1stop.ai

PROJECT TITLE:

HTML TO-DO LIST

Presented by:

Charla Chandana Reddy

TABLE OF CONTENTS

ABSTRACT

In our busy daily lives, keeping track of tasks and responsibilities can sometimes be difficult. Whether it's remembering assignments, personal goals, or small day-to-day tasks, we often forget things if they're not written down. That's why I created this **HTML To-Do List** a basic but useful web application that helps users organize and manage their tasks easily.

The **HTML To-Do List** is a simple project built using **HTML, CSS, and JavaScript**. It allows users to add new tasks, mark them as complete, and delete them once they're done. It works directly in a web browser, so there's no need to download or install anything. The design is clean and user-friendly, making it perfect for anyone students, professionals, or homemakers to use without any technical skills.

The main purpose of this project is to **help users stay organized** by listing their tasks in one place. It improves productivity by giving a visual reminder of what needs to be done. The project also gave me hands-on experience in web development and taught me how to build interactive web pages using JavaScript.

From a technical point of view, the structure of the page is created with HTML, the styling is done using CSS, and the interactivity like adding or removing tasks is powered by JavaScript. The tasks are stored temporarily while the page is open, and the code can be extended in the future to include features like saving tasks permanently using local storage, setting deadlines, or even reminders.

In conclusion, the **HTML To-Do List** is a small yet practical tool that solves a real-world problem. It shows how basic web technologies can be used to build something useful, simple, and effective. This project not only helps users stay on top of their tasks but also encourages better time management in everyday life.

OBJECTIVE

The main objective of my project, the **HTML To-Do List**, is to help people stay organized and manage their tasks better using a very simple and easy-to-use web tool. Let's be honest all of us, at some point, forget things we're supposed to do. Whether it's a homework deadline, a household chore, or even just remembering to call someone back, we often get so busy or distracted that we don't keep track of everything. That's exactly the kind of problem I wanted to solve with this project.

So, I created a basic **To-Do List** app using just **HTML, CSS, and JavaScript**. No complicated software, no logins, no confusion just a straightforward list where you can write down your tasks, check them off when they're done, and remove them when you no longer need them. The objective here is to keep things **simple, fast, and functional**, especially for people who just want a place to organize their day.

One big reason I chose this project is because it's **relatable to everyone**. Whether you're a student, a working professional, or someone managing daily routines at home, having a to-do list can make life feel a little more under control. It feels good to see what tasks you've completed it motivates you to do more, and helps you avoid forgetting important things.

From a technical point of view, my goal was to use basic web development skills to create something useful. I wanted to practice using **HTML** for structure, **CSS** for design, and **JavaScript** to make the list work like adding new tasks, marking them as complete, and deleting them. So not only does this project solve a real-life issue, but it also helped me improve my coding skills.

Another important objective was to make the design **user-friendly and clean**. I didn't want too many buttons or options that might confuse people. Everything is kept neat and simple, with just the main features someone would need like an input box to add a task, a button to add it to the list, and checkboxes or delete options to manage tasks easily.

Looking ahead, this project also has room for more features, like:

- Saving tasks using browser storage (so they don’t disappear when the page refreshes)
- Adding deadlines or reminders
- Categorizing tasks based on importance or type (school, personal, work, etc.)

So overall, the objective of this project is not just to make a task list, but to build a small tool that makes life a bit easier, helps people stay on track, and shows how even **simple code can solve everyday problems**. It’s practical, useful, and something that almost anyone can relate to and benefit from.

INTRODUCTION

Day by day, life gets busy. Whether we're managing school assignments, personal goals, or even simple daily routines, it's easy to forget things when we don't note them down. Sometimes we say, "I'll remember it," but by the end of the day, it's gone. That's exactly why **to-do lists** are such a lifesaver. They help us organize our thoughts, plan our day, and track what we've done and what's left.

