**Test Strategy for Component Testing**

1. **Introduction**
   1. **Objective**

The goal of this test plan is to ensure that recent frontend changes made to the Combo-box List, Date & Time Picker, and Scrollbar components are thoroughly validated for functionality, usability, and compatibility. These enhancements are aimed at improving the user experience across various web browsers and platforms. It is essential that these changes do not introduce any defects or negatively impact existing functionality. The objective is to confirm that these components continue to perform as expected under all possible use cases, including different user interactions and environments.

**1.2 Scope**

This document provides a comprehensive strategy for testing the updates to the Combo-box List, Date & Time Picker, and Scrollbar within a web environment. It covers:

* **Test Design:**
* **Test Data Management:**
* **Risk Analysis:**

1. **Test Design**

**Approach:**

* The testing will take place within a **web environment**, focusing on the core functionality, user interaction, and performance of the Combo-box List, Date & Time Picker, and Scrollbar components.
* The components will undergo testing on various browsers (Chrome, Firefox, Edge, Safari) and devices (desktop, tablet, mobile) to ensure consistency across platforms.
* The test approach will include **functional testing, UI/UX validation, and boundary testing** for each component.

**Test Data Preparation for Common UI Components**

**1. Combo Box**

Test Data Requirements:

* Standard Data:
  + Short strings (e.g., "Apple", "Orange", "Grapes")
  + Numeric entries (e.g., "123", "456", "789")
  + Alphabetically sorted strings (e.g., "Alpha", "Beta", "Gamma")
  + Randomly ordered strings for filtering tests (e.g., "Dog", "Apple", "Zebra")
* Edge Case Data:
  + Strings with special characters (e.g., "@home", "#hash", "\*star")
  + Very long strings (e.g., 200+ character names)
  + Empty string or null values
  + Strings with leading/trailing spaces (e.g., " Apple ")
  + Case-sensitive values (e.g., "apple" vs. "Apple")

**Example Test Data Set:**

|  |  |
| --- | --- |
| **Option ID** | **Combo-box Options** |
| 1 | Honda |
| 2 | Toyota |
| 3 | KIA |
| 4 | BMW |
| 5 | \*\*FORD\*\* |
| 6 | #\*&$(\*#(special CHAR only) |
| 7 | (empty string) |
| 8 | SUZUKI |
| 9 | Mercedes |
| 10 | Ferrari |
| 11 | LONG-STRING-1-2-3-4-5-6-7-8-9-10-11-12-13-14-15-16-17-18-19-20 |

**2. Date & Time Picker**

Test Data Requirements:

* Valid Date and Time Combinations:
  + Future dates (e.g., "12/12/2024")
  + Past dates (if applicable)
  + Leap year dates (e.g., "29/02/2024")
  + Edge dates (e.g., "31/12/2024", "01/01/2025")
  + Valid time combinations (e.g., "12:00 AM", "13:30", "23:59")
* Invalid Date and Time Combinations:
  + Non-existent dates (e.g., "30/02/2024")
  + Invalid months (e.g., "15/13/2024")
  + Incorrect time format (e.g., "25:61")
  + Invalid time ranges (e.g., negative or out of bounds times)
* Formats to Test:
  + Date formats: "DD/MM/YYYY", "MM/DD/YYYY"
  + Time formats: "12-hour AM/PM", "24-hour"

**Example Test Data Set:**

|  |  |  |
| --- | --- | --- |
| **Test Case ID** | **Input Data** | **Expected Outcome** |
| 1 | 01/01/2024 | Valid date |
| 2 | 30/02/2024 | Invalid date (February 30th) |
| 3 | 29/02/2024 | Valid date (leap year) |
| 4 | 31/12/2024 | Valid date |
| 5 | 25:61 | Invalid time |
| 6 | 12:00 PM | Valid time |
| 7 | 13/12/2024 | Valid date |
| 8 | 32/10/2024 | Invalid day (Oct 32) |
| 9 | 13:30 (24-hour) | Valid time (24-hour format) |
| 10 | 12:00 AM (12-hour) | Valid time (12-hour format) |

