TITLE: CodTech IT Solutions Internship - Task Documentation: “To-DO LIST” Using CSS, HTML, JAVASCRIPT.

INTERN INFORMATION:

Name: Makili Lakshmi rani

ID: C0D6252

INTRODUCTION:

The Weather Forecast App is a web-based application designed to provide users with up-to-date weather information for their location and any other location they choose. This project combines HTML, CSS, and JavaScript to create an intuitive and visually appealing interface that allows users to easily access and understand weather data.

Features are;

- **Location-Based Forecast**: The app automatically detects the user's location and displays the current weather conditions, including temperature, humidity, wind speed, and more.

- **Search Functionality**: Users can search for weather forecasts for specific locations by entering the city name or ZIP code.

- **Detailed Weather Information**: The app provides detailed weather information for each location, including hourly and daily forecasts, sunrise and sunset times, and atmospheric pressure.

- **Visual Representation**: Weather data is presented using visually appealing graphics and icons to enhance user experience and understanding.

- **Responsive Design**: The app is designed to be responsive and accessible on various devices, including desktops, tablets, and smartphones, ensuring a seamless user experience across different screen sizes.

- **Customization Options**: Users can customize their experience by choosing between different units of measurement (e.g., Celsius or Fahrenheit) and adjusting display preferences.

By leveraging HTML for structure, CSS for styling, and JavaScript for functionality, the Weather Forecast App provides users with a user-friendly and informative platform for accessing weather forecasts anytime, anywhere.

Implementation

- JavaScript Framework: Utilize a modern JavaScript framework for building the frontend application.
- HTML and CSS: Use HTML and CSS for structuring and styling the user interface, ensuring compatibility with various web browsers.
- Responsive Design: Implement responsive design principles to ensure optimal viewing experience across desktop and mobile devices.
- User Interface Components: Utilize UI libraries for designing interactive and visually appealing components.

CODE EXPLAINATION

HTML Structure:

1.The code starts with <!DOCTYPE html>, which declares the document type as HTML5.

2.Inside the <html> tag, it sets the language to English (<html lang="en">).

3.In the <head> section: It sets the character encoding to UTF-9, which is incorrect and should be UTF-8.It specifies compatibility settings for Internet Explorer.It sets the viewport for responsive design.

It provides a title for the webpage.

It links two CSS files: css1.css and Font Awesome CSS from a CDN.

In the <body> section:

There's a container that holds the entire content of the webpage.

Inside the container, there's an <h1> tag displaying "Todo-List".

There's an input field and a button for adding new tasks, wrapped in a div with class input-container.

There's a set of filters displayed as elements with class filter, allowing users to filter tasks based on completion status.

There's a "Delete-All" button.

A container for displaying todo items (with class todos-container) which includes an empty <ul> for listing tasks and an empty <img> tag for displaying an image when the list is empty.

Finally, there's a <script> tag linking to a JavaScript file named script1.js, which presumably adds functionality to the todo list, such as adding, deleting, and filtering tasks.

The code also contains a commented-out section, which seems to be a duplicate of the main HTML structure with minor differences like the title and some text. It's common to use such commented-out sections for reference or testing purposes.

CSS Styling:

Box Sizing Reset: *::before, *::after selectors reset margin, padding, and set box-sizing to border-box for all elements.

Font and Background: body sets font to 'Roboto', a sans-serif font, and applies a background image with center positioning and cover size.

Container Styling: .container sets width, padding, border, and border radius with a backdrop-filter for blur effect.

Input Styling: .todo-input and .add-button style the input field and button for adding tasks.

Task Styling: .todo styles individual tasks with list-style, background color, border, and padding. .todo label styles task labels.

Checkbox Styling: Styling for checkboxes and their associated labels for task completion indication.

Scrollbar Styling: Styles for customizing the scrollbar appearance within the .todos-container.

Filters Styling: .filter styles filter buttons with padding, border, and color. .filter.active applies styling for active filters.

Delete All Styling: .delete-all styles the button for deleting all tasks.

JavaScript Functionality:

The JavaScript functionality for this todo list application typically involves handling user interactions, such as adding tasks, marking tasks as complete, deleting tasks, and filtering tasks based on completion status.

