PROJECT : TODO APP

1.INTRODUCTION:

To Do List App is a kind of app that generally used to maintain our day-to-day tasks or list everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom. It is helpful in planning our daily schedules. We can add more tasks at any time and delete a task that is completed.

2.ABSTRACT:

This abstract provides an overview of the process involved in building a Todo app—a versatile and essential application for managing tasks and increasing productivity. The Todo app is designed to help users organize their daily activities, set reminders, prioritize tasks, and track their progress effectively. Users can create, edit, and delete tasks, set due dates, and categorize tasks into various lists or tags. the app offers a user-friendly interface to view tasks in different formats, such as a list, calendar, or kanban board. The implementation of the Todo app involves utilizing modern web and mobile technologies. The front-end development will employ HTML, CSS, and JavaScript, with popular frameworks like React or Vue.js to enhance user interaction and responsiveness.

3.purpose of the project:

ToDo List App is a kind of app that generally used to maintain our day-to-day tasks or list everything that we have to do, with the most important tasks at the top of the list, and the least important tasks at the bottom. It is helpful in planning our daily schedules. We can add more tasks at any time and delete a task that is completed.

4.Features:

a.Create (add) a new task or adding a new ToDo in the ToDo List App.

b.See all the tasks or View all the ToDos that were added to the app.

c.Delete any ToDo from the list of ToDos.

5.PROCESS

5.1:SET UP THE PROJECT STRUCTURE:

In this phase of developing the todo list app, you will establish the fundamental framework and organization for the project. Create a directory structure that includes essential folders like "src" for source code, "public" for static files, and potentially others for assets or configuration. Set up necessary build tools and dependencies using tools like Node.js and npm (Node Package Manager). Organize the project to ensure modularity, maintainability, and scalability for future development.

5.2:DESIGN AND IMPLEMENT THE USER INTERFACE:

In this stage, you'll focus on the visual representation and interaction of the todo list app. Design a user-friendly interface that allows users to easily add, edit, and delete tasks. Decide on a consistent and appealing layout, typography, and color scheme. Implement the UI using modern web technologies like HTML, CSS, and JavaScript (and possibly frameworks like React, Angular, or Vue.js). Aim for a responsive design to ensure the app works well on various devices and screen sizes.

5.3:IMPLEMENT TASK MANAGEMENT FUNCTIONALITY:

This step involves adding the core functionality of the todo list app. Develop features that enable users to create new tasks, mark tasks as completed, and update existing tasks. Implement task prioritization, due dates, and task descriptions if needed. Set up data structures to store and manage tasks efficiently. Integrate user interactions with the backend to handle task creation, retrieval, and updates. Ensure that users can perform these operations smoothly and without errors.

5.4:IMPLEMENT TASK FILTERING:

Adding task filtering functionality enhances the user experience by allowing users to view tasks based on specific criteria. Implement filters like "All Tasks," "Completed Tasks," "Active Tasks," and potentially others based on due date, priority, or categories. Make sure the filtering process is intuitive and responsive, providing real-time updates as users select different filters. Efficiently handle the filtering logic on the client-side or interact with the server to retrieve filtered tasks.

6.CODES:

6.1:HTML code:

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>To-Do App</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="heading">

<h1>To-Do-List</h1>

</div>

<div class=" backgroundset">

<div class="container">

<input type="text" id="taskInput" placeholder="Enter your task...">

<input type="date" id="taskDate">

<input type="time" id="taskTime">

<button onclick="addTask()">Add Task</button>

<button onclick="displayCompletedTasks()">Show Completed Tasks</button>

<button onclick="displayActiveTasks()">Show Active Tasks</button>

</div>

<div class="task-grid" id="taskList">

</div>

</div>

<audio id="alarmSound" src="Funny alarm sound.mp3" preload="auto"></audio>

<script src="script.js"></script>

</body>

</html>

6.2:CSS code:

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

}.container {

max-width: 600px;

margin: 50px auto;

padding: 20px;

border: 1px solid #000000;

border-radius: 5px;

padding-right: 20px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.9);

background-color: white;

