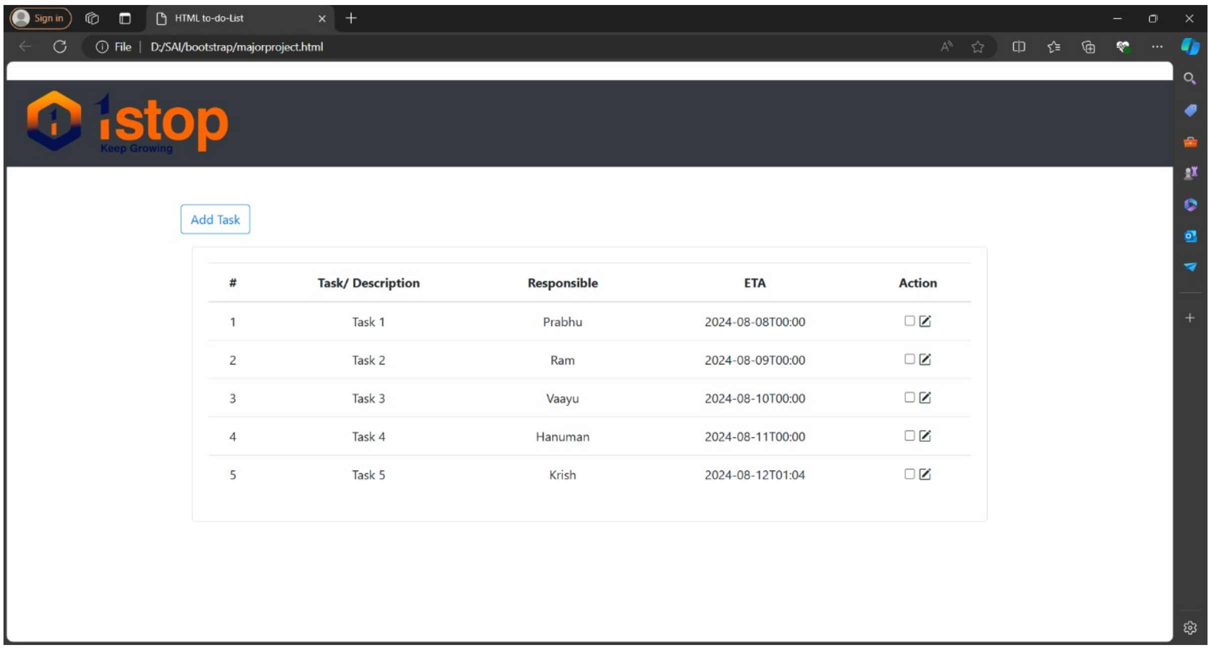
ETA: 08-08-2024 00:00

The data is being entered.



