Front-End UI/UX Mini Project

Project Title: A To-Do List Web Application

Submitted By:

Team Members: Abhishan Francis, Abel Alexander, Mishael Julian

Roll number: 2462835, 2462004, 2462184

College E-mail ID: abhishan.francis@btech.christuniversity.in
abel.alexander@btech.christuniversity.in
mishael.julian@btech.christuniversity.in

Course: UI/UX Design Fundamentals

Instructor Name: Dhiraj A.

Institution: Christ University

Date of Submission: 26/09/2025

Abstract

This report provides a comprehensive overview of the design, development, and implementation of a dynamic, client-side To-Do List web application. The project was built using fundamental front-end technologies: HTML5 for structure, CSS3 for styling, and vanilla JavaScript for all interactive logic. The application delivers essential task management functionalities, allowing users to add new tasks, mark tasks as complete with a visual strikethrough, and delete tasks. A key feature is its implementation of data persistence using the browser's Web Storage API (localStorage), ensuring that user-generated tasks are saved locally and are not lost between browsing sessions. This demonstrates a strong command of core web principles, including DOM manipulation and local storage, to create a practical, standalone productivity tool without dependency on external frameworks or libraries

1. Objectives

The primary objectives of this project were to:

- Develop a fully functional single-page To-Do List application using only HTML, CSS, and vanilla JavaScript.
- Implement core task-management features: adding, completing, and deleting tasks.
- Utilize JavaScript for dynamic DOM manipulation to update the task list in real-time without page reloads.
- Implement data persistence by using the browser's localStorage to save and retrieve tasks.
- Create a clean, intuitive, and responsive user interface that is easy to use on various devices.

2. Scope of the Project

The project is a client-side web application with the following scope:

- In-Scope: The application focuses entirely on front-end functionality. It allows users to manage a list of tasks that are stored directly in their web browser. The core functionalities (add, complete, delete) are handled by JavaScript.
- Out-of-Scope: The project does not include any server-side logic, user authentication, or cloud-based data synchronization. All data is stored locally on the user's machine.

3. Tools & Technologies Used

- HTML5: Used for the semantic structure of the application, including the input form and the list container for tasks.
- CSS3: Used for all styling aspects, such as layout, typography, colors, and visual feedback for user interactions (e.g., the strikethrough on completed tasks).
- Vanilla JavaScript: Used for all the application's logic, including event handling, DOM manipulation (creating, updating, and deleting elements), and interacting with the Web Storage API (localStorage).
- Visual Studio Code (VS Code): The primary code editor used for development.

- Chrome DevTools: Used for debugging JavaScript, inspecting the DOM, and verifying data storage in localStorage.

4. HTML Structure Overview

The HTML is structured semantically to create a clear and accessible application layout.

- A main container holds the entire to-do application.
- A header section includes the application title (<h2>).
- An input area contains an <input type="text"> field for users to enter new tasks and a <button> to submit them.
- An unordered list (<ul>) with a specific ID serves as the container where task list items (<li>) are dynamically added by JavaScript.

5. CSS Styling Strategy

An external stylesheet is used to create a clean and user-friendly interface.