}

h1 {

text-align: center;

height: 150px;

color: rgb(5, 5, 5);

text-align: center;

font-size: 50px;

line-height: 150px;

margin: 0;

font-family: 'Great Vibes', cursive;

}

.task-box.completed {

background-color: #4b4a4a;

text-decoration: line-through;

}

input[type="text"] {

width: 100%;

padding: 10px;

margin-bottom: 10px;

text-align: center;

font-family: 'Great Vibes', cursive;

font-size: 20px;

padding-left: 1px;

padding-right: 1px;

}

button {

padding: 10px;

background-color: #4CAF50;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

}.task-grid {

display: grid;

grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));

grid-gap: 10px;

margin-left: 400px;

margin-right: 400px;

/\* display: inline-block; \*/

}

.task-box {

padding: 10px;

background-color: #f9f9f9;

border: 1px solid black;

border-radius: 20px;

text-align: center;

font-family: 'Indie Flower', cursive;

margin-top: 15px;

box-shadow: 0 2px 4px rgba(0, 0, 0, 0.9)

}

.delete-btn,

.update-btn,

.complete-btn {

padding: 5px 10px;

border: none;

border-radius: 5px;

cursor: pointer;

margin-right: 2%;

}

.complete-btn{

margin-top: 5px;

}

.delete-btn {

background-color: #f44336;

color: white;

}

.update-btn {

background-color: dodgerblue;

color: white;

}

body{

background-image:url(backgrouund\_11zon11.jpeg) ;

background-size: cover;

}

6.3:JS code:

const taskInput = document.getElementById('taskInput');

const taskDateInput = document.getElementById('taskDate');

const taskTimeInput = document.getElementById('taskTime');

const taskList = document.getElementById('taskList');

document.addEventListener('DOMContentLoaded', () => {

loadTasksFromLocalStorage();

});

function addTask() {

const taskText = taskInput.value.trim();

const taskDate = taskDateInput.value;

const taskTime = taskTimeInput.value;

const currentDateTime = new Date();

const taskDateTime = new Date(taskDate);

const [hours, minutes] = taskTime.split(':');

taskDateTime.setHours(hours, minutes);

if (taskText === '') {

alert('Please enter a valid task.');

return;

}

if (taskDateTime < currentDateTime) {

alert('Please check the date and time.');

return;

}

if (taskText !== '') {

const listItem = document.createElement('div');

listItem.className = 'task-box';

listItem.innerHTML = `

<div>

<span><b>${taskText}</b></span>

<br />

<span class="task-datetime">${formatDate(taskDate)} at ${formatTime(taskTime)}</span>

</div>

<button class="update-btn" onclick="updateTask(this)">Update</button>

<button class="delete-btn" onclick="deleteTask(this)">Delete</button>

<button class="complete-btn" onclick="completeTask(this)">Complete</button>

`;

taskList.appendChild(listItem);

taskInput.value = '';

taskDateInput.value = '';

taskTimeInput.value = '';

saveTasksToLocalStorage();

const timeDifference = taskDateTime - currentDateTime;

setTimeout(() => {

playAlarmSound();

}, timeDifference);

}

}

function playAlarmSound() {

const alarmSound = document.getElementById('alarmSound');

alarmSound.play().then(() => {

}).catch((error) => {

console.error('Sound playback failed:', error);

});

}

function completeTask(button) {

const listItem = button.parentElement;

if (button.innerText === 'Complete') {

button.innerText = 'Not-yet-completed';

} else if (button.innerText === 'Not-yet-completed') {

button.innerText = 'Complete';

}

listItem.classList.toggle('completed');

saveTasksToLocalStorage();

}

function updateTask(button) {

const listItem = button.parentElement;

const taskSpan = listItem.querySelector('span');

const taskDatetime = listItem.querySelector('.task-datetime');

const updateInput = document.createElement('input');

updateInput.type = 'text';

updateInput.value = taskSpan.innerText;

const updateDateInput = document.createElement('input');

updateDateInput.type = 'date';

updateDateInput.value = parseDate(taskDatetime.innerText);

const updateTimeInput = document.createElement('input');

updateTimeInput.type = 'time';

updateTimeInput.value = parseTime(taskDatetime.innerText);

taskDatetime.replaceWith(updateDateInput);

taskSpan.replaceWith(updateInput);

listItem.appendChild(updateTimeInput);

button.innerText = 'Save';

button.onclick = function () {

const newTaskText = updateInput.value.trim();

if (newTaskText !== '') {

taskSpan.innerText = newTaskText;

const taskDate = updateDateInput.value;

const taskTime = updateTimeInput.value;

taskDatetime.innerText = formatDate(taskDate) + ' at ' + formatTime(taskTime);

