

## **Internship Project - To-Do List**

- **ABSTRACT:**

A to-do list is a common web application that helps users to manage their tasks and goals. It allows users to create, edit, delete, and mark tasks as done. A to-do list, in general, is a simple but effective way of managing one's tasks and goals. It helps users to prioritize, organize, and track their progress on various activities. This project is focussed on the creation of a web-based to-do list application using basic HTML, CSS, JavaScript, Bootstrap, jQuery, and Popper.js. This application will allow users to add, edit, delete, and mark tasks as completed. The application will have the following features:

- A simple and elegant user interface that uses Bootstrap components and styles.
- A responsive design that adapts to different screen sizes and devices using Bootstrap grid system.
- A dynamic and interactive functionality that uses JavaScript, jQuery, and Popper.js to manipulate the Document Object Model (DOM) and handle user events.
- A persistent data storage that uses the browser's local storage API to save and load the tasks.

The project will demonstrate the fundamental concepts and skills of web development, such as HTML tags, CSS properties, JavaScript variables, functions, loops, conditions, arrays, objects, events, methods, selectors, DOM manipulation, local storage, Bootstrap framework, jQuery library, Popper.js library, and more. The project will also follow the best practices of web design, such as accessibility, usability, and performance. The project aims to provide a practical and enjoyable experience for users who want to organize their tasks efficiently.

I have used HTML to create the basic structure and content of the To-Do List, Bootstrap to style and position the elements, Bootstrap grid system and components to create a responsive layout, Bootstrap typography and colors to create a consistent and attractive design, Bootstrap icons and utilities to add some visual elements, Bootstrap carousels to add some interactivity and functionality. Bootstrap is a free and open-source framework that provides ready-made components and classes that can be used to style and layout HTML elements, and can be used to create responsive and user-friendly web interfaces. I have used JavaScript along with jQuery to add interactivity and functionality, jQuery selectors and methods to target and manipulate the elements dynamically, and jQuery events and callbacks to handle the user actions and responses. jQuery is a free and open-source library that simplifies the manipulation of the DOM, the handling of events, the animation of elements, and the communication with servers. I have used Popper.js to position the tooltips, Popper.js instances and options to create and customize the tooltips. Popper.js is a free and open-source library that calculates the optimal position of an element (such as a popover or a tooltip) based on its reference element (such as a button or an input). I have also used some custom CSS styles to customize the appearance of the website, and override or modify some of the Bootstrap default styles. HTML, CSS, JavaScript, Bootstrap, jQuery and Popper.js are collectively used here to create and design the web-based To-Do List which is responsive, interactive, and functional.

The main challenges of this project were to design a layout that is suitable for a webpage-based To-Do List(application), to organize the content in a clear and concise manner, to ensure the functionality of the carousel, and the tooltip, to optimize the webpage for different devices and browsers, and most importantly, to design and code the brains and looks of the To-Do List in the webpage. I have overcome these challenges by using Bootstrap grid system and components to create a responsive layout, by using headings, paragraphs, lists, images,

icons, buttons, cards, carousels, and tooltips to present the To-Do List in an attractive and professional way, by using jQuery selectors and methods to target and manipulate the elements dynamically, by using jQuery events and callbacks to handle the user actions and responses, by using Popper.js instances and options to create and customize the tooltips, and by using flexbox properties, meta tags, title attributes, and keywords to enhance the responsiveness, accessibility, and SEO-friendliness of the webpage itself.

The main limitations of this project were that the To-Do List does not have any server-side database interaction, and that it does not have any advanced features, such as animations, transitions, or sliders. These limitations can be addressed by adding more languages, such as PHP, MySQL, or MongoDB, and by adding more libraries or frameworks, such as Animate.css, Swiper.js, or Slick.js, to add more visual effects and functionality.

The main outcomes of this project were that I have successfully created a professional, and functional To-Do List using HTML, CSS, JavaScript, Bootstrap, jQuery and Popper.js, that works as expected, and that I have improved my web development skills and knowledge.

- **OBJECTIVE:**

The objective of this project is to design and develop a webpage-based To-Do list application that helps users to manage their tasks and goals effectively. The application will use basic HTML, CSS, JavaScript, Bootstrap, jQuery, and Popper.js to create a user-friendly, responsive, dynamic, interactive, persistent, and functional web application. The project will also demonstrate the fundamental concepts and skills of web development and web design. The webpage-based To-Do List contains all the necessary features within a single HTML

document, without requiring any additional page loading or navigation. The To-Do List is designed to be responsive, user-friendly, professional, and functional.

The project will use HTML to create the structure and content of the website. The project will use Bootstrap to style and position the elements, Bootstrap grid system and components to create the main responsive layout for the To-Do List, Bootstrap typography and colors to give the To-Do List a consistent and attractive design, Bootstrap icons and utilities to add some visual and interactive elements which adds extra functionality to the To-Do List, Bootstrap carousels to add some interactivity and functionality. The project will use JavaScript along with jQuery to add interactivity and functionality, jQuery selectors and methods to target and manipulate the elements dynamically, and jQuery events and callbacks to handle the user actions and responses. The project will use Popper.js instances and options to create, and customize and position the tooltips. The project will also use some custom CSS styles to override or modify some of the Bootstrap default styles. The project will follow the best practices of web development, such as using semantic tags, comments, indentation, validation, responsiveness, and accessibility. The project will test the To-Do List with different test cases and debug any errors or bugs. So, in a quick recap, these are the objectives of this project:

- To create a simple and elegant user interface that uses Bootstrap components and styles to display the tasks.
- To implement a responsive design that adapts to different screen sizes and devices using Bootstrap grid system.
- To add a dynamic and interactive functionality that uses JavaScript, jQuery, and Popper.js to manipulate the Document Object Model (DOM) and handle user events such as adding, editing, deleting, and marking tasks as done.

- To use the browser's local storage API to save and load the tasks in the application.
- To create a professional and clean functional To-Do List.

- INTRODUCTION:

A to-do list is a simple and effective way to organize and manage your tasks. It helps you to prioritize your work, track your progress, and achieve your goals. However, creating and maintaining a to-do list can be challenging, especially if you have multiple projects and deadlines. This is where creating a web-based To-Do list application using HTML, CSS, JavaScript, Bootstrap, jQuery, and Popper.js comes to help. Such a webpage-based To-Do List can be accessed from any browser and device without requiring any installation or download, and, at the same time, can be very polished and function-loaded application. Web-based To-Do List can also be customized and enhanced using various web technologies such as HTML, CSS, and JavaScript.

HTML is a markup language that defines the structure and content of a web page. HTML uses tags and attributes to create elements, such as headings, paragraphs, lists, images, links, forms, and more. HTML is the foundation of web development and is essential for creating any web application.

However, HTML alone is not enough to create a modern and attractive web interface. HTML needs to be combined with other technologies, such as CSS, JavaScript, and various libraries and frameworks, to add style, interactivity, and functionality to the web page. CSS is a style sheet language that specifies how the HTML elements are displayed on the screen. CSS uses properties and selectors to apply different styles, such as colors, fonts, layouts, positions, and more. JavaScript is a scripting language that enables dynamic, and interactive behavior and

communication on the web page. JavaScript uses functions and events to manipulate the DOM (Document Object Model), which is a representation of the HTML elements as objects. It can manipulate the HTML elements, respond to user events, communicate with web servers, and store data locally or remotely. JavaScript can also use various libraries and frameworks, which are collections of pre-written code that provide ready-made features and components for web development.

