To-Do List Web Application Report

Introduction

This report outlines the development and features of a To-Do List web application. The app is designed to help users manage daily tasks efficiently. It allows users to add, update, delete, and mark tasks as complete while maintaining a history of completed tasks. The application is built using HTML, CSS, and JavaScript and is designed to be responsive and user-friendly.

Objectives

The primary objectives of the To-Do List web application are to:

- Provide an intuitive interface for task management.
- Allow users to quickly add, update, and remove tasks.
- Visually indicate task progress and deadlines.
- Maintain a history of tasks for future reference.
- Ensure responsiveness across devices using modern web design techniques.

Technologies Used

- **HTML5:** Provides the basic structure of the application.
- **CSS3:** Used for styling, layout, and animations. Modern CSS features like Flexbox and media queries ensure a responsive design.
- **JavaScript:** Implements dynamic behavior, including task management, local storage (if needed), and interactive UI elements.
- **Optional Libraries/Frameworks:** Although the core of the app uses vanilla JavaScript, additional libraries (such as Font Awesome for icons) can enhance the user experience.

Key Features

1. **Task Management:**
    - **Add Tasks:** Users can enter a new task along with an optional due date/time.
    - **Delete Tasks:** Easily remove tasks that are no longer needed.
    - **Update Tasks:** Edit tasks as priorities change.
    - **Mark as Complete:** Visually distinguish completed tasks.
2. **Time Management:**

    - **Time Remaining:** For tasks with deadlines, the app displays the time remaining until the task is due.
    - **Reminders:** (Optional) Pop-up notifications or alerts can remind users when a task's deadline is reached.
3. **History and Archiving:**
    - **Task History:** Maintains a list of completed tasks for review and reference.
4. **User Interface:**
    - **Responsive Design:** The app uses Flexbox and media queries to adapt to different screen sizes, ensuring a seamless experience on both mobile and desktop devices.
    - **Animations and Transitions:** Subtle animations enhance interactivity and provide feedback for user actions (e.g., hover effects, smooth scrolling).

User Interface and Design

The application follows a clean, minimalistic design approach:

- **Header & Navigation:** A fixed header at the top contains the navigation menu, enabling quick access to various sections such as "Add Task" and "History."
- **Main Area:** The central area displays the to-do list, an input field for adding new tasks, and interactive buttons to manage tasks.
- **Footer:** The footer, which is fixed at the bottom of the page, includes social media links and additional information, ensuring a polished look.

Visual Elements:

- **Input Fields and Buttons:** Styled with padding, border-radius, and hover effects to create a user-friendly experience.
- **Task Items:** Each task is displayed as a list item with options to edit or delete, with visual cues for completed tasks.
- **Animations:** CSS animations are used for fading effects and smooth transitions when tasks are added or removed.

Implementation Overview

The application is structured into several key components:

- **HTML Structure:** The markup defines the structure for the header, main content area, task input, and footer. Semantic tags ensure better accessibility and SEO.

- **CSS Styling:** The stylesheet includes a global reset, typography, and layout rules. Flexbox is heavily used to manage the layout, and media queries ensure responsiveness.
- **JavaScript Functionality:**
    - Event listeners are attached to buttons for adding, updating, and deleting tasks.
    - The app dynamically updates the DOM to reflect task changes.
    - Optional features (such as storing task history in local storage) can be implemented for persistence.
    - Animations and transition effects are triggered on user interactions to provide a smooth user experience.

Source code:-

1.HTML:-

```
<!DOCTYPE html>
<html lang="en">
<head>
 <meta charset="UTF-8" />
 <meta name="viewport" content="width=device-width, initial-scale=1.0" />
 <title>To-Do List with Calendar</title>
 <link rel="stylesheet" href="style2.css" />
</head>
<body>
 <div class="container">
  <header>
   <h1>To-Do List</h1>
  </header>
  <main>
   <div class="input-area">
    <input type="text" id="todo-input" placeholder="Add a new task..." />
```

```
      <!-- Date is optional; if left blank, current date is used -->
      <input type="date" id="task-date" />
      <input type="number" id="task-hours" placeholder="Hours (optional)" min="0" max="12" />
      <input type="number" id="task-minutes" placeholder="Minutes (optional)" min="0" max="59" />
      <input type="number" id="task-seconds" placeholder="Seconds (optional)" min="0" max="59" />
      <select id="task-am-pm">
        <option value="AM">AM</option>
        <option value="PM">PM</option>
      </select>
      <button id="add-btn">Add</button>
    </div>
    <ul id="todo-list"></ul>
    <button id="show-time-remaining" disabled>Show Time Remaining</button>
    <button id="history-btn">Show History</button>
    <div id="history-area" style="display: none;">
      <h3>Task History</h3>
      <ul id="history-list"></ul>
    </div>
  </main>
 </div>

 <!-- Popup Notification Modal -->
 <div id="popup-notification" class="popup">
  <div class="popup-content">
    <span class="close" id="popup-close">&times;</span>
    <p id="popup-message"></p>
```

```
  </div>
 </div>

 <!-- Footer -->
 <footer>
  <p>
   Designed by
   <a href="https://www.instagram.com/vishwachakma" target="_blank">
    Vishwa Chakma
   </a>
  </p>
  <p>
   Follow me on Instagram:
   <a href="https://www.instagram.com/vishwachakma" target="_blank">
    <i class="fab fa-instagram"></i>
   </a>
  </p>
 </footer>

 <script src="script1.js"></script>
</body>
</html>
```

