Abstract:

The To-Do List web application is a simple yet effective tool designed to help users organize their tasks and manage their time more efficiently. Built using HTML, CSS, and JavaScript, this project provides a user-friendly interface for creating, editing, and deleting tasks on the fly.

The application's frontend is developed using HTML and styled with CSS to ensure a visually appealing and intuitive layout. Users can easily navigate through the different sections of the application, including the task list, input form, and action buttons. The design prioritizes simplicity and clarity, allowing users to focus on their tasks without unnecessary distractions.

The core functionality of the To-Do List is implemented using JavaScript, which enables dynamic interaction and real-time updates without the need to reload the page. Users can add new tasks by simply typing in the input field and pressing enter, with the task instantly appearing on the list. They can mark tasks as completed, edit task descriptions, or delete tasks altogether with just a few clicks.

To enhance user experience and convenience, the application utilizes local storage to persist tasks across browser sessions. This means users can close the browser and return to find their tasks exactly as they left them, providing a seamless and uninterrupted workflow.

Additionally, the To-Do List incorporates responsive design principles, ensuring compatibility with various screen sizes and devices. Whether users access the application from a desktop computer, tablet, or smartphone, they can enjoy the same level of functionality and usability.

Overall, the To-Do List web application offers a practical solution for managing tasks and staying organized in today's fast-paced world. By leveraging HTML, CSS, and JavaScript, this project demonstrates the power of web technologies in creating efficient and user-friendly productivity tools. Whether used for personal task management or collaborative projects, the To-Do List empowers users to take control of their schedules and accomplish their goals with ease

OBJECTIVE

The objective of this project is to create a dynamic and user-friendly To Do list application using HTML, CSS, and JavaScript. In today's fast-paced world, staying organized is essential, and a digital To Do list provides a convenient way to manage tasks efficiently. By developing this application, we aim to provide users with a seamless experience for organizing their tasks, increasing productivity, and reducing the cognitive load associated with remembering everything they need to do.

Key Features

User-Friendly Interface :

The To Do list will have a clean and intuitive interface, making it easy for users to add, edit, and delete tasks.

Task Management:

Users can add new tasks with a title, description, due date, priority, and status. They can mark tasks as complete or incomplete, enabling them to track their progress effectively.

Sorting and Filtering:

The application will allow users to sort tasks based on priority, due date, or status. Additionally, users can filter tasks based on various criteria such as priority level or completion status, enabling them to focus on specific tasks or categories.

Responsive Design:

The To Do list will be responsive, ensuring optimal user experience across different devices and screen sizes, including desktops, tablets, and smartphones.

Persistent Data Storage:

Tasks will be stored locally or using a server-side database, ensuring that users can access their To Do list from any device or browser.

Customization Options:

Users will have the option to customize the appearance of the To Do list, such as choosing different themes or color schemes to suit their preferences.

Reminders and Notifications:

The application may include features for setting reminders and receiving notifications for upcoming tasks or deadlines, helping users stay on track and meet their goals.

Collaboration:

In future iterations, we may explore adding collaboration features, allowing users to share their To Do lists with others, assign tasks, and track progress collaboratively.

By developing this To Do list application, we aim to provide users with a powerful tool for organizing their tasks, managing their time effectively, and ultimately enhancing their productivity and overall well-being. Whether it's for personal use, professional tasks, or collaborative projects, our goal is to create a versatile and indispensable tool that empowers users to stay organized and focused on what matters most.

Introduction

In today's fast-paced world, staying organized is key to managing our daily tasks efficiently. Whether it's for work, school, or personal life, having a reliable system to keep track of our to-dos can significantly enhance productivity and reduce stress. Recognizing this need, we embark on a journey to create a dynamic and user-friendly To-Do List application using HTML, CSS, and JavaScript.

Understanding the Purpose

The purpose of our To-Do List project is to provide users with a simple yet effective tool to manage their tasks. We aim to develop a solution that not only allows users to add, edit, and delete tasks but also offers additional features to enhance their organization process. By understanding the needs of our target users and leveraging the capabilities of web development technologies, we strive to deliver a robust and intuitive application.

Key Features

Task Management:

The core functionality of our To-Do List revolves around task management. Users can add new tasks, mark them as completed, edit existing tasks, and remove tasks they no longer need. This feature forms the foundation of our application, enabling users to keep track of their responsibilities effectively.

User Interface Design:

A clean and intuitive user interface is essential for a positive user experience. We prioritize designing an interface that is visually appealing and easy to navigate. Through thoughtful layout design, color schemes, and typography choices, we aim to create a pleasant environment for users to interact with our application.

Responsive Design:

With the increasing prevalence of mobile devices, it's crucial to ensure that our To-Do List is accessible across various screen sizes. Implementing responsive design principles allows our application to adapt seamlessly to different devices, whether it's a desktop computer, tablet, or smartphone. This ensures that users can manage their tasks conveniently regardless of the device they're using.

Data Persistence:

To provide a seamless experience for users, we implement data persistence functionality. This means that tasks added by users will be stored persistently, allowing them to access their task list even after refreshing the page or closing the browser. By leveraging technologies such as local storage or server-side databases, we ensure that users' data is preserved securely.

Customization Options:

Recognizing that different users have different preferences, we incorporate customization options into our To-Do List application. This may include features such as color themes, sorting options, or priority levels for tasks. By allowing users to personalize their experience, we empower them to tailor the application to suit their unique workflow.

Reminders and Notifications:

To further enhance productivity, we explore the possibility of integrating reminders and notifications into our To-Do List. This feature enables users to set deadlines or reminders for specific tasks and receive notifications when those deadlines approach. By keeping users informed and on track, we facilitate better task management and timely completion of goals.

Collaboration and Sharing:

In an increasingly interconnected world, collaboration tools are becoming essential for effective teamwork. While our initial focus is on individual task management, we envision expanding our To-Do List application to support collaboration features in the future. This may include functionalities such as sharing task lists with colleagues, assigning tasks to team members, and tracking progress collectively.

Technological Stack

Our choice of HTML, CSS, and JavaScript as the technological stack for this project offers several advantages. HTML provides the structure and semantics for our web pages, CSS enables us to style the interface and create visually appealing designs, and JavaScript adds interactivity and dynamic functionality to the application. By leveraging these technologies in combination, we can create a powerful and versatile To-Do List application that meets the needs of our users effectively. our To-Do List project represents a commitment to empowering users with a robust and intuitive task management solution. By focusing on key features such as task management, user interface design, responsiveness, data persistence, customization options, reminders and notifications, and future collaboration capabilities, we aim to deliver a comprehensive tool that enhances productivity and organization. Through the strategic utilization of HTML, CSS, and JavaScript, we seek to harness the full potential of web development technologies to create a valuable asset for users in their daily lives.

Methodology

Project Planning and Requirements Gathering

Define the Scope:

Determine the core features and functionalities of the To Do list application.

Identify Target Audience:

Understand who will be using the application and their needs.

List Requirements:

Make a list of features such as adding tasks, marking tasks as completed, deleting tasks, etc.

Designing the User Interface (UI)

Wireframing:

Sketch out the layout of the To Do list application including placement of elements like input fields, buttons, and task items.

UI Design:

Create a visually appealing design using HTML and CSS. Consider usability and accessibility principles.

Setting Up the Project Environment

Create Project Directory:

Set up a folder structure for organizing HTML, CSS, and JavaScript files.

Initialize Git Repository:

Track changes using version control for better collaboration and tracking changes.

Implementing the HTML Structure

Create HTML Skeleton:

Develop the basic structure of the To Do list application using semantic HTML elements.

Add Task Input Field:

Include an input field for users to add new tasks.

Display Task List:

Create a section to display the list of tasks entered by the user.

Styling with CSS

Style Components:

Apply CSS styles to the HTML elements to enhance the visual appearance of the application.

Responsive Design:

Ensure the application is responsive across various devices and screen sizes.

Animations (Optional):

Add animations to provide feedback and improve user experience.

Adding Interactivity with JavaScript

Handle Task Addition:

Implement JavaScript functions to add tasks to the list when the user submits a new task.

Manage Task Status:

Enable users to mark tasks as completed or incomplete.

Delete Tasks:

Allow users to remove tasks from the list.

Local Storage (Optional):

Implement local storage to persist tasks even after the page is refreshed.

Testing

Manual Testing:

Test the application manually to ensure all features work as expected.

Cross-Browser Testing:

Check the compatibility of the application across different web browsers.

User Testing:

Gather feedback from potential users to identify any usability issues and make necessary improvements.

Debugging and Optimization

Debugging:

Identify and fix any bugs or errors in the codebase.

Performance Optimization:

Optimize the code for better performance, including minimizing load times and improving responsiveness.

Documentation

Code Documentation:

Document the codebase, including comments explaining the purpose of each function and section of code.

User Documentation:

Create user guides or documentation explaining how to use the To Do list application.

Deployment

Choose Hosting Platform: Select a hosting platform to deploy the To Do list application.

Deploy Application:

Upload the files to the chosen hosting platform and make the application accessible to users.