One of the most popular and widely used frameworks for web development is Bootstrap. Bootstrap is a free and open-source framework that provides ready-made components and classes to create responsive and user-friendly web interfaces. Bootstrap uses a grid system to create a flexible layout that adapts to different screen sizes and devices. Bootstrap also provides various components, such as buttons, forms, tables, cards, modals, carousels, and more, that can be easily added and customized with HTML attributes and classes. Bootstrap also uses CSS styles and JavaScript plugins to enhance the appearance and performance of the components.

Another popular and widely used library for web development is jQuery. jQuery is a free and open-source library that simplifies the manipulation of the DOM, the handling of events, the animation of elements, and the communication with servers. jQuery uses a simple syntax that allows selecting and modifying the HTML elements with less code. jQuery also provides various methods and effects that can be chained together to create complex animations and transitions. jQuery also uses AJAX (Asynchronous JavaScript and XML) to send and receive data from servers without reloading the page.

A useful library that works well with Bootstrap and jQuery is Popper.js. Popper.js is a free and open-source library that calculates the optimal position of an element (such as a popover or a tooltip) based on its reference element (such as a button or an input). Popper.js uses a

smart algorithm that considers the size, position, overflow, boundaries, and margins of the elements to avoid any clipping or overlapping issues. Popper.js also provides various options and modifiers to customize the behaviour and appearance of the popovers and tooltips.

The purpose of this project is to design and develop a webpage-based To-Do List that contains all the necessary features within a single HTML document, using HTML, CSS, JavaScript, Bootstrap, jQuery and Popper.js.

- **METHODOLOGY:**

A possible methodology for creating an HTML To-Do List is as follows:

- **Step 1:** Create a basic HTML structure for the website using the `<html>`, `<head>`, and `<body>` tags. Include the necessary links to Bootstrap, and CSS in the `<head>` section, and JavaScript, jQuery, and Popper.js libraries in the `<body>` section.
- **Step 2:** Stylize all the elements used in the making of the webpage and the To-Do List using internal CSS. This adds to the professional and polished look of the project itself.
- **Step 3:** Stylize the `<body>` and to it add the navigation bar which contains some important webpage header elements, here, the logo of 1stop which on clicking takes you to 1stop homepage and importantly, an instruction carousel link which, as the name suggests, provides you the instructions on how to use the To-Do List with images and illustrations. To implement this, use `<nav>` element with class “`navbar navbar-expand-lg navbar-light bg-light`” to implement the navigation bar at the top of the webpage. Use `<div>` element with class “`container-fluid`” to stretch the navbar across the entire width of the page at the top. Include the logo associated with the link.

Also include a `<button>` element with class `“navbar-toggler”` with `data-bs-toggle=“collapse”` and `data-bs-target=“#navbar”`. This allows us to interact with the navbar element in a small windowed webpage. Add `<div>` element with class `“collapse navbar-collapse”` and `id=“navbar”`. To it, add another `<div>` element with class `“navbar-nav ms-auto”`. To it add `<a>` element with class `“nav-item nav-link active”`, `id=“Inst”` `data-bs-toggle=“tooltip”`, `data-bs-placement=“bottom”`, and some title as tooltip, which make our “logo” and “instruction” functional as required.

- Step 4:** Create `<div>` element with class `“container”` and stylize it. To it add another `<div>` element with class `“mb-3”` and stylize it. This `<div>` element will contain the different functional buttons in the form of To-Do List. Add four `<button>` elements and stylize it using bootstrap and CSS. Add different id’s to them and call the different functions onclick. This covers up the To-Do List Header. Now add another `<div>` element with class `“d-flex justify-content-center”` which in turn contains `<div>` element with class `“col-sm-12 col-md-12 col-lg-12”`. This contains `<div>` element with class `“card”` which then contains `<div>` element with class `“card-body”`. This finally contains `<table>` element with class `“table table-striped”` which creates a striped format To-Do List. This contains the `<thead>` with class `“text-center”` to contain the section headings for the list, and the `<tbody>` with class `“text-center”` and `id=“taskTableBody”` which contain the contents of the list.
- Step 5:** Add `<div>` element with class `“modal fade”` `id=“addTaskModal”` `data-bs-backdrop=“static”` `data-bs-keyboard=“false”` `tabindex=“-1”` `aria-labelledby=“addTaskModalLabel”` `aria-hidden=“true”`. This then contains `<form>` element which then contains `<div>` element of class `“modal-dialog”` which contains `<div>` element of class `“modal-content”` which then contains `<div>` element of class `“modal-header”` that holds the modal title and close button; `<div>` element with class



= “modal-body” that holds <input> elements for task description fill up as well as data and time; and finally, <div> element with class = “modal-footer” that holds the add task button and the close button.

Similarly, there is another <div> element with class = “modal fade” id = “updateTaskModal” data-bs-backdrop = “static” data-bs-keyboard = “false” tabindex = “-1” aria-labelledby = “updateTaskModalLabel” aria-hidden = “true”; which is similar to the addTask Modal except this picks up the data off the list and offers you to edit the task description and definitions and update the task.

Add <div> element with class = “modal fade” id = “showCalendar” data-bs-backdrop = “static” data-bs-keyboard = “false” tabindex = “-1” aria-labelledby = “showCalendarLabel” aria-hidden = “true”. To it, add <form> element which then contains <div> element of class = “modal-dialog” which then contains <div> element of class = “modal-content” which then contains <div> element of class = “modal-header” that holds the Calendar modal close button; <div> element of class = “modal-body” which holds two <div> element each of them taking up the halves of the modal body, each of which contains <p> elements with different id’s to display date; and finally another <div> element of class = “modal-footer” which holds <span> elements with different id’s to display the current time. This entire component works as the Calendar which displays current Date and Time.

Add <div> element with class = “modal modal-xl fade” id = “showCarousel” data-bs-backdrop = “static” data-bs-keyboard = “false” tabindex = “-1” aria-labelledby = “showCarouselLabel” aria-hidden = “true”. To it, add <form> element which then contains <div> element of class = “modal-dialog” which then contains <div> element of class = “modal-content” which then contains <div> element of class = “modal-

header” that holds the modal close button; contains <div> element with class = “carousel slide” id = “carousel” data-bvs-ride = “carousel” which then contains the carousel indicator buttons to navigate, images to display in the carousel, and carousel control buttons to go to next or previous images, and finally contains the <div> element with class = “modal-footer”. This entire component works as the instruction carousel.

- **Step 6:** Call out functions such as “createHtmlfromStorage()”, “loader()” and “Calendar()” to immediately update the List on load. Define functions such as “showAddTaskModal()”, “showCalendar()”, and “showCarousel()” which on call displays or opens up the corresponding modals.