Adding Tasks (add function):

Checks if the input field (inputactivity) is empty. If not, it proceeds; otherwise, it alerts the user to enter a task.

Creates a new list item (<li>) and sets its content to the value entered in the input field.

Appends a close button () to each task for the removal functionality, with a click event listener that hides the task on click.

Marking Tasks as Completed:

Utilizes event delegation by adding a click event listener to the list container (inputlist).

When a task is clicked, the 'checked' class is toggled on the task, changing its appearance to indicate completion.

Removing Tasks:

The close button (with '×') added to each task allows users to remove tasks from the list.

Initially set up in the add function and further facilitated through a click event listener that sets the task's display style to "none", effectively hiding it.

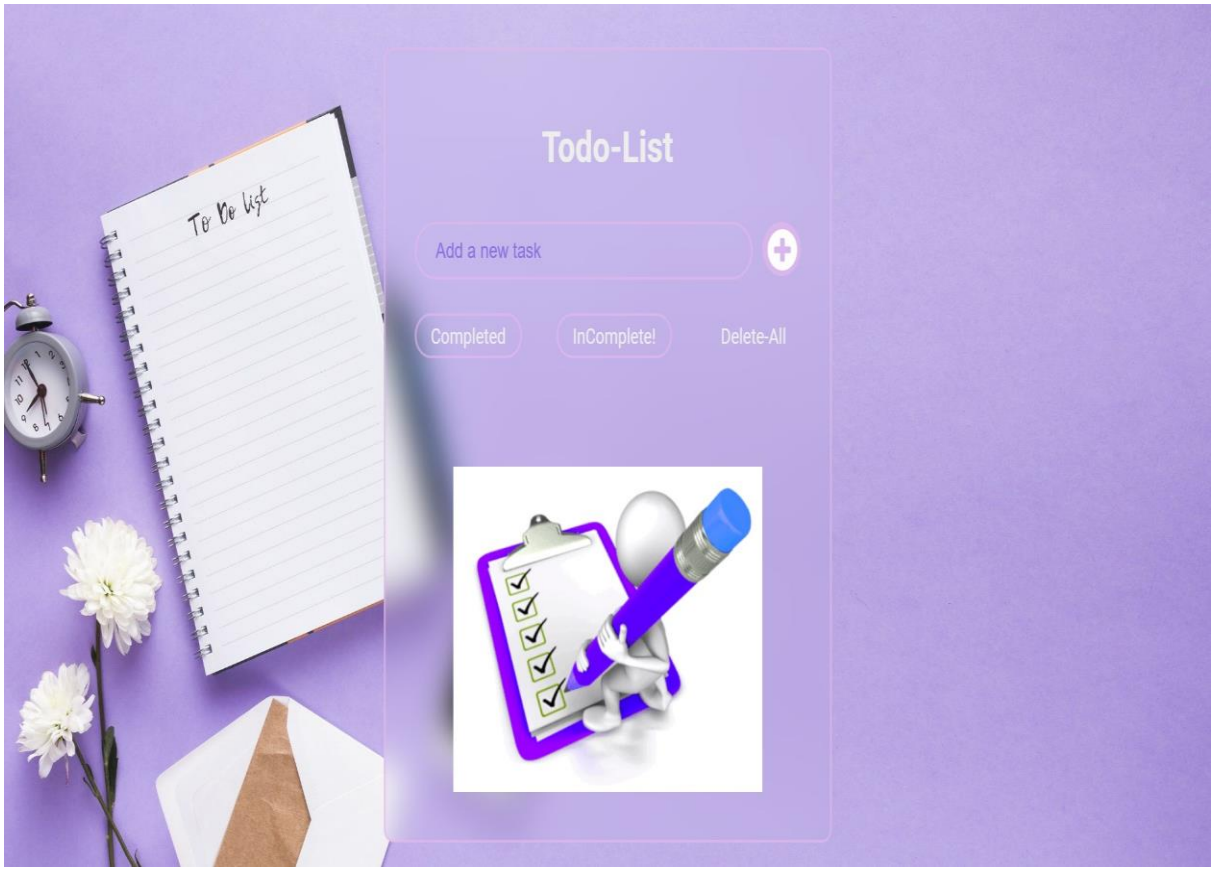
USAGE

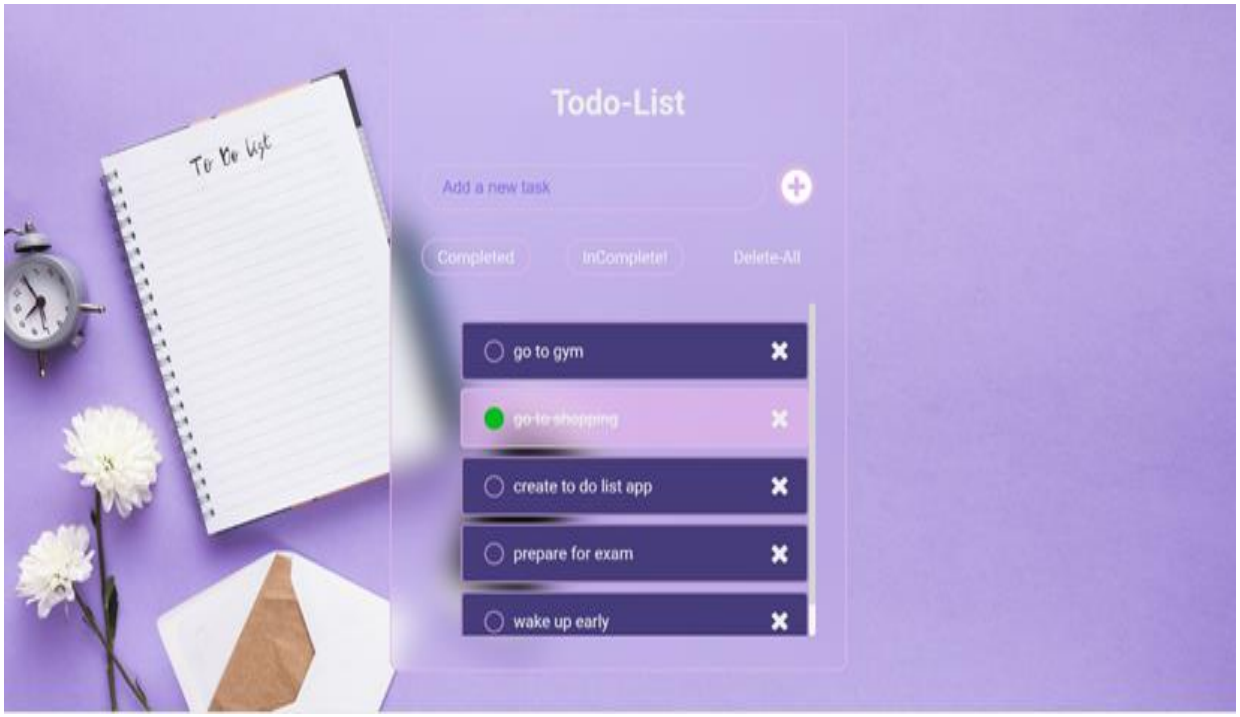
- ✓ Enter a task in the input field and press Enter or click the "Add" button to add it to the list.
- ✓ To mark a task as complete, click the checkbox next to it.
- ✓ To delete a task, click the delete button next to it.
- ✓ To delete all tasks, click the "Delete All" button.
- ✓ Use the filter buttons ("All", "Completed", "Pending") to filter tasks based on their completion status.
- ✓ Interact with the app as needed to manage your tasks effectively.

