Company, product name

Custom Page
Test Plan

Version 1.0

Revision History

Date	Version	Description	Author
10/14/2024	1.0	Create test plan for Custom Page	Hanna Kasmachova

Test Plan

Introduction

1. Purpose

The purpose of this test plan is to describe the process of testing the Custom Page feature in the Publishing module. The Custom Page functionality allows users to create and publish custom web pages. It is also necessary to verify that the page creation and publishing process works smoothly, and that the published page displays without distortion and exactly as the user created it across supported browsers and devices.

2. Background

The target of this test is the Custom Page creation feature within Company Name, a tool designed to help organizations, particularly churches, manage and streamline their operations online. This feature empowers users to build, customize, and publish unique web pages to fit their specific needs. Its intuitive drag-and-drop interface allows non-technical users to create professional-looking pages without writing code. Users can modify layout structures, incorporate media, and embed interactive elements such as forms, creating a seamless experience for managing online content.

Key functions include:
1 Drag-and-Drop Page Builder: Enables users to design pages by arranging text, images, videos, and other elements in customizable layouts.
2 Style Customization Tools: Allow users to adjust fonts, colors, and overall page styling, ensuring a consistent look and feel across their custom pages.
3 Media and Form Integration: Facilitates the addition of images, videos, or embedded forms for data collection, improving engagement with site visitors.
4 Publishing and Scheduling: Supports immediate or scheduled publishing of pages, with version control features that allow rolling back to previous versions.

The architecture of the Custom Page feature is built with a modular front-end framework (HTML5, CSS3, React) for responsive design. It interfaces with backend services via APIs to save and retrieve content changes, while integrating data management and publishing workflows within the Planning Center's cloud-based infrastructure. The platform ensures ease of use and security.

The CompanyName started as a service to simplify church management. Over time, it expanded into a comprehensive platform, including features like PC Publishing, to meet the broader needs of churches and other organizations. The Custom Page feature was introduced to offer flexibility in content creation and management, focusing on empowering users to maintain their web presence without relying on developers.

With continuous updates, the tool has become integral for organizations seeking to engage their communities with custom, dynamic content.

3. Scope

The development proceeds according to the N methodology.
The test scope outlines what will be tested and what will not be tested for the Custom Page feature.
In-Scope includes:

- Creating Custom page: create a new custom web page, drag-and-drop page builder and adding and arranging content blocks (text, images, videos, buttons), save, update and publishing their custom pages;
- Style customization: Font, color, and layout adjustments; Ensuring that style changes are accurately reflected on the published page.
- Media and Form integration: Uploading media (images, videos) and embedding forms for visitor

interaction.
- Publishing and Scheduling: Immediate and scheduled publishing functionality.
- Cross-Browser and Cross-Device compatibility: Ensuring the feature works across popular browsers and devices(the custom page should look good as a layout and work properly on different web browsers (like Chrome, Firefox, Safari) and devices (like computers, tablets, and smartphones)).
- Error handling**:** Verify that appropriate error messages and alerts are shown for issues during publishing (e.g., network failure or server timeout).
- UI and UX Testing: Ensure the "Publish" button, scheduling interfaces, and status notifications are user-friendly, intuitive, and functional.
- QA Team is to test all new in-cope features and enhancements;
- QA Team is to create and maintain next testing artifacts: Test Plan, Checklist, TestCases;
- QA Team is to verify that all Blocked and Critical defects are fixed before each release goes live

Out-of-Scope:

- Testing database structures and configurations.
- Integration with third-party services such as analytics, SEO tools, or other plugins.
- Testing of external systems like content delivery networks.
- QA Team doesn't test Custom Page on other device and operating systems that weren't required before;
- QA Team isn't to execute security and automation testing;
- QA Team isn't to purpose product enhancements that will affect project timeline

4. Risks and Contingencies

This article outlines potential issues or problems that could arise during the testing process, which may impact the project's timeline, quality, or resources, as well as the plans put in place to address these risks if they occur.
Here are some of the ones:

