Daily Organizer

Application Functionality

The Daily Organizer application is a simple yet effective tool designed to help users manage their day-to-day tasks. The application leverages a user-friendly interface and interactive elements to ensure tasks are added, managed, and completed with ease. Below is a detailed breakdown of its features:

1. Task Management:
    - Users can input tasks into the provided text field and click the "Add" button to include them in the list.
    - Each task is dynamically added to the interface without requiring a page refresh, providing a seamless user experience.
2. Task Completion:
    - Tasks can be marked as completed using a checkbox. Once checked, the task will appear visually distinct with a strikethrough and a different color scheme, indicating its completed status.
3. Task Removal:
    - Users can delete a task at any time by clicking the "Remove" button next to the task. This action ensures that unnecessary or irrelevant tasks do not clutter the task list.
4. Reminders:
    - The application provides the ability to set reminders for individual tasks. Users can specify a date and time for the reminder, which triggers a notification when the set time is reached.
    - The reminders are implemented using JavaScript's setTimeout function and alert the user with a message containing the task details.
5. Dynamic User Interface:
    - The application uses interactive design elements like hover effects, animations, and gradients to enhance usability and aesthetics.
    - Completed tasks, active tasks, and reminders are displayed using clear visual cues

Usage Instructions

The Daily Organizer is designed for simplicity, ensuring that even first-time users can navigate and utilize its features without difficulty. Below is a detailed guide to using the application:

1. Adding a Task:

- Locate the text input field labeled "Enter a task."
- Type the description of the task you want to add (e.g., "Buy groceries").
- Click the Add button.
- The task will immediately appear in the task list below.

2. Marking Tasks as Completed:

- Each task in the list has a checkbox on the left. Check this box to mark a task as completed.
- Completed tasks are visually updated with a strikethrough and a distinct color change to help differentiate them from pending tasks.

3. Removing a Task:

- To delete a task, click the Remove button located to the right of the task.
- The task will be permanently removed from the list, and the interface will update dynamically.

4. Setting Reminders:

- Click the Reminder button next to a task to set a notification for that specific task.
- A prompt will appear asking for the reminder's date in YYYY-MM-DD format (e.g., 2025-01-15).

- Another prompt will request the time in HH:MM format (24-hour clock, e.g., 14:30 for 2:30 PM).
- Upon confirming both inputs, the application will calculate the time difference between the current moment and the specified date/time. If valid, a reminder will be set.
- At the specified time, an alert will notify the user of the task.

5. User Interaction Enhancements:

- Hover over the buttons to see a subtle animation effect that provides feedback on button interaction.
- The task list dynamically updates as tasks are added, completed, or removed.

Setup Process

Setting up the Daily Organizer application is straightforward. Here's a step-by-step guide:

1. Code Deployment:

- Copy the provided code into a text editor such as Notepad, VS Code, or Sublime Text.
- Save the file with an .html extension (e.g., daily_organizer.html).
- Open the file in any modern web browser (such as Chrome, Firefox, or Edge) by double-clicking on it.

2. Environment Requirements:

- Browser: Ensure that you use a browser that supports modern JavaScript features (ES6 or later).
- Internet: The application uses Google Fonts (Poppins). Ensure you have an active internet connection to load the font styles correctly.

3. Testing the Application:

- Once opened in a browser, the application should display a clean interface with an input field, a button for adding tasks, and an empty task list.
- Test each feature (e.g., adding a task, marking it as completed, removing it, and setting reminders) to ensure everything works correctly.

4. Customizing the Application:

- Open the file in a code editor to customize its appearance or behavior.
- CSS: Adjust the colors, fonts, or layouts in the <style> section to match your preferences.
- JavaScript: Add more functionality, such as persistent storage using localStorage or sessionStorage.

5. Advanced Deployment (Optional):

- Host the application on a web server or use platforms like GitHub Pages or Netlify for sharing the tool with others.
- For persistent data storage, integrate backend technologies like Node.js, PHP, or Python with a database like SQLite or MySQL.

