CAP:774 STYLING AND SCRIPTING FOR WEB

SUBMIT →

To

→ Simranpreet Kaur Mam
(Assistant Professor)
30724

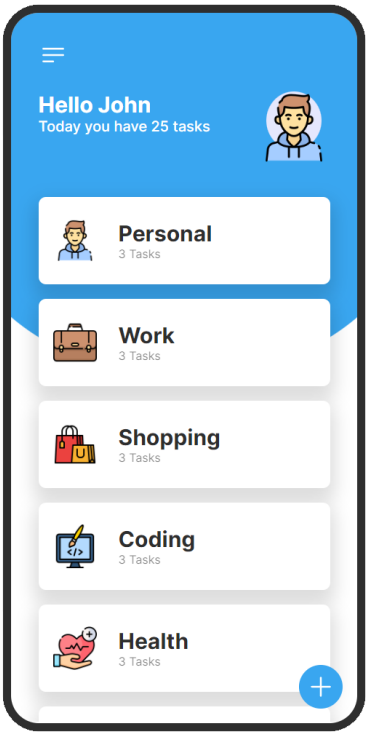
PROJECT TITLE-

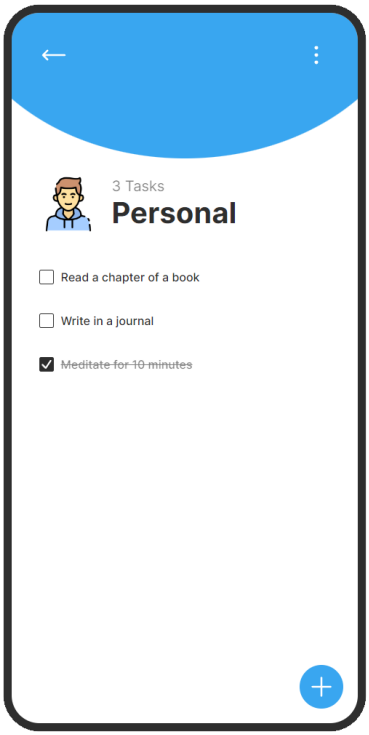
TO-DO LIST APP

PREPARED BY		
STUDENT'S NAME	**ROLL NUMBER**	**REGISTRATION NO.**
SATISH KUMAR	RD2215B65	1220794
EKHLAKH AHMAD	RD2215B58	12209166
SUKHDEEP KAUR	RD2215B57	12209123

School of Computer Application
Lovely Professional University, Phagwara

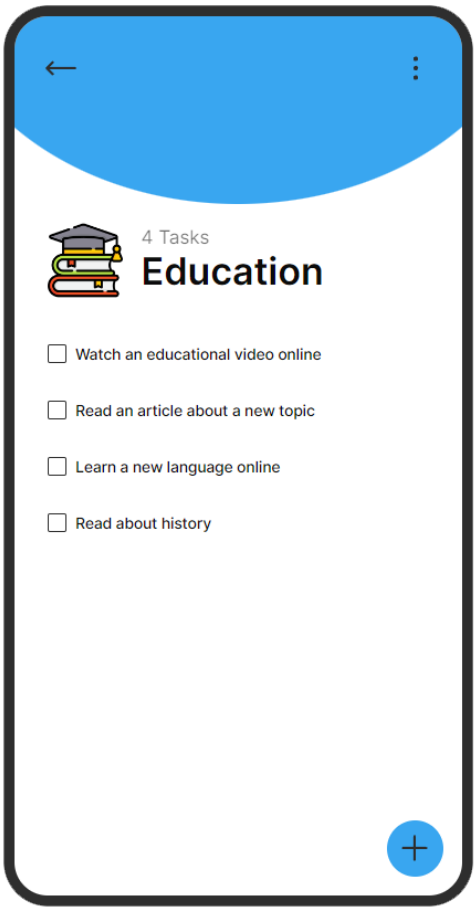
Table of Contents





TODO LIST IN

HTML CSS JS



1.Abstract

The To-Do List Web Application is a user-friendly and intuitive task management tool designed to help individuals organize their daily activities effectively. This report provides an overview of the project, highlighting its key features, advantages, purpose, scope, modules, and a conclusion.

2.Introduction

In today's fast-paced world, staying organized and on top of tasks is essential for productivity. The To-Do List Web Application offers a simple yet powerful solution for managing tasks and categories, allowing users to create, track, and complete tasks efficiently. This report delves into the details of this web application, explaining its functionality and the reasons behind its development.

3.How this program working:

Basically, this program uses local storage to save all the data, allowing tasks and categories to be remembered even when the web page is closed and reopened.

Local storage is a feature provided by web browsers that allows web applications, like my To-Do List App, to store small pieces of data on the user's computer. This data is saved as key-value pairs and remains accessible even if the user closes the web page or refreshes it. In my program, local storage is used to store information about tasks and categories, which means that the tasks you add and the categories you create are stored in the browser's local storage. This ensures that your tasks and categories are not lost when you leave the app and return to it later. It's a convenient way to persist and recall data, making your app more user-friendly and efficient.

Here's how it works in simple words:

1. Saving Data to Local Storage:

- ✓ When you add a new task or make changes to existing tasks (like marking them as completed), my program takes this task data (like the task name, category, and completion status) and stores it in local storage.
- ✓ Think of local storage as a small box where your app can keep things. It's like putting a note in a little box so you can find it later.

2. Retrieving Data from Local Storage:

- ✓ When you open the app again or refresh the page, my program checks this little box (local storage) to see if there are any notes (task data) saved.
- ✓ If it finds some notes (task data), it takes them out of the box and displays them on the screen. This way, you can see the tasks you added in the past.

