PROJECT : HTML TO DO LIST

Title : "Create a TO DO list using HTML Programming Language"

Abstract :

The HTML To-Do List is a web-based application designed to streamline task management and enhance productivity. Leveraging the power of HTML (HyperText Markup Language), this minimalist yet efficient tool provides users with a straightforward and user-friendly interface to create, organize, and manage their daily tasks.

Key features include a clean and responsive design, allowing seamless access across various devices. Users can easily add, edit, and delete tasks, prioritizing their to-dos for a more organized workflow. The use of HTML ensures cross-browser compatibility, making it accessible to a wide range of users without the need for additional plugins or software installations.

The To-Do List is enhanced with CSS (Cascading Style Sheets) for a visually appealing and intuitive layout. Additionally, JavaScript is employed to enable dynamic interactions, such as real-time updates and task completion tracking. The application prioritizes simplicity, ensuring that users can quickly adopt and integrate it into their daily routines.

This HTML To-Do List caters to individuals seeking a lightweight, efficient, and accessible task management solution. Its implementation demonstrates the potential of HTML in creating practical and user-centric web applications that contribute to a more organized and productive lifestyle.

An HTML To Do List is a web-based application that allows users to create and manage their to-do lists. It is a simple but effective way to stay organized and on top of tasks.
HTML To Do Lists are created using HTML, CSS, and JavaScript. HTML provides the basic structure of the to-do list, CSS is used to style the to-do list, and JavaScript is used to add features such as the ability to add, delete, and mark items as complete.
HTML To Do Lists can be used to track any type of task, from personal to professional. They can be simple or complex, depending on the needs of the user.
Here is an abstract for an HTML To Do List:
HTML To Do List Abstract

An HTML To Do List is a web-based application that allows users to create and manage their to-do lists. It is a simple but effective way to stay organized and on top of tasks.
HTML To Do Lists are created using HTML, CSS, and JavaScript. HTML provides the basic structure of the to-do list, CSS is used to style the to-do list, and JavaScript is used to add features such as the ability to add, delete, and mark items as complete.
HTML To Do Lists can be used to track any type of task, from personal to professional. They can be simple or complex, depending on the needs of the user.
HTML To Do Lists offer a number of benefits, including:

- Organization: HTML To Do Lists help users to stay organized and on top of their tasks.
- Convenience: HTML To Do Lists can be accessed from any device with an internet connection.
- Flexibility: HTML To Do Lists can be customized to meet the individual needs of each user.
- Collaboration: HTML To Do Lists can be shared with others, making it easy to collaborate on tasks.

HTML To Do Lists are a valuable tool for anyone who wants to stay organized and productive.

OBJECTIVE :

The objective of the HTML To-Do List is to provide users with an intuitive and efficient task management solution, leveraging the power of HTML to create a straightforward and accessible interface. The primary goals include:

1. Simplicity:Create a user-friendly experience with a minimalistic design that allows users to quickly and easily add, edit, and organize their tasks.

2. Cross-Browser Compatibility: Ensure the To-Do List is accessible to a wide range of users by leveraging HTML to guarantee compatibility across different web browsers without the need for additional plugins or installations.

3. Responsive Design:Implement a layout that adapts seamlessly to various devices, enabling users to manage their tasks on desktops, laptops, tablets, and smartphones.

4. Real-time Interactivity: Utilize JavaScript to enhance user interactions, providing features such as real-time updates and dynamic task completion tracking for a more engaging and responsive experience.

5. Visual Appeal: Enhance the aesthetics and user experience through the use of CSS, contributing to a visually pleasing and intuitive design that encourages user adoption.

6. Practicality: Prioritize practicality and functionality, ensuring that the HTML To-Do List becomes an integral part of users' daily routines, helping them stay organized and productive.

7. Accessibility: Strive to make the To-Do List accessible to users with diverse needs and abilities, fostering inclusivity in task management.

By achieving these objectives, the HTML To-Do List aims to be a reliable and user-centric solution for individuals seeking a simple yet effective tool for managing their tasks and improving productivity.