Now, define the main functions which actually handle the data stored and the functionality of the entire To-Do List. Here we have used the following functions:

```
1. function addTask(){
    $("#addTaskModal").modal('hide');
    var dataArr = $("#taskInputForm").serializeArray();
    var taskObject = new Object();
    var storageObjectArr = [];
    var storageObject = localStorage.getItem('taskStorage');

    for(var i in dataArr){
        var name = dataArr[i]['name']
        var value = dataArr[i]['value']
        taskObject[name] = value
    }

    if(storageObject != null && storageObject != undefined && storageObject != ""){
        storageObjectArr = JSON.parse(storageObject)
```

```

storageObjectArr.push(taskObject)
}
else{
storageObjectArr.push(taskObject)
}
localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr))

createHtmlfromStorage();
loader();
$("#taskInputForm").trigger('reset')
}

```

This function takes the task data from the addTask modal and adds it to the local storage which is then used to display the task with the other tasks.

```

2. function createHtmlfromStorage(){
var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
var storageObjectArr = JSON.parse(storageObject)
var html = "";

if(storageObject != null && storageObject != undefined && storageObject != ""){
if(storageObjectArr && storageObjectArr.length > 0){
for(let i in storageObjectArr){
var date = new Date(storageObjectArr[i]['taskETA'])
html = html + '<tr id='+parseInt(i)+'write'+ ' style="text-decoration: none;">'
+ '<td id='+parseInt(i)+'write2'+ '>' + (parseInt(i)+1) + '</td>'
+ '<td>' + storageObjectArr[i]['taskDescription'] + '</td>'
+ '<td>' + storageObjectArr[i]['taskResponsiblePerson'] + '</td>'
+ '<td>' + date.toUTCString() + '</td>'
+ '<td><i class="bi bi-check-circle-fill" data-bs-toggle="tooltip" data-bs-
placement="bottom" title="Mark as Done" onclick="markAsDone('+i+')"></i>'

```

```

+ '<i class="bi bi-pencil-square" data-bs-toggle="tooltip" data-bs-
placement="bottom" title="Edit Task" onclick="editTask('+i+')"></i>'
+ '<i id='+i+' class="bi bi-star-fill" data-bs-toggle="tooltip" data-bs-
placement="bottom" title="Mark As Important" onclick="impTask('+i+')"
style="cursor: pointer; color: wheat;"></i>'
+ '<i class="bi bi-trash-fill" data-bs-toggle="tooltip" data-bs-placement="bottom"
title="Remove Task" onclick="removeTask('+i+')"></i>'
+ '</td></tr>'
}
}
else{
html = '<tr><td colspan="5">No Tasks Added Yet</td></tr>'
}
}
else{
html = '<tr><td colspan="5">No Tasks Added Yet</td></tr>'
}
$("#taskTableBody").html(html)
}

```

This function takes the data from the local storage and outputs the data in the stylized format in the html document as the To-Do List table or interface.

```

3. function removeTask(index){
  var storageObjectArr = [];
  var storageObject = localStorage.getItem('taskStorage');
  if(storageObject != null && storageObject != undefined && storageObject != ""){
    storageObjectArr = JSON.parse(storageObject);
    var ar1 = storageObjectArr.slice(0,parseInt(index));
    var ar2 = storageObjectArr.slice((parseInt(index)+1));
    storageObjectArr = ar1.concat(ar2);
  }
}

```

```

localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr));
var attriObj = new Object()
var attriArr = []
attriObj = localStorage.getItem('impData');
attriArr = JSON.parse(attriObj)
var arr1 = attriArr.slice(0,parseInt(index))
var arr2 = attriArr.slice((parseInt(index)+1))
attriArr = arr1.concat(arr2)
localStorage.setItem("impData",JSON.stringify(attriArr))
var attriObj2 = new Object()
var attriArr2 = []
attriObj2 = localStorage.getItem('impData2');
attriArr2 = JSON.parse(attriObj2)
var arr12 = attriArr2.slice(0,parseInt(index))
var arr22 = attriArr2.slice((parseInt(index)+1))
attriArr2 = arr12.concat(arr22)
localStorage.setItem("impData2",JSON.stringify(attriArr2))
createHtmlfromStorage();
loader();
}

```

This function extracts data from the local storage, interprets the data in the form of an array, removes certain element of the array and stores the data of the array back to local storage thus effectively removing a task.

```

4. function editTask(index){
    var storageObject = localStorage.getItem('taskStorage');
    var storageObjectArr = [];
    if(storageObject != null && storageObject != undefined && storageObject != ""){
        storageObjectArr = JSON.parse(storageObject);
    }
}

```

```

$("#editTaskTextArea").val(storageObjectArr[index]['taskDescription']);
$("#editResponsiblePerson").val(storageObjectArr[index]['taskResponsiblePerson'])
;
$("#editETA").val(storageObjectArr[index]['taskETA']);
$("#editIndex").val(index);
$("#updateTaskModal").modal('show');
}
}

```

This function fetches the data from the local storage, opens up the updateTask modal with previously filled in task details, which is then used to take data in which will overwrite the original data.

```

5. function updateTask(){
    $("#updateTaskModal").modal('hide');
    var dataArr = $("#taskUpdateForm").serializeArray();
    var taskObject = new Object();
    var storageObjectArr = [];
    var storageObject = localStorage.getItem('taskStorage');

    for(var i in dataArr){
        var name = dataArr[i]['name']
        var value = dataArr[i]['value']
        taskObject[name] = value
    }

    if(storageObject != null && storageObject != undefined && storageObject != ""){
        storageObjectArr = JSON.parse(storageObject)
        storageObjectArr[taskObject['taskIndex']] = taskObject
    }
}

```

```

localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr))
createHtmlfromStorage();
loader();
}

```

This function takes the data from the updateTask modal, extracts data from the local storage, overwrites that data with the data from the updateTask modal and stores it back to the local storage, thus effectively updating the data of the task.

```

6. function clearAll(){
    var storageObjectArr = [];
    var storageObject = localStorage.getItem('taskStorage');
    localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr));

    var attriArr = []
    localStorage.setItem("impData",JSON.stringify(attriArr))
    localStorage.setItem("impData2",JSON.stringify(attriArr))

    createHtmlfromStorage();
    loader();
}

```

This function will actually pass on a blank array to the local storage thus overwriting the array with actual data with blank data, thus effectively removing all tasks.

```

7. function doneAll(){
    var storageObjectArr = [];
    var storageObject = localStorage.getItem('taskStorage');
    storageObjectArr = JSON.parse(storageObject)

```

```

var attriObj = new Object()
var attriArr2 = []
for(var i=0;i<storageObjectArr.length;i=i+1){
attriArr2[i]=1
}
localStorage.setItem('impData2',JSON.stringify(attriArr2))
createHtmlfromStorage();
loader();
}

```

This function sets all elements of the array, that keeps track of the task done, to 1 that effectively reflects on to the fact that all tasks are done.

```

8. function impTask(index){
var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
storageObjectArr = JSON.parse(storageObject)
const imp = document.getElementById(index);
var attriObj = new Object()
var attriArr = []
for(var i=0;i<storageObjectArr.length;i=i+1){
attriArr[i]=0
}
attriObj = localStorage.getItem('impData');
if(attriObj != null && attriObj != undefined && attriObj != ""){
attriArr = JSON.parse(attriObj)
}
if(imp.style.color == "wheat"){
imp.style.color = "gold"
attriArr[index] = 1
}
else{

```



```

imp.style.color = "wheat"
attriArr[index] = 0
}

localStorage.setItem("impData",JSON.stringify(attriArr))
}

