Test Strategy - LSE Transformation

**Version Control**

|  |  |  |  |
| --- | --- | --- | --- |
| **Version No** | **Date** | **Author** | **Change Summary** |
| 0.1 | 10/01/2025 | Smrutipada M | Initial Draft |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

**1. Introduction**

* **Purpose**: This document outlines the comprehensive test strategy for the LSE transformation application. It defines the testing approach, resources, schedule, and scope to ensure the application meets the specified requirements and quality standards.
* **Scope**: The scope includes testing all aspects of the UI transformation, including functional and non-functional testing.

**2. Objectives**

* Validate that the transformed user interface meets all functional requirements and provides a seamless user experience.
* Ensure the user interface is intuitive, user-friendly, and accessible to all users.
* Verify the performance and stability of the UI under various conditions, including high load.
* Ensure compatibility across different browsers, devices, and operating systems.
* Identify and fix defects early in the development cycle to reduce the cost and time of rework.

**3. Test Approach**

* **Types of Testing**:
  + **Functional Testing**: Verify that all UI elements and interactions work as expected.
    - Initial test plan to be prepared and shared for approval. [**Owner - QA Lead]**
    - Test plan to be updated daily to capture on the overall progress. [**Owner - QA Lead**]
    - Test case **MUST** be reviewed by the PO\Product before starting test execution for any user story, if review comment has not been received, test execution will not be commenced. **[Owner – QA Member] For Release regression, Test cases pack must be identified and reviewed and agreed with product so that None of the Major use case is left for Regression execution**
    - QA member to prepare the test matrix which maps test cases to respective requirement. This will ensure the coverage for requirements. **[Owner – QA Member]**
    - Any defects identified to be fixed on priority. Open defects to be discussed with stakeholders. **[Owner - QA lead]**
    - Regression pack **MUST** include all P1 cases. **[Owner – QA Member]**
    - Test artifacts to be signed off. **[Owner - Test Lead/Manager]**
    - Output which here is the test case document should also include the type of test (Sanity, Regression etc...) **[Owner – QA Lead to ensure that the process is followed]**
    - All story level testing **MUST** be executed on QA env unless and otherwise agreed within the scrum team and by PO, and then on the subsequent environments like UAT and PPE
    - Test plan to be updated for every release, with details about the stories, defects etc. All QA artifacts such as Test Plan, test cases, Release report needs to be signed off by PO and QA lead. **[Owner – TPM and QA**
  + **Automation Testing:** Verify the functionalities of the user interface, is automated and tested in web browsers across all devices
    - Regression **MUST** be done using Automation only, if all cases are not automated, the cases getting executed manually must be very clearly documented and agreed with team and PO. PO should seek approval from **key stakeholders**. **[Owner – QA Member]**
    - All new features developed as part of sprint should be automated before marking story as done. **[Owner – QA Member]**
    - QA member to discuss on the automation effort with Lead for any new features and discuss with Scrum Master to mark the task as a candidate for sprint. **[Owner – QA Member]**
    - Automation jobs to be run on timely basis in all the environments such as QA, UAT, PPE. **[Owner – QA Member]**
    - Any failures in automation suites to be picked up on priority and fixed. If automation run has found any issue, the same should be tracked via a ticket. **[Owner – QA Member]**
    - QA member to discuss on the Automation backlogs with their leads and plan to accommodate some of these items in the sprints.
    - All production incidents should be analysed and added to the automation suite on priority. **[Owner – QA Member]**
  + **Non-Functional Testing**:
    - **Performance Testing**
      * **Objective**: Validate system responsiveness, throughput, and stability under varying workloads.
      * **Tests Included**:
        + Baseline Testing: Establish benchmarks for future comparisons.
        + Load Testing: Simulate expected concurrent user loads.
        + Stress Testing: Push the system beyond its capacity to identify breaking points.
        + Spike Testing: Test system response to sudden increases in load.
        + Endurance Testing: Validate performance over extended durations
    - **Scalability Testing**
* **Objective:** Assess the system’s ability to handle increasing workloads or data volumes.
* **Approach**:
* Horizontal scaling: Adding more nodes to the system.
* Vertical scaling: Increasing resource capacity on existing nodes.
  + - **Usability Testing**: Ensure the UI is intuitive, easy to navigate, and meets user expectations. This involves testing the user interface with real users to gather feedback on usability. Assess user-friendliness and compliance with usability standards.
      * + Conduct user testing sessions with diverse user profiles.
        + Evaluate ease of navigation and intuitive design.
    - **Performance Testing**: Assess the responsiveness, load times, and stability of the UI under different conditions. This includes load testing, stress testing, and endurance testing.
    - **Security Testing**
      * **Objective**: Identify vulnerabilities and ensure data protection.
      * **Tools**:
        + TBD
      * **Techniques**:
        + Vulnerability Scanning
        + Penetration Testing
        + Authentication and Authorization Validation
        + Data Encryption Testing

.

