|  |  |
| --- | --- |
| **Project:** | Todo App |
| **Course:** | CSD 380 |
| **Description:** | Test cases for Todo App |
| **Date:** | 6/19/2025 |

To complete the test case plan, fill out the information for Project, Course, Description and Date in the header below.

.

**TABLE OF CONTENTS**

[Access Website via Standard URL 2](#_Toc201226061)

[Create a New Todo Item 4](#_Toc201226062)

[Edit an Existing Todo Item 5](#_Toc201226063)

[Delete a Todo Item 6](#_Toc201226064)

[Prevent Empty Task Submission 7](#_Toc201226065)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 1 | Access Website via Standard URL | | | |
|  | **Test Objective:**  Confirm users can visit the website at the correct URL and it loads without errors. | **Developer:**  **Date tested:** 06/19/2025 | **Peer tester:**  **Date tested:** <yyyy/mm/dd> | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail**  **+ Screenshot** |
| 1 | Open https://buwebdev.github.io/todo/ | Website loads successfully | Yes | <yes/no> |
| 2 | Verify page title | Correct title appears in browser tab | Yes | <yes/no> |
| 3 | Check layout and header | Homepage UI appears with no broken elements | Yes | <yes/no> |
| 4 | Refresh the page | Website reloads with no errors | Yes | <yes/no> |
| 5 | Open DevTools Console | No critical loading errors shown | Yes | <yes/no> |
| **Comments** | This test validates the application's accessibility and ensures that the user can reach the Todo site using the correct public URL. Successful execution confirms that the site is hosted correctly, loads without JavaScript or HTTP errors, and is responsive upon browser refresh. By including checks such as verifying the page title and inspecting the browser's developer console, this test not only confirms the presence of basic content but also ensures that the front-end is initializing properly. If any issues arise during this test (e.g., 404 errors, missing styles, broken elements), they may indicate deployment or hosting configuration problems, which would need to be addressed immediately as they impact all other functionality. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 2 | Create a New Todo Item | | | |
|  | **Test Objective:** Ensure a user can add a new todo task using the input and button. | **Developer:**  **Date tested:** 06/19/2025 | **Peer tester:**  **Date tested:** <yyyy/mm/dd> | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Focus the input field | Cursor appears in the input box | Yes | <yes/no> |
| 2 | Enter 'Buy groceries' | Text is accepted into the field | Yes | <yes/no> |
| 3 | Click the 'Add' button | New task appears in the list | Yes | <yes/no> |
| 4 | Verify task location | New item is placed at the end | Yes | <yes/no> |
| 5 | Refresh the browser | New task is still visible | Yes | <yes/no> |
| **Comments** | This test confirms that the core functionality of task creation works as intended and that user input is handled correctly by the application. By testing each part of the process—from focusing the input field to verifying persistence after a page reload—we ensure the full workflow is functional. A successful outcome indicates that the system is capturing user input, updating the DOM appropriately, and saving data either locally or via backend storage (depending on implementation). If a new task fails to appear or is lost after a refresh, it could indicate issues with event handling, DOM updates, or persistence logic. Verifying the task position also ensures that insertion logic is consistent with user expectations. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 3 | Edit an Existing Todo Item | | | |
|  | **Test Objective:** Confirm users can edit existing tasks. | **Developer:**  **Date tested:** 06/19/2025 | **Peer tester:**  **Date tested:** <yyyy/mm/dd> | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Find an existing task | Edit icon is shown beside the task | Yes | <yes/no> |
| 2 | Click Edit | Task text becomes editable | Yes | <yes/no> |
| 3 | Change text to 'Buy milk' | Updated content appears | Yes | <yes/no> |
| 4 | Click Save | Task displays updated text | Yes | <yes/no> |
| 5 | Refresh the browser | Updated task persists after reload | Yes | <yes/no> |
| **Comments** | This test verifies that the edit functionality behaves correctly and provides a seamless user experience for modifying existing tasks. It checks that tasks can be selected for editing, the interface allows updates, and changes are applied and retained after a refresh. Ensuring these steps work is essential for maintaining task accuracy and user trust in the system. A failure to enter or save updates may point to problems with state management, event binding, or data persistence. This test also indirectly checks UI responsiveness and validates that only the intended task is modified, with no side effects to other list items. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 4 | Delete a Todo Item | | | |
|  | **Test Objective:** Ensure users can remove tasks from their list. | **Developer:**  **Date tested:** 06/19/2025 | **Peer tester:**  **Date tested:** <yyyy/mm/dd> | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Identify a task | Delete icon appears next to it | Yes | <yes/no> |
| 2 | Click Delete | Task is removed from the list | Yes | <yes/no> |
| 3 | Refresh the browser | Task remains deleted | Yes | <yes/no> |
| 4 | Check remaining items | Other tasks remain unaffected | Yes | <yes/no> |
| **Comments** | This test ensures that the delete functionality allows users to reliably remove tasks from their list without affecting unrelated items. Deletion is a critical operation, and verifying it works properly helps maintain data integrity and user satisfaction. This test checks both immediate feedback (i.e., the task disappearing) and long-term behavior (i.e., persistence after refresh). If a deleted task reappears or if another task is mistakenly removed, it would indicate issues with task identification, event handling, or underlying data storage. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 5 | Prevent Empty Task Submission | | | |
|  | **Test Objective:** Ensure users cannot submit empty or invalid todo entries. | **Developer:**  **Date tested:** 06/19/2025 | **Peer tester:**  **Date tested:** <yyyy/mm/dd> | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Leave input field blank | No task text is present | Yes | <yes/no> |
| 2 | Click Add | No task is added to the list | Yes | <yes/no> |
| 3 | Observe feedback | Validation blocks submission or provides warning | Yes | <yes/no> |
| 4 | Enter only spaces | Still blocked from submitting | Yes | <yes/no> |
| 5 | Check list after action | No empty task appears | Yes | <yes/no> |
| **Comments** | This test verifies that input validation is enforced when creating new tasks. It ensures that users cannot add empty or whitespace-only entries, which could clutter the task list and degrade usability. Strong validation is essential to preserve the quality of user-entered data and prevent unnecessary or invalid entries from being saved or displayed. If this test fails, it may indicate missing form checks, improperly bound input events, or weak client-side validation logic. Testing both blank input and whitespace-only input ensures the validation logic goes beyond basic presence and checks for meaningful content. | | | |