Test Plan

**Inventory Management System**

**(Pactfi)**

Prepared By: Sarowar Alam Saidi

Preparation Date: 22/11/2023

Approved By: [Project Manager/Tech manager]

Approval Date:

Table of Contents

[1. Introduction: 4](#_Toc151604431)

[1.1 Purpose: 4](#_Toc151604432)

[1.2 Scope: 4](#_Toc151604433)

[In Scope: Features considered in the scope of this test plan 4](#_Toc151604434)

[2. Test Resource & Environment: 5](#_Toc151604435)

[2.1 Application URL: 5](#_Toc151604436)

[2.2 Browsers: 5](#_Toc151604437)

[2.3 Devices: 5](#_Toc151604438)

[2.4 Human Resource: 5](#_Toc151604439)

[2.5 Test Tools: 5](#_Toc151604440)

[2.6 Test Environment: 5](#_Toc151604441)

[3. Test Methodology & Strategy: 5](#_Toc151604442)

[3.1 Methodology: 5](#_Toc151604443)

[3.1.1 Suspension Criteria 6](#_Toc151604444)

[3.1.2 Test completeness/Exit Criteria 6](#_Toc151604445)

[3.2 Test Strategy: 6](#_Toc151604446)

[3.2.1 Manual Testing: 6](#_Toc151604447)

[3.2.2 Automated Testing: 7](#_Toc151604448)

[3.2.3 Recommendations: 8](#_Toc151604449)

[4. Test Automation Recommendation: 8](#_Toc151604450)

[4.1 Automation Tools: 8](#_Toc151604451)

[4.1.1 Cross-Browser Testing: 8](#_Toc151604452)

[4.1.2 Headless Mode: 8](#_Toc151604453)

[4.1.3 Page Navigation: 8](#_Toc151604454)

[4.1.4 Element Interactions: 8](#_Toc151604455)

[4.1.5 Parallel Execution: 9](#_Toc151604456)

[4.1.6 API for Browser Automation: 9](#_Toc151604457)

[4.1.7 Performance Testing: 9](#_Toc151604458)

[4.1.8 Continuous Integration (CI) Integration: 9](#_Toc151604459)

[4.1.9 Handling Asynchronous Operations: 9](#_Toc151604460)

[4.1.10 Rich Debugging Capabilities: 9](#_Toc151604461)

[4.1.11 Screenshots and Video Recording: 9](#_Toc151604462)

[4.1.12 Cross-Platform Support: 9](#_Toc151604463)

[4.1.13 Human Resource consideration: 9](#_Toc151604464)

[4.2 Programming Language: 10](#_Toc151604465)

[4.3 Page Object Model: 10](#_Toc151604466)

[4.4 Version Control: 10](#_Toc151604467)

[4.5 Maintenance: 10](#_Toc151604468)

[4.6 Training & Skill Development 10](#_Toc151604469)

[5. Risk Analysis 10](#_Toc151604470)

[5.1 Potential Risks: 10](#_Toc151604471)

[5.1.1 Resource Availability: 10](#_Toc151604472)

[5.1.2 Tool Compatibility and Learning Curve: 10](#_Toc151604473)

[5.1.3 Technical Challenges: 11](#_Toc151604474)

[5.1.4 Data Quality: 11](#_Toc151604475)

[5.1.5 Automated Test Maintenance: 11](#_Toc151604476)

[5.1.5 Concurrent Test Execution: 11](#_Toc151604477)

[6 Milestones / Deliverables 12](#_Toc151604478)

[6.1 Test Schedule 12](#_Toc151604479)

[6.2 Deliverables 12](#_Toc151604480)

# 1. Introduction:

## 1.1 Purpose:

The purpose of this test plan is to ensure the proper functioning and reliability of the Inventory Management System for the computer warehouse. We will be following

## 1.2 Scope:

This test plan covers the testing of the key features outlined in the provided user stories for the Inventory Management System.

