QA TEST PLAN: CUSTOM PAGE TESTING FOR PLANNING CENTER PUBLISHING

Table of Contents

Introduction

1.1 Purpose:

The test plan has been created to communicate the test approach to all team members. Its purpose is to validate the functionality, performance, and security of the custom page developed by Planning Center Publishing. This document presents the scope, risks, test approaches, test environments, testing methods, milestones and deliverables, and a completion and exit strategy.

1.2 Scope:

This plan will cover functional testing, user acceptance testing (UAT), performance testing, cross-browser compatibility, security checks, and responsive testing on various devices and browsers.

Performance testing will be considered part of this project due to the large user base and concurrent users using the site hourly or daily.

Rewriting, moving, or archiving existing test cases from the existing Word documents will be revisited during future iterations as features are released. This will keep testing up to date with new updates and remove redundant testing to maintain SLA and feature release schedule.

1.3 Team Members:

Role	Name	Responsibilities
QA Lead	[Name]	Define test plans, review test cases, and manage overall execution.
QA Engineer/Tester	Byron Munoz	Execute manual and automated tests and report defects.
Developer	[Name]	Provide builds for testing and assist with bug fixes.
UI/UX Designer	[Name]	Validate that the custom page adheres to design guidelines.
Product Manager	[Name]	Ensure the custom page meets business and user requirements.
DevOps Engineer	[Name]	Manage the test environment and deployment process.

2 Risks

The following risks have been identified, and appropriate action has been determined to mitigate their impact on the project. The risk's severity is based on how the project would be affected if this particular risk was triggered. The trigger is what event would cause the risk to become an issue and how to mitigate it.

Risk	Impact	Trigger	Mitigation Strategy
Page performance issues when they are under load	Slow page load times or crashes during peak usage.	High traffic volume or inefficient resource handling.	Conduct early performance testing, optimize code, use caching, and CDN to improve performance.
Security vulnerabilities.	Exposure of sensitive data or compromise of the system.	Lack of proper input validation and insecure communication channels.	Perform thorough security testing (e.g., penetration testing), validate with security experts, and use HTTPS.
Cross-browser compatibility	Inconsistent user experience across different browsers.	Page elements render differently or break on specific browsers.	Test the custom page across all targeted browsers early in the development process.
Design inconsistencies	Poor user experience and potential usability issues.	Design elements appear differently than intended across devices.	Regularly validate design with the UI/UX team, ensuring alignment with style guides and responsive design.
Data loss during form submissions	Loss of user input or incomplete data processing.	System crashes or errors during form submission.	Use dummy test data, implement robust backup mechanisms, and log transactions to prevent and recover data loss.
Unclear or incomplete requirements	Incorrect functionality, missing features, or inaccurate testing.	Misinterpretation of requirements or incomplete documentation.	Review requirements with stakeholders, maintain a requirements traceability matrix and clarify any ambiguities.
Deployment and environmental issues	Features may not work as expected due to environmental differences.	Misconfigured test environment and different software versions.	Maintain consistency between test and production environments and use CI pipelines for automated deployment.

Mobile Network Connectivity Issues	Poor performance or broken functionality on slow or unstable mobile connections.	Users are accessing the page from mobile devices on unstable networks.	Optimize for mobile performance (e.g., use lightweight assets) and implement offline fallback for critical functions.
User Behavior Misalignment	Custom page design may not align with the typical user workflow, leading to confusion.	Incorrect assumptions about how users will interact with the page.	Conduct user testing and collect feedback from real users early to ensure the design matches user behavior and expectations.
Feature Scope Creep	An overcomplicated design may confuse users and degrade performance.	Adding too many features beyond the original scope without user validation.	Use agile methodologies to manage scope, prioritize user-focused features, and regularly review the MVP (Minimum Viable Product).
Multiple Control Conflicts	Multiple contributors might cause code conflicts or revert significant changes, impacting the page's stability.	Poor version control management or multiple team members editing simultaneously.	Use a robust version control system (e.g., Git) to enforce clear commit/pull request processes and conduct regular code reviews.
Unforeseen User-Generated Content Issues	Users may post inappropriate or harmful content (if user input is allowed).	Insufficient content moderation or validation mechanisms.	Implement content moderation tools, set up filters to flag inappropriate content, and establish user-reporting mechanisms.

3 Test Approach

The custom page feature on Planning Center Publishing should use either an Agile or Scrum testing approach. Both provide flexibility to adapt to changing requirements and allow continuous testing integration through development.

Exploratory testing will play a large part in the testing, as it focuses on locating unexpected issues by allowing testers to explore the page's functionality freely. This will ensure a smooth user experience by testing real-world scenarios, such as navigating the UI, submitting forms, handling errors, and managing user permissions. Exploratory testing reveals hidden bugs and edge cases that may not be caught using scripts.

