TO-DO LIST APPLICATION

PROJECT DOCUMENTATION

---

1. Introduction

- **1.1 Overview**

The To-Do List application is a simple web-based tool that allows users to manage their tasks efficiently. Users can add new tasks, mark tasks as complete, and delete tasks. It provides a user-friendly interface for organizing daily activities or long-term projects.

- **1.2 Purpose**

The purpose of this project is to create a practical and interactive tool for task management using web technologies. It demonstrates fundamental concepts of DOM manipulation, event handling, and basic user interface design.

2. Project Goals

- **2.1 Core Objectives**
  - Allow users to add new tasks to the list.
  - Enable users to mark tasks as complete.
  - Provide a way for users to delete tasks from the list.
  - Offer a clean and intuitive user interface.
- **2.2 Functional Scope**
  - The application focuses on client-side functionality.
  - Tasks are stored in the browser's memory and are not persisted across sessions (e.g., in a database or local storage).

3. Features and Working

- **3.1 Features**
  - **Add Task:** Users can add new tasks by typing them into an input field and clicking the "Add Task" button or pressing Enter.
  - **Mark as Complete:** Users can mark a task as complete by clicking on the task item. Completed tasks are visually distinguished with a strikethrough.
  - **Delete Task:** Each task item has a delete button (❌) that allows users to remove the task from the list.
  - **Clear All Tasks:** A "Clear All" button removes all tasks from the list.

- **3.2 Working**

1. **Adding a Task:**
    - The user enters the task description in the input field.
    - When the "Add Task" button is clicked (or Enter is pressed), the addTask() function is called.
    - addTask() retrieves the task text from the input field, creates a new <li> element, sets its text content to the task, and adds event listeners for marking as complete and deleting.
    - The new <li> element is appended to the <ul> (task list).
    - The input field is cleared.
2. **Marking a Task as Complete:**
    - When a user clicks on a task (<li>), an event listener toggles the 'done' class on the <li> element.
    - The 'done' class adds a strikethrough to the text, visually indicating completion.
3. **Deleting a Task:**
    - Each task (<li>) has a delete button.
    - When the delete button is clicked, its event listener removes the corresponding <li> element from the <ul> (task list).
4. **Clearing All Tasks:**
    - When the "Clear All" button is clicked, the clearTasks() function is called.
    - clearTasks() sets the innerHTML of the <ul> (task list) to an empty string, effectively removing all tasks.

4. Technology Used

- **4.1 HTML (Hypertext Markup Language)**
    - Provides the structure of the web page, including the input field, task list (<ul>), buttons, and container elements.
- **4.2 CSS (Cascading Style Sheets)**
    - Handles the styling and visual presentation of the page.
    - Defines the layout, colors, fonts, and overall aesthetics.

- **4.3 JavaScript**
  - Implements the dynamic behavior of the application.
  - Handles user interactions, manipulates the DOM (Document Object Model), and manages the task list.

5. Implementation Details

- **5.1 HTML Structure**
  - <div class="container">: Wraps the entire application.
  - <h1>: Displays the title "To-Do List".
  - <input id="taskInput" type="text">: The input field for entering new tasks.
  - <ul id="taskList">: The unordered list that displays the tasks.
  - <div class="buttons">: Contains the "Add Task" and "Clear All" buttons.
  - <button onclick="addTask()">: The "Add Task" button.
  - <button onclick="clearTasks()">: The "Clear All" button.
- **5.2 CSS Styling**
  - **Background:** A gradient background from parrot green to pink.
  - **Container:** A dark-themed container with rounded corners and a shadow.
  - **Task List:** Styled <ul> and <li> elements with appropriate colors and spacing.
  - **Completed Tasks:** The .done class is used to apply a strikethrough to completed tasks.
  - **Buttons:** Styled buttons with hover effects.
- **5.3 JavaScript Logic**
  - **Variables:**
    - taskInput: References the input field.
    - taskList: References the unordered list (<ul>).
  - **Functions:**
    - addTask():
      - Gets the task text from the input field.
      - Creates a new <li> element.
      - Sets the text content of the <li> to the task.

        - Adds a click event listener to the <li> to toggle the 'done' class.
        - Creates a delete button (<button>).
        - Adds an event listener to the delete button to remove the <li> from the list.
        - Appends the delete button to the <li>.
        - Appends the <li> to the taskList (<ul>).
        - Clears the input field.
    - clearTasks(): Clears all tasks by setting the innerHTML of the taskList to an empty string.
    - taskInput.addEventListener('keypress', ...): Adds an event listener to the input field to call addTask() when the Enter key is pressed.