### In Scope: Features considered in the scope of this test plan

1. Add new computer
   1. Confirm the presence of the “Add New” Feature
   2. Confirm the functionality of the feature
   3. Validate the input fields
   4. Validate data saving and canceling
2. Filter computer by name
   1. Confirm the presence of the "Filter by name" button and input field.
   2. Validate filtering with valid and invalid computer names.
   3. Verify clearing of the filter and proper error handling
3. Edit/Update computer information
   1. Confirm the presence of clickable computer names.
   2. Validate navigation to the editing page.
   3. Confirm proper pre-population of computer information for editing.
   4. Validate successful editing and cancellation of changes.
   5. Test editing with invalid data and proper error handling
4. Filter columns by ASC and DESC order
   1. Confirm the presence and click ability of the columns "Introduced," "Discontinued," and "Company."
   2. Validate sorting in both ascending and descending order for each column.
   3. Confirm default sorting orders.
   4. Check handling of missing or null values in sorted columns.
5. Navigate to next pages
   1. Verify the presence and functionality of the "Next" button.
   2. Confirm proper navigation to the next page.
   3. Validate the behavior of the "Next" button on the first and last pages.
6. Concurrent request/race condition
   1. Confirm that multiple users can navigate through pages without conflicts.
   2. Validate that the state of the pagination buttons is consistent across users.

# 2. Test Resource & Environment:

## 2.1 Application URL:

https://computer-database.gatling.io/computers

## 2.2 Browsers:

The application will be tested on the latest versions of Google Chrome, Mozilla Firefox, Microsoft Edge and Safari (web-kit).

Mobile web also will be considered for same browsers.

## 2.3 Devices:

The application will be tested on desktop devices and mobile emulators.

## 2.4 Human Resource:

Resource will be working and roles.

|  |  |
| --- | --- |
| **Role** | **Tasks** |
| QA Manager/Lead | Test Plan, Strategy, Audit |
| Tester | Design and execution manual & automated tests |

## 2.5 Test Tools:

|  |  |
| --- | --- |
| **Tools** | **Description** |
| Jira/Shared sheet | Requirement, RTM, Bug Tracking & Reporting |
| Browser Dev tool | Debug |
| Playwright | Automation |

## 2.6 Test Environment:

A dedicated testing environment will be used for both manual and automation testing purpose.

# 3. Test Methodology & Strategy:

## 3.1 Methodology:

Agile methodology will be followed throughout the project hence considering the test will be following agile. So can team communication can be done simultaneously to achieve the goal.

System testing will be conducted to evaluate the system’s compliance with its specified requirements and scopes

### 3.1.1 Suspension Criteria

If the team members report that there are 40% of test cases failed, suspend testing until the development team fixes all the failed cases.

### 3.1.2 Test completeness/Exit Criteria

* 100% Test coverage.
* 100% Manual and automation test executions.
* Open bugs are fixed or in bug triage.

## 3.2 Test Strategy:

The most suitable testing strategy for the [test scope](#_In_Scope:_Features) outlined above is a combination of manual and automated testing. Each type of testing has its advantages and is appropriate for different aspects of the application.

Manual testing is must needed to make sure the stability while being build and integrated and

By implementing test automation can be achieved efficient and effective testing of the Inventory Management System, ensuring rapid feedback on application quality and reducing the testing cycle time.

### 3.2.1 Manual Testing:

**Objective 1 - Validate Add New Feature:**

Manual Testing: Manually test the presence and functionality of the "Add New" button, ensuring the functionality of the adding feature and validity of the form fields.

Scope: Check the visual appearance, responsiveness, and user experience.

**Objective 2 - Validate Navigation Buttons:**

Manual Testing: Manually test the presence and functionality of the "Next" button, ensuring proper navigation through paginated lists.

Scope: Check the visual appearance, responsiveness, and user experience of the navigation buttons.

**Objective 3 - Confirm Sorting Functionality:**

Manual Testing: Manually click on column headers, observe sorting behavior, and confirm default sorting orders.

Scope: Human validation is effective in checking the visual correctness of sorted columns.

**Objective 4 - Confirm Editability of Computer Names**:

Manual Testing: Manually test the ability to click on computer names, navigate to edit pages, and validate the editing process.