- Layout: Centered layout using basic CSS positioning and box-model properties to create a focused user experience.
- Visual Feedback: Clear visual cues are provided for user actions. For instance, a specific class (.checked) is applied to completed tasks, which adds a strikethrough text-decoration and changes the color.
- Interactive Elements: Hover and active states are styled for buttons and list items to improve usability and provide interactive feedback.
- Responsiveness: Media queries are used to ensure the application's layout adapts well to different screen sizes, particularly for mobile devices.
- **Custom Properties:** CSS variables (custom properties) were used to define a consistent color scheme for theme customization, making it easy to manage colors for elements like text, backgrounds, and links.

```css
/* Custom CSS for To-Do List Application */

body {
    background: linear-gradient(135deg, #667eea 0%, #764ba2 100%);
    min-height: 100vh;
    font-family: 'Arial', sans-serif;
}

.card {
    border-radius: 15px;
    border: none;
}

.card-header {
    border-radius: 15px 15px 0 0 !important;
    background: linear-gradient(45deg, #007bff, #0056b3) !important;
}

#taskInput {
    border-radius: 10px;
    border: 2px solid #e9ecef;
    padding: 12px;
    font-size: 16px;
    transition: all 0.3s ease;
}

#taskInput:focus {
    border-color: #007bff;
    box-shadow: 0 0 10px rgba(0, 123, 255, 0.3);
    outline: none;
}

#addTaskBtn {
    border-radius: 10px;
    padding: 12px;
    font-weight: bold;
    transition: all 0.3s ease;
}

#addTaskBtn:hover {
    background-color: #218838;
    transform: translateY(-2px);
}

.filter-btn {
    border-radius: 0;
    transition: all 0.3s ease;
}

.filter-btn:first-child {
    border-radius: 10px 0 0 10px;
}

.filter-btn:last-child {
    border-radius: 0 10px 10px 0;
}

.filter-btn.active {
    background-color: #007bff;
    color: white;
    border-color: #007bff;
}

.task-item {
    border: none;
    border-radius: 10px;
    margin-bottom: 10px;
    transition: all 0.3s ease;
    background: white;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}
```

```
.task-item:hover {
    transform: translateY(-2px);
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.15);
}

.task-item.completed {
    background-color: #f8f9fa;
    opacity: 0.7;
}

.task-item.completed .task-text {
    text-decoration: line-through;
    color: #6c757d;
}

.task-text {
    font-size: 16px;
    margin: 0;
    transition: all 0.3s ease;
}

.task-checkbox {
    width: 20px;
    height: 20px;
    margin-right: 15px;
    cursor: pointer;
    transform: scale(1.2);
}

.task-actions {
    display: flex;
    gap: 10px;
}

.btn-edit {
    background-color: #ffc107;
    border-color: #ffc107;
    color: #212529;
    padding: 5px 10px;
    font-size: 12px;
    border-radius: 5px;
    transition: all 0.3s ease;
}

.btn-edit:hover {
    background-color: #ffb300;
    transform: scale(1.05);
}

.btn-delete {
    background-color: #dc3545;
    border-color: #dc3545;
    color: white;
    padding: 5px 10px;
    font-size: 12px;
    border-radius: 5px;
    transition: all 0.3s ease;
}

.btn-delete:hover {
    background-color: #c82333;
    transform: scale(1.05);
}

.alert-info {
    border-radius: 10px;
    border: none;
    background: linear-gradient(45deg, #d1ecf1, #bee5eb);
    color: #0c5460;
```

```
@keyframes fadeIn {
    from {
        opacity: 0;
        transform: translateY(20px);
    }
    to {
        opacity: 1;
        transform: translateY(0);
    }
}

.slide-out {
    animation: slideOut 0.3s ease-out;
}

@keyframes slideOut {
    from {
        opacity: 1;
        transform: translateX(0);
    }
    to {
        opacity: 0;
        transform: translateX(100%);
    }
}

/* Responsive Design */
@media (max-width: 768px) {
    .container {
        margin-top: 20px;
    }

    .card-body {
        padding: 20px;
    }

    .btn-group {
        flex-direction: column;
    }

    .filter-btn {
        border-radius: 5px !important;
        margin-bottom: 5px;
    }

    .task-actions {
        flex-direction: column;
        gap: 5px;
    }
}
```

6. Key Features

- Dynamic Task Creation: Users can type a task into the input field and click the "Add" button to see it instantly appear on the list.
- Task Completion Toggle: Clicking on a task toggles its completion status. A "checked" class is applied or removed, providing immediate visual feedback.
- Task Deletion: Each task has a delete icon ("×") that, when clicked, removes the task from the list permanently.