The project I've built is called the **HTML To-Do List**, and as the name suggests, it's a simple task manager that helps users list their daily work or activities. I've used **HTML, CSS, and JavaScript** to build it, which are the three core technologies for creating websites. What's cool about this project is that it runs directly in the web browser — no installation, no login, no database setup — just open the file and use it. It's made for people who just want a clean, basic list without all the complications of big productivity apps.

Now let me explain **why I chose this project**. First of all, everyone can relate to it. Students like me often have multiple assignments, exam dates, and personal goals to keep track of. Teachers have schedules and deadlines. Even at home, we have chores or shopping lists. So this kind of tool can help anyone stay organized. Secondly, from a technical point of view, I wanted to build something that uses what I've learned in web development — and a to-do list was the perfect choice because it covers **HTML structure, CSS styling, and JavaScript logic** in a simple way.

Here's how the to-do list works: you just type in a task in the input box and hit the "Add" button. The task gets added to the list right below. You can then mark the task as completed, and once you're done, you can also delete it. It's really simple — which is what makes it useful. I didn't want to add too many buttons or options that would confuse people. The idea was to keep it clean, neat, and straightforward.

From the coding side, **HTML** helped me set up the basic layout — like the input field, add button, and the list where tasks show up. **CSS** helped me make it look good — I added colors, spacing, and responsive design so that it looks nice on both phones and computers. Then came **JavaScript**, which is what gave the to-do list its real functionality. It handles adding tasks, marking them as done, and removing them. It's the part that makes everything "work" when you interact with it.

I also tried to follow a minimal design using **Bootstrap**, which made the UI look a bit more modern and responsive. While the project doesn’t store data permanently (like if you refresh the page, the list is gone), it’s still a great start for people who just need a **temporary list** or want to learn how these things work. And yes - in the future, I plan to upgrade it by adding **local storage** so the tasks can be saved even after refreshing, and maybe even add features like reminders, categories, or priority levels.

In the end, this project helped me not only revise what I learned in web development but also gave me confidence to build real things that solve daily problems. A to-do list might seem small, but it’s **super useful**, and building it from scratch made me realize how simple ideas can make a big difference.

TOOLS & TECHNOLOGIES USED

1. HTML (HyperText Markup Language)

- **Purpose:** Structure of the webpage.
- **Used for:** Creating buttons, input boxes, task list layout, and basic content of the To-Do List.

2. CSS (Cascading Style Sheets)

- **Purpose:** Styling the webpage.
- **Used for:** Making the To-Do list look attractive with colors, fonts, layout, spacing, etc.
- **Example:** Highlighting completed tasks, hover effects on buttons.

3. JavaScript

- **Purpose:** Adding interactivity and logic.
- **Used for:**
  - Adding new tasks to the list.
  - Marking tasks as completed.
  - Deleting tasks from the list.
  - Saving tasks in browser using Local Storage.

4. Web Browser (e.g., Google Chrome, Firefox, Edge)

- **Purpose:** Testing and running the To-Do list.
- **Used for:** Viewing how your HTML, CSS, and JavaScript code works together in a real environment.

5. Code Editor (e.g., Visual Studio Code, Sublime Text, Notepad++)

- **Purpose:** Writing and managing code.
- **Used for:** Developing the HTML, CSS, and JavaScript files.

6. Local Storage (Browser Feature)

- **Purpose:** Storing data within the browser.
- **Used for:** Saving tasks so that they stay even if the user refreshes or closes the tab.

7. Font Libraries (optional, e.g., Google Fonts)

- **Purpose:** To improve the visual design with custom fonts.
- **Used for:** Making the To-Do list text look stylish and readable.

