Badger-Tech Timeline Test Plan

Table of Contents

1. Introduction

1.1 Purpose

A local museum located on Niagara-On-The-Lake has contacted Brock University, St. Catherines with the task of designing, developing, and implementing an online interactive timeline application. This application will depict the events surrounding the history of Niagara, with special attention given towards the war of 1812.

This document is designed to reference the testing plan, approach, and overall framework that will be used to address the testing process of https://badger-timeline.infinityfreeapp.com/src/ - site. Here, this document will introduce:

- Objectives
- Test Strategies
- Requirements
- Features
- Schedule
- Risks/assumptions
- Tools

Furthermore, this document will introduce the necessary steps taken in the following testing phases:

1. Unit testing
2. Component testing
3. Integration testing
4. System testing (Acceptance testing)

1.2 Project Overview

The https://badger-timeline.infinityfreeapp.com/src/ -site is a powerful web-application that will be responsible for facilitating the process of information passage from the client(Niagara-on-the-lake Museum) to the user(General public).

On the user side, this website will allow users to view information about the museum, the date of any events or workshops sponsored by the museum, create an account for email notifications, and will also have an interactive timeline application that will give the user an experience through Niagara's rich history. This web-app can be accessed and experienced through any device and in any orientation.

On the client side, this website will allow permitted staff members(admins) to edit content directly on the timeline such as texts and images. In addition, admins will also be allowed to edit content on the web such as its services, as well as adding any more additional information about the museum.

2. Objective and Tasks

2.1 Objectives

The objective of this test is to verify that the web-application - https://badger-timeline.infinityfreeapp.com/src/ is working as intended. To verify, all major components of the application must work as intended. The web-application can be considered and thought of as two major components: the webpage component and the timeline component. Both of these components will each be tested separately to check that they are working according to specifications, before finally testing the system as a whole.

The objective of testing the web-page component is to check and verify the various features offered in the homepage. These tests can include and are not limited to checking all links found on the homepage, checking that each button found works as intended, to verify the login/register system, and also to check that the webpages can be readily accessible by various devices such as PCs, Notebooks, and mobile devices. All the while having a reasonable amount of expectations towards the UI/UX associated with each device.

The object of testing the timeline component is to check and verify that the various features implemented in the timeline are all working in accordance to its specifications. These tests can include but are not limited to checking each individual point on the timeline spectrum, checking the various ways to navigate the timeline, checking how the information is presented, editing features for admin users, the sub-timeline itself, image view function, and etc. The timeline application must also be able to operate on various devices such as PCs, Notebooks, and mobile devices, and have a reasonable amount of expectations towards the UI/UX associated with each device.

The final product of this test plan will yield both a production-ready web-application, and also a set of test-cases and scenarios that can be reused whenever a new feature is to be added or changed in the future and/or for maintenance purposes.

2.2 Tasks

This section highlights the various tasks presented in this test plan document. The Tasks are separated and listed by its responsibility as a whole towards the testing plan. These tasks can be subject to change, but can include and not limited to:

- Testing phase
- Post-testing
- Problem reporting
- Retesting
- Acceptance Testing
- Regression Testing

3. Scope

3.1 General

This section will describe which components of the system will be tested, basic and/or added functions that are specific to a component, interface of each component, how the component will interact once integrated with one another, and the expectations of the integrated system.

There are three components that make up the overall web-application system. These components consist of the database, web-pages, and the timeline application.

The database is wholly responsible for storing and retrieving information, and is used by both the web-page component and the timeline component. Once integrated with the web-page component, the database will be responsible for providing information about any upcoming events and workshops hosted by the museum, along with storing and authenticating information for requests involving login/registration. Furthermore, the database is responsible for storing roles that are associated with each user. Once integrated with the timeline component, the database will have the responsibility of providing stored information for each historical time-period and event for the timeline to display. In addition, the database will need to support the editing feature by having the ability to delete old entries and store new entries. All interactions involving the database will be done through either the web-page component or timeline application component. The user cannot directly access the database itself, it will remain independent.

The web-page component will be responsible for greeting the user and will act as the main hub of interaction for information. The web-page will not contain any information about the history of Niagara, but will instead contain information and content about the client(Niagara-On-The-Lake Museum). Content and features presented on the web-page component include information about services offered by the museum, information about the museum itself, up-coming events/workshops that are hosted by the museum, and a login/register feature. Once integrated with the database, the web-page component will retrieve login information, as well as store registration information from the database. The web-page will then get the role of the user, and display the correct actions associated with the role of the user. Once integrated with the timeline component, the web-page will be responsible for providing a user a path to the timeline application either through a link/button, or through the search feature.

