Todo Application Documentation

Introduction

The Todo Application is a web-based task management application that allows users to create, manage, and track their daily tasks. It provides a simple and intuitive interface for users to add, complete, and delete tasks. Additionally, the application includes a set reminder feature that alerts users with an audio notification when the set reminder time matches the current time.

Technologies Used

The Todo Application is built using the following technologies:

- HTML: Markup language used to create the structure of the web application.
- CSS: Styling language used to design the layout and appearance of the application.
- JavaScript: Programming language used to implement the application's logic and functionality.
- Bootstrap: Front-end framework used for responsive design and UI components.
- Font Awesome: Icon toolkit used for displaying icons in the application.
- Local Storage: HTML5 feature used to store user's tasks data locally in the browser.

Features

1. Add Todo Task: Users can add new tasks by entering the task description in the input field and clicking the "Add" button. The task will be displayed in the task list.
2. Set Reminder: Users can set a reminder time for each task by entering the time in the reminder time input field. The reminder time will be displayed next to the task in the task list.
3. Set Reminder Alert: When the set reminder time matches the current time, the application will display an alert notification with the task description and play an audio alert for 3 seconds.
4. Complete Task: Users can mark a task as completed by clicking the checkbox next to the task. The completed task will be displayed with a strikethrough.
5. Delete Task: Users can delete a task by clicking the trash icon next to the task.
6. Save Tasks: The application uses Local Storage to save the user's tasks, so the tasks will persist even after closing the browser.

Usage

1. Adding a Task: Enter the task description in the input field and click the "Add" button to add a new task. Optionally, set a reminder time by selecting the time in the reminder time input field.
2. Completing a Task: Click the checkbox next to a task to mark it as completed. The completed task will be displayed with a strikethrough.
3. Deleting a Task: Click the trash icon next to a task to delete it from the task list.
4. Set Reminder: Enter the reminder time in the reminder time input field to set a reminder for a task. The reminder time will be displayed next to the task in the task list.
5. Set Reminder Alert: When the set reminder time matches the current time, an alert notification with the task description will be displayed, and an audio alert will play for 3 seconds.
6. Save Tasks: Click the "Save" button to save the tasks to Local Storage. The tasks will be loaded automatically when the application is reopened.

Development:

The Todo Application was developed using a combination of HTML, CSS, and JavaScript. The application's UI was designed using Bootstrap for responsiveness and Font Awesome for icons. The application's functionality, including adding, completing, and deleting tasks, was implemented using JavaScript.

The set reminder feature was implemented using JavaScript's `setInterval` function to continuously check if the current time matches the set reminder time. When a match is found, an alert notification is displayed with the task description, and an audio alert is played for 3 seconds.

The application's data is stored locally in the browser's Local Storage, allowing the user's tasks to persist even after closing the browser.

HTML

The HTML structure of the Todo Application includes:
- A container div to hold all the elements of the application.
- Input fields for entering task descriptions and reminder time.
- Buttons for adding tasks and saving tasks to Local Storage.
- An unordered list to display the list of tasks.

CSS

The CSS styling of the Todo Application is designed to create an aesthetically pleasing and user-friendly interface. It includes styles for fonts, colours, margins, padding, and borders to achieve a clean and organized layout.

JavaScript

The JavaScript code is responsible for implementing the functionality of the Todo Application. It includes the following key components:

Task Management:

- A `todoList` array to store the user's tasks data.
- Functions to add new tasks, mark tasks as completed, and delete tasks from the `todoList`.
- A function to get the user's tasks from Local Storage and load them into the application when opened.

```
let todoList = getTodoListFromLocalStorage();
let todosCount = todoList.length;

function onAddTodo() {
  // Get user input values for task description and reminder time
  let userInputElement = document.getElementById("todoUserInput");
  let userInputValue = userInputElement.value;
  let reminderTime = reminderTimeInput.value; // Get reminder time

  // Validate task description input
  if (userInputValue === "") {
    alert("Enter Valid Text");
    return;
  }

  // Increment todosCount to create unique IDs for tasks
```

```
  todosCount = todosCount + 1;
  // Create a new todo object with task details
  let newTodo = {
    text: userInputValue,
    uniqueNo: todosCount,
    isChecked: false,
    reminderTime: reminderTime, // Store reminder time in todo object
  };
  todoList.push(newTodo);
  createAndAppendTodo(newTodo);
  userInputElement.value = "";
  reminderTimeInput.value = ""; // Clear reminder time input after adding todo
}
```

Explanation:
- `todoList`: An array that stores the user's tasks data, which is retrieved from Local Storage using the `getTodoListFromLocalStorage()` function.
- `todosCount`: A variable that keeps track of the number of tasks, used to create unique IDs for each task.
- `onAddTodo()`: A function that handles adding a new task when the "Add" button is clicked. It gets the user's input values for task description and reminder time, validates the task description input, creates a new todo object with task details, adds the new todo to the `todoList`, and updates the UI by calling the `createAndAppendTodo()` function.