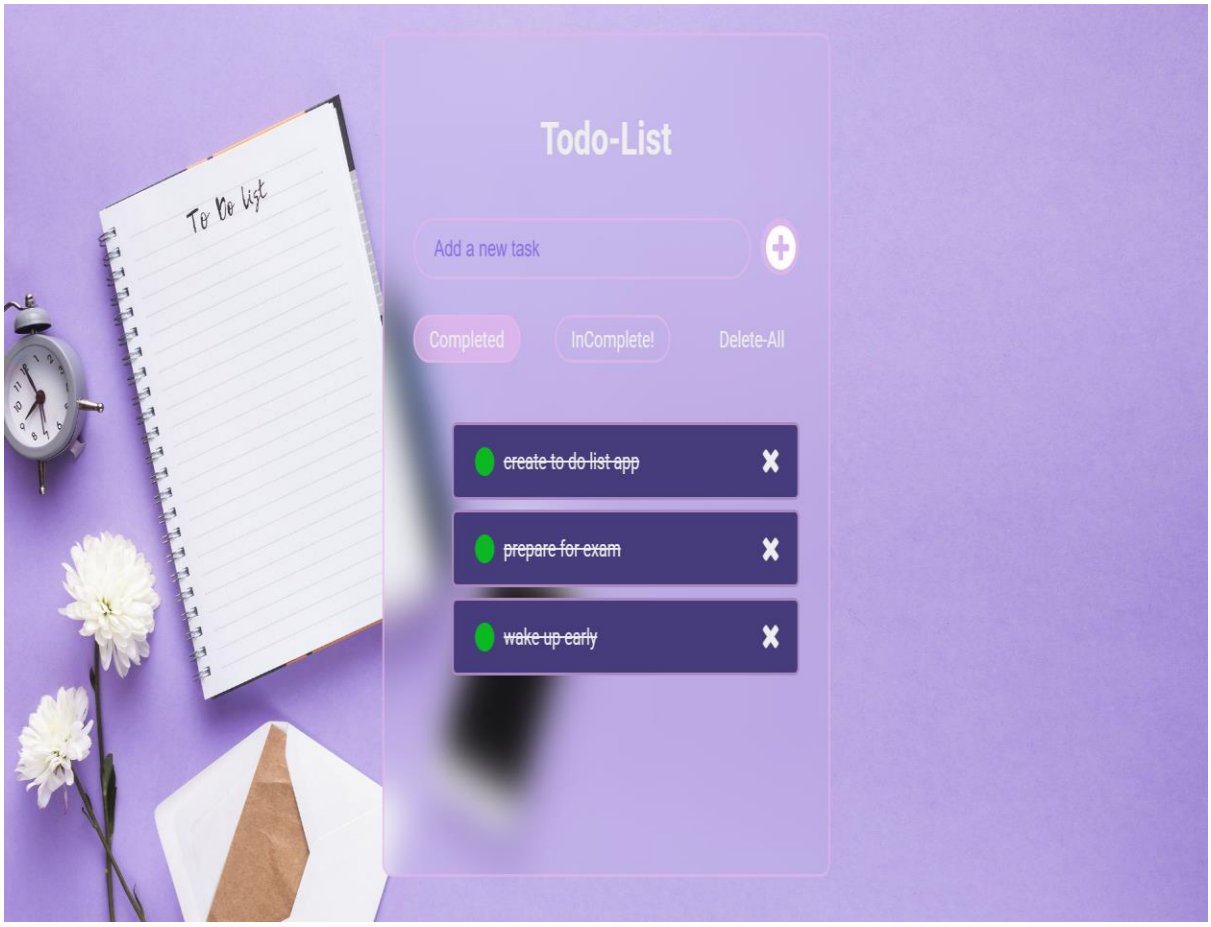
CONCLUSION

In conclusion, the todo list app created using HTML, CSS, and JavaScript provides a simple and intuitive interface for managing tasks efficiently. Through a combination of user-friendly design and interactive functionality, users can easily add, complete, and delete tasks, as well as filter tasks based on their completion status. This project showcases the power of web technologies to create practical and accessible solutions for organizing daily tasks and increasing productivity. With further enhancements and customization, the todo list app can serve as a valuable tool for individuals seeking to streamline their workflow and stay organized.

OUTPUT:




To Do list
Todo-List
Add a new task
Completed
InComplete!
Delete-All
go to gym
go to shopping
create to do list app
prepare for exam
wake up early


To Do list
Todo-List
Add a new task
Completed
InComplete!
Delete-All
create to do list app
prepare for exam
wake up early