* + - **Compatibility Testing**: Test the UI across various browsers (Chrome, Firefox, Safari, Edge) and devices (desktop, tablet, mobile) to ensure consistent behaviour and appearance.
    - Verify that the application renders correctly on various devices/browsers using **BrowserStack**.

|  |  |  |
| --- | --- | --- |
| **Tier 1 browsers – full support for n-2 versions** | | |
| **Tier 2 browsers - P1 and P2 functionality are maintained and work** | | |
| **Tier 3 browsers – No testing carried out and not officially supported** | | |
|  |  |  |
| **Desktop** | | |
| **Tier 1** | **Tier 2** | **Tier 3** |
| Chrome | Firefox | IE |
| Edge | Opera |  |
| Safari | UC |  |
| **Mobile** | | |
| **Tier 1** | **Tier 2** | **Tier 3** |
| Chrome | Firefox | IE |
| Edge | Opera |  |
| Safari | UC |  |

* **Test Levels**:
  + **Integration Testing**: Test interactions between integrated components to ensure they work together as expected. In-Cases where the integrated environment is unavailable use stubs/mocks.
  + **User Acceptance Testing (UAT)**: Validate the UI with end-users to ensure it meets their needs and expectations. This is typically the final phase of testing before the application is released.
    - Conduct UAT sessions with key stakeholders to gather feedback on the new UI and make any necessary adjustments.

**4. Test Environment (QA, UAT & PPE)**

* **Hardware Requirements**: Specify the hardware needed for testing, including different devices for compatibility testing.
* **Software Requirements**: List the software, including operating systems, browsers, and any other tools required for testing.
* **Test Data**: Define the data required for testing, including any specific scenarios or edge cases.

|  |  |  |  |
| --- | --- | --- | --- |
| **Environment** | **IP Address** | **Resources Details** | **Software’s** |
| QA |  |  |  |
| UAT |  |  |  |
| PPE |  |  |  |

**5. Test Tools**

* **Automation Tools**: Selenium, Rest-Assured with Cucumber for automated functional testing. Mobile browsers functional testing to be performed using Appium tool

**\*\*** Existing Cypress automation suite to be migrated to Cucumber Java to align it with LSEG unified framework

**\*\*** Axios API automation suite to be migrated to Cucumber Java

* **Performance Testing Tools**: JMeter or other performance tool to be user to assess performance.
* **Bug Tracking Tools**: JASANA
* **Test Management Tools**

**6. Test Deliverables**

* **Test Plan**: A detailed plan outlining the testing activities, schedule, and resources.
* **Test Cases**: Documented test cases covering all aspects of the UI transformation.
* **Test Scripts**: Automated test scripts for functional and NFR’s.
* **Test Reports**: Reports on test execution, including pass/fail status, defects, and overall quality assessment.
* **Defect Reports**: Detailed documentation of identified defects, including steps to reproduce, severity, and status.

**7. Roles and Responsibilities**

* **Test Manager**: Oversee the testing process, ensure resources are allocated, and manage the testing schedule.
* **Test Lead**: Lead the testing team, coordinate testing activities, and ensure adherence to the test plan.
* **Test Engineers**: Execute test cases, report defects, and collaborate with developers to resolve issues.
* **Developers**: Fix identified defects and collaborate with testers to ensure quality.
* **Business Analysts**: Provide requirements, clarify doubts, and validate that the UI meets business needs.

**8. Schedule**

* **Milestones**: Define key milestones and deadlines, including the start and end of each testing phase.
* **Test Phases**: Outline the phases of testing (e.g., unit testing, integration testing, system testing, UAT) and their timelines.

**9. Risk Management**

* **Risk Identification**: Identify potential risks that could impact the testing process or the quality of the UI.
  + Delays in development, lack of test data, or insufficient test environments.
* **Risk Mitigation**: Define strategies to mitigate identified risks, such as additional testing, contingency plans, or resource allocation.
  + Allocate additional resources to address potential delays or create backup test environments.
* **Contingency Plan**: Plan for handling unforeseen issues, including backup resources and alternative testing approaches.
  + Have a backup test environment ready in case the primary environment becomes unavailable.

**10. Entry and Exit Criteria**

* **Entry Criteria**: Conditions that must be met to start testing, such as completion of development, availability of test environments, and readiness of test data.
* **Exit Criteria**: Conditions that must be met to conclude testing, such as completion of all test cases, resolution of critical defects, and approval from stakeholders.

**11. Suspension Criteria**

**Critical Defects**:

Testing is suspended if a critical defect is discovered that blocks further testing activities.

Example: The application crashes frequently, making other features inaccessible.

**Environment Unavailability**:

Testing is halted if the test environment is unstable, not properly configured, or unavailable.

Example: Network outages, server crashes, or incomplete test setup.

**Unavailability of Resources**:

Testing stops if critical resources, such as test data, tools, or team members, are unavailable.

Example: Delays in receiving required datasets or the absence of key testers.

**Incomplete Deliverables:**

Testing is suspended if the prerequisites, such as code, builds, or design documents, are incomplete or not delivered on time.

Example: Missing API endpoints or incomplete user interfaces.

**Non-Compliance with Entry Criteria:**

Testing cannot proceed if entry criteria defined in the test plan are not met.

Example: The development team did not fix high-priority bugs from the previous phase.

**Performance Issues:**

Testing is paused if severe performance degradation prevents meaningful test execution.

Example: Extremely slow system response times during load testing.

**Test Tool Failures:**

Testing is halted if automation or performance testing tools fail or generate incorrect results.

Example: A critical testing framework crashes and cannot be repaired immediately.

**Major Changes in Requirements:**

Testing stops if there are significant requirement changes that invalidate current test cases.

Example: The application design changes midway through testing.

**12. Communication Plan**

* **Status Meetings**: Schedule regular meetings to discuss progress, issues, and next steps.
* **Reporting**: Define the format and frequency of test reports, including daily or weekly status updates, defect reports, and final test summary.