**3. Scrollbar**

Test Data Requirements:

* Short Content:
  + Content that fits within a single page without requiring a scrollbar
* Long Content:
  + Text that requires vertical scrolling (e.g., a list with 100+ items)
  + Content that requires horizontal scrolling (e.g., a wide table or image)
  + Mixed content that requires both horizontal and vertical scrolling
* Edge Case Content:
  + Content exactly at the boundary between fitting on one page and requiring scrolling (to test threshold behavior)
  + Rapidly changing dynamic content (e.g., a chat window or live feed)

**Example Test Data Set:**

|  |  |  |
| --- | --- | --- |
| **Content ID** | **Content Type** | **Expected Scroll Behavior** |
| 1 | Short text (e.g., 100 words) | No scrollbar |
| 2 | Long text (e.g., 2000 words) | Vertical scrollbar enabled |
| 3 | Wide image (2000px) | Horizontal scrollbar enabled |
| 4 | Wide table (e.g., 10 columns) | Horizontal scrollbar enabled |
| 5 | Tall image (3000px) | Vertical scrollbar enabled |
| 6 | Mixed content (text & image) | Both vertical and horizontal scroll |

**General Considerations:**

* Data Localization: Prepare test data for various date formats, numeric formats, and text in multiple languages if the application supports different locales.
* Performance Testing: Create large data sets (e.g., 10,000+ items for Combo-box) to test how well the components perform under high load or when handling extensive content.
* Accessibility: Ensure that all test data, especially for Combo-box and Date & Time Picker, is compatible with screen readers, keyboard navigation, and other accessibility tools.

**Example Test Cases:**

**COMBO-BOX LIST**

|  |  |  |
| --- | --- | --- |
| TC ID | TITLE | EXPECTED RESULTS: |
| TC01 | Confirm that clicking on the combo-box displays all available options | |  | | --- | | All available options should be displayed when the combo-box is clicked. |  |  | | --- | |  | |
| TC02 | Test whether typing in the combo-box dynamically filters the options. | |  | | --- | | Typing should filter the combo-box options in real time. |  |  | | --- | |  | |
| TC03 | Verify, combo-box list remains responsive against the long string options | |  | | --- | | Combo-box should remain responsive and not lag with long string options. |  |  | | --- | |  | |
| TC04 | Verify, a scroller is added into combo-box if a collection is long enough. | |  | | --- | | A scroll bar should appear if the list exceeds the viewable height. |  |  | | --- | |  | |
| TC05 | Ensure the list automatically collapses after making a selection. | |  | | --- | | The combo-box should close after a selection is made. |  |  | | --- | |  | |
| TC06 | Check that keyboard navigation (using arrow keys) properly highlights and selects options. | |  | | --- | | Arrow keys should navigate the combo-box and highlight/select options correctly. |  |  | | --- | |  | |
| TC07 | Verify, names in collection appears in alphabetical order | |  | | --- | | Names should be listed alphabetically within the combo-box. |  |  | | --- | |  | |
| TC08 | Verify, items starting with number in collection appears in numerical order | |  | | --- | | Numeric items should be ordered in ascending order. |  |  | | --- | |  | |
| TC09 | Verify, dates appearing in collections must be in chronological order or reverse chronological order | |  | | --- | | Dates should be displayed in proper chronological or reverse order. |  |  | | --- | |  | |
| TC10 | Verify that a user can select a specific option and that selection is reflected correctly. | The selected option should be accurately reflected in the combo-box. |