2.css code:-

```
/* Ensure full height for body and use flex layout */
html, body {
 height: 100%;
 margin: 0;
}
```

```
body {
  display: flex;
  flex-direction: column;
  font-family: Arial, sans-serif;
  background-color: #f4f4f9;
}

/* Container takes available space */
.container {
  flex: 1;
  width: 80%;
  margin: 0 auto;
}

header {
  background-color: #4CAF50;
  color: white;
  padding: 20px;
  text-align: center;
}

h1 {
  margin: 0;
}

.input-area {
  margin: 20px 0;
```

```
  text-align: center;
}

.input-area input,
.input-area select,
.input-area button {
  padding: 10px;
  margin: 5px;
  font-size: 16px;
}

#todo-list {
  list-style: none;
  padding: 0;
}

.todo-item {
  background-color: #fff;
  padding: 10px;
  margin: 10px 0;
  border: 1px solid #ddd;
  border-radius: 5px;
  display: flex;
  flex-wrap: wrap;
  align-items: center;
  justify-content: space-between;
}
```

```
.todo-item span {
  display: inline-block;
}

.countdown {
  margin-left: 10px;
  font-style: italic;
  color: #555;
}

button {
  cursor: pointer;
}

#history-area {
  margin-top: 20px;
  background-color: #fff;
  padding: 20px;
  border-radius: 5px;
  border: 1px solid #ddd;
}

#history-list {
  list-style: none;
  padding: 0;
}

#history-list li {
```

```
  margin: 5px 0;
  padding: 10px;
  background-color: #f9f9f9;
  border: 1px solid #ddd;
  border-radius: 5px;
}

#history-list li button {
  background-color: red;
  color: white;
  border: none;
  padding: 5px;
  cursor: pointer;
  border-radius: 5px;
}

#history-btn,
#show-time-remaining {
  padding: 10px 20px;
  font-size: 16px;
  background-color: #4CAF50;
  color: white;
  border: none;
  cursor: pointer;
  margin: 5px;
}

/* Popup modal styling */
```

```
.popup {
  display: none;
  position: fixed;
  z-index: 100;
  left: 0;
  top: 0;
  width: 100%;
  height: 100%;
  overflow: auto;
  background-color: rgba(0,0,0,0.4);
}

.popup-content {
  background-color: #fff;
  margin: 15% auto;
  padding: 20px;
  border: 1px solid #888;
  width: 80%;
  max-width: 400px;
  border-radius: 10px;
  text-align: center;
}

.close {
  color: #aaa;
  float: right;
  font-size: 28px;
  font-weight: bold;
```

```
  cursor: pointer;
}

.close:hover,
.close:focus {
  color: black;
  text-decoration: none;
  cursor: pointer;
}

/* Footer stays at bottom */
footer {
  background-color: #333;
  color: white;
  text-align: center;
  padding: 10px;
}

footer p {
  margin: 5px;
}

footer a {
  color: #fff;
  text-decoration: none;
  font-weight: bold;
}
```

```
footer a:hover {
  text-decoration: underline;
}

footer i {
  font-size: 24px;
  margin-left: 8px;
  color: #fff;
}

footer i:hover {
  color: #E4405F;
}
```