Code

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Daily Organizer</title>
    <style>
                                                                    @import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@400;600&display=swap');

        body {
            font-family: 'Poppins', sans-serif;
            margin: 0;
            padding: 0;
            display: flex;
            justify-content: center;
            align-items: center;
            height: 100vh;
            background: linear-gradient(135deg, #B666D2, #d8a7ea);
            color: #fff;
        }
        .container {
```

```
    width: 100%;
    max-width: 450px;
    background-color: #2c2c2c;
    border-radius: 20px;
    box-shadow: 0 10px 30px rgba(0, 0, 0, 0.5);
    padding: 25px;
    text-align: center;
    color: #fff;
}
h1 {
    font-size: 2rem;
    margin-bottom: 20px;
    color: #a5a5a5;
}
.form-group {
    display: flex;
    flex-direction: column;
    margin-bottom: 20px;
}
.form-group input {
    padding: 10px;
    border: 2px solid #444;
    border-radius: 30px;
    outline: none;
```

```
    transition: all 0.3s;
    background-color: #444;
    color: #fff;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);
    margin-bottom: 10px;
}
.form-group input:focus {
    border-color: #85929e;
    background-color: #555;
}
.form-group button {
    padding: 10px 20px;
    border: none;
    background: linear-gradient(to bottom, #85929e, #6b7a84);
    color: #fff;
    border-radius: 15px;
    font-weight: 600;
    cursor: pointer;
    transition: transform 0.2s, box-shadow 0.2s;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);
}
.form-group button:hover {
    transform: translateY(-2px);
    box-shadow: 0 6px 10px rgba(0, 0, 0, 0.4);
```

```
}
ul {
    list-style: none;
    padding: 0;
    margin: 0;
}
li {
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 12px;
    margin-bottom: 10px;
    border: 1px solid #444;
    border-radius: 10px;
    background-color: #3a3a3a;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.3);
    transition: all 0.3s;
}
li.completed {
    text-decoration: line-through;
    color: #8bffa1;
    background-color: #2a2a2a;
}
li:hover {
```

```
    transform: translateY(-2px);
    box-shadow: 0 6px 10px rgba(0, 0, 0, 0.4);
}
li input[type="checkbox"] {
    margin-right: 10px;
}
li button {
    background: linear-gradient(to bottom, #85929e, #6b7a84);
    border: none;
    color: #fff;
    border-radius: 10px;
    padding: 5px 10px;
    cursor: pointer;
    transition: transform 0.2s, box-shadow 0.2s;
    box-shadow: 0 4px 6px rgba(0, 0, 0, 0.2);
}
li button:hover {
    transform: translateY(-2px);
    box-shadow: 0 6px 10px rgba(0, 0, 0, 0.4);
}
li .remove-btn {
    background: linear-gradient(to bottom, #e74c3c, #c0392b);
}
li .remove-btn:hover {
```

```
            transform: translateY(-2px);
            background: linear-gradient(to bottom, #c0392b, #a93226);
        }
        li .reminder-btn {
            background: linear-gradient(to bottom, #85929e, #6b7a84);
        }
        li .reminder-btn:hover {
            transform: translateY(-2px);
            background: linear-gradient(to bottom, #6b7a84, #54616d);
        }
    </style>
</head>
<body>
    <div class="container">
        <h1>Daily Organizer</h1>
        <div class="form-group">
            <input type="text" id="taskInput" placeholder="Enter a task">
            <button onclick="addTask()">Add</button>
        </div>
        <ul id="taskList"></ul>
    </div>

