Test Case Template

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

Project: Test Plan	
Course: CSD380-A311 DevOps	
Description: creation of a detailed test plan based on the business requirements for a Todo application.	
Date: 2025/06/18	

.

Table of Contents

Test Case Template

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 Case Template

Test 1	View a List of Todo Task Items			
	Test Objective: Verify that users can view a list of Todo task items.	**Developer: Steve Stylin** **Date tested:** 2025/06/19	**Peer tester:** **Date tested:** <yyyy/mm/dd>	
Step	**Action**	**Expected results:**	**Developer pass/fail**	**Tester pass/fail + Screenshot**
1	Navigate to the Todo application URL: https://buwebdev.github.io/todo/	The page should load without errors, displaying the list of Todo items.	yes	<yes/no>
2	Check for the presence of the Todo list section	The Todo list section should be visible and populated with items	yes	<yes/no>
3	Refresh the browser window.	The Todo list should reload correctly without losing any data.	yes	<yes/no>
4	Verify that the number of items displayed matches the expected count	The displayed count should match the number of items in the database	yes	<yes/no>
5	Check for any error messages during loading	No error messages should be displayed.	yes	<yes/no>
Comments	The test successfully confirmed that users can view the list of Todo items. The application loaded without errors, and the data was consistent upon refresh. However, implementing pagination would be beneficial for a better user experience if the list grows significantly.			

Test Case Template

Test 2	Create a New Todo Task Item			
	Test Objective: Ensure that users can create a new Todo task item	**Developer: Steve Stylin** **Date tested:** 2025/06/19	**Peer tester:** **Date tested:** <yyyy/mm/dd>	
Step	**Action**	**Expected results:**	**Developer pass/fail**	**Tester pass/fail**
1	Click on the "Add Todo" button	The input form for a new Todo item should appear	yes	<yes/no>
2	Enter a valid Todo item description in the input field.	The input field should accept the text without errors.	yes	<yes/no>
3	Click the "Submit" button.	The new item should be visible on the list	yes	<yes/no>
4	Verify that the new item appears in the Todo list.	The new item should be visible on the list	yes	<yes/no>
5	Check for any error messages.	No error messages should be displayed	yes	<yes/no>
Comments	The test confirmed that users can successfully create a new Todo item. The input form functioned as expected, and the new item was displayed immediately. It is recommended to add validation for duplicate entries to enhance data integrity and accuracy.			

Test Case Template

Test 3	Edit a Todo Task Item			
	Test Objective: Verify that users can edit an existing Todo task item	**Developer: Steve Stylin** **Date tested:** 2025/06/19	**Peer tester:** **Date tested:** <yyyy/mm/dd>	
Step	**Action**	**Expected results:**	**Developer pass/fail**	**Tester pass/fail**
1	Click on the "Edit" button next to an existing Todo item.	The edit form should appear pre-filled with the current item description.	yes	<yes/no>
2	Modify the description in the input field.	The input field should accept the new text without errors.	yes	<yes/no>
3	Click the "Update" button	The updated Todo item should reflect the changes in the list.	yes	<yes/no>
4	Verify the updated item appears correctly in the Todo list.	The updated description should be visible in the list.	yes	<yes/no>
5	Check for any error messages.	No error messages should be displayed.	yes	<yes/no>
Comments	The editing functionality worked as intended, allowing users to modify existing items. The changes were reflected immediately in the list. It may be beneficial to implement an undo feature for accidental edits.			

Test Case Template

Test 4	Delete a Todo Task Item			
	Test Objective: Ensure that users can delete a Todo task item.	**Developer: Steve Stylin** **Date tested:** 2025/06/19	**Peer tester:** **Date tested:** <yyyy/mm/dd>	
Step	**Action**	**Expected results:**	**Developer pass/fail**	**Tester pass/fail**
1	Click on the "Delete" button next to a Todo item.	A confirmation dialog should appear asking for confirmation.	yes	<yes/no>
2	Confirm the deletion.	The Todo item should be removed from the list.	yes	<yes/no>
3	Verifying that the item is no longer displayed in the Todo list.	The deleted item should not be visible	yes	<yes/no>
4	Check for any error messages.	No error messages should be displayed.	yes	<yes/no>
5	Attempt to delete the same item again	An appropriate error message should be displayed.	yes	<yes/no>
Comments	The deletion process functioned correctly, with items being removed as expected. The confirmation dialog is a good feature to prevent accidental deletions. Consider adding a feature to restore deleted items for user convenience.			

Test Case Template

Test 5	Custom 404 Error Page for Invalid URLs			
	Test Objective: Verify that users are presented with a custom 404 error page for invalid URLs	**Developer: Steve Stylin** **Date tested:** 2025/06/19	**Peer tester:** **Date tested:** <yyyy/mm/dd>	
Step	**Action**	**Expected results:**	**Developer pass/fail**	**Tester pass/fail**
1	Enter an invalid URL in the browser.	The custom 404 error page should load.	yes	<yes/no>
2	Check the content of the 404 page.	The page should contain a user-friendly message and navigation options.	yes	<yes/no>
3	Click on the home link.	The user should be redirected to the main Todo application page.	yes	<yes/no>
4	Verify that the 404 page does not display any broken links.	All links should function correctly.	yes	<yes/no>
5	Check for any error messages.	No error messages should be displayed.	yes	<yes/no>
Comments	The custom 404 error page was displayed correctly for invalid URLs, providing a user-friendly experience. The navigation options were functional, enabling users to easily return to the main page. It may be beneficial to include a search feature on the 404 page to enhance user experience. .			