3.1 Testing Objectives:

1. **Functionality Verification:** Ensures all features on the custom page (social, buttons, links, event schedule, etc.) work as intended and align with the user's requirements.
2. **UI/UX Validation:** Confirm that the custom page provides an intuitive, user-friendly experience and is visually consistent with the rest of the Planning Center platform.
3. **Performance Testing:** Verify that a custom page loads quickly and efficiently and handles regular and peak traffic.
4. **Cross-browser and Device Compatibility:** Ensures the custom page works across various browsers (Chrome, Firefox, Edge, Safari) and devices (desktop, tablet, mobile).
5. **Security Testing:** Ensure that the custom page is secure, protecting sensitive user data, and adhering to best practices for web security.
6. **Regression Testing:** Confirm that the custom page's functionalities remain stable after updates or new releases.

3.2 Test Scope:

In-scope testing

- Functional testing of all interactive elements (forms, buttons, links).
- Verifying page layout consistency across different browsers and devices.
- Ensuring the page adheres to the design and functionality specifications.
- Load and stress testing for performance validation.
- Security validation to protect user data and prevent vulnerabilities.
- Multi-user collaboration to ensure no crashes while multiple users update/edit the custom page(s)

Out-of-Scope

- API or back-end server testing (assumed to be handled separately).

3.3 Test Strategy:

Test strategy will require both manual and automated testing methods

- Manual Testing:
    - Primarily exploratory, functional, and UI/UX testing
- Automated Testing:
    - Covers regression and performance testing through scripts

Testing Tools:

- Selenium for automated UI testing.
- BrowserStack for cross-browser and device testing.
- TestRail for test case management.
- LoadRunner for load testing.
- Metasploit for basic security testing.

4 Test Environment

- **Operating Systems:** Windows 10, Windows 11, Linux, macOS, iOS, Android
- **Browsers:** Chrome, Firefox, Safari, Edge

- **Devices:** Desktop, Tablet, Mobile (various screen resolutions)
- **Network:** High-speed and slow internet conditions will be incorporated.

4.1 Test Cases

4.1.1 Functional Tests

Test Case Description	Precondition	Tests Steps	Expected Result
Verify the custom page loads without errors	The user is logged in	1. Navigate to the custom page. 2. Observe page load behavior.	The custom page loads without errors in under 3 seconds.
Test form submission functionality	Form fields are empty	1. Fill in the form. 2. Submit the form.	The form is submitted successfully, and the user sees a confirmation message.
Validate error handling on invalid form inputs	Form fields are present	1. Enter invalid inputs. 2. Submit the form.	Proper error messages are shown next to the invalid fields.

4.1.2 UI/UX Tests

Test Case Description	Precondition	Tests Steps	Expected Result
Validate page layout on desktop, tablet, and mobile	User is logged in	1. Open the custom page on desktop, tablet, and mobile devices.	The layout adapts correctly to different screen sizes.
Verify button and link functionality	Page is loaded	1. Click each button and link. 2. Observe the actions.	All buttons/links trigger the correct actions or navigate properly.

4.1.3 Performance Tests

Test Case Description	Precondition	Tests Steps	Expected Result
Validate page load time under normal conditions	User is logged in	1. Load the page and measure load time.	Page loads within 2-3 seconds under normal conditions.
Stress test with 500 concurrent users	User traffic is simulated	1. Simulate 500 users accessing the page concurrently.	Page performance remains stable, with minimal latency.

4.1.4 Cross-Browser Compatibility

Test Case Description	Precondition	Tests Steps	Expected Result
Test compatibility on Chrome, Firefox, Safari, and Edge	Page is live	1. Open the page on different browsers. 2. Verify UI consistency.	The page works consistently and without issues across all browsers.

4.1.5 Security Tests

Test Case Description	Precondition	Tests Steps	Expected Result
Verify form input validation against SQL injection	Form is active	1. Enter SQL injection string in input fields. 2. Submit form.	The form should sanitize input and prevent SQL injection attacks.
Check HTTPS encryption for a secure connection	User accesses the page	1. Navigate to the page and check the URL for HTTPS.	The page should be secured with HTTPS encryption.

5 Milestones & Deliverables

5.1 Test Schedule:

The initial test schedule follows…….

Phase	Duration	Team Member
Test Plan Review	3 Days	QA Lead, Product Manager
Environment Setup	1 Day	DevOps Engineer
Test Case Design	3 Days	QA Engineers
Test Case Execution	5 Days	QA Engineers
Bug Reporting & Triage	Ongoing	QA Engineers, Developers
Regression Testing	2 Days	QA Engineers
Final Review	1 Day	QA Lead, Product Manager

5.2 Deliverables:

- Detailed Test Plan
- Tests Case Documentation
- Test Execution Report
- But Reports (tracking in the bug tracking system)
- Final Test Summary

6 Entry & Exit Criteria

6.1 Entry Criteria:

- Custom Page Development is complete and ready for testing
- Test environment is set up and accessible
- Test data is available for all input fields and scenarios

6.2 Exit Criteria:

- All test cases executed with 100% coverage.
- No critical or high-severity bugs remain unresolved.
- The test summary report has been reviewed and approved by stakeholders.