The timeline application component is responsible for giving the user a rich walkthrough experience on the events and historical periods surrounding the Niagara region. Unlike the web-page component, this timeline component will be solely responsible for displaying any and all information regarding historical periods and any events surrounding those periods to the user. Once integrated with the database, the timeline component will retrieve the necessary information regarding history periods and events and display the information for the user to view. Any/all interactions involving the database will be done through the timeline

component, such as when an admin wants to make an edit request for a historical event. Once integrated with the web-page application, the timeline component will have very minimal interaction with the web-page, since the timeline component will not need to rely on the web-page for anything. The two components will mostly be responsible for providing a link to each other.

3.2 Tactics

This section will describe how the testing process of the scope will be carried out in regards to the exploratory, functional, and acceptance level of testing.

Exploratory level will be responsible for ensuring that unit level defects are removed before the next level of testing can start. This level of testing will be carried out in the web-application with the use of test scripts. This testing can be done by analyzing the software code and determining whether or not the unit will work as intended. Some examples include basic navigation and modules.

Functional level will be responsible for testing the function of each feature. This level of testing will be carried out on the web-application with the use of test-scripts. This testing can be done by feeding in an input for a given feature and then validating the output of the feature. Some examples include seeing whether the database has saved an entry, whether the user is able to log in using the correct credentials, and etc.

Acceptance level will be responsible for testing the overall functionality of the system. This level of testing will be carried out in the web-application without the use of test-scripts, as the goal is to test from a user point of view. This testing can be done by taking a user story and attempting to validate the user story.

4. Testing Strategy

This section will describe the strategies used during the testing phase. The goal of this section is to verify the functionality of https://badger-timeline.infinityfreeapp.com/src/ -site is working according to its specifications. Here, the testing is divided into distinct stages, with each stage having a clearly defined object and goal.

To ensure a complete and realistic test, all test data and environments will be produced with the sole purpose of emulating a production environment as much as possible. Furthermore, each test will be a repeatable and quantifiable process for the purpose of producing a reliable and consistent testing kit.

4.1 Unit Testing

The unit testing stage will cover individual components in isolation. This is the defect testing stage, where individual functions and methods are tested to ensure that they are working as intended before being integrated to create a component.

This stage is done in both a black box and white box method of testing. The black-box tests will be conducted on the site, where testers will merely check for the existence of a unit. The white-box tests will involve a thorough look into the source code, where testers will be checking the functionality of an object or method via code.

The following includes test cases outlining the process of unit testing

ID	Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass /Fail
case1	Check homepage buttons	Go to homepage Click on all clickable buttons	All available buttons on the homepage	User is able to click on the button.	As expected	Pass
case2	Check footer links	Go to homepage Go to footer Click on all clickable links	All available links on the footer	User is able to click on the button.	As expected	Pass
case3	Check header links	Go to homepage Go to footer Click on all clickable links	All available links on the header	User is able to click on the button.	As expected	Pass

case4	Check photo feature	Go to homepage Click on any photo from carousal/slideshow	Available photos from a carousal/slideshow on the homepage	A larger image is displayed as a hover and/or user gets redirected somewhere	As expected	Pass
case5	Check search bar	Go to homepage Click on search bar	Any String value	The search bar is editable.	As expected	Pass
case6	Mouse and keyboard functionality	Go to horizontal/vertical scroll Click Use arrow keys to the move direction	horizontal/vertical scroll signifier	Scroller should work as if you were using the mouse.	As expected	Pass
case7	Check main timeline points is clickable	Go to timeline Hover mouse over one of the points on the timeline	Any one of the points on the timeline	Point should be clickable	As expected	Pass
case8	Check that subtimeline is accessible	Go to timeline Hover mouse over space between two points	In between two points on the timeline	Space between two points should be clickable	As expected	Pass
case9	Check photocard feature	Go to timeline Select one of the points on the timeline	One of the points on the timeline	A empty photocard should pop up	As expected	Pass
case10	Check that filter feature is accessible	Go to timeline Click on the field next to filter	Any string value	Field should be editable	As expected	pass

case11	Check image view feature	Go to timeline Shuffle through images displayed on the overlay	Any photo of any type as test data	Slider should shuffle through images corresponding to an event.	As expected	Pass
case12	Check text-to-speech	Go to timeline Click on section with text-to-speech feature	A section that allows for text to speech	Text should be replayed back in speech form.	As expected	Pass
case13	Check speech-to-text	Go to timeline Click on section with speech-to-text feature	A section that allows for speech to text	Speech should be written in text form.	As expected	Pass
case14	Check calendar feature	Go to timeline Click on section with ‘open calendar feature’	A section that allows for a calendar pop-up overlay	Calender with the correct date should pop-up	As expected	Pass

4.2 Component Testing

The component testing phase will cover components which are combined to form a composite component. This phase will come after the unit testing phase, to ensure that no potential problems will arise if a unit is not working properly.