Domain Configuration (Optional):

Configure a custom domain name for the application if necessary.

Maintenance and Updates

Monitor Performance:

Regularly monitor the performance of the application and make necessary updates to improve user experience.

Address Feedback:

Listen to user feedback and make updates or additions to the application based on their suggestions.

Security Updates:

Stay updated with security patches and implement necessary measures to protect user data.

Continuous Improvement

Feature Enhancements:

Continuously add new features or improve existing ones to meet evolving user needs.

Technology Upgrades:

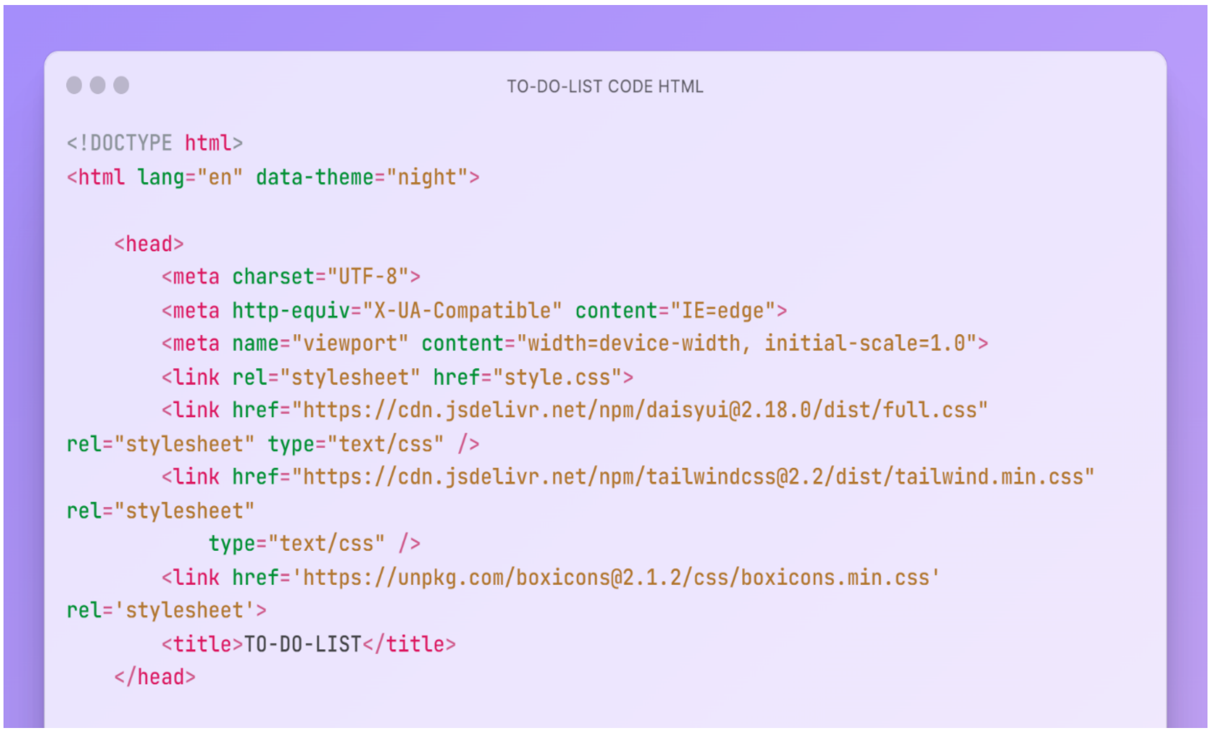
Keep abreast of new technologies and frameworks to enhance the To Do list application.

Conclusion

By following this methodology, you can develop a functional and user-friendly To Do list web application using HTML, CSS, and JavaScript. Remember to prioritize user experience and regularly update the application to meet changing requirements and standards.

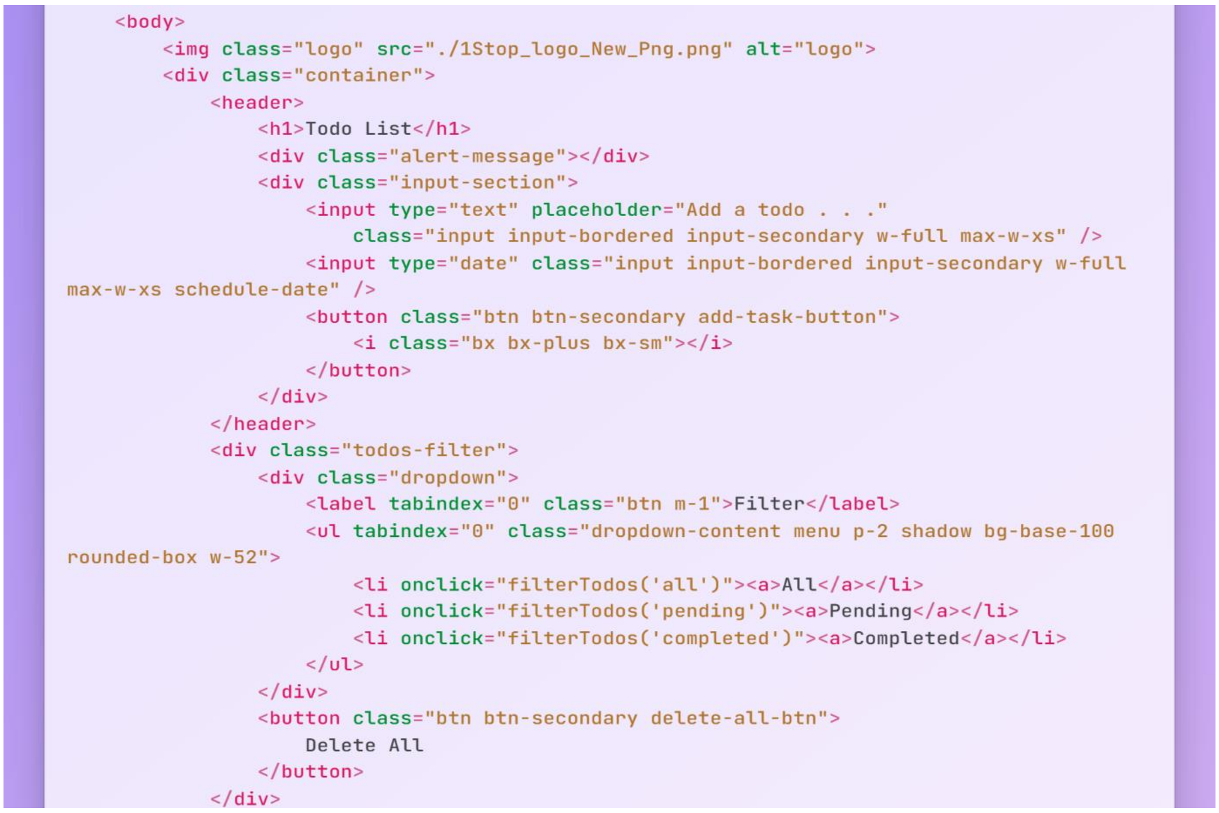
CODE - Include the screenshots of RESULTS AND OUTPUT