```

This function maintains an array that keeps track on which task is important. With click events, the elements of the array are toggled between 0 and 1 that decides which task is important.

```

9. function markAsDone(index){
  var storageObjectArr = [];
  var storageObject = localStorage.getItem('taskStorage');
  storageObjectArr = JSON.parse(storageObject)
  const imp = document.getElementById(parseInt(index)+"write");
  var attriObj2 = new Object()
  var attriArr2 = []
  for(var i=0;i<storageObjectArr.length;i=i+1){
    attriArr2[i]=0
  }
  attriObj2 = localStorage.getItem('impData2');
  if(attriObj2 != null && attriObj2 != undefined && attriObj2 != ""){
    attriArr2 = JSON.parse(attriObj2)
  }
  if(imp.style.textDecoration == "none"){
    imp.style.textDecoration = "line-through"
    attriArr2[index] = 1
  }
  else{

```

```

imp.style.textDecoration = "none"
attriArr2[index] = 0
}
localStorage.setItem("impData2",JSON.stringify(attriArr2))
}

```

This function maintains an array that keep track on which task is done. With click events, the elements of the array are toggled between 0 and 1 that decides which task is done.

```

10. function loader(){
    var attriObj = new Object()
    var attriArr = []
    attriObj = localStorage.getItem('impData');
    attriArr = JSON.parse(attriObj)

    for(var i in attriArr){
        if(attriArr[i]==1){
            const imp = document.getElementById(i)
            imp.style.color = "gold"
        }
    }

    var attriObj2 = new Object()
    var attriArr2 = []
    attriObj2 = localStorage.getItem('impData2');
    attriArr2 = JSON.parse(attriObj2)

    for(var i in attriArr2){
        if(attriArr2[i]==1){
            const imp = document.getElementById(parseInt(i)+"write")

```

```

imp.style.textDecoration = "line-through"
}
}
}

```

This function extract data from the array that decides which task is important and done, and accordingly changes the colors of the icons or background or text decorations.

```

11. function Calendar(){
    const date = document.getElementById("date")
    const day = document.getElementById("day")
    const month = document.getElementById("mon")
    const year = document.getElementById("year")
    const hrs = document.getElementById("hrs")
    const min = document.getElementById("min")
    const sec = document.getElementById("sec")
    const week = ["SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT"]
    const mont =
["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV",
"DEC"]

    setInterval(()=>{
    const full_date = new Date()
    date.innerHTML = (full_date.getDate()<10?'0:') + full_date.getDate()
    day.innerHTML = week[full_date.getDay()]
    month.innerHTML = mont[full_date.getMonth()]
    year.innerHTML = full_date.getFullYear()
    hrs.innerHTML = (full_date.getHours()<10?'0:') + full_date.getHours()
    min.innerHTML = (full_date.getMinutes()<10?'0:') + full_date.getMinutes()
    sec.innerHTML = (full_date.getSeconds()<10?'0:') + full_date.getSeconds()

```

```

var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
if(storageObject !== null && storageObject !== undefined && storageObject !== ""){
var storageObjectArr = JSON.parse(storageObject)
if(storageObjectArr && storageObjectArr.length > 0){
for(let i in storageObjectArr){
const dat = new Date(storageObjectArr[i]['taskETA'])
if(full_date.getTime() > dat.getTime()){
const imp = document.getElementById(parseInt(i)+"write2");
imp.style.backgroundColor = "blue"
imp.style.color = "white"
}
}
}
},1000)
}

```

This function reads the current date and time and displays the data in the Calendar modal. Also, by reading the deadline data from the local storage, it can compare it with the current date and time and accordingly change the CSS property to reflect on to the fact that a task's deadline has been met.

- **Step 7:** Test the To-Do List and debug any errors or bugs. Use the browser's console or debugger tools to inspect the code and find any syntax or logical errors. Use different test cases to check if the list works as expected.

- CODE:

(Note: Indents have been removed from the code below so as to fit it in the page)

```
<!DOCTYPE html>
<html>
<head>
<meta charser="utf-8">
<meta name="viewport" content="width=device-width, initial-scale=1">
<title>HTML ToDo List | Major Project</title>
<link href="vendor/bootstrap/css/bootstrap.css" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/bootstrap-
icons/1.11.1/font/bootstrap-icons.min.css" integrity="sha512-
oAvZuuYVzkcTc2dH5z1ZJup50mSQ000qlfRvu0TTiyTBjwX1faoyearj8KdMq0LgsBTHMrRuMek7s+CxF8yE+w=="
crossorigin="anonymous" referrerpolicy="no-referrer" />

<style>

.bi-check-circle-fill{
color: green;
padding-right: 5px;
cursor: pointer;
}
.bi-pencil-square{
color: #2d5dbe;
padding-right: 5px;
cursor: pointer;
}
.bi-trash-fill{
color: red;
padding-left: 5px;
cursor: pointer;
}
.bi-check-circle-fill:hover{
color: lightgreen;
}
.bi-pencil-square:hover{
color: #97b6f3;
}
.bi-star-fill:hover{
color: gold!important
}
.bi-trash-fill:hover{
color:rgb(240, 136, 136)
}
.left{
```

```
width: 50%;
height: 100%;
display: flex;
align-items: center;
justify-content: center;
background: rgb(228, 107, 107);
flex-direction: column;
font-size: 42px;
color:aliceblue;
border-top-left-radius: 20px;
border-bottom-left-radius: 20px;
}

.right{
width: 50%;
height: 100%;
display: flex;
align-items: center;
justify-content: center;
background:#97b6f3;
flex-direction: column;
font-size: 42px;
border-top-right-radius: 20px;
border-bottom-right-radius: 20px;
}

.modal-footer span{
font-size: 42px;
width: 50px;
display: inline-block;
text-align: center;
position: relative;
}

.modal-footer span::after{
font-size: 10px;
color:aliceblue;
position: absolute;
bottom: -8px;
left: 50%;
transform: translateX(-50%);
}

#hrs::after{
content: 'HOURS';
}

#min::after{
content: 'MINUTES';
}

#sec::after{
content: 'SECONDS'
}
```

```

.modal-header,.modal-footer{
background: linear-gradient(45deg, #153677, #4e0f58);
color:aliceblue;
}
.modal-body{
background-color: antiquewhite;
}
#btn2:hover{
background-color: #78407f!important
}
#btn3:hover{
background-color: #85e03a!important
}
#btn1:hover{
background-color: rgb(255, 225, 0)!important;
}
#Inst:hover{
cursor: pointer;
}

</style>

</head>
<body style="background: linear-gradient(135deg, #153677, #4e0f58); width: 100%; min-height: 100vh">
<nav class="navbar navbar-expand-lg navbar-light bg-light">
<div class="container-fluid">
<a href="https://www.1stop.ai/" class="navbar-brand" data-bs-toggle="tooltip" data-bs-
placement="bottom" title="Visit 1stop">

</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">
<a class="nav-item nav-link active" id="Inst" data-bs-toggle="tooltip" data-bs-placement="bottom"
title="To-Do List Instructions" onclick="showCarousel()">Instructions</a>
</div>
</div>
</div>
</nav>

<h1 style="text-align: center; padding-top: 20px; color:antiquewhite">To-Do List</h1>
<div class="container" style="width: 100%; background: antiquewhite; margin: 50px auto 20px;
padding: 40px 30px 70px; border-radius: 10px;">
<div class="mb-3" style="background-color: rgb(240, 69, 69); border-radius: 10px; height: 50px;
padding: 6px 10px 5px">