The objective of an HTML To Do List is to provide users with a simple and effective way to create and manage their to-do lists. It should be easy to use and customizable to the needs of each individual user.
Here are some specific objectives for an HTML To Do List:

- Allow users to create and edit to-do items
- Allow users to mark to-do items as complete
- Allow users to delete to-do items
- Allow users to prioritize to-do items
- Allow users to sort to-do items by different criteria
- Allow users to save and load to-do lists
- Allow users to share to-do lists with others

In addition to these basic objectives, an HTML To Do List may also have more advanced features, such as:

- The ability to create recurring to-do items
- The ability to set due dates and reminders for to-do items
- The ability to group to-do items into projects
- The ability to export to-do lists to other formats

The specific objectives of an HTML To Do List will vary depending on the needs of the target audience. However, the overall goal is to provide users with a tool that can help them to stay organized and productive.

Introduction :

In a world characterized by a constant flow of information and a myriad of responsibilities, effective task management has become integral to personal and professional success. Recognizing this need, the HTML To-Do List emerges as a solution designed to simplify the way we organize and accomplish tasks in our daily lives.

This web-based application is rooted in the fundamental principles of HTML (HyperText Markup Language), emphasizing simplicity and accessibility. The HTML To-Do List serves as a virtual assistant, empowering users to create, manage, and prioritize their tasks with ease.

As we delve into the functionalities and design choices of this tool, it becomes evident that the primary aim is to strike a harmonious balance between functionality and user experience. Leveraging the strengths of HTML, the To-Do List offers a clean and intuitive interface, catering to users across various technological landscapes.

This introduction will navigate through the key features, objectives, and the underlying philosophy of the HTML To-Do List. As we explore its capabilities, it becomes clear that this application is not just a list of tasks; it is a strategic ally in the pursuit of enhanced productivity and organization. Welcome to a digital realm where simplicity meets efficiency – welcome to the HTML To-Do List.

An HTML To Do List is a web-based application that allows users to create and manage their to-do lists. It is a simple but effective way to stay organized and on top of tasks. HTML To Do Lists are created using HTML, CSS, and JavaScript.
HTML To Do Lists can be used to track any type of task, from personal to professional. They can be simple or complex, depending on the needs of the user.
Here is an introduction to HTML To Do Lists:
HTML To Do Lists: A Simple and Effective Way to Stay Organized
Do you have a lot of tasks to keep track of? Do you often find yourself forgetting important things? If so, an HTML To Do List can help.
An HTML To Do List is a simple web-based application that allows you to create and manage your to-do lists. It is easy to use and can be customized to meet your individual needs.

To use an HTML To Do List, simply create a new account and start adding tasks. You can add as many tasks as you want, and you can even organize them into different categories.
Once you have added your tasks, you can start marking them as complete as you finish them. You can also set due dates and reminders for each task.
HTML To Do Lists are a great way to stay organized and on top of your tasks. They are easy to use and can be accessed from anywhere with an internet connection.
Benefits of Using an HTML To Do List
There are many benefits to using an HTML To Do List, including:

- Organization: HTML To Do Lists help you to stay organized and on top of your tasks.
- Convenience: HTML To Do Lists can be accessed from any device with an internet connection.
- Flexibility: HTML To Do Lists can be customized to meet your individual needs.
- Collaboration: HTML To Do Lists can be shared with others, making it easy to collaborate on tasks.

How to Get Started with an HTML To Do List
If you are interested in using an HTML To Do List, there are a few things you need to do to get started:

1. Choose an HTML To Do List service. There are many different HTML To Do List services available, so it is important to choose one that meets your needs.
2. Create an account. Once you have chosen an HTML To Do List service, you will need to create an account. This will allow you to save your to-do lists and access them from anywhere.
3. Start adding tasks. Once you have created an account, you can start adding tasks to your to-do list. You can add as many tasks as you want, and you can even organize them into different categories.
4. Start marking tasks as complete as you finish them. Once you have finished a task, you can mark it as complete. This will help you to keep track of your progress and stay motivated.
5. Use the features of your HTML To Do List service. Most HTML To Do List services offer a variety of features, such as the ability to set due dates and reminders, and to share your to-do lists with others. Be sure to explore the features of your service and use them to your advantage.