HTML

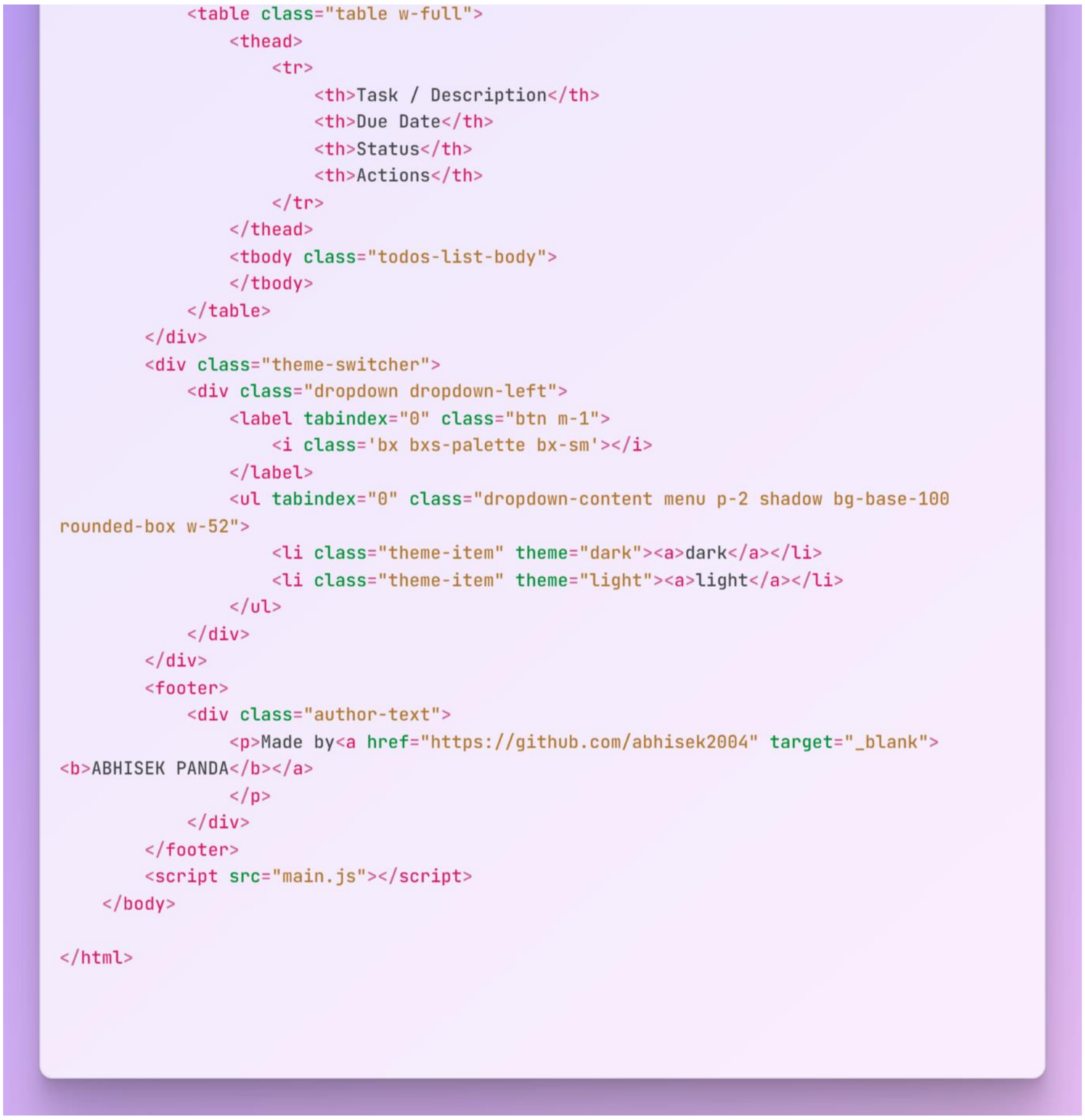


TO-DO-LIST CODE HTML

```
<!DOCTYPE html>
<html lang="en" data-theme="night">

    <head>
        <meta charset="UTF-8">
        <meta http-equiv="X-UA-Compatible" content="IE=edge">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" href="style.css">
        <link href="https://cdn.jsdelivr.net/npm/daisyui@2.18.0/dist/full.css"
rel="stylesheet" type="text/css" />
        <link href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2/dist/tailwind.min.css"
rel="stylesheet"
            type="text/css" />
        <link href='https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css'
rel='stylesheet'>
        <title>TO-DO-LIST</title>
    </head>
```

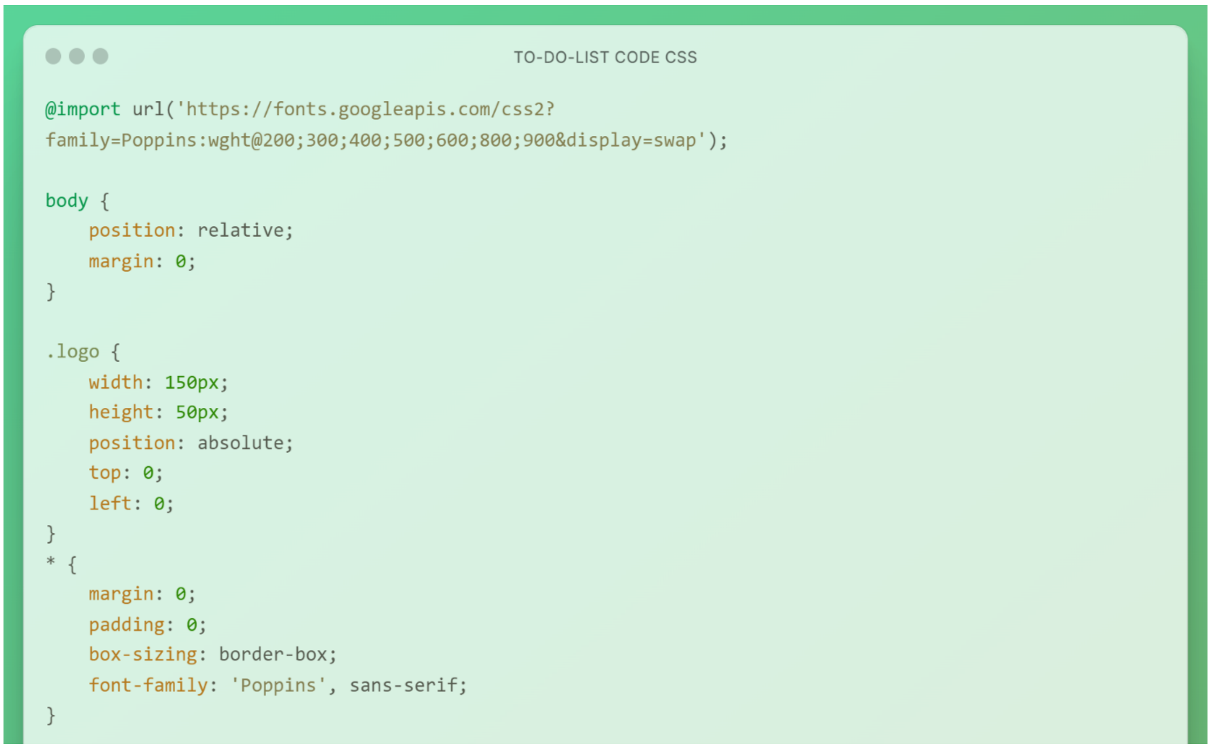


```
    <body>
        <img class="logo" src="./1Stop_logo_New_Png.png" alt="logo">
        <div class="container">
            <header>
                <h1>Todo List</h1>
                <div class="alert-message"></div>
                <div class="input-section">
                    <input type="text" placeholder="Add a todo . . ."
                        class="input input-bordered input-secondary w-full max-w-xs" />
                    <input type="date" class="input input-bordered input-secondary w-full
max-w-xs schedule-date" />
                    <button class="btn btn-secondary add-task-button">
                        <i class="bx bx-plus bx-sm"></i>
                    </button>
                </div>
            </header>
            <div class="todos-filter">
                <div class="dropdown">
                    <label tabindex="0" class="btn m-1">Filter</label>
                    <ul tabindex="0" class="dropdown-content menu p-2 shadow bg-base-100
rounded-box w-52">
                        <li onclick="filterTodos('all')"><a>All</a></li>
                        <li onclick="filterTodos('pending')"><a>Pending</a></li>
                        <li onclick="filterTodos('completed')"><a>Completed</a></li>
                    </ul>
                </div>
                <button class="btn btn-secondary delete-all-btn">
                    Delete All
                </button>
            </div>
```



```html
            <table class="table w-full">
                <thead>
                    <tr>
                        <th>Task / Description</th>
                        <th>Due Date</th>
                        <th>Status</th>
                        <th>Actions</th>
                    </tr>
                </thead>
                <tbody class="todos-list-body">
                </tbody>
            </table>
        </div>
        <div class="theme-switcher">
            <div class="dropdown dropdown-left">
                <label tabindex="0" class="btn m-1">
                    <i class='bx bxs-palette bx-sm'></i>
                </label>
                <ul tabindex="0" class="dropdown-content menu p-2 shadow bg-base-100
rounded-box w-52">
                    <li class="theme-item" theme="dark"><a>dark</a></li>
                    <li class="theme-item" theme="light"><a>light</a></li>
                </ul>
            </div>
        </div>
        <footer>
            <div class="author-text">
                <p>Made by<a href="https://github.com/abhisek2004" target="_blank">
<b>ABHISEK PANDA</b></a>
                </p>
            </div>
        </footer>
        <script src="main.js"></script>
    </body>

</html>
```