6. Theme

- **6.1 Color Palette**
  - Background: Gradient from #00FF7F (Parrot Green) to #FF69B4 (Pink)
  - Container Background: #111 (Dark Gray)
  - Text: #eee (Off-White) for tasks, #777 (Gray) for completed tasks
  - Task Background: #222 (Darker Gray)
  - Button Background: #00FF7F (Parrot Green), #00cc66 (Darker Green on hover)
- **6.2 Typography**
  - Font: 'Segoe UI', sans-serif
- **6.3 Visual Style**
  - Modern and clean design.
  - Vibrant gradient background with a dark-themed container for contrast.
  - Clear visual distinction between active and completed tasks.
  - User-friendly button styles.

7. Code Snippets

- **7.1 Adding a Task**

JavaScript

```
function addTask() {
  const taskText = taskInput.value.trim();
  if (taskText === '') return;

  const li = document.createElement('li');
  li.textContent = taskText;

  li.addEventListener('click', () => {
    li.classList.toggle('done');
  });

  const deleteBtn = document.createElement('button');
  deleteBtn.textContent = '❌';
  deleteBtn.onclick = () => taskList.removeChild(li);
  li.appendChild(deleteBtn);

  taskList.appendChild(li);
  taskInput.value = '';
}
```

- **7.2 Clearing Tasks**

JavaScript

```
function clearTasks() {
  taskList.innerHTML = '';
}
```

- **7.3 HTML Structure for Task Input and List**

HTML

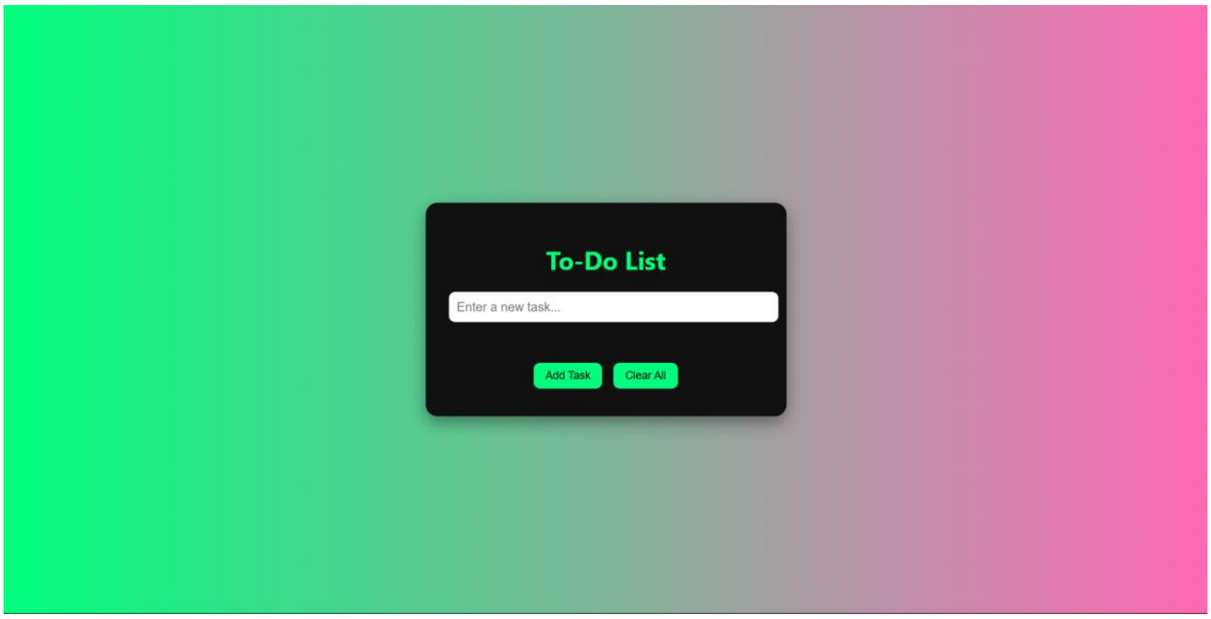
```
<input id="taskInput" type="text" placeholder="Enter a new task..." />
<ul id="taskList"></ul>
```