Javascript code:-

```
let history = [];            // To store task history
let tasks = [];              // To store active tasks
let countdownInterval = null; // To store the countdown interval

// Request notification permission on load
if ("Notification" in window && Notification.permission !== "granted") {
  Notification.requestPermission();
}

// Add event listener for the "Add" button
document.getElementById('add-btn').addEventListener('click', function() {
  const taskInput = document.getElementById('todo-input');
```

```
  const taskDate = document.getElementById('task-date').value; // Optional – if blank,
current date is used
  const hoursVal = document.getElementById('task-hours').value;
  const minutesVal = document.getElementById('task-minutes').value;
  const secondsVal = document.getElementById('task-seconds').value;
  const taskAmPm = document.getElementById('task-am-pm').value;

  if (!taskInput.value) {
    alert('Please provide a task description.');
    return;
  }

  const now = new Date();
  let reminderTime;

  if (taskDate) {
    // Use the provided date
    reminderTime = new Date(taskDate);
    // Check if the selected date is today
    const isToday = reminderTime.toDateString() === now.toDateString();
    let finalHours;
    if (hoursVal === "") {
      // If hours are not provided and the date is today, default to the current hour;
      // otherwise, default to 0 (midnight) for future dates.
      finalHours = isToday ? now.getHours() : 0;
    } else {
      finalHours = parseInt(hoursVal);
      // Apply AM/PM conversion if hours are provided
```

```
    if (taskAmPm === 'PM' && finalHours !== 12) {
      finalHours += 12;
    } else if (taskAmPm === 'AM' && finalHours === 12) {
      finalHours = 0;
    }
  }
  const finalMinutes = minutesVal === "" ? 0 : parseInt(minutesVal);
  const finalSeconds = secondsVal === "" ? 0 : parseInt(secondsVal);
  reminderTime.setHours(finalHours, finalMinutes, finalSeconds, 0);
} else {
  // If no date is provided, default to the current date/time
  reminderTime = new Date();
  let finalHours = hoursVal === "" ? reminderTime.getHours() : parseInt(hoursVal);
  if (hoursVal !== "") {
    if (taskAmPm === 'PM' && finalHours !== 12) {
      finalHours += 12;
    } else if (taskAmPm === 'AM' && finalHours === 12) {
      finalHours = 0;
    }
  }
  const finalMinutes = minutesVal === "" ? 0 : parseInt(minutesVal);
  const finalSeconds = secondsVal === "" ? 0 : parseInt(secondsVal);
  reminderTime.setHours(finalHours, finalMinutes, finalSeconds, 0);
}

// Validate that the reminder time is in the future
if (reminderTime <= now) {
  alert('You cannot set a reminder for a past time!');
```

```
  return;
}

// Create list item for the task
const li = document.createElement('li');
li.classList.add('todo-item');
const formattedTime = formatTime(reminderTime);

// Create a span for the task text and one for the countdown
const taskText = document.createElement('span');
taskText.textContent = `${taskInput.value} - Due: ${formattedTime}`;

const countdownSpan = document.createElement('span');
countdownSpan.classList.add('countdown');
countdownSpan.textContent = ""; // Will be updated with remaining time

// Create a delete button
const deleteBtn = document.createElement('button');
deleteBtn.textContent = 'Delete';
deleteBtn.addEventListener('click', function() {
  li.remove();
  tasks = tasks.filter(t => t.reminderTime !== reminderTime);
});

// Append elements to the list item
li.appendChild(taskText);
li.appendChild(countdownSpan);
li.appendChild(deleteBtn);
```

