|  |  |
| --- | --- |
| **Project:** | **Assignment 4.2** |
| **Course:** | **CSD 380** |
| **Description:** | **Test plan for todo application** |
| **Date:** | **2025/06/17** |

.

**TABLE OF CONTENTS**

[**Create a new todo task item** 2](#_Toc201062479)

[**Delete a todo task item** 2](#_Toc201062480)

[**For invalid URLs, provide users with a custom 404 error page** 3](#_Toc201062481)

[**Prevent users from submitting a task form without entering a todo item** 4](#_Toc201062482)

[**Prevent users from submitting the edit form without entering a todo item** 5](#_Toc201062483)

For each test, the developer should: provide a test description, a test objective, the developer name and date tested. For each step, fill out the actions to be taken and describe the expected results, and indicate yes or no for each step depending upon the result. The peer tester should provide their name, date tested, indicate yes or no for each step depending upon the result, and a screenshot (thumbnail) of the result.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 1 | **Create a new todo task item** | | | |
|  | **Test Objective: Verifies that creating a new task item adds it to the list** | **Developer: Megan Wheeler**  **Date tested:** 2025/06/17 | **Peer tester: Anton DeCesare**  **Date tested:** 06/20/2025 | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail**  **+ Screenshot** |
| 1 | Open Web browser and go to <https://buwebdev.github.io/todo/> | Todo application loads successfully with an empty task list or existing tasks displayed | yes | Yes |
| 2 | Find the text input field where new todo items can be entered | Input field is visible and accessible for text entry | Yes | Yes |
| 3 | Type a new todo item text (e.g., "Buy groceries") into the input field | Text appears in the input field as typed | Yes | Yes |
| 4 | Click the "Add Task" button | New todo item appears in the task list below the input field | yes | Yes |
| 5 | Check that the new todo item is displayed in the list with appropriate formatting | Todo item is visible in the list, typically with a checkbox and the entered text | yes | Yes |
| **Comments** | This test confirms that the application supports the creation of new todo items, a fundamental feature of the system. The steps demonstrate a successful flow: locating the input field, entering valid text, and observing the task appear in the list with appropriate formatting. The presence of visual indicators like a checkbox and proper layout reinforces usability. All expected outcomes are achieved, and the UI responds promptly to user input. This reflects strong front-end handling and good interaction design. Continued reliability in this process will be critical as it serves as the foundation for most user interactions with the app. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 2 | **Delete a todo task item** | | | |
|  | **Test Objective: Verifies that a task item on the todo list can be deleted** | **Developer: Megan Wheeler**  **Date tested:** 2025/06/17 | **Peer tester: Anton DeCesare**  **Date tested:** 6/20/2025 | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Open Web browser and go to <https://buwebdev.github.io/todo/> | Todo application loads successfully with an empty task list or existing tasks displayed | yes | Yes |
| 2 | Either create a new todo item or locate an existing one in the list | At least one todo item is present in the list | yes | Yes |
| 3 | Look for the trash icon associated with the todo item | Icon is visible and clickable | yes | Yes |
| 4 | Click the icon for the selected todo item | Todo item is removed from the list immediately | yes | Yes |
| 5 | Confirm the todo item is no longer visible in the task list | Deleted item is completely removed from the display | yes | Yes |
| **Comments** | This test confirms that the delete functionality works as expected, allowing users to remove tasks from the list cleanly and immediately. The visibility and usability of the trash icon make it easy for users to perform this action without confusion. Once the item is deleted, it is no longer visible, which suggests proper DOM manipulation or backend update depending on the implementation. This supports a clean and manageable user experience. Continued consistency in this feature will ensure users can maintain an accurate and relevant task list without leftover or ghost entries. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 3 | **For invalid URLs, provide users with a custom 404 error page** | | | |
|  | **Test Objective: Verifies that an invalid URL displays a custom 404 error page** | **Developer: Megan Wheeler**  **Date tested:** 2025/06/17 | **Peer tester: Anton DeCesare**  **Date tested:** 6/19/2025 | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Open Web browser and go to <https://buwebdev.github.io/todo/> | Todo application loads successfully with an empty task list or existing tasks displayed | yes | Yes |
| 2 | Modify the URL to include an invalid path (e.g., <https://buwebdev.github.io/todo/invalid-page>) | Custom 404 error page is displayed instead of a generic browser 404 | yes | Yes |
| 3 | Try accessing a non-existent subdirectory (e.g., <https://buwebdev.github.io/todo/admin/nonexistent>) | Custom 404 error page is displayed | no | Yes |
