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.

```
55      const form = document.getElementById("registrationForm");
56      form.onsubmit = function () {
57        // The missing preventDefault() is causing the page to
          reload before validation and custom actions can complete.
58        validateForm();
59      };
```

  - 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.

```
61      function validateForm() {
62        const inputs = document.querySelectorAll(".input-field");
63        let isValid = true;
64
65        inputs.forEach((input) => {
66          if (!input.value.trim()) {
67            isValid = false;
68            // CSS for highlighting missing fields should be
              implemented here.
69          }
70        });
71
72        // The validateForm() function should only handle input
          valiation and any actions like console logs of "Form
          Submitted" should be implemented in the onsubmit function
          above.
73        if (isValid) {
74          console.log("Form submitted");
75        }
76
77        // Should return a boolean value of true or false after
          validation is  complete.
78      }
```

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.

```
.form-container {
  display: flex;
  flex-direction: column;
  max-width: 500px;
  margin: 0 auto;
  padding: 20px;
}

div.form-container {
  /* display: block; is overrriding the dispplay: flex; above and may cause resposive issues in browsers and screen sizes. */
  display: block;
}
```

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

```
.submit-button {
  background-color: #007bff;
  /* Expected Text color */
  color: white;
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}

@media (prefers-color-scheme: dark) {
  .submit-button {
    background-color: #007bff;
    /* White text is being changed to the same color as the button, making it invisible. */
    color: #007bff;
  }
}
```

  - 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.

```
.input-field {
  margin: 10px 0;
  padding: 8px;
  /* Expected default border color */
  border: 1px solid #ccc;
}


div.form-container form .input-field {
  /* Border color is getting overridden and displaying red borders by default */
  border: 2px solid red;
}
```

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.

```
49        form.onsubmit = function (e) {
50          // Added preventDefault() to stop default behavior
51          e.preventDefault();
```

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

```
64        function validateForm() {
65          const inputs = document.querySelectorAll(".input-field");
66          let isValid = true;
67
68          inputs.forEach((input) => {
69            if (!input.value.trim()) {
70              isValid = false;
71              input.style.border = "2px solid red";
72
73            } else {
74              input.style.border = "1px solid #ccc"
75            }
76          });
77          // Return Boolean value of either true or false
78          return isValid;
79        }
```

- Ensured form submission only occurs when validation is successful.

```
49        form.onsubmit = function (e) {
50          // Added preventDefault() to stop default behavior
51          e.preventDefault();
52
53          // Print console log and displays alert if form is
            successfully submitted
54          if(validateForm()){
55            console.log("Form Submitted")
56            alert("Form Submitted Successfully!")
57            // Reload page after alert is dismissed by clicking "OK"
58            location.reload();
59          } else {
60            console.log("Please fill out all fields.");
61          }
62        };
```

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.

```css
/* Removed css classes  which are not requred*/
.form-container {
  display: flex;
  flex-direction: column;
  max-width: 500px;
  margin: 0 auto;
  padding: 20px;
}

/* Remove
  div.form-container {
  display: block;
} */

.input-field {
  margin: 10px 0;
  padding: 8px;
  border: 1px solid #ccc;
}

/* Remove
  div.form-container form .input-field {
  border: 2px solid red;
} */

.submit-button {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}

@media (prefers-color-scheme: dark) {
  .submit-button {
    background-color: #007bff;
    color: white; /* Fixed overriden text color */
  }
}
```

- Corrected dark mode styling by changing text color from **#007bdd** to **white** for **.submit-button**.

```css
.submit-button {
  background-color: #007bff;
  color: white;
  padding: 10px 20px;
  border: none;
  cursor: pointer;
}

@media (prefers-color-scheme: dark) {
  .submit-button {
    background-color: #007bff;
    color: white; /* Fixed overriden text color */
  }
}
```

- Added CSS to handle empty input boxes and make border red.

```
78          inputs.forEach((input) => {
79            if (!input.value.trim()) {
80              isValid = false;
81              input.style.border = "2px solid red";
82            } else {
83              input.style.border = "1px solid #ccc"
84            }
85          });
```

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.
- Resolved cross-browser styling issues by removing conflicts and removing unrequired CSS classes.
- Implemented debugging and preventive strategies to avoid similar issues in the future.