	Risks	Descriptions	Actions for solutions of the problem
1	Cross-Browser Issues	The custom page may not display correctly in different web browsers, leading to inconsistent user experiences.	Start testing early in multiple browsers to identify and fix issues quickly.
2	Environment Setup Delays	Delays in setting up the testing environment may hinder the testing schedule.	Coordinate with developers early to ensure that the test environment is ready on time.
3	Resource Availability	Team members may become unavailable due to other commitments or emergencies.	Prioritize critical testing tasks to ensure coverage.
4	API Integration Failures	The Custom Page feature relies on APIs (e.g., for saving, publishing, or embedding media), which could fail or cause errors	Set up monitoring for API performance and reliability, and create fallback mechanisms in case an API fails (e.g., error handling and retry options).
5	Browser/Device Version Compatibility	Older versions of browsers or devices might not fully support the Custom Page feature, leading to	Identify the minimum supported versions of browsers and devices early on and communicate this to

		compatibility issues.	users. Perform regression testing on these versions to ensure key functionality works as expected.
6	Incomplete Test Data	If the test data used during testing is incomplete or inaccurate, this could result in missed bugs or incomplete feature testing.	Ensure that comprehensive, real-world test data is created and reviewed before starting testing to cover a variety of user scenarios.
7	Insufficient Test Coverage	Not all possible user scenarios or edge cases may be covered during testing, leading to missed bugs or functionality issues in production.	Perform exploratory testing in addition to scripted tests to catch unexpected issues. Regularly review and update test cases based on user feedback and new requirements.
8	Test Environment Instability	The test environment may not be stable or may differ significantly from the production environment, leading to issues that won't be present in the live system.	Have a backup test environment available to avoid downtime if issues arise in the main environment.
9	Delays in Bug Fixing	Bugs identified during testing may take too long to resolve, causing delays in the testing process and project timeline.	Ensure close collaboration between the QA and development teams to address critical issues quickly. Schedule regression testing cycles to verify that bug fixes do not introduce new issues.
10	Lack of Regression Testing	Changes or updates to the Custom Page feature could introduce new bugs or break existing functionality, especially if previous tests are not rerun.	Schedule regular manual regression testing for areas that cannot be easily automated, ensuring full coverage after updates.
11	Troubles with environment		Quality Assurance Engineer will contact with developer for troubleshooting
12	Troubles with build for testing		Quality Assurance Engineer will contact with developer for troubleshooting
13	Blocker issue for working with application		Quality Assurance Engineer will report a defect in JIRA, contact with Lead and developer for troubleshooting
14	Troubles with device		Quality Assurance Engineer will contact with administration for troubleshooting, contact with Project Manager
15	Trouble with personal computer		Quality Assurance Engineer will contact with administration for troubleshooting. While Quality

			Assurance Engineer wait for troubleshooting Quality Assurance Engineer can test application on device.
16	Start to support another OS/Browser in the end of the project		It needs more time for developers and then for Quality Assurance Engineer to cover the new OS/Browser.
17	Loss of network in the office/home (or trouble with Wi-Fi)		The Quality Assurance Engineer will contact the administration for troubleshooting. While Quality Assurance Engineer wait for troubleshooting Quality Assurance Engineer can test application on behavior if network is lost
18	Lack of devices for testing		Need to find devices as soon as possible. As a general rule the devices will be prepared for the QA team before the test process starts.
19	Lack of QA resources		Need to involve to test process more QA resources
20	Showstopper issues which can block verification		The Quality Assurance Engineer will report a defect in JIRA. The issue should be fixed immediately.
21	Lack of completed requirements		BAs will discuss requirements with customers as soon as possible and will create more completed requirements.

5. Constraints

This topic will describe the limitations. Constraints help define the boundaries within which the testing for the Custom Page feature will be conducted, ensuring that the most critical features are tested while acknowledging the limitations of the testing environment and resources.

- **Image and Video Sizing**: Images are recommended to be at least 700px wide, and videos must be hosted on supported platforms (YouTube, Vimeo, or Boxcast). Testing must verify proper scaling and resolution of images across devices.
- **Content Compatibility**: Custom Pages allow users to add various types of media (text, images, video, and social media links). However, the performance and display of these media types might differ across devices and browsers. Testing must ensure that all media types are displayed correctly on both desktop and mobile devices, but certain older browsers and unsupported media formats might not be fully tested.
- **Limited User Access:** Custom Pages can be set to limited access, restricting visibility to specific membership types. Testing might focus only on predefined roles, and testing edge cases (e.g., mixed permissions) could be restricted.
- **Block Type Customization:** Pages support a variety of block types, including buttons, contact info, and social media links. Testing must verify that each block functions as expected, but some integrations (like social media APIs) may have limitations.Testing of blocks will focus on the core functionality, but external links and third-party integrations might not be tested fully, depending on

the availability of those services during the testing period.