CSS

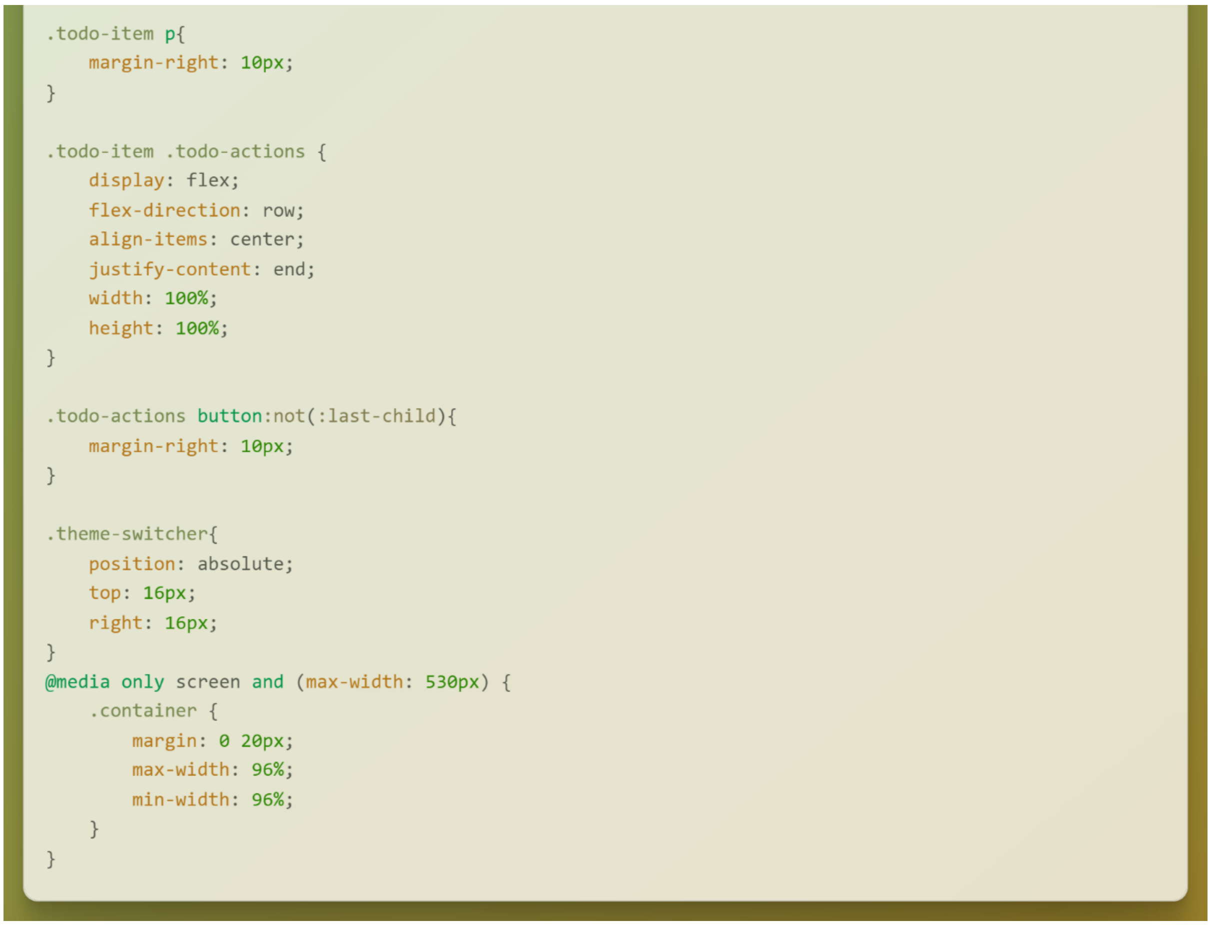


```css
@import url('https://fonts.googleapis.com/css2?
family=Poppins:wght@200;300;400;500;600;800;900&display=swap');

body {
    position: relative;
    margin: 0;
}

.logo {
    width: 150px;
    height: 50px;
    position: absolute;
    top: 0;
    left: 0;
}
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Poppins', sans-serif;
}
```

```css
:root {
    --color-primary: #5a78ff;
    --color-secondary: #0957ff;
}

body{
    position: relative;
    height: 100vh;
    display: flex;
    justify-content: center;
    align-items: center;
}

.author-text{
    position: absolute;
    bottom: 0;
    left: 0;
    width: 100%;
    display: flex;
    justify-content: center;
    margin-bottom: 20px;
    text-align: center;
}


.container {
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;
    margin: 0 400px;
    min-width: 640px;
    max-width: 1000px;
    background: rgba( 255, 255, 255, 0.15 );
    box-shadow: 0 8px 32px 0 rgba( 31, 38, 135, 0.3 );
    backdrop-filter: blur( 3px );
    -webkit-backdrop-filter: blur( 3px );
    border-radius: 10px;
    border: 1px solid rgba( 255, 255, 255, 0.18 );
    padding: 20px;
}
```

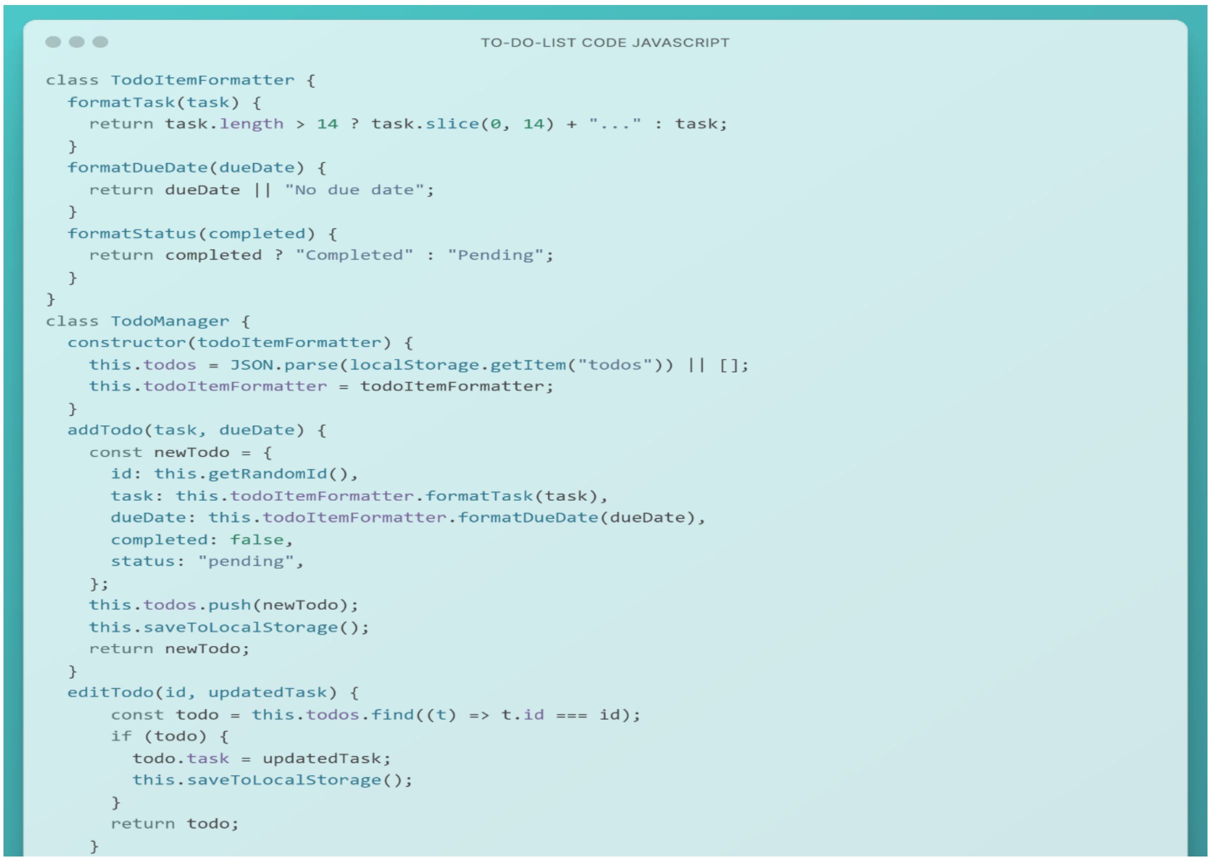
```
.container header {
    display: flex;
    justify-content: center;
    align-items: center;
    flex-direction: column;
    width: 100%;
    margin-bottom: 20px;
}

header h1 {
    font-size: 30px;
    font-weight: 700;
    margin-bottom: 10px;
}

.alert-message{
    width: 100%;
    transition: all 0.3s ease;
    transform: scale(0.9);
}

.alert-message.show{
    display: block;
    transform: scale(1);
}

.alert-message.hide{
    display: none;
}

header .input-section {
    width: 100%;
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: center;
    width: 100%;
    height: 100%;
}
```

```
.input-section input{
    margin-right: 10px;
    max-width: 100%;
}

.todos-filter{
    display: flex;
    justify-content: space-between;
    align-items: center;
    width: 100%;
    height: 100%;
    margin-bottom: 10px;
}

.todos-list {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    align-items: center;
    min-height: 100%;
    max-height: 54vh;
    overflow-y: auto;
    width: 100%;
}

.todos-list .todo-item{
    display: flex;
    flex-direction: row;
    justify-content: space-between;
    align-items: center;
    padding: 10px;
    width: 100%;
    border-bottom: 1px solid rgba( 255, 255, 255, 0.18 );
}
```