```

```

<button type="button" class="btn btn-primary" style="border-radius: 20px;"
onclick="showAddTaskModal()">Add New Task</button>
<button type="button" class="btn btn-warning" id="btn1" style="border-radius: 20px;"
onclick="clearAll()">Clear All Tasks</button>
<button type="button" class="btn" id="btn2" style="border-radius: 20px; background-color: #9b54a4;
color: white" onclick="doneAll()">Mark All Done</button>
<button type="button" class="btn" id="btn3" style="border-radius: 20px; background-color: #76ca31;
color: black" onclick="showCalendar()">Show Calendar</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 table-striped">
<thead class="text-center">
<th>#</th>
<th>Task Description</th>
<th>Responsible Person</th>
<th>Due Date&Time</th>
<th>Actions</th>
</thead>
<tbody class="text-center" id="taskTableBody">

</tbody>
</table>

</div>
</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"
style="background-color: aliceblue;"></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="addResponsiblePerson" class="form-label">Responsible Person</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="addTaskResponsible" class="form-label">Due Date&Time</label>
<input type="datetime-local" class="form-control" id="addETA" name="taskETA">
</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"
style="background-color: aliceblue;"></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="Edit your Task Description"></textarea>
</div>
<div class="mb-1">
<label for="addResponsiblePerson" class="form-label">Responsible Person</label>
<input type="text" class="form-control" id="editResponsiblePerson" name="taskResponsiblePerson"
placeholder="Edit the Responsible Person's Name">
</div>
<div class="mb-1">
<label for="addTaskResponsible" class="form-label">Due Date&Time</label>
<input type="datetime-local" class="form-control" id="editETA" name="taskETA">
</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()">Update Task</button>
</div>
</div>
</div>
</form>
</div>

<div class="modal fade" id="showCalendar" data-bs-backdrop="static" data-bs-keyboard="false"
tabindex="-1" aria-labelledby="showCalendarLabel" aria-hidden="true">
<form id="Calendar">
<div class="modal-dialog">
<div class="modal-content">
<div class="modal-header">
<h5 class="modal-title" id="CalendarModalLabel">Calendar</h5>
<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"
style="background-color: aliceblue;"></button>
</div>
<div class="modal-body" style="height: 250px; width: 100%; display:flex; align-items: center;">
<div class="left">
<p id="date"></p>
<p id="day"></p>
</div>
<div class="right">
<p id="mon"></p>
<p id="year"></p>
</div>
</div>
<div class="modal-footer" style="display: flex; justify-content: center;">
<span id="hrs">00</span>
<span>:</span>
<span id="min">00</span>
<span>:</span>
<span id="sec">00</span>
</div>
</div>
</div>
</div>
</form>
</div>

<div class="modal modal-xl fade" id="showCarousel" data-bs-backdrop="static" data-bs-
keyboard="false" tabindex="-1" aria-labelledby="showCarouselLabel" aria-hidden="true">
<form id="To-Do Instructions">
<div class="modal-dialog">
<div class="modal-content">

```

```

<div class="modal-header" style="background: linear-gradient(45deg, #153677, #4e0f58);
color:aliceblue;">
<h5 class="modal-title" id="InstructionsModallabel">To-Do List Instructions</h5>
<button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"
style="background-color:aliceblue"></button>
</div>
<div id="carousel" class="carousel slide" data-bvslide="carousel">
<div class="carousel-indicators">
<button type="button" data-bs-target="#carousel" data-bs-slide-to="0" class="active"></button>
<button type="button" data-bs-target="#carousel" data-bs-slide-to="1"></button>
<button type="button" data-bs-target="#carousel" data-bs-slide-to="2"></button>
<button type="button" data-bs-target="#carousel" data-bs-slide-to="3"></button>
<button type="button" data-bs-target="#carousel" data-bs-slide-to="4"></button>
<button type="button" data-bs-target="#carousel" data-bs-slide-to="5"></button>
<button type="button" data-bs-target="#carousel" data-bs-slide-to="6"></button>
</div>
<div class="carousel-inner">
<div class="carousel-item active">

</div>
<div class="carousel-item">

</div>
<div class="carousel-item">

</div>
<div class="carousel-item">

</div>
<div class="carousel-item">

</div>
<div class="carousel-item">

</div>
<div class="carousel-item">

</div>
</div>
<button class="carousel-control-prev" type="button" data-bs-target="#carousel" data-bs-slide="prev">

```

```

<span class="carousel-control-prev-icon" style="border-radius: 40px; background-color: rgb(196, 164, 226);"></span>
</button>
<button class="carousel-control-next" type="button" data-bs-target="#carousel" data-bs-slide="next">
<span class="carousel-control-next-icon" style="border-radius: 40px; background-color: rgb(196, 164, 226);"></span>
</button>
</div>
<div class="modal-footer" style="display: flex; justify-content: center; background: linear-gradient(45deg, #153677, #4e0f58); color: aliceblue;">
</div>
</div>
</form>
</div>

<script src="https://code.jquery.com/jquery-3.7.1.js" integrity="sha256-eKhayi8LEQwp4NKxN+CfCh+3qOVUtJn3QNZ0TciwLP4=" crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/2.9.2/umd/popper.min.js" integrity="sha512-2rNj2KJ+D8s1ceNasTIex6z4HwyOnEYLVC3FigGOmyQCZc2eBXKgOxQmo3oKlHycj53uz4QMsRCWNbLd32Q1g==" crossorigin="anonymous" referrerpolicy="no-referrer"></script>
<script src="vendor/bootstrap/js/bootstrap.js"></script>

<script>

createHtmlfromStorage();
loader();
Calendar();

function showAddTaskModal(){
$("#addTaskModal").modal('show');
}

function showCalendar(){
$("#showCalendar").modal('show');
}

function showCarousel(){
$("#showCarousel").modal('show');
}

function addTask(){

$("#addTaskModal").modal('hide');
var dataArr = $("#taskInputForm").serializeArray();
var taskObject = new Object();

```

```

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

for(var i in dataArr){
var name = dataArr[i]['name']
var value = dataArr[i]['value']
taskObject[name] = value
}

if(storageObject != null && storageObject != undefined && storageObject != ''){
storageObjectArr = JSON.parse(storageObject)
storageObjectArr.push(taskObject)
}
else{
storageObjectArr.push(taskObject)
}
localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr))

createHtmlfromStorage();
loader();
$("#taskInputForm").trigger('reset')
}

function createHtmlfromStorage(){
var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
var storageObjectArr = JSON.parse(storageObject)
var html = '';

if(storageObject != null && storageObject != undefined && storageObject != ''){
if(storageObjectArr && storageObjectArr.length > 0){
for(let i in storageObjectArr){
var date = new Date(storageObjectArr[i]['taskETA'])
html = html + '<tr id='+parseInt(i)+'write'+ ' style="text-decoration: none;">'
+ '<td id='+parseInt(i)+'write2'+>' + (parseInt(i)+1) + '</td>'
+ '<td>' + storageObjectArr[i]['taskDescription'] + '</td>'
+ '<td>' + storageObjectArr[i]['taskResponsiblePerson'] + '</td>'
+ '<td>' + date.toUTCString() + '</td>'
+ '<td><i class="bi bi-check-circle-fill" data-bs-toggle="tooltip" data-bs-placement="bottom"
title="Mark as Done" onclick="markAsDone('+i+')"></i>'
+ '<i class="bi bi-pencil-square" data-bs-toggle="tooltip" data-bs-placement="bottom" title="Edit
Task" onclick="editTask('+i+')"></i>'
+ '<i id='+i+' class="bi bi-star-fill" data-bs-toggle="tooltip" data-bs-placement="bottom"
title="Mark As Important" onclick="impTask('+i+')" style="cursor: pointer; color: wheat;"></i>'
+ '<i class="bi bi-trash-fill" data-bs-toggle="tooltip" data-bs-placement="bottom" title="Remove
Task" onclick="removeTask('+i+')"></i>'
+ '</td></tr>'
}
}
}