Set Reminder Feature:
- A `reminderTimeInput` element to capture the user's set reminder time.
- A `reminderAudio` element to play the audio alert when the set reminder time matches the current time.
- Functions to set and check the reminder time continuously using `setInterval`.
- Functions to play and stop the reminder audio alert.

```
let isReminderSet = false;
function onSetReminder() {
  let reminderTime = reminderTimeInput.value;
  if (reminderTime === "") {
    alert("Enter Valid Reminder Time");
    return;
  }
  isReminderSet = true;
  reminderTimeInput.disabled = true;
  addTodoButton.disabled = true;
  saveTodoButton.disabled = true;
  checkReminderTime();
}
function checkReminderTime() {
  setInterval(function () {
    let currentTime = new Date();
    for (let todo of todoList) {
      let [hour, minute] = todo.reminderTime.split(":");
```

```
      let reminderTime = new Date(currentTime);
      reminderTime.setHours(parseInt(hour), parseInt(minute), 0, 0);

      if (
        !todo.isChecked &&
        currentTime >= reminderTime &&
        currentTime <= reminderTime.getTime() + 3000
      ) {
        playReminderAudio();
        alert(`Task: ${todo.text}\nReminder: Complete Task`);
        stopReminderAudio
    }
  }, 1000);
}
function playReminderAudio() {
  reminderAudio.play();
}
function stopReminderAudio() {
  reminderAudio.pause();
  reminderAudio.currentTime = 0;
  isReminderSet = false;
  reminderTimeInput.disabled = false;
  addTodoButton.disabled = false;
  saveTodoButton.disabled = false;
  stopAudioButton.style.display = "none";
```

Explanation:

- `isReminderSet`: A boolean variable that keeps track of whether the user has set a reminder time.
- `onSetReminder()`: A function that handles setting a reminder time when the "Add" button is clicked. It validates the reminder time input, sets `isReminderSet` to true, disabled input and buttons, and calls the `checkReminderTime()` function to continuously check the reminder time.
- `checkReminderTime()`: A function that uses `setInterval` to continuously check if the set reminder time matches the current time. If the condition is met, it plays the reminder audio, displays an alert with the task description, and stops the reminder audio after 3 seconds.
- `playReminderAudio()`: A function that plays the reminder audio when called.
- `stopReminderAudio()`: A function that stops the reminder audio, resets `isReminderSet`, and enables input and buttons when called.

Event Listeners:

- Event listeners to handle user actions such as adding tasks, completing tasks, deleting tasks, and saving tasks.

```
reminderTimeInput.addEventListener("change", function () {
    if (isReminderSet) {
      isReminderSet = false;
      reminderTimeInput.disabled = false;
      addTodoButton.disabled = false;
```

```
      saveTodoButton.disabled = false;
      stopAudioButton.style.display = "none";
    }
  });
  addTodoButton.onclick = function () {
    onAddTodo();
  };
  stopAudioButton.onclick = function () {
    stopReminderAudio();
  };
```

Explanation:
- `reminderTimeInput.addEventListener`: An event listener that listens for changes in the reminder time input field. If `isReminderSet` is true (i.e., a reminder time is set), it resets the reminder settings and enables input and buttons.
- `addTodoButton.onclick`: An event listener that listens for clicks on the "Add" button and calls the `onAddTodo()` function to add a new task.
- `stopAudioButton.onclick`: An event listener that listens for clicks on the "Stop Audio" button and calls the `stopReminderAudio()` function to stop the reminder audio.

Local Storage:
The application uses Local Storage to store the user's tasks data locally in the browser. When the user adds, completes, or deletes a task, the `todoList` is updated and saved to Local Storage. When the application is reopened, the user's tasks are loaded from Local Storage.

```
function getTodoListFromLocalStorage() {
    let stringifiedTodoList = localStorage.getItem("todoList");
    let parsedTodoList = JSON.parse(stringifiedTodoList);
    if (parsedTodoList === null) {
      return [];
    } else {
      return parsedTodoList;
    }
  }
  saveTodoButton.onclick = function () {
    localStorage.setItem("todoList", JSON.stringify(todoList));
  };
```

Explanation:
- `getTodoListFromLocalStorage()`: A function that retrieves the user's tasks data from Local Storage. If no data is found, it returns an empty array.
- `saveTodoButton.onclick`: An event listener that listens for clicks on the "Save" button and saves the user's tasks data to Local Storage in JSON format when clicked.

Conclusion
The Todo Application provides users with a simple and efficient way to manage their daily tasks. With features such as adding, completing, and deleting tasks, as well as the set reminder alert, users can stay organised and on top of their tasks. The application's use of Local Storage ensures that tasks are saved and easily accessible even after closing the browser.

The application can be further enhanced with additional features such as task filtering, task prioritisation, and user authentication to provide a comprehensive task management solution. As a tech professional, this project demonstrates your ability to develop a practical and user-friendly application using modern web technologies.