Scope: Human intervention is crucial to assess the user interface, pre-population of data, and the overall editing experience.

**Objective 5 - Validate Filtering by Computer Name:**

Manual Testing: Manually test the "Filter by name" functionality with various inputs, ensuring proper filtering and error handling.

Scope: Human validation is effective in assessing the correctness of filtered results and error messages.

**Objective 6 - Test Pagination with Concurrent Requests:**

Manual Testing: Manually test the pagination functionality with multiple users, ensuring there are no conflicts.

Scope: Human observation is essential to understand and validate the behavior of pagination buttons in a concurrent user scenario.

### 3.2.2 Automated Testing:

**Objective 1 - Confirm Sorting Functionality**:

Automated Testing: Use automated tests to validate sorting functionality, especially for large datasets and various scenarios.

Scope: Automated tests can efficiently cover a wide range of scenarios, including sorting and default order checks.

**Objective 2 - Validate Filtering by Computer Name:**

Automated Testing: Automate tests for the "Filter by name" functionality with various inputs, checking for correct filtering and error handling.

Scope: Automated tests can efficiently cover a large number of test cases, ensuring robust validation of the filtering feature.

**Objective 3 - Test Pagination with Concurrent Requests:**

Automated Testing: Automated tests can simulate concurrent user scenarios, verifying that the pagination functionality works seamlessly under load.

Scope: Automated tests are efficient in simulating multiple users and assessing system behavior under concurrency.

### 3.2.3 Recommendations:

1. Begin with manual testing for user interface-related aspects, visual correctness, and overall user experience.
2. Use automated testing for repetitive and data-intensive scenarios, such as sorting and filtering with large datasets.

# 4. Test Automation Recommendation:

## 4.1 Automation Tools:

Considering the web-based nature of the application, using Playwright for automation testing is recommended for several reason.

Implementing test automation with Playwright, and TypeScript, following best practices like POM and integrating with CI/CD, will provide a robust foundation for automating the testing of the application. Regular collaboration, documentation, and ongoing training will contribute to the success and sustainability of the test automation effort.

Reason playwright is recommended:

### 4.1.1 Cross-Browser Testing:

Playwright supports cross-browser testing, allowing you to verify that the functionality of your application is consistent across different browsers (e.g., Chrome, Firefox, Safari).

### 4.1.2 Headless Mode:

Playwright can run tests in headless mode, meaning the browser interface is not displayed during testing. This is useful for running tests in the background, optimizing execution speed, and minimizing resource usage.

### 4.1.3 Page Navigation:

Playwright simplifies page navigation and interaction, making it easy to simulate user actions like clicking buttons, navigating between pages, and interacting with form fields.

### 4.1.4 Element Interactions:

Playwright provides robust support for interacting with elements on a webpage, making it suitable for validating the behavior of buttons, links, and input fields in the user stories.

### 4.1.5 Parallel Execution:

Playwright supports parallel test execution, allowing you to run multiple tests concurrently. This can significantly reduce the overall test execution time.

### 4.1.6 API for Browser Automation:

Playwright's API for browser automation is easy to use and provides a wide range of functionalities. It allows you to script interactions with the browser, making it suitable for both simple and complex test scenarios.

### 4.1.7 Performance Testing:

Playwright can be used for performance testing by simulating multiple users navigating through the application simultaneously. This is valuable for testing the pagination functionality under load.

### 4.1.8 Continuous Integration (CI) Integration:

Playwright can be integrated into CI/CD pipelines, enabling automated testing as part of the development and deployment process.

### 4.1.9 Handling Asynchronous Operations:

Playwright handles asynchronous operations well, which is crucial for testing modern web applications that often involve asynchronous JavaScript.

### 4.1.10 Rich Debugging Capabilities:

Playwright provides detailed debugging information, making it easier to identify and fix issues during test development.

### 4.1.11 Screenshots and Video Recording:

Playwright allows you to capture screenshots and record videos during test execution. This is helpful for visually verifying the state of the application at different points in the test.