```

```

}
else{
html = '<tr><td colspan="5">No Tasks Added Yet</td></tr>'
}
}
else{
html = '<tr><td colspan="5">No Tasks Added Yet</td></tr>'
}
$("#taskTableBody").html(html)
}

function removeTask(index){

var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
if(storageObject != null && storageObject != undefined && storageObject != ''){
storageObjectArr = JSON.parse(storageObject);
var ar1 = storageObjectArr.slice(0,parseInt(index));
var ar2 = storageObjectArr.slice((parseInt(index)+1));
storageObjectArr = ar1.concat(ar2);
}

localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr));
var attriObj = new Object()
var attriArr = []
attriObj = localStorage.getItem('impData');
attriArr = JSON.parse(attriObj)
var arr1 = attriArr.slice(0,parseInt(index))
var arr2 = attriArr.slice((parseInt(index)+1))
attriArr = arr1.concat(arr2)
localStorage.setItem("impData",JSON.stringify(attriArr))

var attriObj2 = new Object()
var attriArr2 = []
attriObj2 = localStorage.getItem('impData2');
attriArr2 = JSON.parse(attriObj2)
var arr12 = attriArr2.slice(0,parseInt(index))
var arr22 = attriArr2.slice((parseInt(index)+1))
attriArr2 = arr12.concat(arr22)
localStorage.setItem("impData2",JSON.stringify(attriArr2))
createHtmlfromStorage();
loader();
}

function editTask(index){

```

```

var storageObject = localStorage.getItem('taskStorage');
var storageObjectArr = [];
if(storageObject != null && storageObject != undefined && storageObject != ''){
storageObjectArr = JSON.parse(storageObject);
$("#editTaskTextArea").val(storageObjectArr[index]['taskDescription']);
$("#editResponsiblePerson").val(storageObjectArr[index]['taskResponsiblePerson']);
$("#editETA").val(storageObjectArr[index]['taskETA']);
$("#editIndex").val(index);
$("#updateTaskModal").modal('show');
}
}

function updateTask(){
$("#updateTaskModal").modal('hide');
var dataArr = $("#taskUpdateForm").serializeArray();
var taskObject = new Object();
var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');

for(var i in dataArr){
var name = dataArr[i]['name']
var value = dataArr[i]['value']
taskObject[name] = value
}

if(storageObject != null && storageObject != undefined && storageObject != ''){
storageObjectArr = JSON.parse(storageObject)
storageObjectArr[taskObject['taskIndex']] = taskObject
}

localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr))
createHtmlfromStorage();
loader();

}

function clearAll(){

var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr));

var attriArr = []
localStorage.setItem("impData",JSON.stringify(attriArr))
localStorage.setItem("impData2",JSON.stringify(attriArr))

```

```

createHtmlfromStorage();
loader();

}

function doneAll(){
var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
storageObjectArr = JSON.parse(storageObject)
var attriObj = new Object()
var attriArr2 = []
for(var i=0;i<storageObjectArr.length;i=i+1){
attriArr2[i]=1
}
localStorage.setItem('impData2',JSON.stringify(attriArr2))
createHtmlfromStorage();
loader();

}

function impTask(index){
var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
storageObjectArr = JSON.parse(storageObject)
const imp = document.getElementById(index);
var attriObj = new Object()
var attriArr = []
for(var i=0;i<storageObjectArr.length;i=i+1){
attriArr[i]=0
}
attriObj = localStorage.getItem('impData');
if(attriObj != null && attriObj != undefined && attriObj != ''){
attriArr = JSON.parse(attriObj)
}
if(imp.style.color == "wheat"){
imp.style.color = "gold"
attriArr[index] = 1
}
else{
imp.style.color = "wheat"
attriArr[index] = 0
}

localStorage.setItem("impData",JSON.stringify(attriArr))

}

function markAsDone(index){

```



```

var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
storageObjectArr = JSON.parse(storageObject)
const imp = document.getElementById(parseInt(index)+"write");
var attriObj2 = new Object()
var attriArr2 = []
for(var i=0;i<storageObjectArr.length;i=i+1){
attriArr2[i]=0
}
attriObj2 = localStorage.getItem('impData2');
if(attriObj2 != null && attriObj2 != undefined && attriObj2 != ''){
attriArr2 = JSON.parse(attriObj2)
}
if(imp.style.textDecoration == "none"){
imp.style.textDecoration = "line-through"
attriArr2[index] = 1
}
else{
imp.style.textDecoration = "none"
attriArr2[index] = 0
}
localStorage.setItem("impData2",JSON.stringify(attriArr2))
}

function loader(){
var attriObj = new Object()
var attriArr = []
attriObj = localStorage.getItem('impData');
attriArr = JSON.parse(attriObj)

for(var i in attriArr){
if(attriArr[i]==1){
const imp = document.getElementById(i)
imp.style.color = "gold"
}
}
}

var attriObj2 = new Object()
var attriArr2 = []
attriObj2 = localStorage.getItem('impData2');
attriArr2 = JSON.parse(attriObj2)

for(var i in attriArr2){
if(attriArr2[i]==1){
const imp = document.getElementById(parseInt(i)+"write")
imp.style.textDecoration = "line-through"
}
}