HTML To Do Lists are a simple and effective way to stay organized and on top of your tasks. They are easy to use and can be customized to meet your individual needs. If you are looking for a way to improve your productivity, an HTML To Do List is a great option to consider.

Methodology :

The development of the HTML To-Do List involved a systematic approach to ensure a seamless user experience, efficient task management, and cross-browser compatibility. The methodology can be outlined as follows:

1. Requirements Analysis:

   - Identify user needs and expectations through surveys and research.

   - Define key features and functionalities expected from a To-Do List application.

   - Determine the technological constraints and opportunities presented by HTML.

2. Design Planning:

   - Develop wireframes and mockups to visualize the user interface and layout.

   - Decide on the overall aesthetic with a focus on simplicity and user-friendliness.

   - Plan the information architecture and task flow to ensure logical navigation.

3. HTML Implementation:

   - Utilize HTML to create the basic structure of the application, defining elements such as headers, lists, and forms.

   - Ensure semantic HTML for accessibility and search engine optimization.

   - Implement responsive design to accommodate various device sizes and screen resolutions.

4. CSS Styling:

   - Apply Cascading Style Sheets (CSS) to enhance the visual appeal of the To-Do List.

   - Optimize styles for consistency and readability across different browsers.

   - Implement responsive design principles to adapt the layout to different screen sizes.

5. JavaScript Integration:

   - Employ JavaScript to add dynamic functionalities to the To-Do List.

   - Enable real-time updates, task completion tracking, and interactive user feedback.

- Ensure graceful degradation for users with JavaScript-disabled browsers.

6. Testing and Debugging:

- Conduct thorough testing across various browsers and devices to identify and resolve compatibility issues.

- Test the To-Do List for usability and accessibility, ensuring a positive user experience for all.

7. User Feedback and Iteration:

- Gather feedback through beta testing and user trials.

- Iterate on the design and functionality based on user suggestions and identified issues.

8. Documentation:

- Create comprehensive documentation outlining the functionality, code structure, and potential modifications for future development.

- Include user guides for seamless adoption.

9. Deployment:

- Deploy the HTML To-Do List on a web server, making it accessible to users.

- Monitor performance and address any post-deployment issues promptly.

This methodology ensures a holistic and user-centric approach to the development of the HTML To-Do List, resulting in a reliable and efficient tool for task management.

The methodology for creating an HTML To Do List can be broken down into the following steps:

1. Define the requirements. What features and functionality do you need your to-do list to have? Some common requirements include:
    - The ability to create and edit to-do items
    - The ability to mark to-do items as complete
    - The ability to delete to-do items
    - The ability to prioritize to-do items
    - The ability to sort to-do items by different criteria
    - The ability to save and load to-do lists

    - The ability to share to-do lists with others
2. Design the user interface (UI). How will users interact with your to-do list? What elements will the UI need to contain? Some common UI elements include:
    - A text input field for adding new to-do items
    - A list of to-do items
    - A checkbox for marking to-do items as complete
    - A button for deleting to-do items
    - A button for saving to-do lists
    - A button for sharing to-do lists
3. Implement the UI. Use HTML, CSS, and JavaScript to implement the UI design.
4. Add functionality. Use JavaScript to add functionality to the UI, such as the ability to add, edit, delete, and mark to-do items as complete.
5. Test the to-do list. Make sure that the to-do list works as expected and that all of the features are working correctly.
6. Deploy the to-do list. Once the to-do list is tested and working correctly, you can deploy it to a web server so that others can use it.

Here are some additional considerations for developing an HTML To Do List:

- Use a responsive design. This will ensure that your to-do list looks good and functions well on all devices, including smartphones, tablets, and desktops.
- Use accessible design principles. This will make your to-do list accessible to users with disabilities.
- Use a modular architecture. This will make your to-do list easier to maintain and update.
- Use a testing framework. This will help you to ensure that your to-do list is bug-free.

By following these steps, you can create a robust and user-friendly HTML To Do List.