**DATE & TIME PICKER:**

|  |  |  |
| --- | --- | --- |
| TC ID | Title | Expected Results: |
| TC01 | Verify the functionality of picking dates and times | |  | | --- | | Date and time should be correctly selected and displayed. |  |  | | --- | |  | |
| TC02 | Ensure the selected date and time appear in the correct format (e.g., DD/MM/YYYY, 12-hour/24-hour). | |  | | --- | | Date and time should appear in the required format (e.g., DD/MM/YYYY). |  |  | | --- | |  | |
| TC03 | Test if the picker renders properly across different browsers and screen sizes. | |  | | --- | | Date & time picker should render consistently across all platforms. |  |  | | --- | |  | |
| TC04 | Ensure invalid date/time selections (e.g., February 30) are not allowed and proper error messages are shown. | |  | | --- | | Invalid dates should be rejected, and appropriate error messages shown. |  |  | | --- | |  | |
| TC05 | Confirm that users can interact with the picker using both mouse and keyboard for better accessibility. | |  | | --- | | Full interaction using both mouse and keyboard should be possible. |  |  | | --- | |  | |
| TC06 | Verify, user cannot select the past date in future date picker | |  | | --- | | Past dates should be disabled in the future date picker. |  |  | | --- | |  | |
| TC07 | Verify, a proper date & time picker icon is shown in the field | |  | | --- | | The correct icon should be displayed. |  |  | | --- | |  | |
| TC08 | Verify, user remains able to select different options among “AM” and “PM” | |  | | --- | | The user should be able to toggle between AM and PM. |  |  | | --- | |  | |
| TC09 | Check if user is able to select current time | |  | | --- | | Current time should be selectable in the picker. |  |  | | --- | |  | |
| TC10 | Check if user is able to select different months and years | |  | | --- | | Month and year selection should work smoothly. |  |  | | --- | |  | |
| TC11 | Check if system handles the selection of date February 29 in a leap year. | |  | | --- | | February 29 should be selectable only in a leap year. |  |  | | --- | |  | |
| TC12 | Check that date & time picker remains dynamic across mobile platform and web platform | |  | | --- | | Picker should function consistently across mobile and web platforms. |  |  | | --- | |  | |
| TC13 | Verify, that the time appears in the correct format (e.g., HH:MM AM/PM or 24-hour format) based on the requirements. | |  | | --- | | Time should be displayed in the required format (e.g., 24-hour format). |  |  | | --- | |  | |
| TC14 | Verify, system handles the invalid time input which is entered manually | Invalid time inputs should be rejected, and proper validation shown. |

**SCROLLBAR**

|  |  |  |
| --- | --- | --- |
| TC ID | Title | Expected Results: |
| TC01 | Test scrolling behavior for long content, ensuring smooth and responsive scrolling. | Scrolling should be smooth and responsive with no lag for long content. |
| TC02 | Check that the scrollbar appears and functions correctly on various screen sizes and resolutions. | |  |  |  | | --- | --- | --- | | |  | | --- | | Scrollbar should appear and function properly across all screen sizes and resolutions. |  |  | | --- | |  | |  |  | | --- | |  | |
| TC03 | Ensure that scroll bar is not shown if the content is not more than 1 page | |  |  |  | | --- | --- | --- | | |  | | --- | | No scrollbar should be displayed if the content fits within one page. |  |  | | --- | |  | |  |  | | --- | |  | |
| TC04 | Verify that the scroll position is maintained if the page is refreshed or reloaded. | |  |  |  | | --- | --- | --- | | |  | | --- | | Scroll position should be retained after a page refresh/reload. |  |  | | --- | |  | |  |  | | --- | |  | |
| TC05 | Test the ability to drag the scrollbar and validate proper scrolling in both directions (vertical and horizontal). | |  | | --- | | Scrollbar should allow dragging in both vertical and horizontal directions. |  |  | | --- | |  | |

**2. TEST DATA MANAGEMENT**

* **Data Preparation**: Ensure the data required for testing (like dates in the Date Picker or items in the Combo-box List) is varied and realistic, covering regular, boundary, and edge cases as well as negative if required.
* **Consistency and Reset**: After each test, the system will reset to ensure no residual data from previous tests affect subsequent ones, maintaining the integrity of the results.

**3. RISK ANALYSIS**

**Potential Risks**:

* **Cross-browser Compatibility**: There is a risk that certain components, like the Date & Time Picker or Scrollbar, may not work uniformly across different browsers, leading to inconsistent user experiences.
* **Mobile/Tab Responsiveness**: Components may not scale correctly on mobile and other Tab devices, leading to issues such as improper display or inability to interact with elements.
* **Accessibility Challenges**: Ensuring that these components are accessible for all users, including those using keyboard navigation, could be a challenge if not tested adequately.