8. Icons (optional, e.g., Font Awesome)

- **Purpose:** Adding icons for delete, edit, checkmark, etc.
- **Used for:** Improving user experience and design appeal

METHODOLOGY

1. Planning the Project

- First, we decided the **main goal** of our project: to create a basic **To-Do List** that helps users add, complete, and delete tasks.
- We noted down the **features** we wanted:
    - Add Task
    - Mark as Completed
    - Delete Task
    - Save in Local Storage

2. Designing the Layout (Using HTML)

- We created the **structure** of the webpage using **HTML**.
- This included:
    - An input box for entering tasks
    - An "Add Task" button
    - A list area to display all the tasks
- HTML acted like the **skeleton** of our To-Do List.

3. Styling the To-Do List (Using CSS)

- After the structure was ready, we styled it using **CSS**.
- We used colors, fonts, padding, borders, etc., to make it look clean and attractive.
- We also gave different styles to **completed tasks** and **pending tasks**.
- CSS helped in improving the **User Interface (UI)**.

4. Making It Work (Using JavaScript)

- Now, we added **JavaScript** to bring the list to life.
- JavaScript helped us perform actions when a user:
    - Clicks the "Add" button

    - Marks a task as completed
    - Deletes a task
- We used addEventListener() to catch user actions and run functions.

5. Storing Tasks (Using Local Storage)

- To make sure the tasks don't disappear when we refresh the page, we used **Local Storage**.
- JavaScript stores the tasks in the browser so that the list stays the same even after reloading.
- We used methods like:
    - localStorage.setItem() – to save tasks
    - localStorage.getItem() – to get saved tasks
    - JSON.stringify() and JSON.parse() – to store arrays properly

6. Testing the Project

- After building the To-Do List, we **tested it** by:
    - Adding, completing, and deleting different tasks
    - Refreshing the browser to check if tasks were saved
    - Using different devices or browsers
- We fixed small errors like styling problems or logic bugs.

7. Finalizing and Improving

- Once everything worked well, we cleaned the code and improved the design a bit.
- We also thought of **future improvements** like:
    - Task editing
    - Notifications or due dates
    - Sync with other devices using a backend

CODE

Index.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>HTML TO-DO LIST | MAJOR PROJECT</title>
  <link href="./bootstrap-5.0.2-dist/css/bootstrap.css" rel="stylesheet">
  <link href="./fontawesome-free-6.7.2-web/css/all.css" rel="stylesheet">
  <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap-
icons@1.13.1/font/bootstrap-icons.min.css">
</head>
<body>

  <!-- Navbar -->
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <div class="container-fluid">
      <a href="#" class="navbar-brand">
        <img src="image.jpg" class="img-fluid" alt="logo" width="50">
      </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>


    <!-- Main Content -->
    <div class="container p-5">
        <div class="mb-3 text-center">
            <button type="button" class="btn btn-outline-primary" data-bs-toggle="modal" data-
bs-target="#addTaskModal">
                Add task
            </button>
        </div>


        <!-- Task List -->
        <div class="row justify-content-center">
            <div class="col-12 col-md-8">
                <div class="card">
                    <div class="card-body">
                        <table class="table table-bordered table-striped text-center">
                            <thead class="table-light">
                                <tr>
                                    <th>#</th>
                                    <th>Task Description</th>
                                    <th>Responsible</th>
                                    <th>ETA</th>
                                    <th>Action</th>
                                </tr>
                            </thead>
                            <tbody id="taskTableBody">
                                <!-- Task items will be dynamically added here -->
                            </tbody>
                        </table>
                    </div>
                </div>
            </div>
        </div>
    </div>