- Data Persistence with localStorage: The application saves the entire task list to the browser's localStorage after every change (add, complete, or delete). When the user reopens or refreshes the page, their tasks are reloaded, ensuring no data is lost**.**

7. Overcoming Layout Challenges

The primary challenge was ensuring that dynamically created elements (the list items and their delete buttons) were styled correctly and behaved as expected. This was managed by ensuring that JavaScript created elements with the correct structure and that event listeners were attached properly. Event delegation was used on the parent <ul> container to efficiently handle clicks on any list item or delete button, which is more performant than adding a separate event listener to every single task.

8. Outcomes & Learning

The project successfully resulted in a clean, consistent, and visually engaging front-end layout that functions as intended across all target devices. Through this project, we gained practical insights into responsive design, semantic layout structuring, and modern user interface aesthetics. The hands-on implementation of design principles using only HTML5 and CSS3 enhanced our understanding of core web technologies and the importance of building responsive, mobile-first websites from the ground up.

9. Future Enhancements

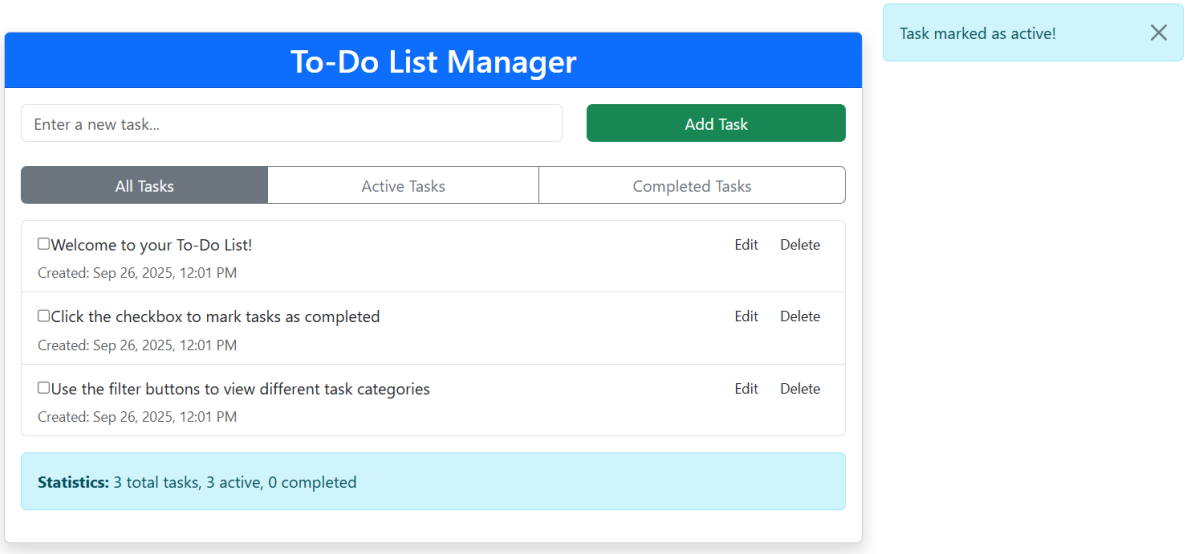
While the current project meets its objectives, several enhancements could be implemented in the future:

- Add JavaScript for Interactivity: Integrate JavaScript to add dynamic features such as client-side validation for the contact form, smooth scrolling, and dynamic content rendering.
- Backend Integration: Implement a backend service (e.g., Node.js or EmailJS) to make the contact form fully functional, allowing submissions to be sent to an email address.
- Integrate Animations: Use CSS animations or a JavaScript library like AOS (Animate On Scroll) to add more sophisticated transitions and make the user experience more engaging.
- Theme Toggler: Add a light/dark mode theme toggler to improve user accessibility and personalization.