```

```

}

}

function Calendar(){

const date = document.getElementById("date")
const day = document.getElementById("day")
const month = document.getElementById("mon")
const year = document.getElementById("year")
const hrs = document.getElementById("hrs")
const min = document.getElementById("min")
const sec = document.getElementById("sec")
const week = ["SUN", "MON", "TUE", "WED", "THU", "FRI", "SAT"]
const mont = ["JAN", "FEB", "MAR", "APR", "MAY", "JUN", "JUL", "AUG", "SEP", "OCT", "NOV", "DEC"]

setInterval(()=>{
const full_date = new Date()
date.innerHTML = (full_date.getDate()<10?'0':'')+full_date.getDate()
day.innerHTML = week[full_date.getDay()]
month.innerHTML = mont[full_date.getMonth()]
year.innerHTML = full_date.getFullYear()
hrs.innerHTML = (full_date.getHours()<10?'0':'')+full_date.getHours()
min.innerHTML = (full_date.getMinutes()<10?'0':'')+full_date.getMinutes()
sec.innerHTML = (full_date.getSeconds()<10?'0':'')+full_date.getSeconds()

var storageObjectArr = [];
var storageObject = localStorage.getItem('taskStorage');
if(storageObject != null && storageObject != undefined && storageObject != ''){
var storageObjectArr = JSON.parse(storageObject)
if(storageObjectArr && storageObjectArr.length > 0){
for(let i in storageObjectArr){
const dat = new Date(storageObjectArr[i]['taskETA'])
if(full_date.getTime() > dat.getTime()){
const imp = document.getElementById(parseInt(i)+"write2");
imp.style.backgroundColor = "blue"
imp.style.color = "white"
}
}
}
}
},1000)

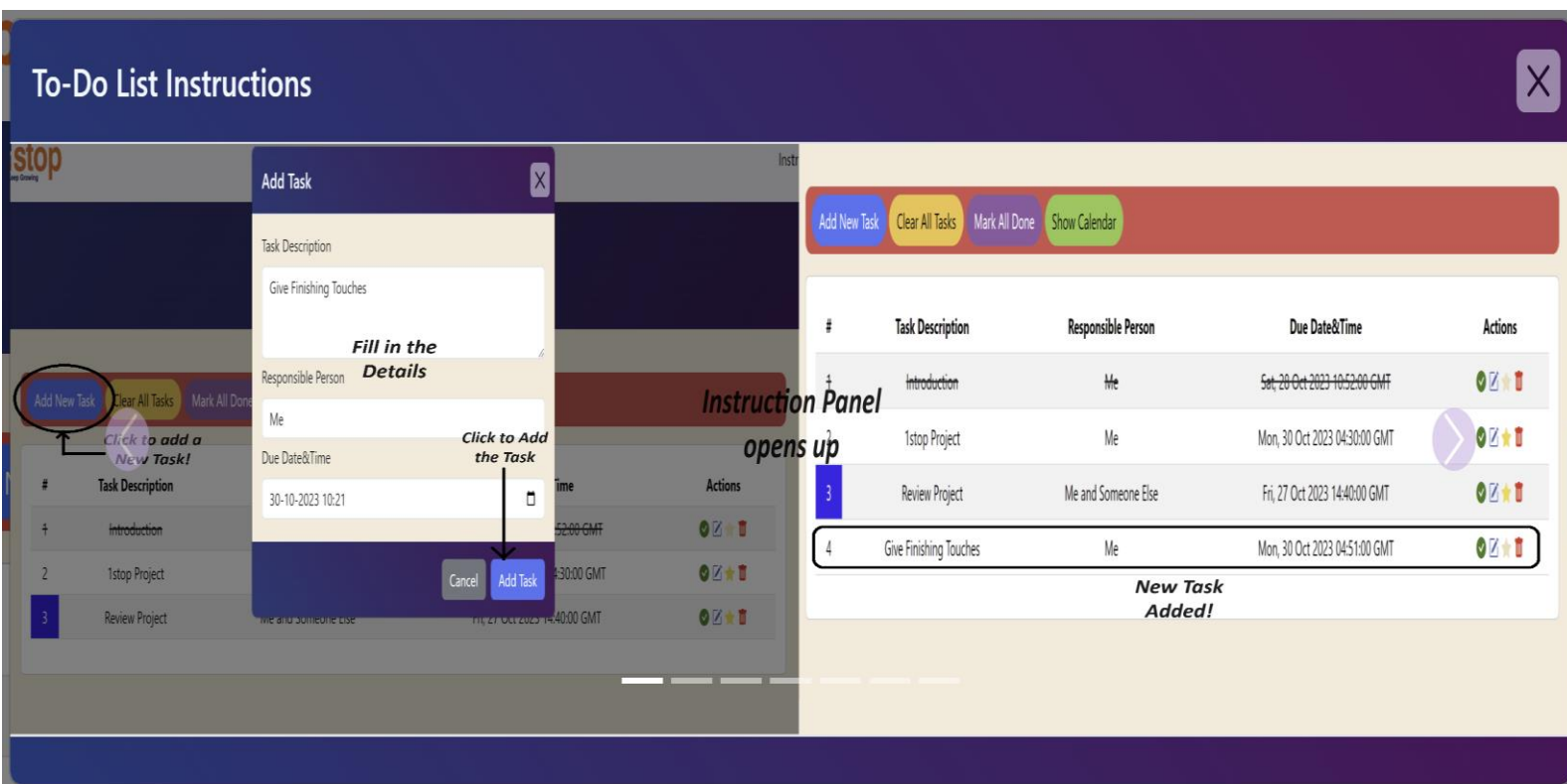
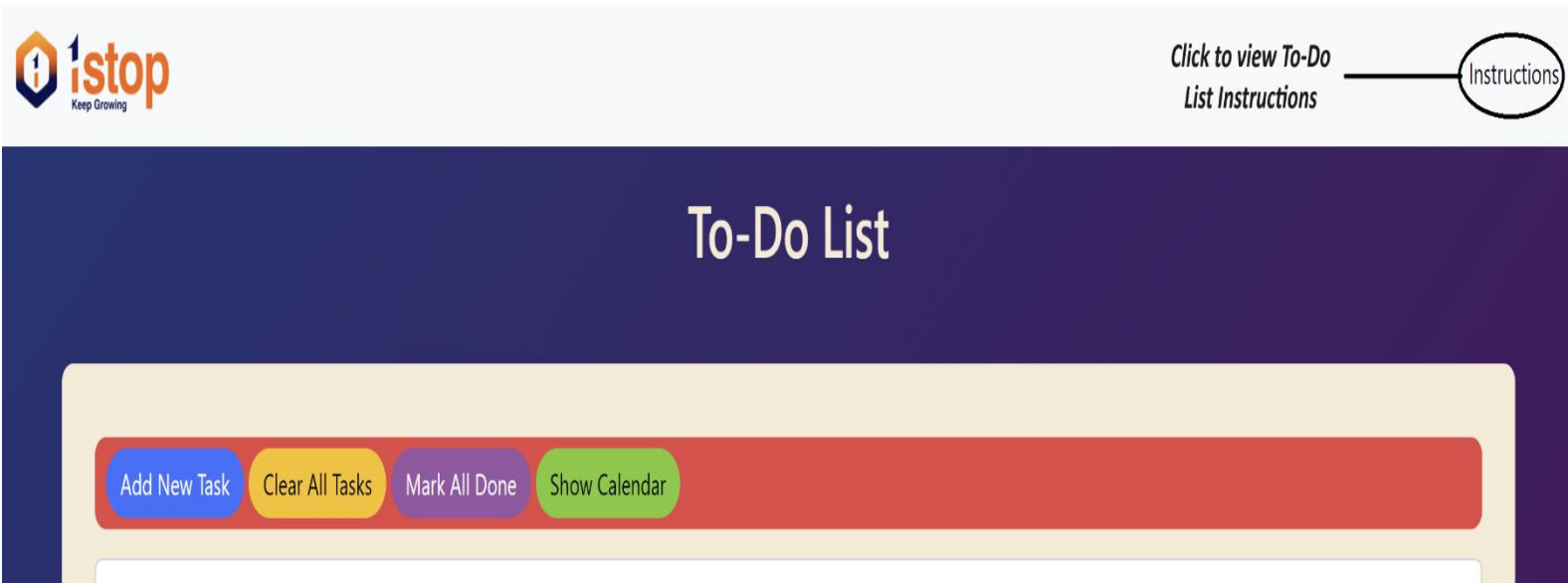
}

</script>
</body>
</html>

```

- RESULTS/OUTPUT:

## Navigation Bar of the To-Do List WebPage




# Features of the To-Do List

## Adding a New Task

Add New TaskClear All TasksMark All DoneShow Calendar

Click to add  
New Task

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   

Add Task

Task Description

Submit Project

Enter details for the Task. Click on Add Task.