- **7.4 CSS Styling for Task Items**

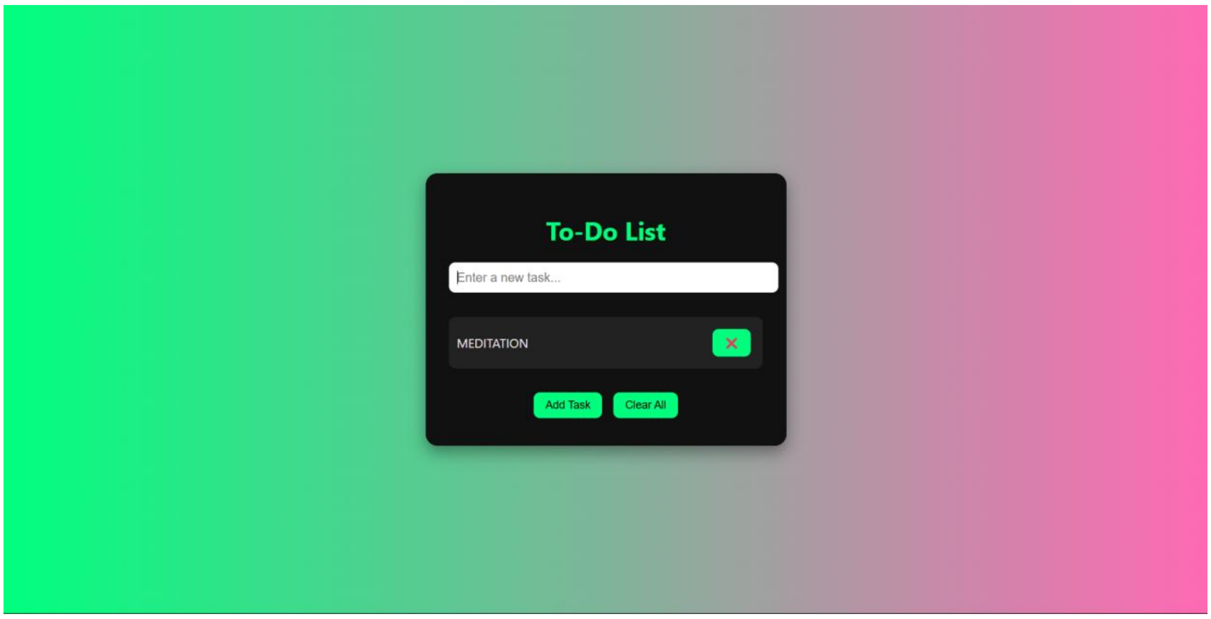
```
CSS
li {
  background: #222;
  color: #eee;
  padding: 10px;
  margin-bottom: 10px;
  border-radius: 8px;
  display: flex;
  justify-content: space-between;
  align-items: center;
}

li.done {
  text-decoration: line-through;
  color: #777;
}
```

8. Output Photos

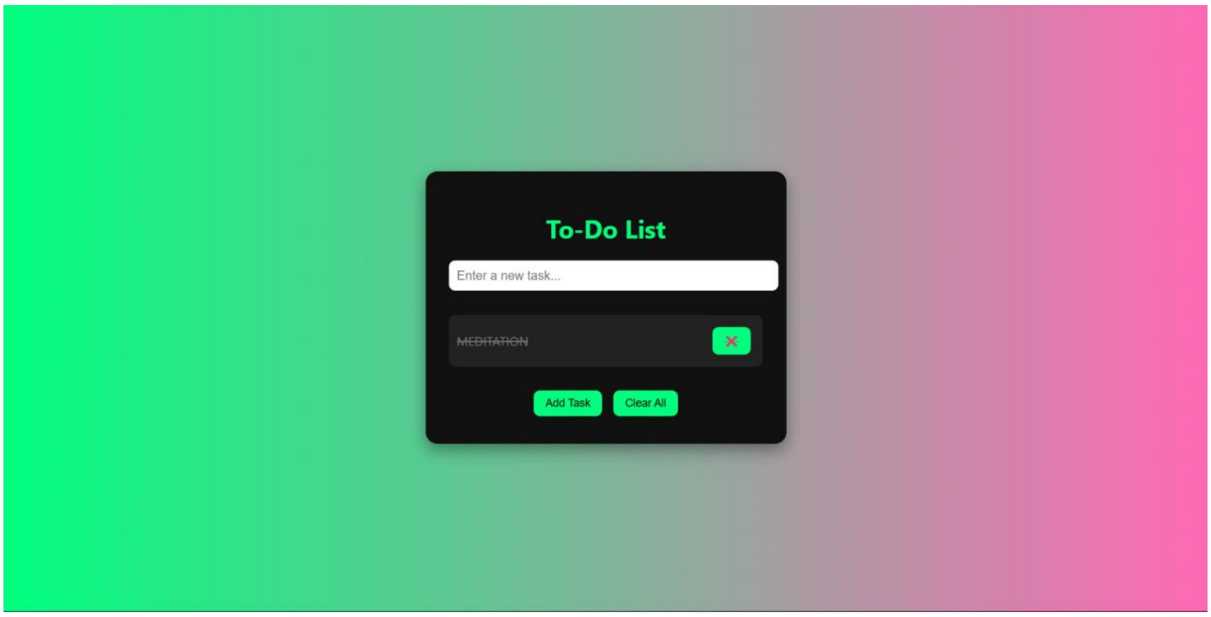
- **8.1 Initial State**
  - The page displays the title "To-Do List."
  - There's an input field where users can enter tasks.
  - There's an empty list area.
  - The "Add Task" and "Clear All" buttons are visible.
  - The background has the parrot green to pink gradient.