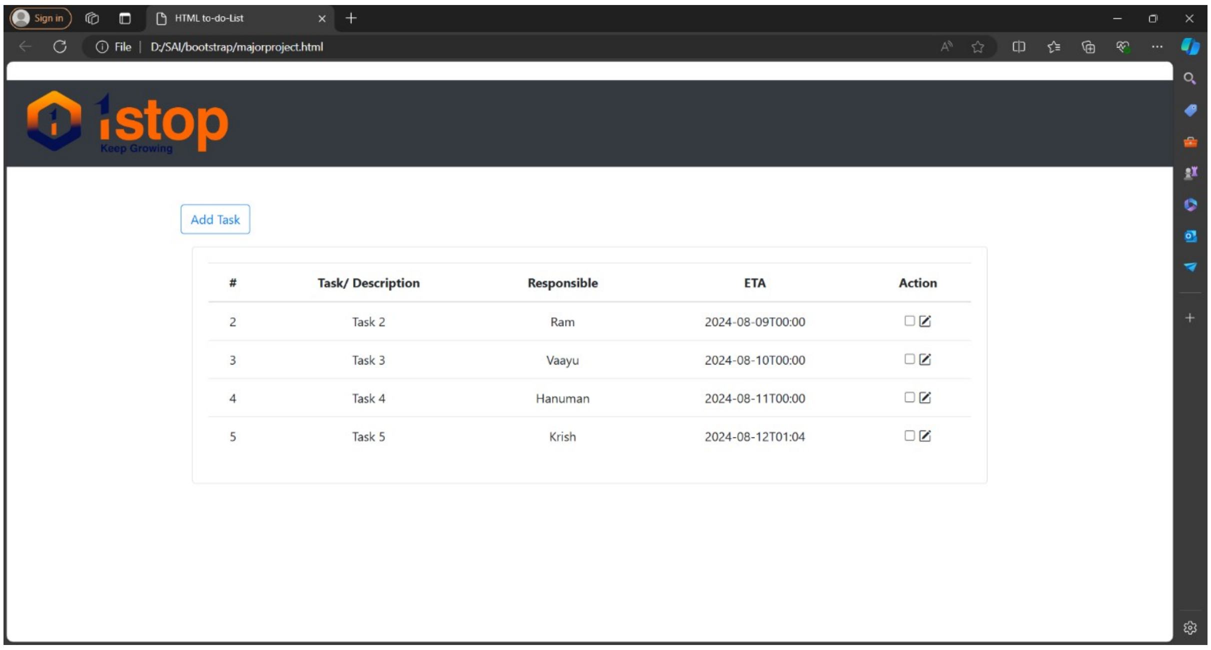
#	Task/ Description	Responsible	ETA	Action
1	Task 1	Prabhu	2024-08-08T00:00	☐ ✎

Task 1 has been uploaded on the website.



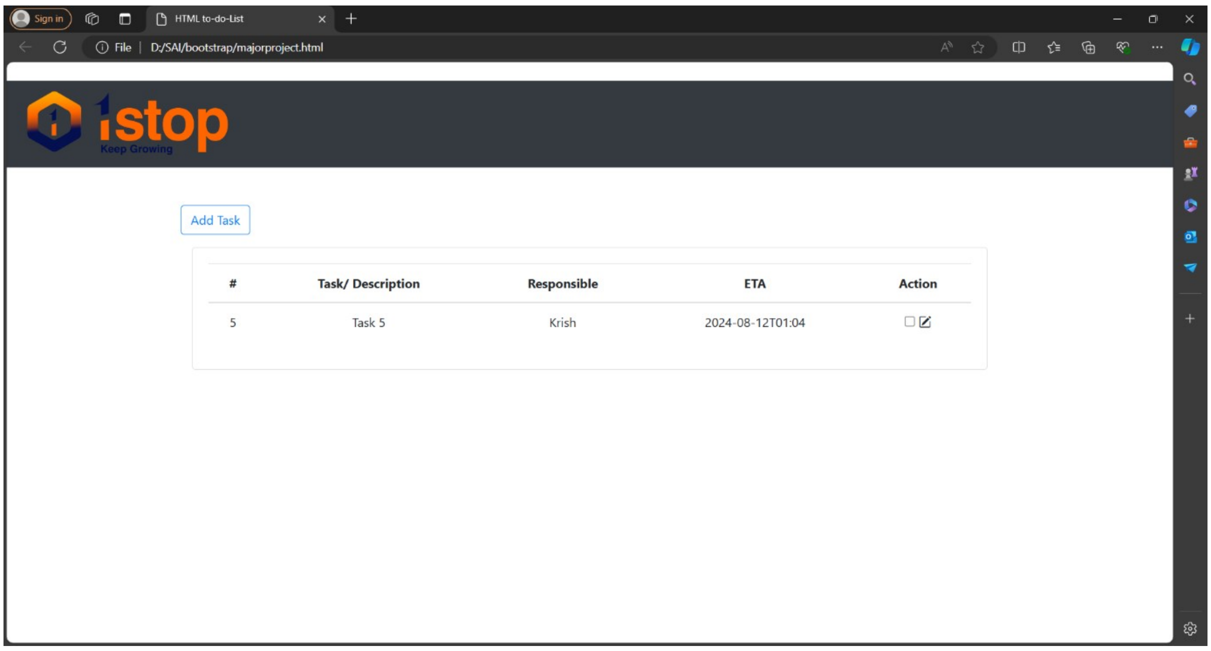
#	Task/ Description	Responsible	ETA	Action
1	Task 1	Prabhu	2024-08-08T00:00	☐ ✎
2	Task 2	Ram	2024-08-09T00:00	☐ ✎
3	Task 3	Vaayu	2024-08-10T00:00	☐ ✎
4	Task 4	Hanuman	2024-08-11T00:00	☐ ✎
5	Task 5	Krish	2024-08-12T01:04	☐ ✎