Responsible Person

Me

















Due Date&Time

09-11-2023 08:44

Cancel

Add Task

Add New TaskClear All TasksMark All DoneShow Calendar













#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   
4	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

## Mark any Task as Done

[Add New Task](#)[Clear All Tasks](#)[Mark All Done](#)[Show Calendar](#)

#	Task Description	Responsible Person	Due Date&Time	<i>Click to mark Task as Done</i>	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT		  
2	1stop Project	Me	Mon, 30 Oct 2023 09:43:00 GMT		  
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT		  

[Add New Task](#)[Clear All Tasks](#)[Mark All Done](#)[Show Calendar](#)

#	Task Description	Responsible Person	Due Date&Time	<i>Selected Task marked as Done</i>	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT		  
2	1stop Project	Me	Mon, 30 Oct 2023 09:43:00 GMT		  
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT		  

## Edit any Task Details

[Add New Task](#)[Clear All Tasks](#)[Mark All Done](#)[Show Calendar](#)

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	<a href="#">Click to edit the Task</a>    
2	1stop Project	Me	Mon, 30 Oct 2023 09:43:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   

### Edit Task



Task Description

Finalize 1stop Project

**Enter new details for  
the Selected Task.  
Click Update Task.**













Responsible Person

Me

Due Date&Time


31-10-2023 10:22

[Cancel](#)[Update Task](#)[Add New Task](#)[Clear All Tasks](#)[Mark All Done](#)[Show Calendar](#)

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   

# Mark any Task as Important

Add New Task Clear All Tasks Mark All Done Show Calendar

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	1stop Project	Me	Mon, 30 Oct 2023 09:43:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   

Add New Task Clear All Tasks Mark All Done Show Calendar

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	1stop Project	Me	Mon, 30 Oct 2023 09:43:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   













Remove any Task

Add New Task Clear All Tasks Mark All Done Show Calendar

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   
4	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

Click to remove  
the Task

Add New Task Clear All Tasks Mark All Done Show Calendar

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

***Selected Task  
Removed***



## Auto Task Deadline Detector and Marker

Add New Task

Clear All Tasks

Mark All Done

Show Calendar













*Tasks marked blue have  
crossed the deadline*

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introduction	Me	Sat, 28 Oct 2023 03:00:00 GMT	   
2	Testing the Project	Me	Sat, 28 Oct 2023 14:57:00 GMT	   
3	Finalize and Debug	Me	Sun, 29 Oct 2023 04:00:00 GMT	   
4	Submit the Project	Me	Mon, 30 Oct 2023 06:02:00 GMT	   

Remove all of the Tasks from the List

Add New TaskClear All TasksMark All DoneShow Calendar

Click to remove  
All Tasks

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

Add New TaskClear All TasksMark All DoneShow Calendar

#	Task Description	<div>All of the Tasks have been Removed</div>	Responsible Person	Due Date&Time	Actions
No Tasks Added Yet					

## Marks all of the Tasks as Done







Add New Task

Clear All Tasks

Mark All Done

Show Calendar

Click to mark all  
Tasks as Done

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   
4	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

















Add New Task

Clear All Tasks

Mark All Done

Show Calendar

All Tasks are  
marked as Done

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   
4	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

In-Built Calendar that displays Current Date and Time

















Add New Task

Clear All Tasks

Mark All Done

Show Calendar

Click to view  
Calendar

#	Task Description	Responsible Person	Due Date&Time	Actions
1	Introductions	Me	Sun, 29 Oct 2023 02:35:00 GMT	   
2	Finalize 1stop Project	Me	Tue, 31 Oct 2023 04:52:00 GMT	   
3	Review Project	Me and Someone Else	Mon, 30 Oct 2023 15:41:00 GMT	   
4	Submit Project	Me	Thu, 09 Nov 2023 03:14:00 GMT	   

Calendar

28

SAT

OCT

2023

Opens up a Calendar  
that displays current  
Date and Time

22

:

27

:

27

HOURS

MINUTES

SECONDS

- CONCLUSION:

- In this project, I have created a simple and interactive web application that allows users to create and manage their personal to-do lists. The application has the following features:
  - Users can add new tasks by clicking on the Add New Task button, entering in the task data in the Add Task window, and clicking the Add Task button.
  - Users can mark tasks as completed by clicking on the checkbox next to each task. Completed tasks are crossed out.
  - Users can edit tasks by clicking on the edit icon next to each task and making changes in the data that appears in the Edit Task window. Users can save their changes by clicking the Update Task button.
  - Users can mark tasks as important by clicking on the star icon next to each task which turns yellow to mark any task as important.
  - Users can delete tasks from their list by clicking on the trash icon next to each task.
  - Users can clear all the tasks from their list by clicking on the Clear All Tasks button at the header of the list.
  - Users can mark all the tasks as done in their list by clicking on the Mark All Done button at the header of the list.
  - Users can see tasks that have crossed the deadline as they will be highlighted or marked blue.
  - User can open up a Calendar that shows current Date and Time by clicking on the Show Calendar button from the header of the list.

- The To-Do List web application is built using HTML, CSS, JavaScript, jQuery, Bootstrap, and Popper.js. HTML is used to define the structure and content of the Webpage and To-Do List. CSS is used to style and format the webpage elements and the list. Bootstrap is a CSS framework that provides responsive design and pre-built components for web development. I have used Bootstrap to style and position the elements, Bootstrap grid system and components to create a responsive layout, Bootstrap typography and colors to create a consistent and attractive design, Bootstrap icons and utilities to add some visual elements, Bootstrap carousels to add some interactivity and functionality. JavaScript is used to add interactivity and functionality to the List along with jQuery, a JavaScript library that simplifies DOM manipulation and event handling. I have used Popper.js, a JavaScript library that helps in creating, positioning, and customizing tooltips. I have also used some custom CSS styles to customize the appearance of the website, and override or modify some of the Bootstrap default styles. All these are used to design a well-polished and functional To-Do list which looks professional and works without any flaws and as expected.
- The main challenges of this project were to design a layout that is suitable for a webpage-based To-Do List(application), to organize the content in a clear and concise manner, to ensure the functionality of the carousel, and the tooltip, to optimize the webpage for different devices and browsers, and most importantly, to design and code the brains and looks of the To-Do List in the webpage. I have overcome these challenges by using Bootstrap grid system and components to create a responsive layout, by using headings, paragraphs, lists, images, icons, buttons, cards, carousels, and tooltips to present the To-Do List in an attractive and professional way, by using jQuery selectors and methods to target and manipulate the elements dynamically, by using jQuery events and callbacks to handle the user actions and responses, by using Popper.js instances and options to create and customize the tooltips, and

by using flexbox properties, meta tags, title attributes, and keywords to enhance the responsiveness, accessibility, and SEO-friendliness of the webpage itself.

- The main limitations of this project were that the To-Do List does not have any server-side database interaction, and that it does not have any advanced features, such as animations, transitions, or sliders. These limitations can be addressed by adding more languages, such as PHP, MySQL, or MongoDB, and by adding more libraries or frameworks, such as Animate.css, Swiper.js, or Slick.js, to add more visual effects and functionality.
- The project demonstrates how to use various web technologies to create a user-friendly and dynamic web application. The project also shows how to apply basic principles of web design such as layout, color, typography, and usability. The project can be further improved by adding more features such as:
  - Allowing users to sort their tasks by priority, due date, or alphabetical order.
  - Allowing users to create multiple lists for different categories or projects.
  - Allowing users to sync their lists with a database or a cloud service so that they can access them from any device or browser.
  - Allowing users to share their lists with other users or export them as PDF or CSV files
- The main outcomes of this project were that I have successfully created a professional, and functional To-Do List using HTML, CSS, JavaScript, Bootstrap, jQuery and Popper.js, that works as expected, and that I have improved my web development skills and knowledge.