}

updateInput.replaceWith(taskSpan);

updateDateInput.replaceWith(taskDatetime);

updateTimeInput.remove();

button.innerText = 'Update';

button.onclick = function () {

updateTask(button);

};

};

saveTasksToLocalStorage();

}

function deleteTask(button) {

const listItem = button.parentElement;

taskList.removeChild(listItem);

saveTasksToLocalStorage();

}

function displayCompletedTasks() {

const taskBoxes = document.getElementsByClassName('task-box');

for (const taskBox of taskBoxes) {

if (taskBox.classList.contains('completed')) {

taskBox.style.display = 'block';

} else {

taskBox.style.display = 'none';

}

}

}

function displayActiveTasks() {

const taskBoxes = document.getElementsByClassName('task-box');

for (const taskBox of taskBoxes) {

if (taskBox.classList.contains('completed')) {

taskBox.style.display = 'none';

} else {

taskBox.style.display = 'block';

}

}

}

function saveTasksToLocalStorage() {

const taskBoxes = taskList.getElementsByClassName('task-box');

const tasks = [];

for (const taskBox of taskBoxes) {

const taskText = taskBox.querySelector('b').innerText;

const taskDate = parseDate(taskBox.querySelector('.task-datetime').innerText);

const taskTime = parseTime(taskBox.querySelector('.task-datetime').innerText);

const isCompleted = taskBox.classList.contains('completed');

const task = {

text: taskText,

date: taskDate,

time: taskTime,

completed: isCompleted,

};

tasks.push(task);

}

localStorage.setItem('tasks', JSON.stringify(tasks));

}

function loadTasksFromLocalStorage() {

const tasksJson = localStorage.getItem('tasks');

if (tasksJson) {

const tasks = JSON.parse(tasksJson);

for (const task of tasks) {

const listItem = document.createElement('div');

listItem.className = 'task-box';

listItem.innerHTML = `

<div>

<span><b>${task.text}</b></span>

<br />

<span class="task-datetime">${formatDate(task.date)} at ${formatTime(task.time)}</span>

</div>

<button class="update-btn" onclick="updateTask(this)">Update</button>

<button class="delete-btn" onclick="deleteTask(this)">Delete</button>

<button class="complete-btn" onclick="completeTask(this)">Complete</button>

`;

if (task.completed) {

listItem.classList.add('completed');

}

taskList.appendChild(listItem);

}

}

}

function parseDate(datetimeString) {

return datetimeString.split(' at ')[0];

}

function parseTime(datetimeString) {

return datetimeString.split(' at ')[1];

}

function formatDate(dateString) {

const options = {year: 'numeric' , month: 'long',day: 'numeric' };

const date = new Date(dateString);

return date.toLocaleDateString(undefined, options);

}

function formatTime(timeString) {

const [hours, minutes] = timeString.split(':');

return `${hours.padStart(2, '0')}:${minutes.padStart(2, '0')}`;

}

function isFutureDateTime(dateTimeString) {

const givenDateTime = new Date(dateTimeString);

const currentDateTime = new Date();