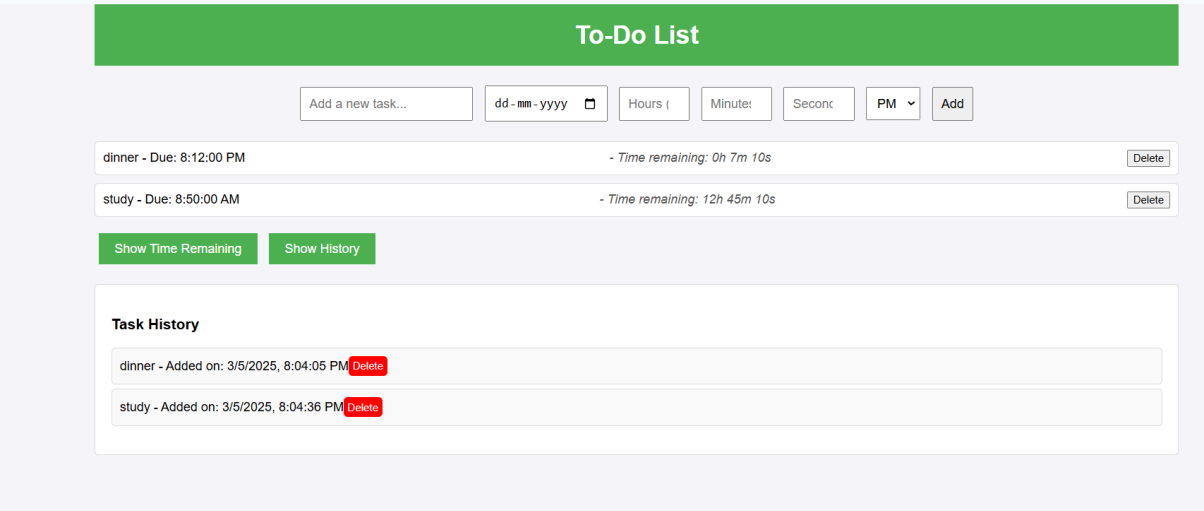
```
  document.getElementById('todo-list').appendChild(li);

  // Save the task with a "notified" flag and a reference to its countdown span
  tasks.push({ task: taskInput.value, reminderTime, notified: false, countdownSpan });
  document.getElementById('show-time-remaining').disabled = false;
  history.push({ task: taskInput.value, timestamp: new Date().toLocaleString() });

  // Reset input fields
  taskInput.value = '';
  document.getElementById('task-date').value = '';
  document.getElementById('task-hours').value = '';
  document.getElementById('task-minutes').value = '';
  document.getElementById('task-seconds').value = '';
});

// Function to format time in a 12-hour format
function formatTime(date) {
  let hours = date.getHours();
  let minutes = date.getMinutes();
  let seconds = date.getSeconds();
  let ampm = hours >= 12 ? 'PM' : 'AM';
  hours = hours % 12 || 12;
  minutes = minutes < 10 ? '0' + minutes : minutes;
  seconds = seconds < 10 ? '0' + seconds : seconds;
  return `${hours}:${minutes}:${seconds} ${ampm}`;
}

// Show time remaining functionality
```

```
document.getElementById('show-time-remaining').addEventListener('click', function() {
  if (tasks.length === 0) return;
  clearInterval(countdownInterval);
  countdownInterval = setInterval(function() {
    let currentTime = new Date();
    tasks.forEach(task => {
      let remainingTime = task.reminderTime - currentTime;
      if (remainingTime <= 0) {
        if (!task.notified) {
          task.notified = true;
          showPopup(`Task "${task.task}" is due!`);
          if (Notification.permission === "granted") {
            new Notification("Task Reminder", { body: `Task "${task.task}" is due!` });
          }
          playNotificationSound();
        }
        task.countdownSpan.textContent = " - Reminder time has passed!";
      } else {
        let remainingHours = Math.floor(remainingTime / (1000 * 60 * 60));
        let remainingMinutes = Math.floor((remainingTime % (1000 * 60 * 60)) / (1000 * 60));
        let remainingSeconds = Math.floor((remainingTime % (1000 * 60)) / 1000);
        task.countdownSpan.textContent = ` - Time remaining: ${remainingHours}h
${remainingMinutes}m ${remainingSeconds}s`;
      }
    });
  }, 1000);
});
```

```
// Function to play a notification sound
function playNotificationSound() {
  const audio = new Audio('https://www.soundjay.com/button/beep-07.wav');
  audio.play();
}

// Function to show the pop-up modal with a custom message for 10 seconds
function showPopup(message) {
  const popup = document.getElementById('popup-notification');
  const popupMessage = document.getElementById('popup-message');
  popupMessage.textContent = message;
  popup.style.display = "block";
  // Automatically hide the popup after 10 seconds (10000ms)
  setTimeout(() => {
    popup.style.display = "none";
  }, 10000);
}

// Close the popup when the close button is clicked
document.getElementById('popup-close').addEventListener('click', function() {
  document.getElementById('popup-notification').style.display = "none";
});

// Show History functionality
document.getElementById('history-btn').addEventListener('click', function() {
  const historyArea = document.getElementById('history-area');
  const historyList = document.getElementById('history-list');
```

```
  historyArea.style.display = historyArea.style.display === 'none' || historyArea.style.display === '' ? 'block' : 'none';
  historyList.innerHTML = '';
  history.forEach((item, index) => {
    const historyItem = document.createElement('li');
    historyItem.textContent = `${item.task} - Added on: ${item.timestamp}`;
    const deleteHistoryBtn = document.createElement('button');
    deleteHistoryBtn.textContent = 'Delete';
    deleteHistoryBtn.addEventListener('click', function() {
      history.splice(index, 1);
      historyItem.remove();
    });
    historyItem.appendChild(deleteHistoryBtn);
    historyList.appendChild(historyItem);
  });
});
```

Output:-



Future Enhancements

To further improve the application, the following enhancements could be considered:

- **User Authentication:** Allow users to create accounts and store their tasks on a server.
- **Cloud Storage:** Implement server-side storage for task data, enabling access across multiple devices.
- **Advanced Notifications:** Integrate browser notifications or email alerts for upcoming deadlines.
- **Analytics:** Provide insights into task completion rates and productivity trends.
- **Collaboration:** Enable task sharing and collaboration among team members.

Conclusion

The To-Do List web application is a robust and efficient tool designed to streamline task management. Built with HTML, CSS, and JavaScript, it provides an engaging and responsive user experience. With its intuitive interface, smooth animations, and practical features, this application serves as an excellent example of modern web development practices. Future enhancements can further extend its functionality and adaptability to user needs.