**1. Introduction**

This test plan outlines the strategy, scope, objectives, resources, and schedule for the testing activities required for both the UI (demoqa.com) and API (reqres.in) platforms. It includes both manual and automation testing approaches to ensure thorough validation of the specified features.

**2. Scope**

The document mainly targets the UI testing and API Testing as per Requirements Specifications provided by Client.

* **UI Testing (demoqa.com)**

- Registration: https://demoqa.com/register -> Register button

- Login: https://demoqa.com/login -> Login

- Search: https://demoqa.com/books -> Search for any of the existing books

* **API Testing (reqres.in)**

- Create user: POST /api/users

- Get list of users: GET /api/users (to check new registered users)

**3. Objectives**

- Validate the functionality, performance, and reliability of the UI and API features.

- Ensure that the UI elements and API endpoints function as expected with different input scenarios.

- Identify and report any defects or issues found during the testing process.

**4. Testing Approach**

**Manual Testing**

- **Objective**: Perform detailed manual testing to validate the core functionalities of the UI and API features.

- **Scope**:

- Execute test cases to cover positive and negative scenarios.

- Validate user interactions, data flow, and error handling.

- Ensure usability and interface consistency.

**- Tools**: Jira and X-ray for test management, test case maintenance, and execution and bug reporting.

**Automation Testing**

- **Objective**: Automate critical test cases to ensure quick and efficient regression testing.

- **Scope**:

- Automate high-priority test cases for both UI and API.

- Use automation frameworks to execute tests and validate results.

- Integrate automated tests into the CI/CD pipeline for continuous testing.

**- Tools**: Jira for tracking automation progress and bug reporting.

**5. Frameworks and Tools**

**UI Automation**

- **Framework**: TestNG

- **Language**: Java

- **Tools**: Selenium WebDriver (for test management), Maven(for build management)

**API Automation**

- **Framework**: TestNG

- **Language**: Java

- **Tools**: RestAssured (for test management), Maven(for build management)

**Manual Test Case Creation and Execution**

- **Tool**: Jira integrated with X-ray, Postman(for API test execution)

**Bug Reporting**

- **Tool**: Jira

**6. Test Deliverables**

- **Test Plan Document**: Outline of the testing strategy and approach.

- **Manual Test Cases**: Detailed test cases covering all functionalities.

- **Automation Test Scripts**: Scripts for automated test execution.

- **Bug Reports**: Reports of any defects found during testing.

- **Test Summary Report**: Summary of the testing activities and results.

**7. Test Environment**

- **UI Testing**:

- Browser: Chrome, Firefox

- OS: Windows, macOS

- **API Testing**:

- Postman for manual API testing

- Java environment for RestAssured

**8. Defect Management Process**

**Defect Lifecycle**

1. **New**
   * When a defect is first identified, it is logged into the defect tracking system (e.g., Jira) and assigned a status of "New". The details of the defect, including steps to reproduce, screenshots, and any relevant logs, are provided.
2. **Assigned**
   * The defect is reviewed by the Test Lead or Project Manager and assigned to the appropriate developer for resolution. The status is updated to "Assigned".
3. **Open**
   * The assigned developer begins working on the defect, changing the status to "Open". During this phase, the developer analyzes the issue, reproduces it, and works on a fix.
4. **Fixed**
   * Once the developer has resolved the defect, they change the status to "Fixed" and provide details about the fix, including code changes and any relevant comments.
5. **Retest**
   * The defect is then assigned back to the tester for verification. The status is updated to "Retest". The tester retests the defect using the provided steps to ensure it has been resolved.
6. **Closed**
   * If the defect has been successfully fixed and passes the retest, the tester changes the status to "Closed". The defect is considered resolved.
7. **Reopen**
   * If the defect is not fixed or if it reoccurs during retesting, the tester reopens the defect, changing the status to "Reopen", and assigns it back to the developer. The lifecycle then follows the same steps from "Open".
8. **Deferred**
   * In some cases, a defect may be deferred to a future release due to low priority or resource constraints. The status is changed to "Deferred" and documented with reasons for deferral.

**Severity and Priority Definitions**

**Severity**

Severity refers to the impact of the defect on the system's functionality.

1. **Critical**
   * The defect causes a system crash, data loss, or severe functionality impairment, rendering the system unusable. Immediate attention is required.
2. **High**
   * The defect significantly impacts major functionality, but the system is still usable. It needs to be fixed as soon as possible.
3. **Medium**
   * The defect affects minor functionality or causes inconvenience, but a workaround is available. It should be fixed in the normal course of development.
4. **Low**
   * The defect has minimal impact, such as a minor UI issue or typo. It does not affect system functionality and can be fixed at a later time.

**Priority**

Priority refers to the order in which defects should be resolved based on business needs.