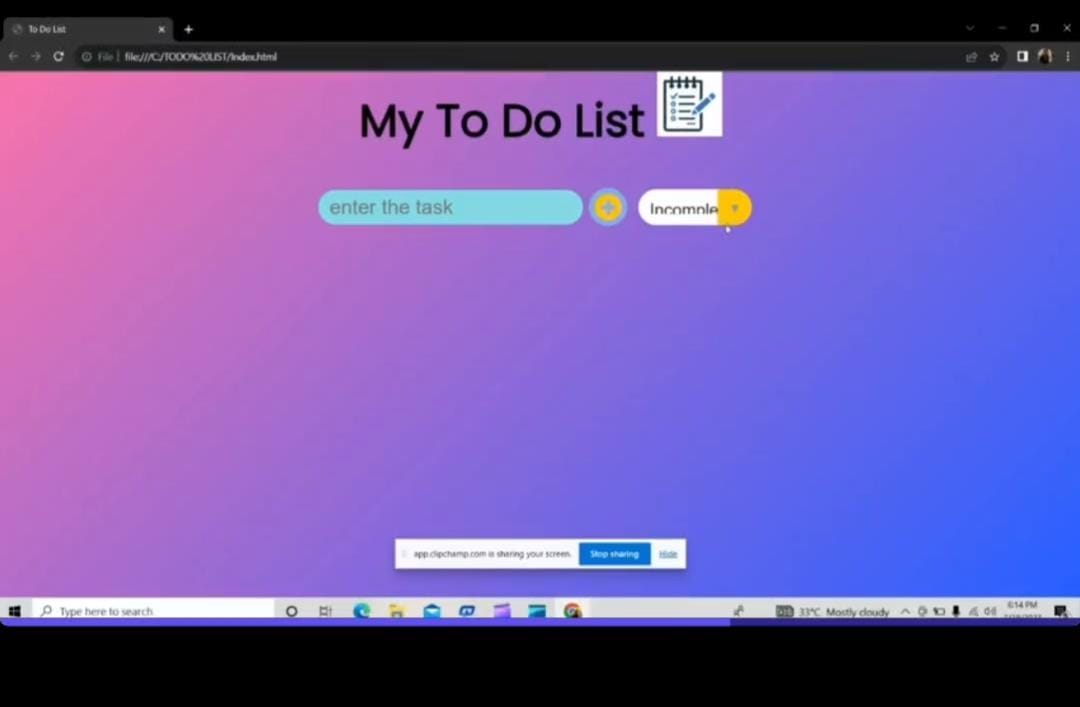
if( givenDateTime < currentDateTime){

prompt('please enter valid date and time');

}

}

Output:



8.Advantages of a To-Do App:

a.Task Organization: To-do apps allow users to organize their tasks efficiently, creating lists, categories, or tags to prioritize and categorize tasks according to their importance and deadlines.

b.Accessibility: Most to-do apps are available on multiple platforms, including smartphones, tablets, and desktop computers, making it easy for users to access their tasks from anywhere with an internet connection.

c.Reminders and Notifications: To-do apps often come with built-in reminders and notifications, helping users stay on top of their tasks and deadlines. These reminders can be set up to notify users at specific times or when they are in a particular location.

d.Collaboration: Many to-do apps offer collaboration features, allowing users to share task lists with others, delegate tasks, and track progress together. This is particularly useful for teams or families who need to coordinate tasks.

e.Time Management: With to-do apps, users can allocate estimated times for each task, helping them better manage their time and prioritize tasks effectively.

9.Disadvantages of a To-Do App:

a.Learning Curve: For some users, learning how to use a new to-do app can take time, and the initial setup may require some effort.

b.Technical Issues: Like any digital tool, to-do apps can experience technical glitches, server outages, or compatibility issues, disrupting users' ability to access and manage their tasks.

c.Distractions: Constant notifications from a to-do app can also become a distraction, pulling users away from their work or other activities.

d.Dependency on Technology: Relying solely on a digital to-do app might be problematic if the user loses access to their devices or if there's a technical failure.

e.Privacy and Security: Some to-do apps store sensitive personal and work-related information, so users need to be cautious about the app's privacy and security measures.

10.CONCLUSION:

In conclusion, the Todo app stands as an essential tool for individuals seeking to optimize their time and maximize their productivity. By assisting users in organizing, prioritizing, and completing tasks efficiently, it empowers them to lead more balanced and fulfilling lives. With continuous updates and improvements, the Todo app will remain a steadfast companion on the journey towards increased productivity and success.users can easily create, edit, and prioritize tasks, ensuring that nothing slips through the cracks. The inclusion of due dates and reminders will help users stay on top of their commitments and deadlines, reducing the risk of forgetting important responsibilities.