**Raisely TSE Assessment: Bug Investigation & Fixes Report**

**1. Introduction**

**Scenario**: "A customer support team member has reported that users are experiencing

issues with a web form. The form submission button becomes unresponsive after certain

user interactions, and the styling breaks in specific browsers." This document details the investigation, root causes, debugging process, solutions, and preventive measures.

**2. Identified Issues & Root Causes**

**Issue 1: Button Unresponsiveness**

* **Root Cause:**

The missing **preventDefault()** is causing the page to reload before validation and custom actions are completed.

A screen shot of a computer code

AI-generated content may be incorrect.

* + The **validateForm()** function does not return a boolean value, preventing proper control over form submission. Actions after validation should be implemented in the onsubmit function, outside of **validateForm()** function.

A screen shot of a computer screen

AI-generated content may be incorrect.

**Issue 2: Cross-Browser Styling Inconsistencies**

* **Root Causes:**
  + Conflicting CSS in **.form-container** class initially uses **display: flex;** but is overridden by **display: block;**, causing potential responsiveness issues.

A screen shot of a computer program

AI-generated content may be incorrect.

* + Incorrect dark mode styling for the **.submit-button**, where the text color matched the background color, making the text unreadable.

A computer screen shot of a program code

AI-generated content may be incorrect.

* + Overriding CSS selector (**div.form-container form .input-field { border: 2px solid red; }**) displaying red borders by default which are intended to be shown only if input is invalid.

A screen shot of a computer code

AI-generated content may be incorrect.

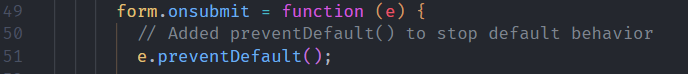
**3. Step-by-step Debugging Process**

1. Reproduced the issues by testing the form in multiple browsers like Chrome, Edge, and Mozilla Firefox.
2. Used browser DevTools to inspect event listeners and applied styles.
3. Checked the browser console for errors and unexpected behaviors.
4. Reviewed CSS rules for conflicts and inconsistencies.
5. Implemented fixes iteratively and verified functionality after each change.
6. Performed final testing to confirm all fixes worked as expected.

**4. Solutions Implemented**

**Fix for Button Unresponsiveness**

* Added **e.preventDefault()** in the **onsubmit** handler to prevent premature form submission.



* Updated **validateForm()** to return **true** if valid and **false** otherwise.

A screen shot of a computer code

AI-generated content may be incorrect.

* Ensured form submission only occurs when validation is successful.

A screen shot of a computer code

AI-generated content may be incorrect.

**Fix for Cross-Browser Styling Issues**

* Removed **div.form-container** & **div.form-container form .input-field** styles to maintain consistency and removed unnecessary & conflicting CSS rules.

A screenshot of a computer program

AI-generated content may be incorrect.

* Correcteddark mode styling by changing text color from **#007bdd** to **white** for **.submit-button**. A screen shot of a computer program

  AI-generated content may be incorrect.
* Added CSS to handle empty input boxes and make border red.

A screen shot of a computer code

AI-generated content may be incorrect.

**5. Preventive Measures**

1. Always Use **preventDefault()** for Form Submissions
   * When handling form validation in JavaScript, always call **e.preventDefault()**.
   * This stops the form from refreshing the page before validation is completed.
2. Make Sure **onsubmit** Handlers Return Correct Value
   * If validation function doesn’t return true or false, the form may behave unexpectedly.
   * Always return a value so the script knows whether to proceed or not.
3. Test in Different Browsers
   * Different browsers (Chrome, Firefox, Edge, Safari) may interpret styles differently.
   * Check form on multiple browsers to catch layout issues early.
4. Use Clear and Non-Conflicting CSS Rules
   * Avoid writing CSS rules that accidentally override each other.
   * Best to stay careful when using overly specific selectors (e.g., **div.form-container form .input-field {}**) because they may cause unexpected changes.
   * Instead, keep styles clean and structured

**6. Summary of Changes**

* Fixed form submission issue by using **preventDefault()**.
* Updated **validateForm()** to correctly handle empty inputs and return appropriate values.
* Resolvedcross-browser styling issues by removing conflicts and removing unrequired CSS classes.
* Implemented debugging and preventive strategies to avoid similar issues in the future.