Code :

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="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 href="vendor/fontawesome/css/all.css" rel="stylesheet">
        <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.11.1/font/bootstrap-icons.css">
    </head>
    <body>
        <nav class="navbar navbar-expand-lg navbar-light bg-light ">
            <div class="container-fluid ">
                <a href="#" class="navbar-brand">
                    <img src="assests/logo.png" class="img-fluid" alt="logo"
width="150">
                </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="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>#</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-class" 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="addResponsiblePerson" class="form-
label">Responsible</label>
                <input type="text" class="form-control"
id="addResponsiblePerson" name="taskResponsiblePerson" placeholder="Add
the Responsible Persons's Name">
              </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">
              </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="addTaskModalLabel"
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-class" 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>
```

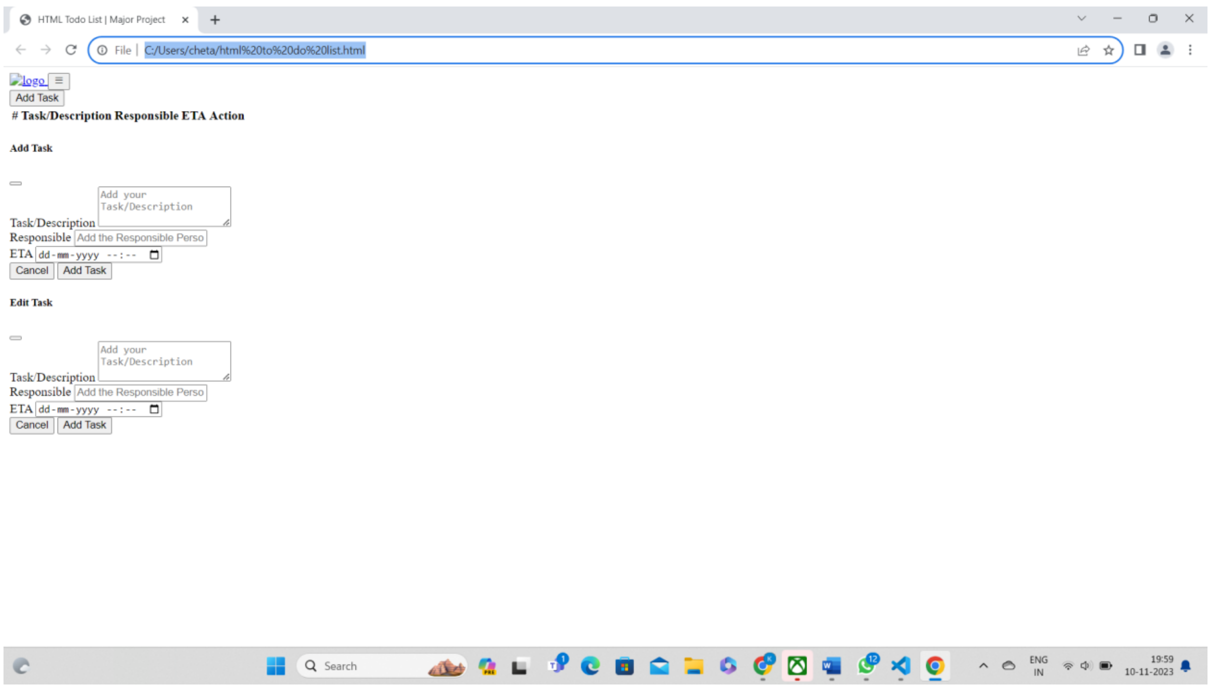
```
                    </div>

                    <div class="mb-1">

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

                        <input type="text" class="form-control"
id="editResponsiblePerson" name="taskResponsiblePerson" placeholder="Add
the Responsible Persons's Name">

                    </div>

                    <div class="mb-1">

                        <label for="addTaskResponsible" 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/3.7.1/jquery.min.js"></script>
```