    <!-- Add Task Modal -->
    <div class="modal fade" id="addTaskModal" data-bs-backdrop="static" data-bs-
keyboard="false" tabindex="-1" aria-labelledby="addTaskLabel">
        <div class="modal-dialog">
            <div class="modal-content">
                <div class="modal-header">
                    <h5 class="modal-title" id="addTaskLabel">Add Task</h5>
                    <button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
                </div>
                <div class="modal-body">
                    <form id="taskInputForm">
                        <div class="mb-3">
                            <label for="taskDescription" class="form-label">Task Description</label>
                            <textarea class="form-control" id="taskDescription"
name="taskDescription" rows="3"></textarea>
                        </div>
                        <div class="mb-3">
                            <label for="taskResponsiblePerson" class="form-
label">Responsible</label>
                            <input type="text" class="form-control" id="taskResponsiblePerson"
name="taskResponsiblePerson">
                        </div>
                        <div class="mb-3">
                            <label for="taskETA" class="form-label">ETA</label>
                            <input type="datetime-local" class="form-control" id="taskETA"
name="taskETA">
                        </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>
                    </form>
                </div>
            </div>
        </div>
    </div>

    <!-- JavaScript -->
    <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.js"></script>
    <script src="./bootstrap-5.0.2-dist/js/bootstrap.bundle.min.js"></script>

    <script>
        function addTask() {
            console.log("Add Task clicked");

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

            var dataArr = $("#taskInputForm").serializeArray();
            var taskObject = {};
            for (var i in dataArr) {
                taskObject[dataArr[i]['name']] = dataArr[i]['value'];
            }

            var storageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];
            storageObjectArr.push(taskObject);
            localStorage.setItem("taskStorage", JSON.stringify(storageObjectArr));

            createHtmlFromStorage();
            $("#taskInputForm").trigger("reset");
        }

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

            if (storageObjectArr.length > 0) {
                storageObjectArr.forEach((task, index) => {
                    html += `
                        <tr>
                            <td>${index + 1}</td>
                            <td>${task.taskDescription}</td>
                            <td>${task.taskResponsiblePerson}</td>
                            <td>${task.taskETA}</td>
                            <td>
                                <i class="bi bi-check-circle-fill text-success"
onclick="markAsDone(${index})"></i>
                                <i class="bi bi-pencil-square text-primary"
onclick="editTask(${index})"></i>
```

```
                        </td>
                    </tr>`;
                });
            } else {
                html = '<tr><td colspan="5">No Tasks Added Yet</td></tr>';
            }

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

        function markAsDone(index) {
            var storageObjectArr = JSON.parse(localStorage.getItem('taskStorage')) || [];
            storageObjectArr.splice(index, 1);
            localStorage.setItem("taskStorage", JSON.stringify(storageObjectArr));
            createHtmlFromStorage();
        }

        $(document).ready(() => {
            createHtmlFromStorage();
        });
    </script>