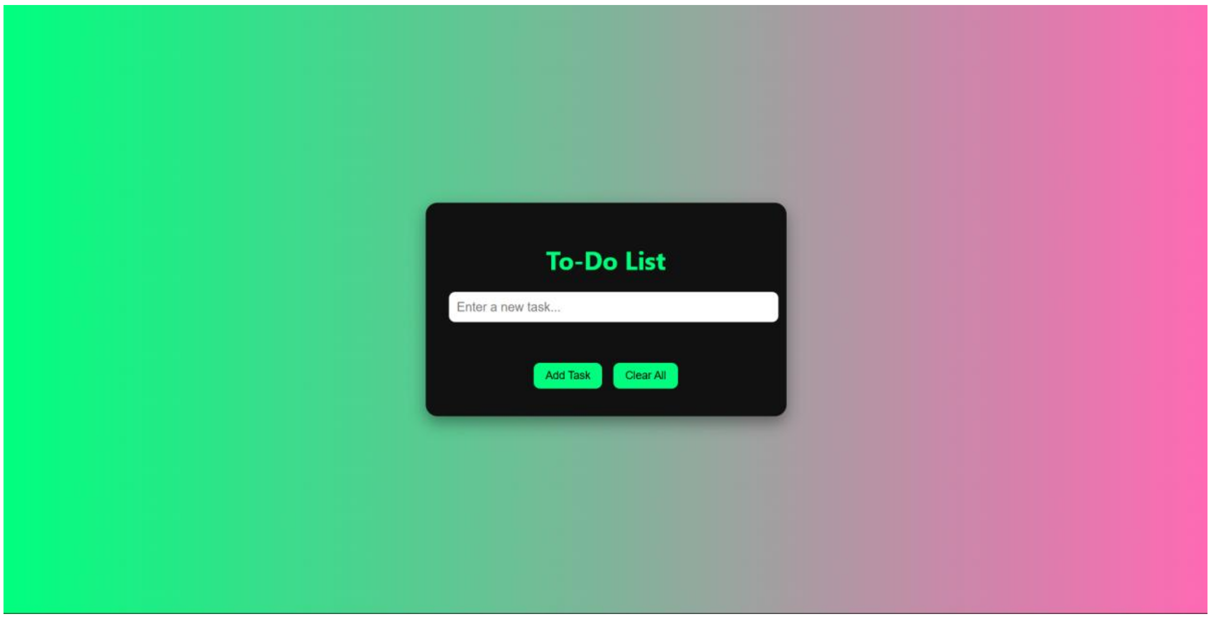
- **8.2 After Adding Tasks**
    - Tasks entered by the user are displayed as list items.
    - Each task item shows the task text and a delete button (❌).



- **8.3 After Marking a Task as Complete**
    - When a task is clicked, it's displayed with a strikethrough, indicating completion. The text color changes to gray.



- **8.4 After Clicking "Clear All"**
    - All tasks are removed from the list, and the list area is empty.



9. Limitations and Feature Enhancements

- **9.1 Limitations**
    - **No Data Persistence:** Tasks are not saved. If the browser is refreshed, all tasks are lost.
    - **Basic Functionality:** The application provides only basic CRUD (Create, Read, Update, Delete) functionality.
    - **No Task Prioritization:** There's no way to prioritize tasks.

    - **No Categories or Tags:** Tasks cannot be categorized or tagged.
    - **Simple Styling:** The styling is basic and could be enhanced.
- **9.2 Feature Enhancements**
    - **Local Storage Persistence:** Use local storage to save tasks so they persist across browser sessions.
    - **Task Prioritization:** Add the ability to prioritize tasks (e.g., using drag-and-drop or priority levels).
    - **Task Categories/Tags:** Allow users to categorize or tag tasks.
    - **Due Dates:** Implement the ability to set due dates for tasks.
    - **Reminders:** Add functionality to set reminders for tasks.
    - **User Authentication:** Support multiple users with accounts to manage their own task lists.
    - **Improved UI/UX:** Enhance the user interface and user experience with better styling, animations, and interactions.
    - **Undo/Redo:** Implement undo/redo functionality for task actions.
    - **Search/Filter:** Add the ability to search and filter tasks.

10. Conclusion

The To-Do List application provides a simple and effective way to manage tasks. It demonstrates core web development principles using HTML, CSS, and JavaScript. While it has limitations in terms of data persistence and advanced features, it serves as a solid foundation for further development and enhancements. The application's clean design and intuitive functionality make it a useful tool for basic task management.