    <script>
        const tasks = [];
```

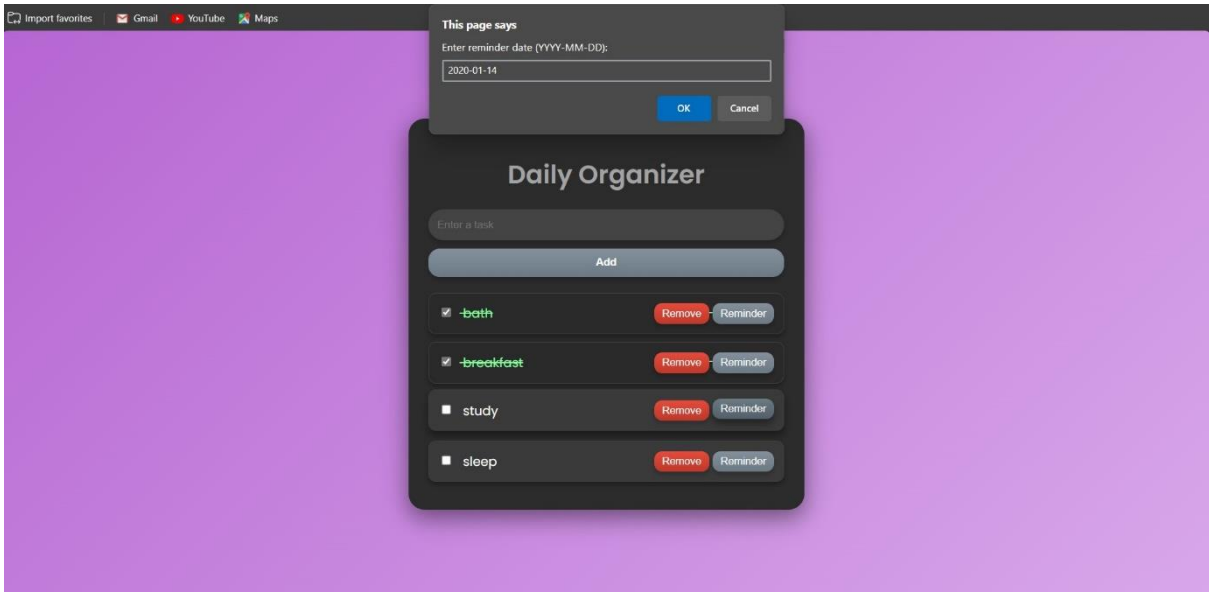
```
function addTask() {
    const taskInput = document.getElementById('taskInput');
    const taskValue = taskInput.value.trim();

    if (taskValue === '') {
        alert('Please enter a valid task.');
        return;
    }

    const taskList = document.getElementById('taskList');
    const listItem = document.createElement('li');
    const taskId = Date.now();

    listItem.setAttribute('data-id', taskId);
    listItem.innerHTML = `
        <div>
            <input type="checkbox" onchange="markComplete(this)">
            <span>${taskValue}</span>
        </div>
        <div>
            <button class="remove-btn" onclick="removeTask(this)">Remove</button>
                                                    <button        class="reminder-btn"
onclick="setReminder(${taskId})">Reminder</button>
        </div>
```

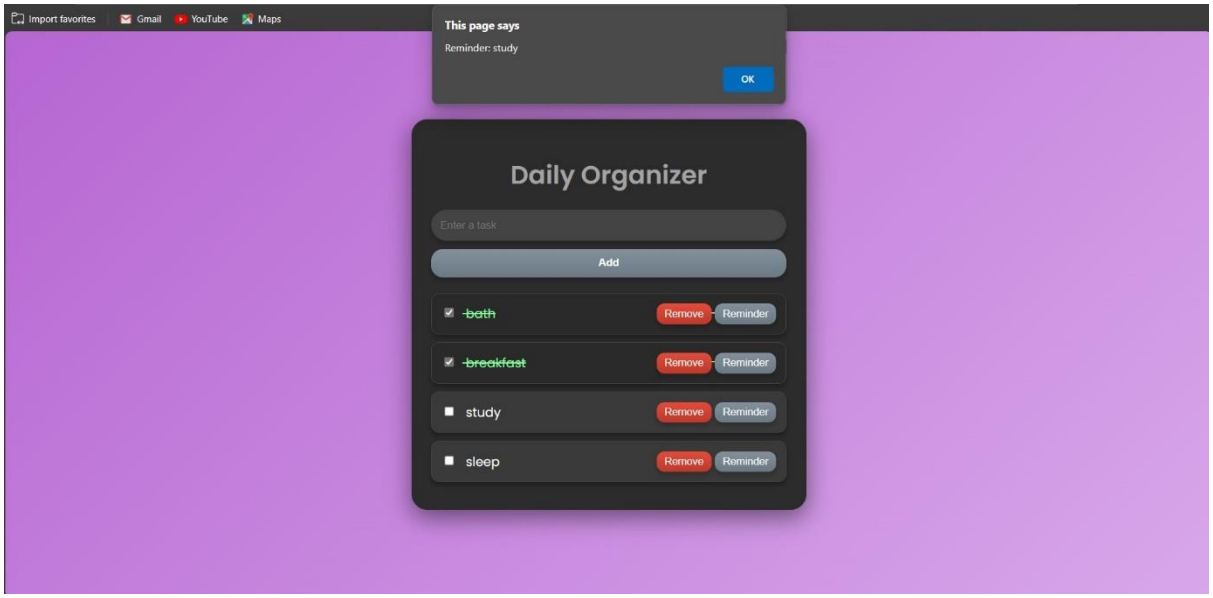
```
    `;