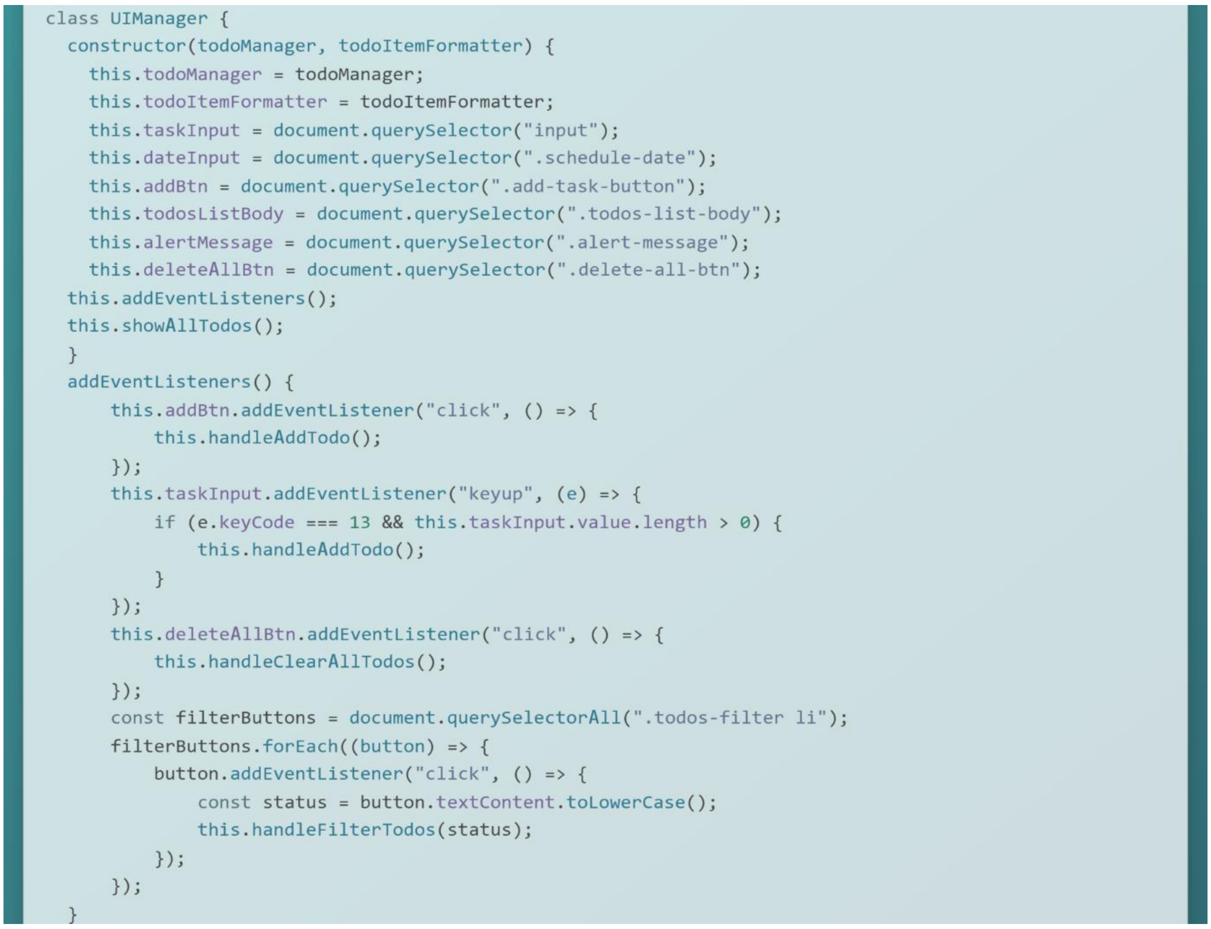


```css
.todo-item p{
    margin-right: 10px;
}

.todo-item .todo-actions {
    display: flex;
    flex-direction: row;
    align-items: center;
    justify-content: end;
    width: 100%;
    height: 100%;
}

.todo-actions button:not(:last-child){
    margin-right: 10px;
}

.theme-switcher{
    position: absolute;
    top: 16px;
    right: 16px;
}
@media only screen and (max-width: 530px) {
    .container {
        margin: 0 20px;
        max-width: 96%;
        min-width: 96%;
    }
}
```