</body>
</html>
```

POPPER.JS:

https://unpkg.com/@popperjs/core@2/dist/umd/popper.js

you can download or use code of popper.js by accessing the above link.

jQuery:

you can download or use code of jQuery by accessing the link below.

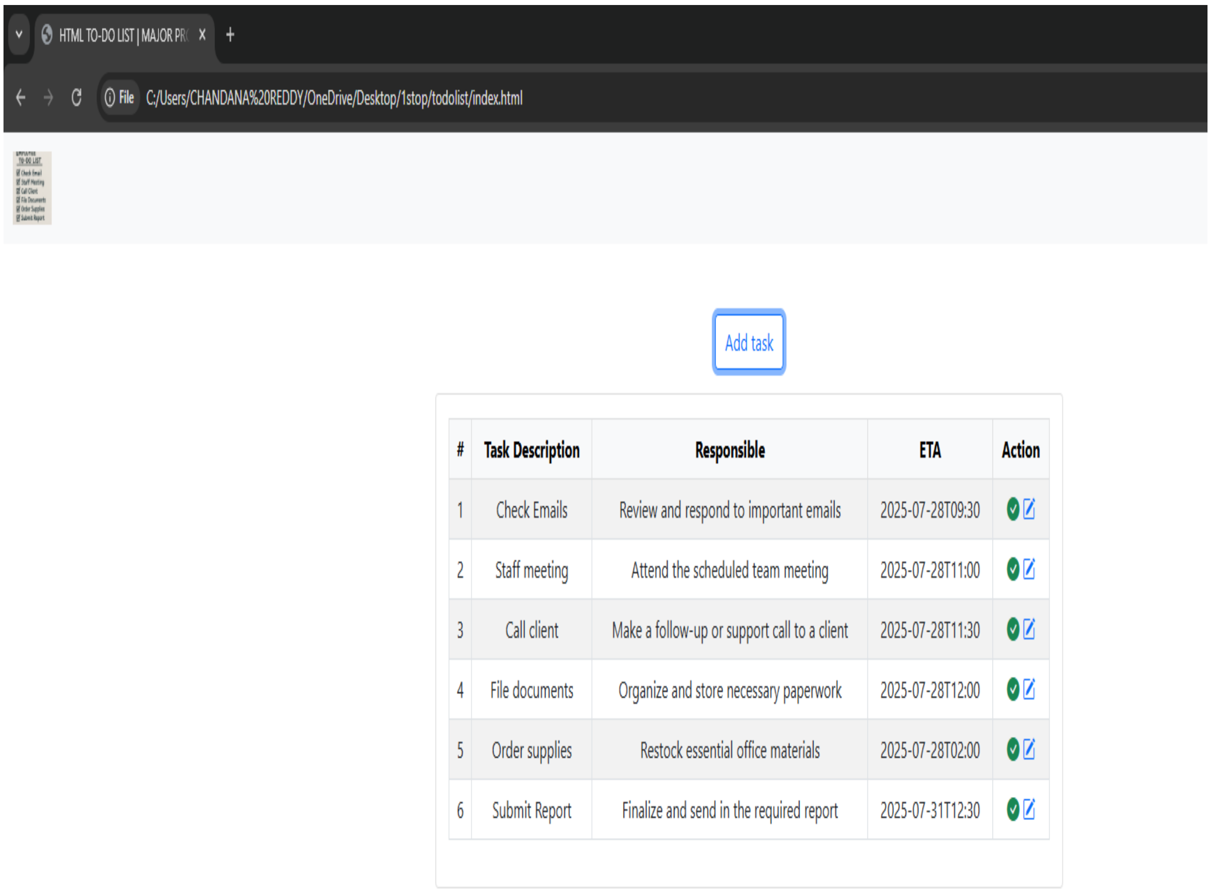
“ https://jquery.com/download/”

Bootstrap:

you can download or use the code of Bootstrap.css and Bootstrap.js by accessing the link below.

https://getbootstrap.com/docs/5.3/getting-started/download/

OUTPUT


HTML TO-DO LIST | MAJOR PR
File C:/Users/CHANDANA%20REDDY/OneDrive/Desktop/1stop/todolist/index.html
Add task

#	Task Description	Responsible	ETA	Action
1	Check Emails	Review and respond to important emails	2025-07-28T09:30	✓ ✎
2	Staff meeting	Attend the scheduled team meeting	2025-07-28T11:00	✓ ✎
3	Call client	Make a follow-up or support call to a client	2025-07-28T11:30	✓ ✎
4	File documents	Organize and store necessary paperwork	2025-07-28T12:00	✓ ✎
5	Order supplies	Restock essential office materials	2025-07-28T02:00	✓ ✎
6	Submit Report	Finalize and send in the required report	2025-07-31T12:30	✓ ✎

FEATURES OF THE TO-DO LIST

1. **Add New Tasks**
   - Users can type a new task and add it to the list by clicking a button.
   - Simple and user-friendly input box.

2. **Mark Tasks as Completed**
   - Tasks can be marked as **completed** (usually shown with a strike-through line or tick mark).
   - Helps track progress easily.

3. **Delete Tasks**
   - Each task has a **delete button** to remove it from the list.
   - Keeps the list clean and updated.

4. **Save Tasks Automatically (Using Local Storage)**
   - Tasks remain saved in the browser even after the page is refreshed or closed.
   - No need to re-enter tasks again and again.

5. **Attractive and Clean User Interface**
   - Styled using CSS for a neat and modern look.
   - Different colors for completed and pending tasks.

6. **Responsive Design (Optional)**
   - Works on both desktop and mobile devices if responsive CSS is added.
   - Useful for accessing the to-do list on the go.

7. **Interactive Buttons**
   - Add, Complete, and Delete buttons are interactive and give quick feedback.
   - Improves overall user experience.

8. Clear All Tasks

- A single button to delete all tasks at once.
- Useful for starting fresh.

9. Editable Tasks

Tasks can be edited after adding them (if implemented).

- Helps fix mistakes easily.

ADVANTAGES & DISADVANTAGES

➢ ADVANTAGES

1. Simple & Easy to Use

- Clean interface that allows users to add, complete, or delete tasks quickly.

2. Improves Productivity

- Helps users stay organized and manage their daily tasks efficiently.

3. No Installation Needed

- Can run directly in any browser. No need to install software or apps.

4. Fast Loading

- As it's made using HTML, CSS, and JavaScript, the page loads quickly.

5. Saves Tasks Automatically

- Uses **Local Storage** to keep tasks even after refreshing or closing the tab.

6. Customizable

- Easy to modify the design or add new features like edit, task count, dark mode, etc.

7. Good for Learning Web Development

- Great beginner project to understand how HTML, CSS, and JavaScript work together.

➢ DISADVANTAGES

1. Data Stored Only in Browser

- Tasks are saved in Local Storage, not on a server — so it's limited to one device/browser**.**