| 4 | Try accessing a file that doesn't exist (e.g., <https://buwebdev.github.io/todo/random.html>) | Custom 404 error page is displayed | no | No |
| 5 | Check the content and styling of the displayed 404 page | Page shows custom branding, helpful message, and navigation back to main application | no | No |
| **Comments** | This test evaluates the application's handling of invalid URLs by checking whether a custom 404 error page is served instead of the browser's generic fallback. It confirms that the app responds appropriately to incorrect paths, nonexistent subdirectories, and missing files. The initial steps confirm that the custom 404 page is correctly triggered, but failures in Steps 3–5 indicate inconsistent implementation or lack of routing fallback for deeper URL structures. Additionally, the lack of clear branding or helpful navigation in the 404 page suggests a missed opportunity to guide users back to the main application. Enhancing the 404 page with visual consistency and actionable links would improve the user experience and help mitigate navigation errors. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 4 | **Prevent users from submitting a task form without entering a todo item** | | | |
|  | **Test Objective: Verifies that users cannot submit a task form without a todo item** | **Developer: Megan Wheeler**  **Date tested:** 2025/06/17 | **Peer tester: Anton DeCesare**  **Date tested:** 06/19/2025 | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Open Web browser and go to <https://buwebdev.github.io/todo/> | Todo application loads successfully with an empty task list or existing tasks displayed | yes | Yes |
| 2 | Leave the input field empty and press Enter or click submit button | Form submission is prevented, no new empty item is added to the list | yes | Yes |
| 3 | Enter only spaces or tabs in the input field and attempt to submit | Form submission is prevented, whitespace-only items are not added | no | Yes |
| 4 | Enter only special characters (e.g., "!!!") and attempt to submit | Form either accepts valid special characters or prevents submission based on validation rules | no | Yes |
| 5 | Check if any error message or visual feedback is provided when submission is prevented | User receives clear feedback about why submission failed (if applicable) | no | Yes |
| **Comments** | This test confirms that the application handles all invalid task submissions effectively. When users attempt to submit the form with an empty field, only whitespace, or special characters, the system prevents the submission and provides immediate visual feedback by highlighting the “Task” label in red. This feedback mechanism is both intuitive and user-friendly, clearly signaling input issues without disrupting the experience. The consistent enforcement of input validation across all test scenarios ensures high data quality and a polished user interface. This reflects thoughtful design and reliable front-end validation logic. You can, however enter a blank data field by editing an existing “Todo” and deleting the contents. | | | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test 5 | **Prevent users from submitting the edit form without entering a todo item** | | | |
|  | **Test Objective: Verifies that a user cannot submit the edit form without a todo item** | **Developer: Megan Wheeler**  **Date tested:** 2025/06/17 | **Peer tester: Anton DeCesare**  **Date tested:** 06/20/2025 | |
| **Step** | **Action** | **Expected results:** | **Developer pass/fail** | **Tester pass/fail** |
| 1 | Open Web browser and go to <https://buwebdev.github.io/todo/> | Todo application loads successfully with an empty task list or existing tasks displayed | yes | Yes |
| 2 | Either create a new todo item or locate an existing one in the list | At least one todo item is present in the list | yes | Yes |
| 3 | Click on the todo item edit button to enable editing | Todo item becomes editable | yes | Yes |
| 4 | Delete all text from the todo item, leaving it empty | Input field becomes empty | yes | Yes |
| 5 | Click save button while the field is empty | Edit submission is prevented, empty todo item is not saved | no | No |
| 6 | Enter only spaces in the edit field and attempt to save | Edit submission is prevented, whitespace-only content is not saved | no | No |
| 7 | Check if any error message or visual feedback is provided when edit submission is prevented | User receives clear feedback about why the edit failed to save | no | No |
| **Comments** | This test confirms whether the application enforces input validation during the editing of existing todo items. While the system properly recognizes when the field becomes empty or contains only whitespace, it fails to block the submission in these cases. As a result, invalid updates may be saved, degrading the quality of the task list. Additionally, no visual feedback or error messaging is provided to inform the user that their input is invalid, which can lead to confusion or data loss. To fully meet Requirements #6 and #7, the edit form should mirror the validation behavior of the task creation form, including consistent prevention logic and red visual indicators when input is invalid. | | | |