**Prioritization of Risks**:

* **High Risk**: Inconsistent behavior across different browsers, especially lesser-used browsers like Microsoft Edge and Mozilla Firefox, which could result in significant user frustration and traffic loss to business.
* **Medium Risk**: Responsiveness issues on mobile and other handy devices like iPad and Tabs, as users accessing the platform on these devices may encounter layout problems or difficulty interacting with components.
* **Low Risk**: Performance challenges with large data sets (e.g., too many items in the Combo-box List), potentially slowing down the component.

**Risk Matrix**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Risk ID | Description | Impact | Frequency | Priority |
| R1 | Combo-box does not display all options when clicked. | 4 | 3 | 12 |
| R2 | Typing in combo-box does not dynamically filter options. | 3 | 4 | 12 |
| R3 | Combo-box becomes unresponsive with long string options. | 5 | 2 | 10 |
| R4 | Scroller is missing when combo-box options are too long to fit in the dropdown. | 3 | 3 | 9 |
| R5 | Incorrect selection behavior in combo-box (selection not reflected correctly). | 4 | 4 | 16 |
| R6 | Combo-box does not automatically collapse after a selection is made. | 3 | 2 | 6 |
| R7 | Keyboard navigation in combo-box fails (arrow keys do not work). | 3 | 3 | 9 |
| R8 | Items in combo-box not sorted alphabetically or numerically. | 2 | 4 | 8 |
| R9 | Date/time picker allows invalid dates or formats (e.g., February 30, wrong time format). | 5 | 3 | 15 |
| R10 | Date/time picker is not responsive across different browsers and screen sizes. | 4 | 2 | 8 |
| R11 | Date/time picker allows past date selection when it should be future only. | 4 | 3 | 12 |
| R12 | Date/time picker does not show an icon in the field. | 2 | 3 | 6 |
| R13 | AM/PM selection not functioning correctly in date/time picker. | 3 | 2 | 6 |
| R14 | User unable to select current time in date/time picker. | 4 | 3 | 12 |
| R15 | Scrollbar does not appear when needed (content is longer than a page). | 3 | 3 | 9 |
| R16 | Scroll position is not retained after page refresh. | 4 | 3 | 12 |
| R17 | Scrollbar does not allow smooth scrolling, impacting user experience. | 3 | 3 | 9 |
| R18 | Scrollbar behavior is inconsistent across different screen sizes or resolutions. | 4 | 2 | 8 |
| R19 | Invalid manual input of date/time is not handled by the system. | 5 | 3 | 15 |

**Explanation of Key Columns:**

* **Impact** (1-5): The level of negative impact the risk would have on the system if it occurred.
  + **1**: Minimal impact
  + **5**: Critical system failure or major user disruption
* **Frequency** (1-5): The probability of the issue occurring.
  + **1**: Very unlikely to occur
  + **5**: Very likely to occur
* **Priority**: The product of **Impact × Frequency**. It helps prioritize risks. The higher the score, the more critical the issue is to address.

**MITIGATION AND CONTINGENCY STRATEGIES:**

**1. Cross-Browser Testing**

**Mitigation Measures:**

* Browser Compatibility List: Maintain a list of supported browsers and their versions based on user analytics to prioritize testing efforts.
* Automated Testing Tools: Utilize tools like Selenium, BrowserStack, or Cross Browser Testing to automate tests across various browsers and operating systems.
* Regular Updates: Keep the test environment updated with the latest browser versions and ensure compatibility with older versions if they are still in use.

**Contingency:**

* If issues are detected in specific browsers, prioritize them based on user analytics. Roll out patches or alternative solutions, such as polyfills, for browsers that do not support certain features.

**2. Mobile and Tablet Testing**

**Mitigation Measures:**

* Device Coverage: Use a wide range of real devices and emulators (e.g., Android Studio Emulator, Xcode Simulator) to cover different operating systems and screen sizes.
* Responsive Design Frameworks: Implement CSS frameworks like Bootstrap or Materialize that support responsive design principles to ensure consistent behavior across devices**.**
* Touch Gestures Testing: Validate touch gestures and ensure that they are intuitive and responsive on mobile devices.