3. Why It's Useful:

- ✓ Local storage is handy because it allows my app to remember your tasks. You don't have to start from scratch each time you open the app.
- ✓ It's like having a to-do list that you can access anytime, even after you've closed the app or turned off your computer.

4. Advantages

The To-Do List Web Application offers several advantages for users, including:

- ✓ **Simplicity:** The app provides an easy-to-use interface for adding, managing, and completing tasks.
- ✓ **Customization:** Users can create and manage categories, tailoring the app to their specific needs.
- ✓ **Persistence:** Tasks are stored in local storage, ensuring data retention even after closing the browser.
- ✓ **Efficiency:** Users can quickly add and track tasks, improving time management.
- ✓ **Visual Representation:** Categories and tasks are visually presented, making it easy to grasp and organize activities.

5. Purpose

The primary purpose of the To-Do List Web Application is to streamline task management and enhance productivity. It enables users to categorize their tasks, set priorities, and visualize their progress. The application serves as a digital assistant, helping individuals remember and efficiently complete their tasks. Whether it's work-related, personal, or educational tasks, this app simplifies the process of staying organized.

6. Scope

The scope of the To-Do List Web Application includes the following features:

- ✓ **Task Management:** Users can add, edit, complete, and delete tasks.
- ✓ **Category Management:** Users can create, select, and switch between categories.
- ✓ **Local Storage:** Tasks are stored locally, ensuring data persistence.
- ✓ **User-Friendly Interface**: The app offers an intuitive and visually appealing interface.
- ✓ **Efficient Organization:** Categories help users organize tasks, making it easy to focus on specific areas of life or work.

7. Modules

1. User Interface (UI)

The UI module is responsible for rendering the web page, including categories, tasks, buttons, and input forms. It interacts with the user, allowing for task and category management.

2. Task Management

This module manages the creation, modification, and deletion of tasks. It also handles the completion status of tasks and saves data to local storage.

3. Category Management

The category management module handles the creation and selection of task categories. It ensures that tasks are associated with the appropriate categories.

4. Local Storage

Local storage is used to save and retrieve task data, providing data persistence across page reloads and browser sessions.

5. Event Handling

This module captures user interactions, such as clicks and form submissions, and triggers appropriate actions in response to these events.

8. Functions:

1. toggleScreen()

- ✓ **Description:** This function toggles the visibility of the category selection screen by adding or removing the **show-category** class from the **.wrapper** element.
- ✓ **Usage:** It's used when the menu button or the back button is clicked to show or hide the category selection screen.

2. toggleAddTaskForm()

- ✓ **Description:** This function toggles the visibility of the add task form, black backdrop, and the add task button by adding or removing the **active** class from these elements.
- ✓ **Usage:** It's used when the add task button is clicked or when the black backdrop is clicked to show or hide the add task form.

3. calculateTotal()

- ✓ **Description:** This function calculates and displays the total number of tasks in the currently selected category and the total number of tasks in all categories.
- ✓ **Usage:** It's called whenever the category or tasks change, ensuring that the totals are up-to-date.

4. renderCategories()

- ✓ **Description:** This function populates the categories section of the app by creating HTML elements for each category based on the data in the **categories** array.
- ✓ **Usage:** It's called during the initialization of the app and when a category is selected.

5. renderTasks()

- ✓ **Description:** This function populates the tasks section of the app, displaying tasks based on the selected category. It also includes event listeners for completing tasks and deleting tasks.
- ✓ **Usage:** It's called when the app initializes and when a category is selected, ensuring that tasks are displayed for the chosen category.

6. saveLocal()

- ✓ **Description:** This function saves the current tasks data to local storage, allowing task data to persist across page reloads and browser sessions.
- ✓ **Usage:** It's called whenever a task is added, modified, or deleted to update the local storage data.

7. getLocal()

- ✓ **Description:** This function retrieves task data from local storage, allowing previously saved tasks to be loaded into the app.
- ✓ **Usage**: It's called during the app's initialization to load any existing task data from local storage.

Event Listeners:

1. menuBtn.addEventListener("click", toggleScreen)

- ✓ **Description:** This event listener triggers the **toggleScreen** function when the menu button is clicked, toggling the category selection screen's visibility.

2. backBtn.addEventListener("click", toggleScreen)

- ✓ **Description**: This event listener triggers the **toggleScreen** function when the back button is clicked, hiding the category selection screen.

3. addTaskBtn.addEventListener("click", toggleAddTaskForm)

- ✓ **Description:** This event listener triggers the **toggleAddTaskForm** function when the add task button is clicked, displaying the add task form.

4. blackBackdrop.addEventListener("click", toggleAddTaskForm)

- ✓ **Description:** This event listener triggers the **toggleAddTaskForm** function when the black backdrop is clicked, hiding the add task form.

5. Checkbox change event listeners

- ✓ **Description:** Event listeners are added to checkboxes for each task to detect changes in the completion status of the task. When a checkbox is clicked, the function associated with it updates the completion status of the task and saves the data.

6. Delete button event listeners

- ✓ **Description:** Event listeners are added to the delete buttons for each task. When a delete button is clicked, the associated task is removed, and the data is updated in local storage.

7. addBtn.addEventListener("click", ...)

- ✓ **Description:** This event listener triggers the addition of a new task when the "Add" button in the add task form is clicked. It checks if the input is valid and, if so, adds a new task and updates the local storage data.

10. Conclusion

The To-Do List Web Application offers a user-friendly solution for efficient task management and organization. It simplifies the process of creating, tracking, and completing tasks, making it an essential tool for individuals seeking better time management and productivity. By incorporating features like category management and local storage, the app enhances the user experience and ensures that task data remains accessible. The project demonstrates the value of user-cantered design and the effective use of web technologies to create a practical and valuable application for daily life.