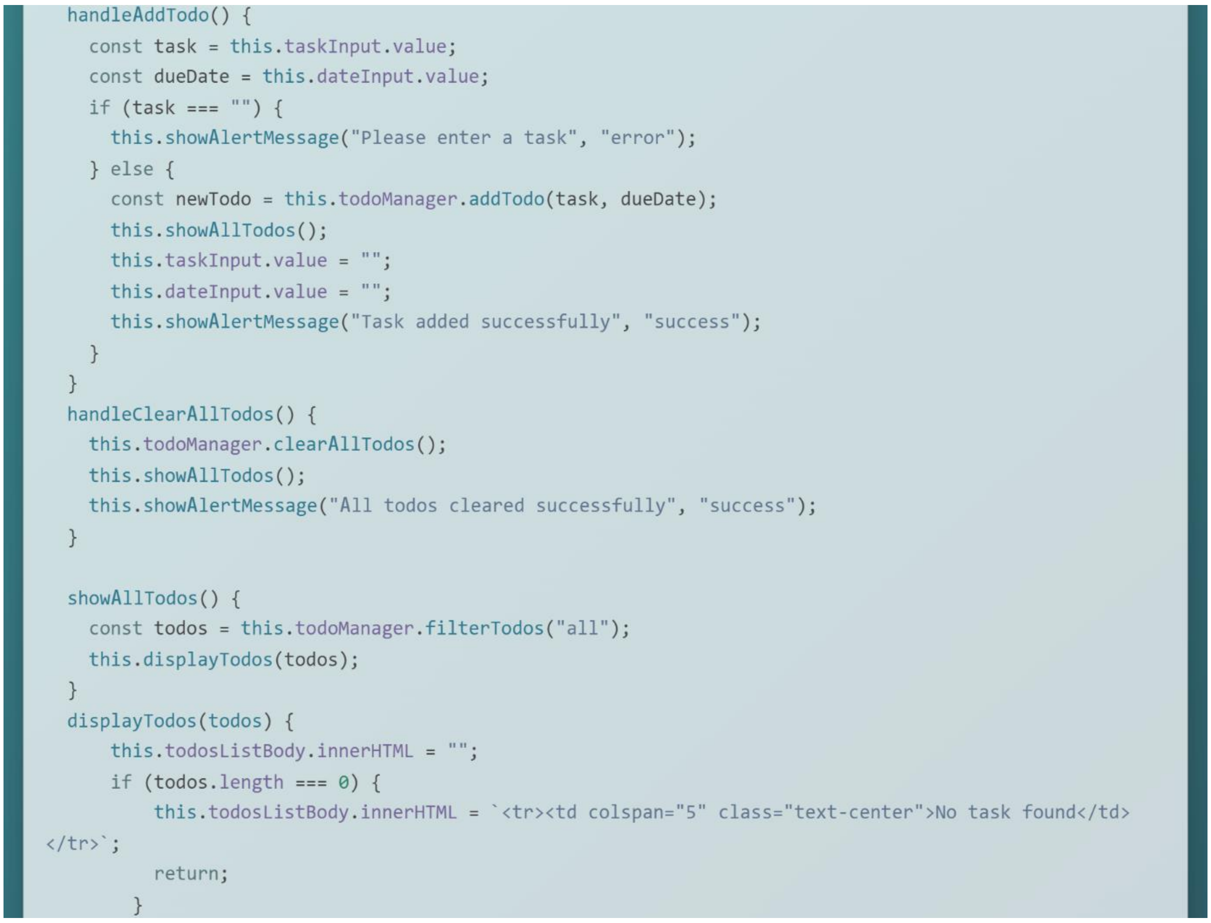
JAVASCRIPT


TO-DO-LIST CODE JAVASCRIPT

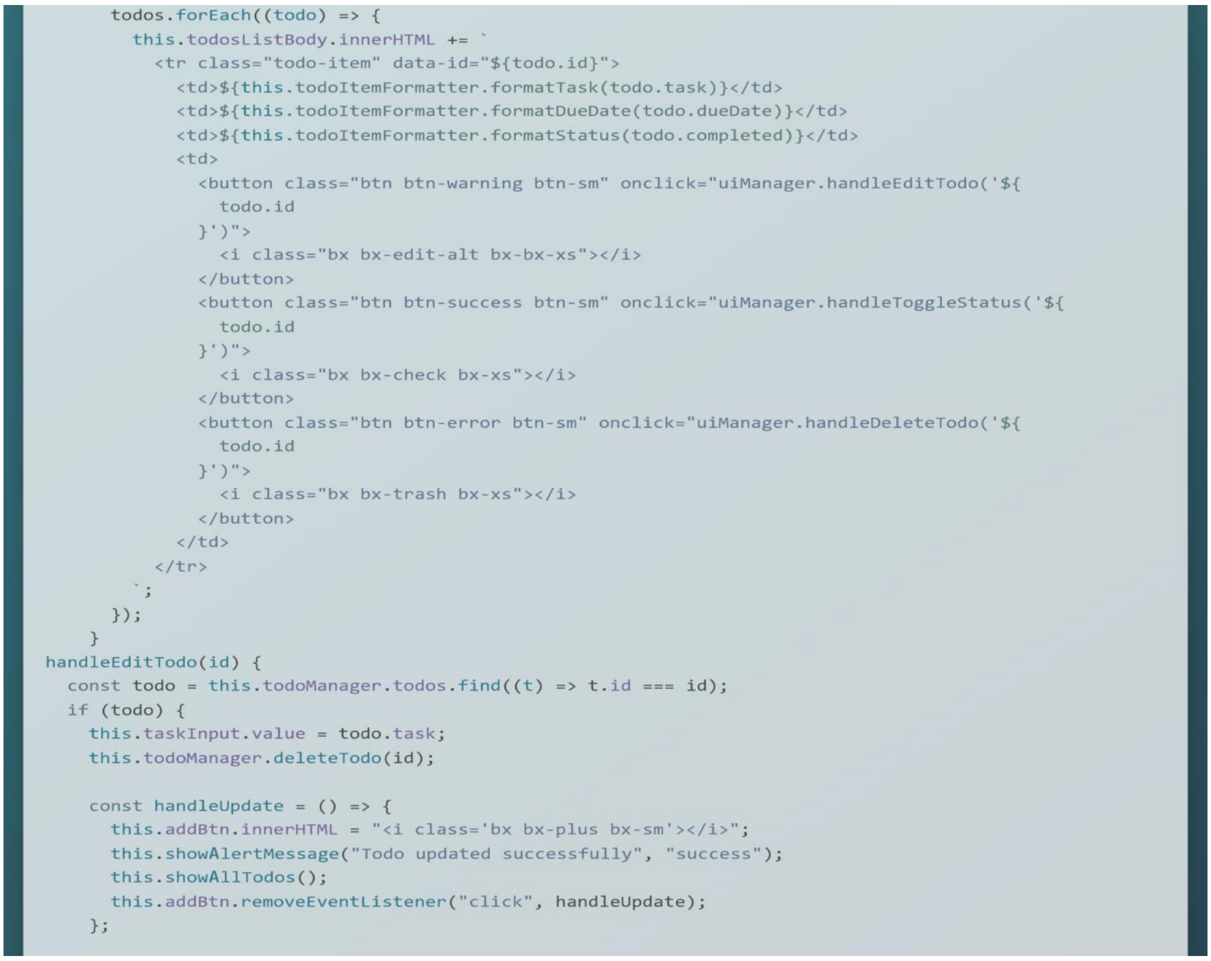
```
class TodoItemFormatter {
  formatTask(task) {
    return task.length > 14 ? task.slice(0, 14) + "..." : task;
  }
  formatDueDate(dueDate) {
    return dueDate || "No due date";
  }
  formatStatus(completed) {
    return completed ? "Completed" : "Pending";
  }
}
class TodoManager {
  constructor(todoItemFormatter) {
    this.todos = JSON.parse(localStorage.getItem("todos")) || [];
    this.todoItemFormatter = todoItemFormatter;
  }
  addTodo(task, dueDate) {
    const newTodo = {
      id: this.getRandomId(),
      task: this.todoItemFormatter.formatTask(task),
      dueDate: this.todoItemFormatter.formatDueDate(dueDate),
      completed: false,
      status: "pending",
    };
    this.todos.push(newTodo);
    this.saveToLocalStorage();
    return newTodo;
  }
  editTodo(id, updatedTask) {
      const todo = this.todos.find((t) => t.id === id);
      if (todo) {
        todo.task = updatedTask;
        this.saveToLocalStorage();
      }
      return todo;
    }
    deleteTodo(id) {
      this.todos = this.todos.filter((todo) => todo.id !== id);
      this.saveToLocalStorage();
    }
    toggleTodoStatus(id) {
      const todo = this.todos.find((t) => t.id === id);
      if (todo) {
        todo.completed = !todo.completed;
        this.saveToLocalStorage();
      }
    }
    clearAllTodos() {
      if (this.todos.length > 0) {
        this.todos = [];
        this.saveToLocalStorage();
      }
    }
    filterTodos(status) {
      switch (status) {
        case "all":
          return this.todos;
        case "pending":
          return this.todos.filter((todo) => !todo.completed);
        case "completed":
          return this.todos.filter((todo) => todo.completed);
        default:
          return [];
      }
    }
    getRandomId() {
      return (
        Math.random().toString(36).substring(2, 15) +
        Math.random().toString(36).substring(2, 15)
      );
    }
    saveToLocalStorage() {
      localStorage.setItem("todos", JSON.stringify(this.todos));
    }
}
```



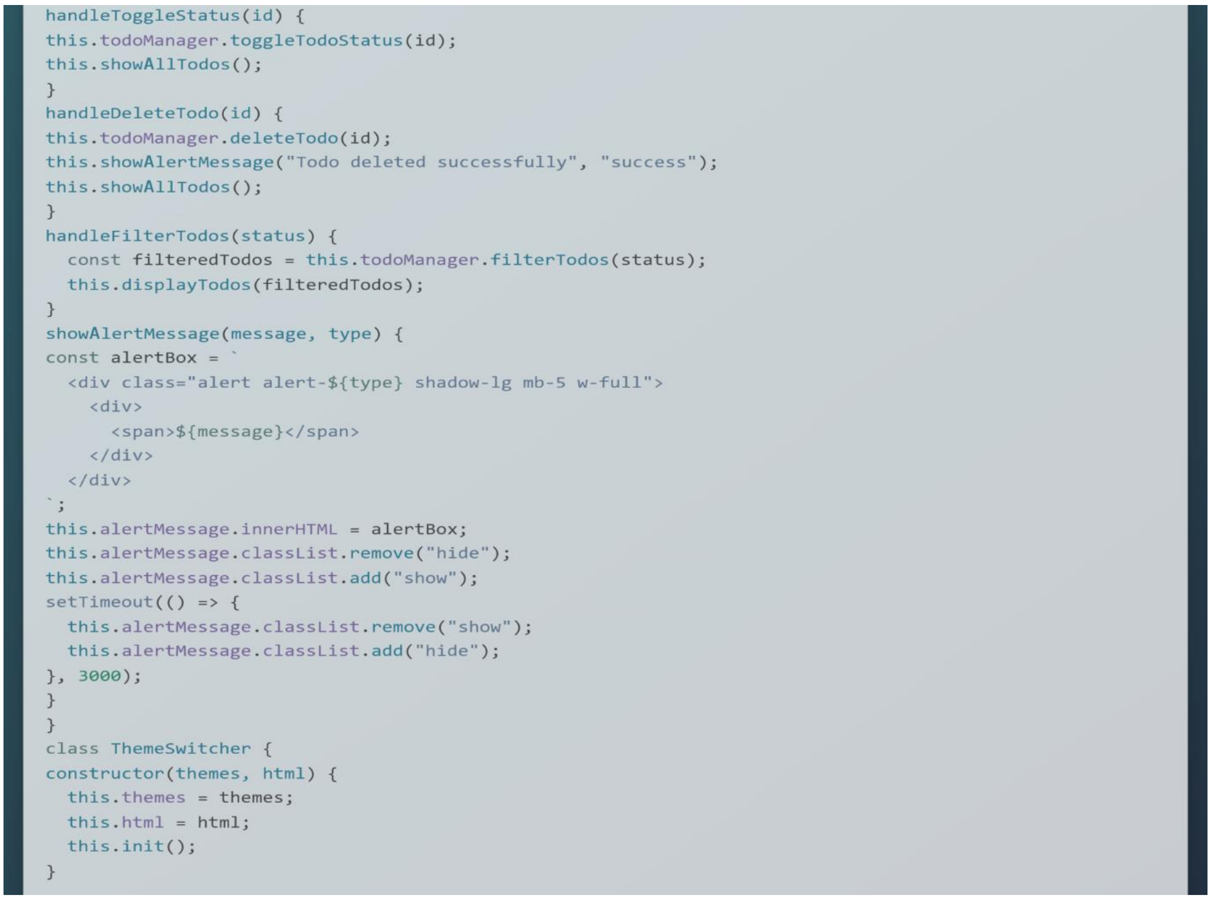
```
class UIManager {
  constructor(todoManager, todoItemFormatter) {
    this.todoManager = todoManager;
    this.todoItemFormatter = todoItemFormatter;
    this.taskInput = document.querySelector("input");
    this.dateInput = document.querySelector(".schedule-date");
    this.addBtn = document.querySelector(".add-task-button");
    this.todosListBody = document.querySelector(".todos-list-body");
    this.alertMessage = document.querySelector(".alert-message");
    this.deleteAllBtn = document.querySelector(".delete-all-btn");
  this.addEventListeners();
  this.showAllTodos();
  }
  addEventListeners() {
      this.addBtn.addEventListener("click", () => {
          this.handleAddTodo();
      });
      this.taskInput.addEventListener("keyup", (e) => {
          if (e.keyCode === 13 && this.taskInput.value.length > 0) {
              this.handleAddTodo();
          }
      });
      this.deleteAllBtn.addEventListener("click", () => {
          this.handleClearAllTodos();
      });
      const filterButtons = document.querySelectorAll(".todos-filter li");
      filterButtons.forEach((button) => {
          button.addEventListener("click", () => {
              const status = button.textContent.toLowerCase();
              this.handleFilterTodos(status);
          });
      });
  }
```