**Contingency:**

* If a layout issue is discovered on a specific device, develop and test alternative layouts or CSS adjustments tailored to that device to ensure usability.

**3. Accessibility Testing**

**Mitigation Measures:**

* Automated Accessibility Tools: Integrate tools like Lighthouse, Axe, or WAVE into the testing pipeline to automatically check for accessibility issues.
* Manual Testing: Conduct manual testing by individuals with diverse abilities to identify real-world accessibility challenges.
* Training: Provide regular training for the development team on accessibility standards (e.g., WCAG) and best practices.

**Contingency:**

* If accessibility issues are found, prioritize them based on severity and impact on users, and create a remediation plan that includes timeline estimates for fixes.

**4. Frontend Unit Tests**

**Mitigation Measures:**

* Testing Frameworks: Utilize frameworks such as Jest, Mocha, or Jasmine for writing and executing frontend unit tests.
* Continuous Integration: Integrate unit tests into the CI/CD pipeline to ensure that tests are executed automatically with each build.
* Code Coverage: Monitor code coverage metrics to identify untested components and increase coverage gradually.

**Contingency:**

* In the event of failing unit tests, implement a rollback strategy to revert to a stable version of the application while investigating and resolving the test failures.

**5. Performance Risk Mitigation**

**Mitigation Measures:**

* **Load Testing:** Use tools like Apache JMeter or LoadRunner to simulate user loads and identify how the application performs under various conditions. This helps in finding bottlenecks and areas that may need optimization.
* **Performance Monitoring:** Implement monitoring tools (e.g., New Relic, Datadog) to continuously track application performance in real-time, measuring metrics such as response time, server load, and error rates.
* **Code Optimization:** Regularly review and optimize code for performance, focusing on reducing load times, minimizing render-blocking resources, and ensuring efficient database queries.
* **Caching Strategies:** Utilize caching (e.g., Redis, Memcached) to reduce server load and improve response times for frequently requested data. Implement Content Delivery Networks (CDNs) to deliver static content quickly to users worldwide.
* **Image and Asset Optimization:** Ensure that images and other assets are optimized for web delivery by compressing them without losing quality and using appropriate file formats.

**Contingency:**

* If performance issues are identified during testing or monitoring, have a rollback plan to revert to the last known stable version while addressing the problems. Additionally, prioritize fixes based on user impact, and establish performance benchmarks to measure improvements after optimizations are implemented.

**Deliverables:**

* **Test Strategy Document**: A concise document detailing the testing plan and approach for each component.
* **Test Cases**: A comprehensive set of test cases covering key functional and edge scenarios for each component.
* **Risk Analysis Report**: A report that outlines the identified risks, their priority level, and corresponding mitigation strategies for each risk.

**Conclusion:**This detailed risk analysis report highlights the potential risks associated with the frontend changes to the Combo box List, Date & Time Picker, and Scrollbar components. These risks span across usability, performance, and compatibility, which are crucial aspects for ensuring a seamless user experience. By proactively addressing these risks, we can prevent common issues like inconsistent input behavior, accessibility problems, performance bottlenecks, and cross-browser incompatibilities.

**Key Mitigation Strategies include:**

* Thorough Usability Testing: This ensures that changes do not hinder user interaction and that accessibility remains compliant with industry standards.
* Cross-Browser & Cross-Device Testing: Ensuring compatibility across multiple browsers, operating systems, and devices.
* Performance Monitoring: Regular monitoring will help catch any degradation in performance caused by new features or changes.
* User Feedback Loops: Implement a feedback mechanism to promptly address any usability concerns or bugs that arise post-deployment.

By implementing these strategies and prioritizing risk mitigation, we can minimize defects and enhance the overall performance and functionality of the platform, ensuring it continues to meet user expectations. Continuous testing and monitoring will further ensure that changes to these key components are handled effectively without compromising the quality of the product.