This is a snapshot of the final website.



#	Task/ Description	Responsible	ETA	Action
2	Task 2	Ram	2024-08-09T00:00	☐ ✎
3	Task 3	Vaayu	2024-08-10T00:00	☐ ✎
4	Task 4	Hanuman	2024-08-11T00:00	☐ ✎
5	Task 5	Krish	2024-08-12T01:04	☐ ✎

Once the Task has been completed by the person responsible, we can click on the button under action and the task 1 is portrayed as complete.



#	Task/ Description	Responsible	ETA	Action
5	Task 5	Krish	2024-08-12T01:04	☐

Here task 1, task 2, task 3 and task 4 have been completed.

CONCLUSION

The development of the To-Do list application has provided a comprehensive exploration of essential web technologies, including HTML, CSS, and JavaScript. This project has not only reinforced fundamental HTML concepts but also demonstrated the practical application of CSS for styling and JavaScript for interactivity, resulting in a fully functional and visually appealing web application.

HTML, the backbone of the project, was used to structure the content and define the layout of the To-Do list. The HTML code established the basic framework, including the input field, task list, and buttons, providing a solid foundation for the application.

CSS was employed to enhance the visual presentation of the application. Through CSS, the project achieved a cleaner and more engaging user interface. Styling elements such as colors, fonts, spacing, and layouts contributed to a more intuitive and aesthetically pleasing user experience. CSS allowed for responsive design adjustments, ensuring that the application remained functional and user-friendly across various screen sizes and devices. The use of CSS helped transform the simple HTML structure into a polished and professional-looking application.

JavaScript added dynamic functionality to the To-Do list, making it interactive and user-friendly. JavaScript was used to handle user actions, such as adding and removing tasks. By incorporating event listeners and manipulating the DOM (Document Object Model), JavaScript enabled real-time updates to the task list, allowing users to interact with the application in a meaningful way. This scripting capability brought the static HTML and styled CSS elements to life, demonstrating the power of JavaScript in creating interactive web experiences.

The integration of HTML, CSS, and JavaScript in this project highlights the importance of each technology in web development. HTML provides the structure, CSS enhances the visual appeal, and JavaScript adds interactivity. This combination ensures that the application is not only functional but also engaging and user-centric. Apart from this we have also used JSON applications such as JSON.parse and JSON.stringify

Overall, this project has achieved its objectives by creating a functional To-Do list application that demonstrates the effective use of HTML for structure, CSS for design, and JavaScript for interactivity. The skills gained through this project are fundamental for advancing in web development, offering a strong foundation for tackling more complex challenges. The successful implementation of these technologies underscores the importance of a holistic approach to web development, where each component plays a crucial role in delivering a seamless and comprehensive user experience.

In conclusion, this project has been a valuable learning experience, providing practical insights into the integration of HTML, CSS, and JavaScript. It has prepared the developer for future projects and further exploration in the field, emphasizing the importance of mastering these core technologies for building sophisticated and effective web applications.

THANK YOU

THANK YOU

SAISHA SURESH BORE

INDIAN INSTITUTE OF INFORMATION

TECHNOLOGY, DHARWAD

WEB DEVELOPMENT - FRONT END