- **Navigation and Page Visibility:** Custom Pages can be added to or removed from the Church Center site navigation. However, some pages with limited access may be visible but not accessible to non-eligible users.
- **Save, Publish, and Archive Functions:** Pages can be saved as drafts, published, or archived. These states must be tested, but testing will be limited to a predefined number of state transitions (e.g., save, publish, discard) for practical reasons.
- **Mobile and Desktop Views:** The layout and appearance of Custom Pages may differ between mobile and desktop views. Testing will focus on ensuring that pages render correctly on both platforms, but extreme layout customizations may not be fully tested.
- **Performance Testing:** The Custom Page feature must handle multiple users accessing and editing pages simultaneously. However, full load testing may be restricted due to limited access to production-level infrastructure during testing.
- **User Interface Changes:** The Custom Page builder allows users to drag and drop blocks to create layouts. While most block types and layouts will be tested, very complex or unusual page structures may not be fully validated due to time constraints and focus on common use cases.

6. Test requirements

This is the product page for CompanyName.
You can find specific details about Custom Pages here.
Requirements for image sizing here, for navigation here.

Test Strategy

1. Testing Types

1.1 Functional Testing

Test Objective:	Verify that users can successfully create, edit, and save custom pages with different content blocks (text, images, videos, etc.).
Technique:	Execute each use case, use-case flow, or function, using valid and invalid data, to verify the following: The expected results occur when valid data is used(For example: Creating a new custom page with valid blocks (text, images, videos) should result in successful creation and display.). The appropriate error or warning messages are displayed when invalid data is used. Each business rule is properly applied.
Completion Criteria:	All planned tests have been executed. All identified defects have been addressed. All blocks (text, image, video, etc.) can be added, edited, and removed without errors.
Special Considerations:	Ensure different block types are correctly handled across browsers and devices. Ensure that all relevant stakeholders (e.g., developers, business analysts) are involved in identifying use cases and business rules to ensure comprehensive test coverage. Be aware of any dependencies on other systems or modules that could affect the functionality of the Custom Page feature during testing.

1.2 UI/UX Testing

Test Objective:	Verify that the Custom Page feature provides a user-friendly interface that allows users to navigate seamlessly through the functions and features of the application. This includes ensuring that all UI elements function as expected.
Technique:	Create or modify tests for each window within the Custom Page feature to verify proper navigation and object states. This includes: Assessing all interactive elements (buttons, dropdowns, text fields) to ensure they respond correctly to user actions. Confirming that visual feedback (like highlighting or enabling/disabling buttons) aligns with user expectations. Checking that all window transitions occur smoothly without errors or delays.
Completion Criteria:	All navigation paths work as intended. All UI elements meet design specifications and usability standards.
Special Considerations:	Ensure testing accounts for various user roles and access levels to confirm that navigation and UI elements function correctly based on permissions.

1.4 System testing

Test Objective:	Verify the overall behavior and interaction between different modules of the system as a whole
Technique:	Execute test cases that focus on how different system components interact with each other(for example different subscriptions on custom pages). Ensure that the integrated system meets the specified requirements and behaves as expected. Test with both valid and invalid inputs, ensuring smooth interaction among modules.
Completion Criteria:	All integrated modules work as intended without errors. System-level requirements and user scenarios are covered and tested. All identified defects are addressed.
Special Considerations:	Ensure data flows smoothly between different system components.

1.5 Compatibility testing

Test Objective:	Ensure the system or application works as expected across different devices, browsers, operating systems, and configurations
Technique:	Execute test cases on various combinations of operating systems (Windows, macOS, Linux, etc.), browsers (Chrome, Firefox, Safari, Edge), and device types (mobile, tablet, desktop). Test the application with different screen resolutions and form factors. Use tools to simulate different environments and configurations if physical devices are unavailable.
Completion Criteria:	The system functions correctly across all targeted platforms, devices, and browsers. UI elements display consistently, and functionality is not broken across various environments. No critical defects related to compatibility remain unresolved.
Special Considerations:	Ensure the system adapts to different screen sizes (responsive design).

1.7 User Acceptance Testing (UAT)

Test Objective:	Verify that the system meets business requirements and is ready for release, based on real-world scenarios.
Technique:	Perform testing from the end user's perspective, using real-world scenarios. Use actual business use cases, confirming that the system supports all necessary operations.
Completion Criteria:	All critical business processes have been successfully tested. Stakeholders have signed off on the tested functionality. All high-severity bugs found during UAT are resolved.
Special Considerations:	Ensure key stakeholders and end-users participate in the UAT process. Perform testing in an environment that closely mimics production. Consider any edge cases specific to the business processes.