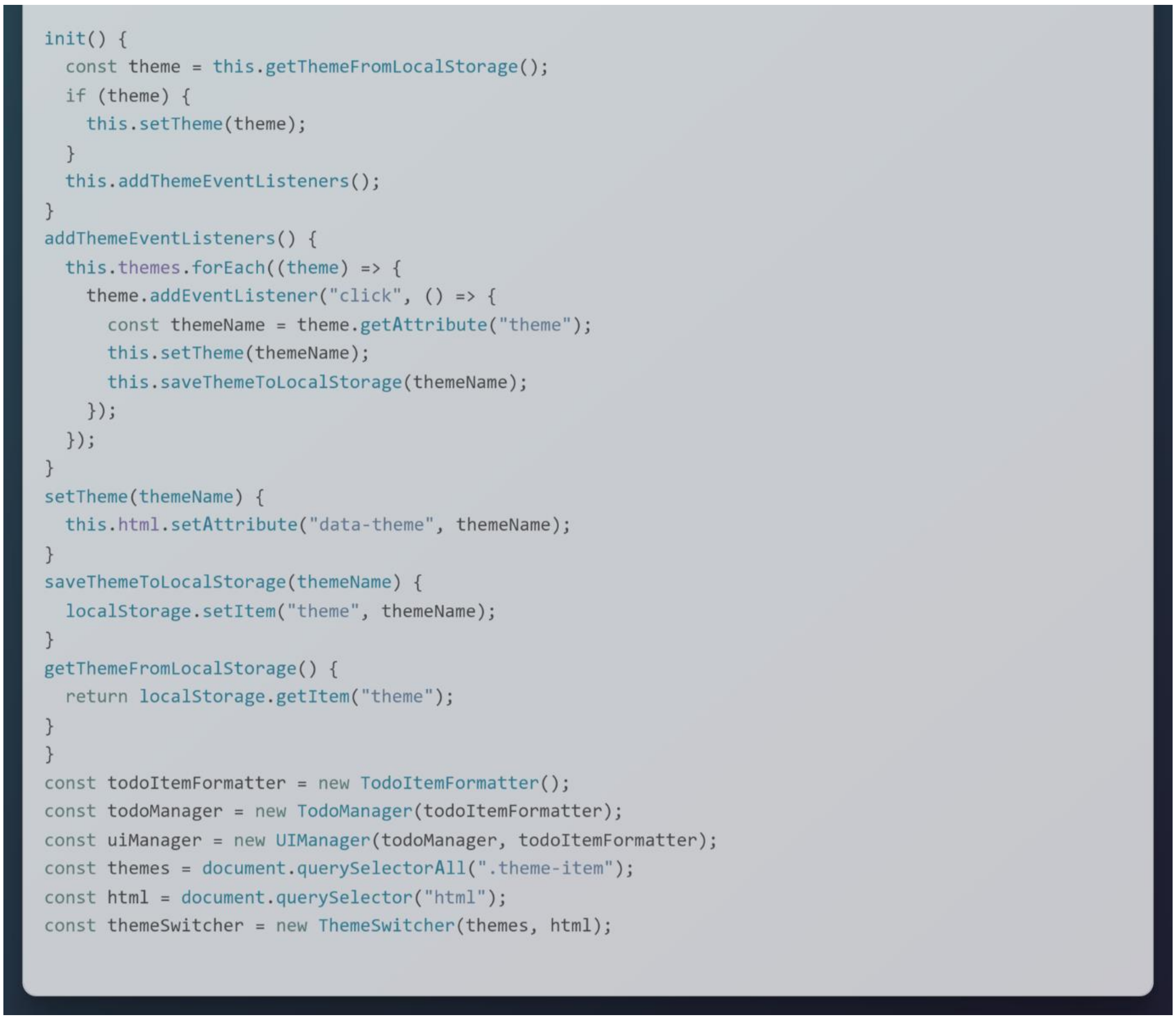
```
  handleAddTodo() {
    const task = this.taskInput.value;
    const dueDate = this.dateInput.value;
    if (task === "") {
      this.showAlertMessage("Please enter a task", "error");
    } else {
      const newTodo = this.todoManager.addTodo(task, dueDate);
      this.showAllTodos();
      this.taskInput.value = "";
      this.dateInput.value = "";
      this.showAlertMessage("Task added successfully", "success");
    }
  }
  handleClearAllTodos() {
    this.todoManager.clearAllTodos();
    this.showAllTodos();
    this.showAlertMessage("All todos cleared successfully", "success");
  }

  showAllTodos() {
    const todos = this.todoManager.filterTodos("all");
    this.displayTodos(todos);
  }
  displayTodos(todos) {
      this.todosListBody.innerHTML = "";
      if (todos.length === 0) {
          this.todosListBody.innerHTML = `<tr><td colspan="5" class="text-center">No task found</td>
</tr>`;
          return;
        }
```



```
      todos.forEach((todo) => {
        this.todosListBody.innerHTML += `
          <tr class="todo-item" data-id="${todo.id}">
            <td>${this.todoItemFormatter.formatTask(todo.task)}</td>
            <td>${this.todoItemFormatter.formatDueDate(todo.dueDate)}</td>
            <td>${this.todoItemFormatter.formatStatus(todo.completed)}</td>
            <td>
              <button class="btn btn-warning btn-sm" onclick="uiManager.handleEditTodo('${
                todo.id
              }')">
                <i class="bx bx-edit-alt bx-bx-xs"></i>
              </button>
              <button class="btn btn-success btn-sm" onclick="uiManager.handleToggleStatus('${
                todo.id
              }')">
                <i class="bx bx-check bx-xs"></i>
              </button>
              <button class="btn btn-error btn-sm" onclick="uiManager.handleDeleteTodo('${
                todo.id
              }')">
                <i class="bx bx-trash bx-xs"></i>
              </button>
            </td>
          </tr>
        `;
      });
    }
handleEditTodo(id) {
  const todo = this.todoManager.todos.find((t) => t.id === id);
  if (todo) {
    this.taskInput.value = todo.task;
    this.todoManager.deleteTodo(id);

    const handleUpdate = () => {
      this.addBtn.innerHTML = "<i class='bx bx-plus bx-sm'></i>";
      this.showAlertMessage("Todo updated successfully", "success");
      this.showAllTodos();
      this.addBtn.removeEventListener("click", handleUpdate);
    };
```

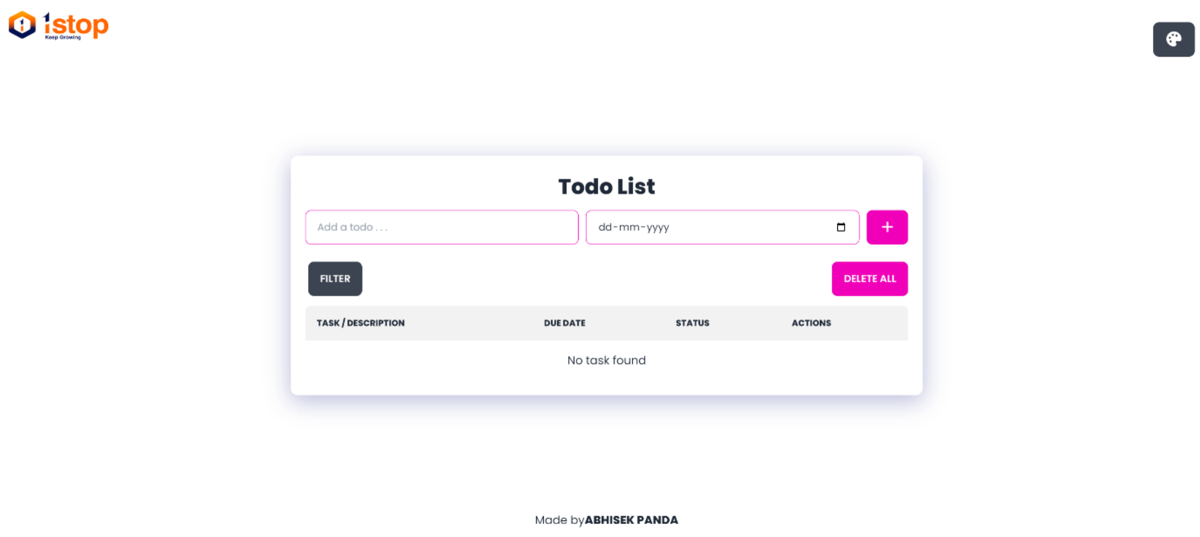


```
handleToggleStatus(id) {
this.todoManager.toggleTodoStatus(id);
this.showAllTodos();
}
handleDeleteTodo(id) {
this.todoManager.deleteTodo(id);
this.showAlertMessage("Todo deleted successfully", "success");
this.showAllTodos();
}
handleFilterTodos(status) {
  const filteredTodos = this.todoManager.filterTodos(status);
  this.displayTodos(filteredTodos);
}
showAlertMessage(message, type) {
const alertBox = `
  <div class="alert alert-${type} shadow-lg mb-5 w-full">
    <div>
      <span>${message}</span>
    </div>
  </div>
`;
this.alertMessage.innerHTML = alertBox;
this.alertMessage.classList.remove("hide");
this.alertMessage.classList.add("show");
setTimeout(() => {
  this.alertMessage.classList.remove("show");
  this.alertMessage.classList.add("hide");
}, 3000);
}
}
class ThemeSwitcher {
constructor(themes, html) {
  this.themes = themes;
  this.html = html;
  this.init();
}
```



```
init() {
  const theme = this.getThemeFromLocalStorage();
  if (theme) {
    this.setTheme(theme);
  }
  this.addThemeEventListeners();
}
addThemeEventListeners() {
  this.themes.forEach((theme) => {
    theme.addEventListener("click", () => {
      const themeName = theme.getAttribute("theme");
      this.setTheme(themeName);
      this.saveThemeToLocalStorage(themeName);
    });
  });
}
setTheme(themeName) {
  this.html.setAttribute("data-theme", themeName);
}
saveThemeToLocalStorage(themeName) {
  localStorage.setItem("theme", themeName);
}
getThemeFromLocalStorage() {
  return localStorage.getItem("theme");
}
}
const todoItemFormatter = new TodoItemFormatter();
const todoManager = new TodoManager(todoItemFormatter);
const uiManager = new UIManager(todoManager, todoItemFormatter);
const themes = document.querySelectorAll(".theme-item");
const html = document.querySelector("html");
const themeSwitcher = new ThemeSwitcher(themes, html);
```