1. **P1 (High)**
   * The defect must be fixed immediately as it affects critical functionality and business operations.
2. **P2 (Medium)**
   * The defect should be fixed soon, as it affects important functionality, but does not require immediate resolution.
3. **P3 (Low)**
   * The defect can be fixed at a later time, as it has minimal impact on functionality and business operations.

**Escalation Procedures**

1. **Initial Notification**
   * When a critical or high-severity defect is identified, the tester immediately notifies the Test Lead and Project Manager via email and the defect tracking system.
2. **Review and Assignment**
   * The Test Lead reviews the defect, confirms its severity and priority, and assigns it to the relevant developer for resolution. If necessary, the Test Lead escalates the issue to the Development Manager.
3. **Development Manager Involvement**
   * For critical defects, the Development Manager prioritizes the defect above all other tasks and allocates additional resources if required. A resolution timeline is established.
4. **Stakeholder Communication**
   * The Project Manager communicates the impact of the critical defect to stakeholders, including clients, outlining the steps being taken to resolve it and providing an estimated timeline for the fix.
5. **Daily Status Meetings**
   * For critical defects, daily status meetings are held to track progress, address any roadblocks, and ensure timely resolution. The Test Lead, Development Manager, and relevant team members participate in these meetings.
6. **Escalation to Senior Management**
   * If the defect remains unresolved beyond the established timeline, the issue is escalated to senior management for further intervention and decision-making.
7. **Post-Resolution Review**
   * Once the defect is resolved, a post-resolution review is conducted to analyze the root cause, the resolution process, and any improvements needed in the defect management process to prevent future occurrences.

#### 9. Test Case Development and Management

* **Test Case Development Approach:** Test cases will be developed based on the requirements documents, user stories, and design specifications. Each test case will include a clear description, preconditions, test steps, expected results, and postconditions. The test cases will be reviewed by the QA Lead to ensure completeness and accuracy.
* **Test Case Management:** All test cases will be documented and managed using Jira integrated with X-ray. This tool will be used to track the development, execution, and results of each test case. Updates and maintenance of test cases will be performed as necessary to reflect changes in requirements or system behaviour.
* **Test Case Coverage:** The test cases will cover all major functionalities and features of the application, including both UI and API testing. Specific focus will be given to critical business workflows, security, performance, and edge cases. Both positive and negative scenarios will be included.
* **Test Case Prioritization:** Test cases will be prioritized based on the risk and impact of the functionality being tested. High-risk and high-impact areas will be tested first, followed by medium and low-priority test cases. This prioritization will ensure that critical functionalities are validated early in the testing cycle.
* **Test Case Traceability:** A traceability matrix will be maintained to ensure that all requirements are covered by corresponding test cases. Each test case in Jira will be linked to its respective requirement or user story to ensure comprehensive coverage and easy tracking.

**10. Test Schedule**

- **Test Planning**: 2 days

- **Manual Test Case Creation**: 3 days

- **Manual Testing Execution**: 3 days

- **Automation Test Script Development**: 5 days

- **Automation Testing Execution**: 2 days

- **Bug Reporting and Validation**: 5 days

- **Test Summary Report Preparation**: 1 day

**11. Roles and Responsibilities**

- **Test Manager**: Oversee the entire testing process, ensure resources and environment are in place.

* + - Number of resources: 1

- **Manual Testers**: Create and execute manual test cases, report bugs.

* + - Number of resources: 1

- **Automation Engineers**: Develop and execute automation scripts, integrate with CI/CD pipeline.

* + - Number of resources: 1

- **Developers**: Develop code and fix defects found during testing.

* + - Number of resources: 2

**12. Entry and Exit Criteria**

**Entry Criteria**

- Development of features is complete and code has been merged successfully.

- Development has completed unit testing.

- Test scripts are completed, reviewed and approved by the Project Team.

- Test environment is set up and accessible.

- Test data is available.

**Exit Criteria**

- All planned test cases have been executed.

- All critical and major defects have been fixed and verified.

- Test summary report has been prepared.

- All expected and actual results are captured and documented with the test scripts.

- All test metrics collected based on reports from daily and Weekly Status reports.

- All defects are logged in Jira.

**13. Risk and Mitigation**

- **Risk**: Unavailability of test environment.

- **Mitigation**: Coordinate with the dev team to ensure environment readiness.

- **Risk**: Delay in test script development.

- **Mitigation**: Prioritize critical test cases and parallelize script development.

**14. Communication Plan**

- **Daily Standup Meetings**: Discuss progress, blockers, and plan for the day.

- **Weekly Status Reports**: Summarize the testing activities and progress.

- **Bug Triage Meetings**: Review and prioritize reported defects.

**15. Approvals**

|  |  |  |
| --- | --- | --- |
|  | **Project Manager** | **Test Manager** |
| **Name** |  |  |
| **Signature** |  |  |