1.8 Exploratory testing

Test Objective:	Identify defects or gaps by exploring the application without predefined test cases.
Technique:	Allow testers to navigate through the system intuitively, focusing on discovering issues that might be missed during scripted testing. Focus on unusual user flows, rare edge cases, and non-standard inputs. Track and document any issues or behavior deviations discovered during testing.
Completion Criteria:	All significant parts of the system have been explored. Documented issues are logged for further analyzing
Special Considerations:	Exploratory testing should be done after initial scripted testing to allow testers to think outside predefined use cases. Include testers with both domain knowledge and fresh perspectives to ensure diverse coverage.

1.3 Regression testing

Conduct regression tests for every new feature to ensure previous features (e.g., editing, saving drafts) continue to work after new code is deployed.
Regression testing is to be performed at the end of each iteration (release) by selecting the number of test cases / checklist items which cover part of code / modules that were changed during defects fixing or/and new feature implementation.

1.6 Smoke testing

Quality Assurance Engineer is to select and run a set of checklist items that cover the most important functionality of a single component or system, to ascertain, if a crucial function of the software works correctly. Smoke Test will be performed after any planned delivery and before main testing activities.

2. Entry and exit criteria

Entry criteria:

- Requirements are defined and approved
- Test Environment is ready
- Test Data is prepared
- Test Plan and Test Cases are complete
- Build is stable and development is ending

Exit criteria:

- All test cases are executed, and 95% of them pass
- No critical or high-severity bugs remain open
- Published websites display correctly across all browsers and devices

3. Review and Sign-Off

The test plan will be reviewed by the QA team, developers, and project stakeholders. Once all feedback is addressed, the plan will be signed off by the project manager before starting test execution.

Test Items

1. Testing Environments

Mobile Devices:
iPhone 12 with iOS 15+ (Safari, Chrome, Firefox)
iPad mini
Samsung pad
Samsung Galaxy S21 with Android 11+ (Chrome, Firefox, Samsung browsers)
Browsers:
Google Chrome (latest 2 versions)
Mozilla Firefox (latest 2 versions)
Safari (latest)
Microsoft Edge (latest)

2. Test Management System

Confluence, TestRail, Jira/YouTrack.
QA Team use this type for prioritizing bugs:
Severity Levels:

- Critical: Publishing pages, adding and editing blocks fails, leading to major data loss or crashes.
- High: Publishing pages, adding and editing blocks succeeds, but the website content is not visible or incorrect.
- Medium: Minor UI issues or scheduling delays.
- Low: Non-critical issues like minor notifications delays or visual discrepancies.

Bug Lifecycle:
New → In Progress → Resolved → Retested → Verified → Closed

3. Roles and Responsibilities

Test Lead: Oversee the entire testing process, including managing test schedules, reviewing test cases, and ensuring timely delivery of testing deliverables.
QA Engineers: Execute functional, regression, and cross-browser tests. Report and track defects, verify bug fixes, and retest.
Developers: Fix reported bugs, provide support for setting up the environment, and respond to queries from the QA team.
Project Manager: Approve the test plan, monitor progress, and sign off on the final test results.

4. Test Schedule

- **Test Plan Review & Approval:** Oct 18
- **Test Case Development**: Oct 21 - Oct 23
- **Environment Setup & Configuration**: Oct 24
- **Functional Testing**: Oct 25 - Oct 24
- **Cross-Browser Testing**: Oct 25 - Oct 30
- **Responsiveness Testing**: Oct 31 - Nov 4
- **Bug Fixing & Retesting**: Nov 5 - Nov 11
- **Final Regression Testing & Sign-Off**: Nov 12

5. Test Deliverables

- **Test Plan Document**: Detailed test plan for the Custom Page feature.
- **Test Cases**: Test cases for creating web-pages, scheduled publishing, cross-browser, and cross-device tests.
- **Bug Reports**: Detailed bug reports logged in JIRA, including screenshots, logs, and steps to reproduce.
- **Test Summary Report**: A final report summarizing the test results, defect statistics, and coverage of test cases.
- **Published Test Artifacts**: Websites published through the testing process (sample websites).