OUTPUT

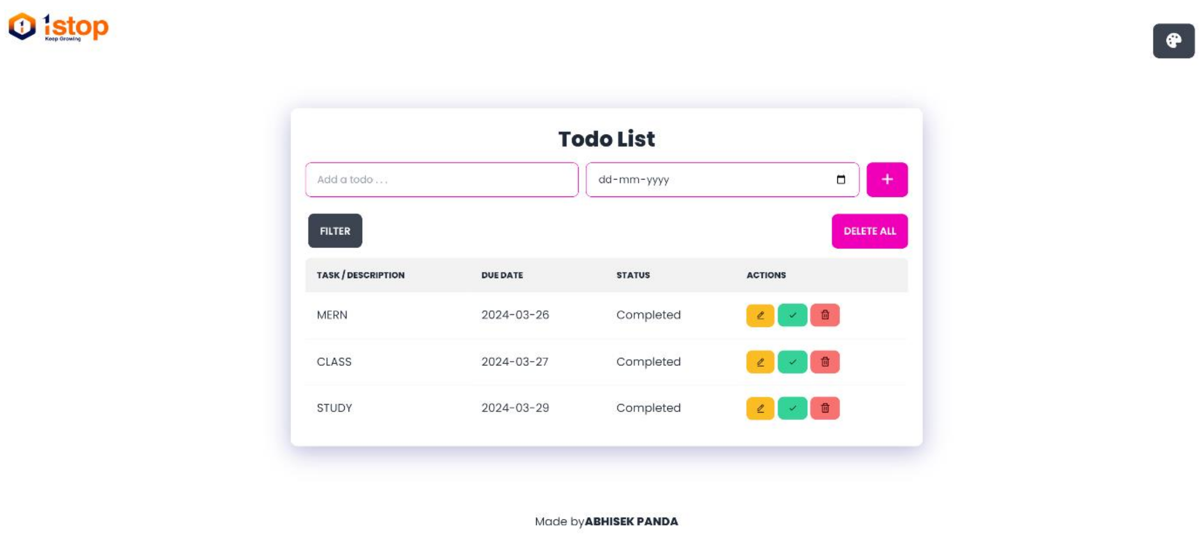


Todo List

TASK / DESCRIPTION	DUE DATE	STATUS	ACTIONS

No task found

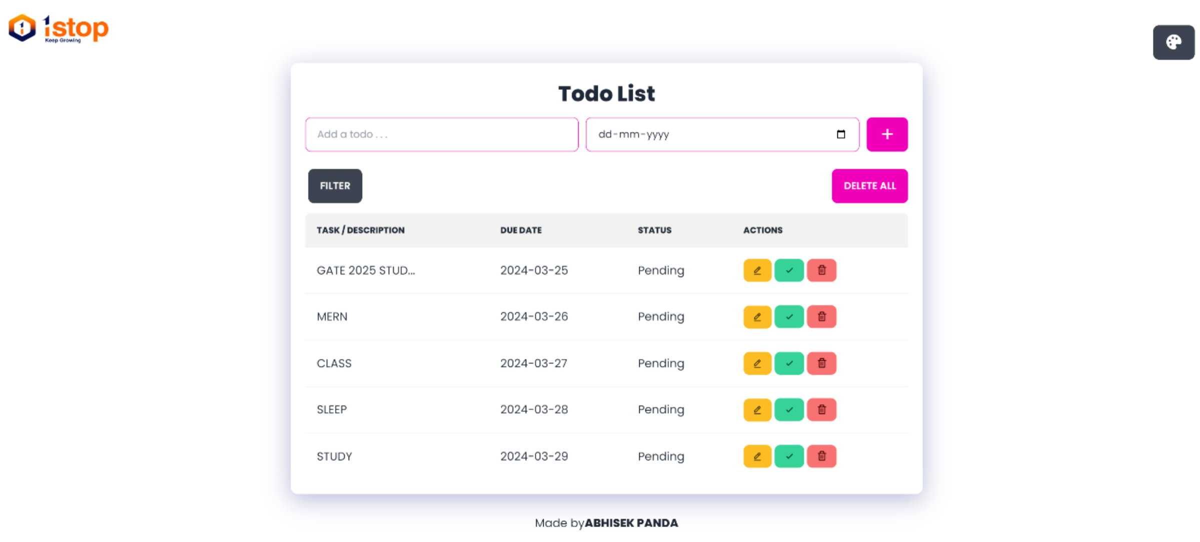
Made by**ABHISEK PANDA**



Todo List

TASK / DESCRIPTION	DUE DATE	STATUS	ACTIONS
MERN	2024-03-26	Completed	
CLASS	2024-03-27	Completed	
STUDY	2024-03-29	Completed	

Made by**ABHISEK PANDA**



Todo List

TASK / DESCRIPTION	DUE DATE	STATUS	ACTIONS
GATE 2025 STUD...	2024-03-25	Pending	
MERN	2024-03-26	Pending	
CLASS	2024-03-27	Pending	
SLEEP	2024-03-28	Pending	
STUDY	2024-03-29	Pending	

Made by**ABHISEK PANDA**

1stop
Keep Growing

Todo List

Add a todo . . .

dd-mm-yyyy

+

FILTER

DELETE ALL

TASK / DESCRIPTION	DUE DATE	STATUS	ACTIONS
GATE 2025 STUD...	2024-03-25	Pending	✓
SLEEP	2024-03-28	Pending	✓

Made by**ABHISEK PANDA**

CONCLUSION

In creating this To Do list application, we've utilized fundamental web development technologies: HTML, CSS, and JavaScript. This project exemplifies the power and simplicity of these languages in crafting interactive and dynamic web experiences.

HTML provided the structure of our application, defining the layout and content hierarchy. With the use of semantic tags like `<div>`, `<input>`, `<button>`, and `<ul>`, we organized our elements in a meaningful way, enhancing accessibility and maintainability.

CSS came into play for styling our application, making it visually appealing and user-friendly. Through CSS rules, we defined the appearance of elements, including fonts, colors, margins, and padding. The styling not only enhances the aesthetic appeal but also improves the usability and readability of the application.

JavaScript added interactivity to our To Do list, making it functional and dynamic. By leveraging DOM manipulation, we were able to interact with HTML elements, respond to user actions, and update the UI in real-time. The `addTask()` function, for instance, enables users to add tasks dynamically to the list, providing instant feedback and enhancing the user experience.

Moreover, this project serves as a practical exercise in web development, reinforcing key concepts such as event handling, element manipulation, and input validation. By building a To Do list, we've gained hands-on experience in implementing common features found in many web applications, laying a solid foundation for more complex projects in the future.

As we conclude this project, it's important to recognize that the journey doesn't end here. Web development is a constantly evolving field, with new technologies and best practices emerging regularly. Thus, this project serves as a stepping stone for further learning and exploration, empowering us to tackle more advanced challenges and create innovative web solutions.

In essence, this To Do list project encapsulates the essence of web development: creativity, problem-solving, and continuous growth. It's a testament to the limitless possibilities offered by HTML, CSS, and JavaScript, and a reminder of the endless opportunities awaiting those who embark on the journey of web development.

MY PROJECT LIVE LINK- https://65ffd86789da71a6df192eae--tiny-kelpie-7652ec.netlify.app/

MY GIT HUB LINK WHERE THE CODE IS UPLOADED-
https://github.com/abhisek2004/-1Stop-Internship--WD.git

MY LINKEDIN LINK WHERE I POSTED THE VIDEO OF PROJECT WHICH I DID-
https://www.linkedin.com/posts/abhisekpanda2004_webdevelopment-internship-codingjourney-activity-7178370178982797313-zdA-?utm_source=share&utm_medium=member_desktop

DOCUMENTATION POST LINK IN LINKEDIN-

https://www.linkedin.com/posts/abhisekpanda2004_html-to-do-list-documentation-activity-7178373456856584193-ukWl?utm_source=share&utm_medium=member_desktop