2. No User Login

- Can't be used by multiple users or across different devices unless backend is added.

3. Not Scalable

- Not suitable for professional task management unless major features are added.

4. No Notifications or Reminders

- Doesn't alert or remind users of due tasks (unless integrated with advanced tools).

5. Limited Security

- Local Storage is not secure for sensitive task data (if used in real apps).

6. No Backup or Sync

- If the browser data is cleared, all saved tasks are lost**.**

CONCLUSION

In this project, we created a **To-Do List** using simple web development tools like **HTML, CSS, and JavaScript**. It is a basic but very useful project, especially for beginners like us who are learning how websites work and how we can build something useful from scratch.

The main goal of this project was to help users **manage their daily tasks** in an easy and organized way. With this To-Do list, anyone can write down tasks, mark them as completed, or delete them when they are done. This kind of simple application can be very helpful in our everyday lives – whether it's for students tracking their homework, employees managing their work, or even homemakers planning their day.

While working on this project, we understood how important the structure (HTML), design (CSS), and behavior (JavaScript) are in creating a complete working web application. HTML helped us build the basic layout like buttons, input boxes, and task lists. CSS made it look neat and attractive. JavaScript made the To-Do List functional – like adding tasks, completing them, or deleting them.

One of the best parts of our To-Do list is the **use of Local Storage**. This feature helps in saving the tasks even after refreshing or closing the page. So, if a user adds 5 tasks today and closes the browser, those tasks will still be there when they open the page again. This is a simple example of **client-side storage**, and it adds real value to the user experience.

This project also taught us the importance of a **good user interface (UI)**. Even though our project is simple, we tried to make it look good using CSS. A clean and friendly UI makes it easier for people to use and understand the app. We also tried to make the buttons responsive so that users get immediate feedback when they click something.

Another important learning from this project is how we can break big problems into small parts. For example, to build this project, we first created the structure, then styled it, and finally added functions step by step. This taught us how to approach a real-world project in an organized way.

We also faced a few challenges during this project. At times, the tasks were not saving, or the styling was not showing properly. But by debugging the code and reading online tutorials, we solved these problems. This process helped us become better at **problem-solving and debugging**, which are very important skills in the world of programming.

To conclude, this HTML To-Do List project gave us a very good experience in learning **front-end development**. It showed us how coding can be used to build helpful tools. It improved our confidence, taught us how websites are made, and encouraged us to learn more about web development.