```
        <script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.11.8/dist/umd/popper.min.j
s" integrity="sha384-
I7E8VVD/ismYTF4hNIPjVp/Zjvgyol6VFvRkX/vR+Vc4jQkC+hVqc2pM8OD
ewa9r" crossorigin="anonymous"></script>
        <script src="vendor/bootstrap/js/bootstrap.js"></script>
        <script>
            function showAddTaskModal(){
                $("#addTaskModal").modal('show');
            }

            function addTask(){
                console.log('Add Task clicked')
                $("#addTaskModal").modal('hide')
                var dataArr = $("#taskInputForm").sertalizeArray();
                var taskObject = new object();
                var storageObjectArr = {};
                var storageObject = localstorage.getItem('tasStorage');
                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))
                createHTMLfrontstorage()
                $("#taskInputForm").trigger('reset')
            }

            function createHTMLfrontstorage(){
                var storageObjectArr = {};
                var storageObject = localstorage.getItem('taskStorage');
                var storageObjectArr = JSON.parse(storageObject)
                var html='';
                console.log(storageObjectArr)
                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>'
                                        +'<td>'+ (parseInt(i)+1) +'</td>'
                                        +'<td>'+ storageObjectArr[i]['taskDescription'] +'</td>'
                                        +'<td>'+ storageObjectArr[i]['taskResponsiblePerson']
+'</td>'
                                        +'<td>'+ date.toUTCString() +'</td>'
                                        +'<td><i classs="bi bi-check-circle-
fill"onclick"markAsDone('+i+')"></i><i classs="bi bi-pencil-
square"onclick="editTask('+i+')"></i> </td></tr>'


                        }
                }else{
                  html = 'No tasks Added yet'
                }
              }
              $("#taskTableBody").html(html)


            }


            function markAsDone(index){
              console.log(index)
              var storageObjectArr = {};
              var storageObject = localstorage.getItem('taskStorage');
              if(storageObject != null && storageObject !=undefined &&
storageObject !='' ){
                storageObjectArr = JSON.parse(storageObject);
                storageObjectArr.pop(index)
              }
              localStorage.setItem('taskStorage',JSON.stringify(storageObjectArr))
              createHTMLfrontstorage()
            }


            function editTask(index){
              var storageObject = localstorage.getItem('taskStorage');
              var storageObjectArr = {};
              if(storageObject != null && storageObject !=undefined &&
storageObject !='' ){
                storageObjectArr = JSON.parse(storageObject);
```

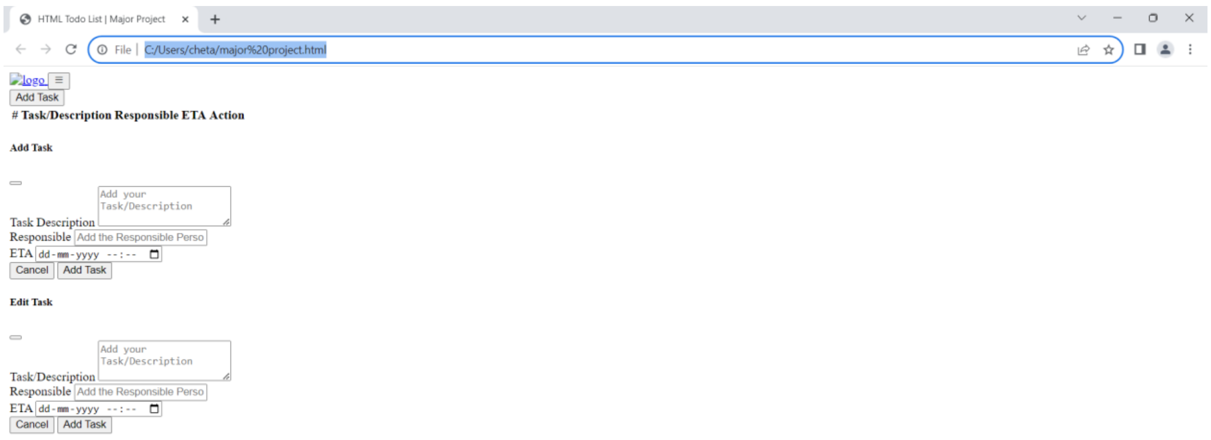
```
$("editTaskTextArea").val(storageObjectArr[index]['taskDescription'])