    taskList.appendChild(listItem);
    tasks.push({ id: taskId, name: taskValue, reminder: null });
    taskInput.value = '';
}

function markComplete(checkbox) {
    const listItem = checkbox.closest('li');
    listItem.classList.toggle('completed', checkbox.checked);
}

function removeTask(button) {
    const listItem = button.closest('li');
    const taskId = parseInt(listItem.getAttribute('data-id'));
    const taskIndex = tasks.findIndex(task => task.id === taskId);

    if (taskIndex > -1) {
        tasks.splice(taskIndex, 1);
    }

    listItem.remove();
}
```

```
function setReminder(taskId) {
    const reminderDate = prompt('Enter reminder date (YYYY-MM-DD):');
    const reminderTime = prompt('Enter reminder time (HH:MM, 24-hour format):');

    // Ensure inputs are provided
    if (!reminderDate || !reminderTime) {
        alert('Invalid date or time. Please try again.');
        return;
    }

    // Parse the input into a valid Date object
    const dateTime = new Date(`${reminderDate}T${reminderTime}:00`);

    // Validate the parsed date
    if (isNaN(dateTime.getTime())) {
        alert('Invalid date and time format. Please try again.');
        return;
    }
    const now = new Date();
    const reminderTimeMs = dateTime - now;

    // Debug log: Current time, reminder time, and time difference in milliseconds
    console.log('Current Time:', now);
    console.log('Reminder Time:', dateTime);
```

```
            console.log('Time difference (ms):', reminderTimeMs);
            // Validate that the reminder is in the future
            if (reminderTimeMs <= 0) {
                alert('The reminder time must be in the future.');
                return;
            }
            const task = tasks.find(task => task.id === taskId);
            if (task) {
                // Clear any previous reminder for the task
                if (task.reminder) {
                    clearTimeout(task.reminder);
                }

                // Set the new reminder
                task.reminder = setTimeout(() => {
                    alert(`Reminder: ${task.name}`);
                }, reminderTimeMs);

                alert(`Reminder set for "${task.name}" at ${dateTime.toLocaleString()}.`);
            }
        }
    </script>
</body>
</html>
```

Output:


Import favorites
Gmail
YouTube
Maps
This page says
Enter reminder date (YYYY-MM-DD):
2020-01-14
OK
Cancel
Daily Organizer
Enter a task
Add
bath
breakfast
study
sleep
Remove
Reminder


Import favorites
Gmail
YouTube
Maps
This page says
Reminder: study
OK
Daily Organizer
Enter a task
Add
bath
breakfast
study
sleep
Remove
Reminder