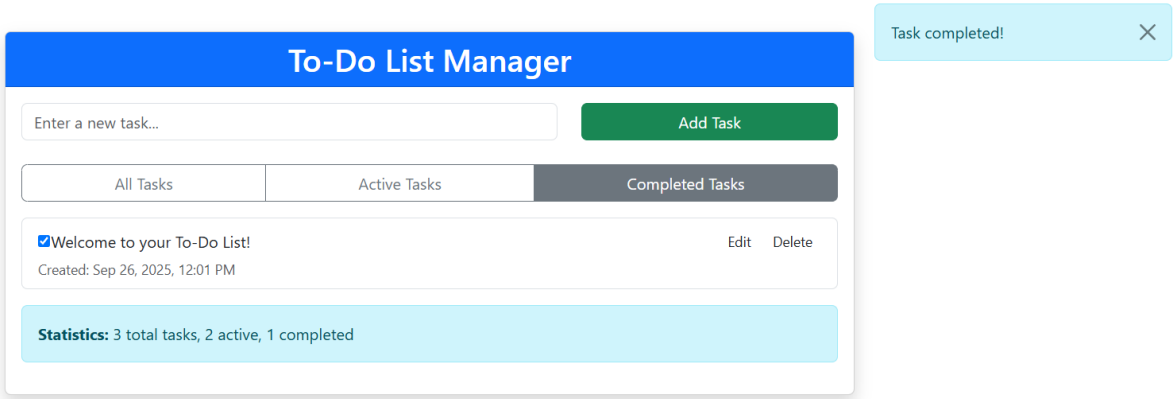
10. Screenshots of Final Outcome

(This section would include 3-4 screenshots of the final project.)

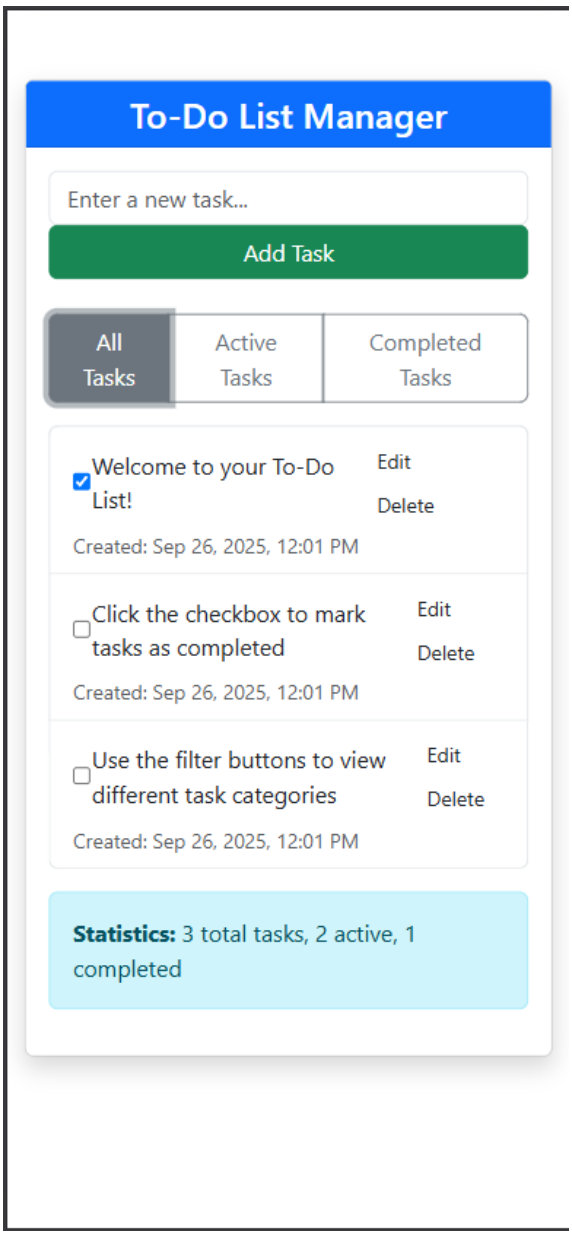
- Screenshot 1: The initial view of the application with an empty task list and input field.



- Screenshot 2: The application with several tasks added, showing one task marked as complete with a strikethrough.



- Screenshot 3: A view of the browser's developer tools showing the task list's HTML saved as a string in localStorage.



11. References

- MDN Web Docs for JavaScript and Web Storage API.