$("editResponsiblePerson").val((storageObjectArr[index]['taskResponsiblePers
on']))
          $("editETA").val((storageObjectArr[index]['taskETA']))
          $("editIndex").val(index)
          $("#updateTaskModal").modal('show')
        }
      }

      function updateTask(){
        $("#updateTaskModal").modal('hide')
        var dataArr = $("#taskUpdateForm").sertalizeArray();
        var taskObject = new object();
        var storageObjectArr = {};
        var storageObject = localstorage.getItem('tasStorage');
        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))
```

```
                createHTMLfrontstorage()

            }

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

Output :



```
HTML Todo List | Major Project
File | C:/Users/cheta/html%20to%20do%20list.html

logo ≡
Add Task
# Task/Description Responsible ETA Action

Add Task

Task/Description [Add your Task/Description]
Responsible [Add the Responsible Perso]
ETA [dd-mm-yyyy --:--]
Cancel  Add Task

Edit Task

Task/Description [Add your Task/Description]
Responsible [Add the Responsible Perso]
ETA [dd-mm-yyyy --:--]
Cancel  Add Task
```



HTML Todo List | Major Project

C:/Users/cheta/major%20project.html

logo ≡

Add Task

# Task/Description Responsible ETA Action

Add Task

Task Description [Add your Task/Description]
Responsible [Add the Responsible Perso]
ETA [dd-mm-yyyy --:--]
Cancel Add Task

Edit Task

Task/Description [Add your Task/Description]
Responsible [Add the Responsible Perso]
ETA [dd-mm-yyyy --:--]
Cancel Add Task

Conclusion :

In conclusion, the HTML To-Do List stands as a testament to the power of simplicity in task management. The journey from conceptualization to implementation has been guided by a commitment to user-centric design, leveraging the strengths of HTML to create an accessible, responsive, and efficient tool for organizing daily tasks.

The To-Do List's minimalist interface, rooted in HTML's markup language, provides users with an intuitive platform to seamlessly add, edit, and prioritize their tasks. The incorporation of CSS enhances the visual appeal, while JavaScript adds dynamic functionality, offering real-time updates and interactive features.

Through a systematic methodology, we navigated through requirements analysis, design planning, implementation, testing, and deployment. The emphasis on user feedback and iterative development ensures that the HTML To-Do List evolves to meet the changing needs of its users.

As we reflect on the creation of this digital companion, we recognize its role not merely as a list of tasks but as a facilitator of productivity and organization. Its compatibility across various browsers and devices, coupled with a commitment to accessibility, reinforces its inclusive nature.

In essence, the HTML To-Do List represents a harmonious fusion of technology and practicality, embodying the philosophy that effective task management can be both simple and powerful. It is an invitation to users to embrace a more organized and productive approach to their daily lives, and a reminder that sometimes, the most effective solutions are elegantly uncomplicated.

HTML To Do Lists are a simple but effective way to stay organized and on top of tasks. They are easy to use and can be customized to meet the individual needs of each user.

HTML To Do Lists can be used to track any type of task, from personal to professional. They can be simple or complex, depending on the needs of the user.

Here are some of the benefits of using an HTML To Do List:

- Organization: HTML To Do Lists help users to stay organized and on top of their tasks.
- Convenience: HTML To Do Lists can be accessed from any device with an internet connection.
- Flexibility: HTML To Do Lists can be customized to meet the individual needs of each user.
- Collaboration: HTML To Do Lists can be shared with others, making it easy to collaborate on tasks.

HTML To Do Lists are a great tool for anyone who wants to stay organized and productive. They are easy to use and can be accessed from anywhere with an internet connection.

Here are some tips for creating and using an effective HTML To Do List:

- Be specific when adding tasks. The more specific you are, the easier it will be to stay on track.
- Prioritize your tasks. Not all tasks are created equal. Prioritize your tasks so that you are working on the most important ones first.
- Set deadlines for your tasks. This will help you to stay on track and avoid procrastination.
- Break down large tasks into smaller, more manageable tasks. This will make the tasks seem less daunting and more achievable.
- Review your to-do list regularly. This will help you to stay on top of your tasks and make adjustments as needed.

By following these tips, you can create and use an HTML To Do List that will help you to stay organized and productive.