### 4.1.12 Cross-Platform Support:

Playwright is cross-platform, supporting testing on different operating systems. This ensures that your tests are reliable and consistent across various environments.

### 4.1.13 Human Resource consideration:

Playwright is ready to use framework and has rich library to support a number of built in automation features which allows human resource with similar knowledge to learn and master the tool faster comparing to other tools.

## 4.2 Programming Language:

Playwright support multiple language. TypeScript or JavaScript can be used as programming languages for playwright scripts. The choice may depend on the team's expertise and preferences.

## 4.3 Page Object Model:

POM promotes a modular and maintainable structure for automation code. It separates page-specific elements and interactions into dedicated classes, improving code readability and reusability. Though not mandatory in playwright but adopting its principles can significantly improve the structure and maintainability of automated code.

## 4.4 Version Control:

Utilize Git for version control, enabling collaboration among team members and maintaining a versioned history of test scripts.

## 4.5 Maintenance:

Automation scripts should be regularly reviewed and updated to accommodate changes in the application. This ensures that tests remain reliable and aligned with the evolving application.

## 4.6 Training & Skill Development

Need to ensure that team members are proficient in the selected tools and technologies. Continuous skill development helps maximize the benefits of automation.

# 5. Risk Analysis

## 5.1 Potential Risks:

### 5.1.1 Resource Availability:

Risk: The single resource assigned to testing may face unexpected interruptions, such as illness or urgent assignments.

Mitigation: Cross-train another team member to be familiar with testing tasks, and maintain open communication to identify potential resource constraints.

### 5.1.2 Tool Compatibility and Learning Curve:

Risk: The selected testing tool (Playwright) may have compatibility issues with the application or may require a significant learning curve.

Mitigation: Allocate time for tool setup and learning. Conduct training sessions if necessary and have a backup plan in case the tool proves unsuitable.

### 5.1.3 Technical Challenges:

Risk: Technical challenges, such as difficulties in setting up the testing environment or unexpected behavior in the application, may delay testing progress.

Mitigation: Conduct a thorough environment setup and testing beforehand. Collaborate closely with the development team to address technical issues promptly.

### 5.1.4 Data Quality:

Risk: Incomplete or inaccurate test data may impact the effectiveness of testing, especially for scenarios like filtering and sorting.

Mitigation: Ensure the availability of realistic and representative test data. Implement data validation checks to detect discrepancies.

### 5.1.5 Automated Test Maintenance:

Risk: Maintenance of automated tests may become challenging, especially if the application undergoes frequent changes.

Mitigation: Plan for regular reviews and updates of automated test scripts. Use version control for scripts to track changes and facilitate collaboration.

### 5.1.5 Concurrent Test Execution:

Risk: Simulating concurrent user scenarios in automated tests may uncover performance issues or unexpected behavior in the application.

Mitigation: Gradually increase the load in automated tests to identify and address potential performance bottlenecks. Collaborate with the development team to optimize performance.

# 6 Milestones / Deliverables

## 6.1 Test Schedule

Here 1 story point is equivalent to 8 hours = 1sp. Considering 7 days sprint.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task Name** | **Start** | **Finish** | **Effort** | **Comments** |
| Requirement review & Test Planning |  |  | 1sp |  |
| Test Case Design |  |  | 1sp |  |
| Environment Setup |  |  | 0.5sp |  |
| Manual System Testing |  |  | 1sp |  |
| Automate test scripts |  |  | 1sp |  |
| Automated testing execution |  |  | 0.5sp |  |
| Bug reporting & Documenting |  |  | 1sp |  |
| Regression |  |  | 0.5sp |  |
| Formal Run |  |  | 0.5sp |  |
| Release to Production |  |  | 0.5sp |  |

## 6.2 Deliverables

|  |  |  |
| --- | --- | --- |
| **Deliverable** | **For** | **Date** |
| Test Plan | Project Manager, QA Lead, Test Team |  |
| Traceability matrix | Project Manager, QA Lead |  |
| Test Result | Project Manager |  |
| Metrics | All team members |  |
| Release Note | Production |  |