**Test Strategy for Voosh Dashboard Alert Page**

**1. Test Objectives**

The purpose of this testing strategy is to ensure the **Voosh Dashboard's alert system functions correctly**, covering:

* **UI functionality** (tab navigation, dropdown selections, form submission)
* **Data validation** (percentage inputs, special characters, error messages)
* **Alert setup automation** across different rate tabs (Inaccuracy, Unfulfilled, M/I Rate)

**2. Test Approach**

✅ **Manual & Automated Testing:**

* **Automated Testing (Cypress):** Covers E2E workflow, UI interactions, validations.
* **Manual Testing (Exploratory):** Verifies usability and edge cases.

✅ **Functional Testing:**

* UI interactions (dropdowns, tab switching, section expansion)
* Validations for required fields
* Alert creation

✅ **Boundary Testing:**

* Percentage values (0-100, special decimals)
* Special character inputs

✅ **Negative Testing:**

* Invalid inputs (out-of-range values, empty fields)
* Edge case scenarios (submitting multiple times, duplicate alerts)

**3. Test Scope**

* **Covered Features:**
  + Alerts setup & management
  + UI interactions (dropdowns, tooltips, tab navigation)
  + Data validation (numeric inputs, special characters)
  + Alert triggering & notification delivery

**4. Test Execution Plan**

✅ **Automated Test Execution:**

* **Pre-Deployment Testing:** Runs Cypress suite before release.
* **Post-Deployment Testing:** Confirms UI stability after updates.

✅ **Test Environment:**

* **Browsers:** Chrome, Edge
* **Devices:** Desktop
* **Data:** Simulated test credentials

✅ **Test Case Prioritization:**

* **Critical tests first:** Login, alert creation, dropdown selections
* **Medium-priority tests:** Validations, tab navigation, multiple form submission
* **Low-priority tests:** Special character testing, boundary cases

**5. Defect Management**

* ✅ **Bug Reporting:**
* Issues are **logged manually** or within the preferred tracking system.
* Categorized based on impact: **Critical (blocking), Major, Minor (cosmetic or usability).**
* ✅ **Debugging Approach:**
* Use Cypress **cy.debug() or cy.log()** for troubleshooting failing tests.
* Screenshots & logs captured automatically in **Cypress screenshots/ and videos/ folders** for test failures.
* Verification through **re-running test cases** and validating expected vs actual results.
* Screenshots & logs captured on failures

**6. Success Criteria**

✔ **All critical flows pass across multiple test runs**  
✔ **Proper validation messages are displayed** for incorrect inputs  
✔ **Alerts are created successfully** and trigger when expected  
✔ **Test execution time is optimized** for fast results