This stage can be done using a black-box and white-box testing, where black-box tests will be conducted on the site itself, and the white-box tests will involve a thorough look into the source code and validating the input and output of test cases.

The following includes test cases outlining the process of component testing

ID	Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass /Fail
case1	Check homepage buttons	Go to homepage Click on all clickable buttons	All available buttons on the homepage	User is successfully redirected to desired site/feature	As expected	Pass
case2	Check footer links	Go to homepage Go to footer Click on all clickable links	All available links on the footer	User is successfully redirected to desired site/feature	As expected	Pass
case3	Check header links	Go to homepage Go to footer Click on all clickable links	All available links on the header	User is successfully redirected to desired site/feature	As expected	Pass
case4	Check main timeline	Go to timeline Select one of the points on the timeline	One of the points on the timeline	A photocard with a picture should pop up	As expected	Pass
case5	Check photocard information reveal feature	Hover over photocard	Using the photocard from above	The photocard should ‘flip’ and text should appear	As expected	Pass

case6	Check subtimeline	Go to timeline Click on space between two points	Click on space between 1791, 1812	An overlay will appear	As expected	Pass
case7	Check text-to-speech	Hover over photocard Click on text to speech signifier	A photocard event	The text should then be heard over speaker	As expected	Pass
Case8	Check filter	Go to timeline Fill out filter option	Using values 1791, 1812	All other time periods should disappear, and the timeline should only display the specified time period	As expected	Pass
Case9	Check image view	Go to timeline Click on space between two points	Click on space between 1791, 1812	The sub-timeline should appear, and user will be greeted the option of shuffling through photocards	As expected	Pass
Case10	Check speech-to-text	Go to homepage Click on search bar Select speech-to-text signifier	“1812”	Search bar should display the string “1812” in its text field	As expected	Pass
Case11	Check calendar	Go to timeline Select one of the points on the timeline Click on calendar signifier	One of the points on the timeline	A photocard with a picture should pop up with a calendar signifier attached to it. A calendar will popup if clicked	As expected	Pass

4.3 System and Integration Testing

Integration testing, tests independent components and models of our system as a group. The testing is done after unit testing and before system testing. We will make use of integration testing for our Login page. When a user enters their username and password and clicks login this will direct them towards the home page for the niagara museum website. This is an example of integration testing because we test the overall login page functionality, not each individual component such as username and password. Therefore, combinations will need to be checked such as:

Invalid username invalid password (correct error message)
Invalid username valid password (correct error message)
Valid username invalid password (correct error message)
Valid username valid password (no error message and directs user to home page)

ID	Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
Case1	Test login	Go to login page Enter username Enter password Click ‘Sign in’	1) Invalid username and password (error message) 2) Invalid username and valid password (error message) 3) Valid username and invalid password (error message) 4) Matching username and password	No error message Directs user to home page	As expected	Pass
Case2	Test Sign Up	Go to sign up page Enter Full Name Enter Email Enter Username Enter Password Enter Re-type Password Click Sign Up	1) Using a fake mail service when entering your email (error message) 2) Password and re-type	No error message Directs user to home page	As expected	Pass

			password do not match (error message) 3) Enter a username that already exists (error message) 4) Password and re-type password match. Email is a real mail service.			

S*ystem testing,* tests the system as a whole. The Niagara museum website with all its functionalities/components interacting together.

Sign up functionality
Login functionality
Home page functionality
Timeline functionality
Sub-timeline functionality

Once all these functionalities are integrated together and working. You are now testing the integrated system with all of its functionalities as a whole.

ID	Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
Case1	Sign up functionality	Go to sign up page Enter Full Name Enter Email Enter Username Enter Password Enter Re-type Password Click Sign Up	Password and re-type password match. Email is a real mail service.	Directs user to home page	As Expected	Pass
Case2	Login functionality	Go to login page Enter username Enter password Click ‘Sign in’	Matching username and password	Directs user to home page	As Expected	Pass

Case3	Home page functionality	Go to home page Click ‘View details’ for Indigenous Click ‘View details’ for African Canadians Click ‘View details’ for European Colonialists Click ‘Visit Now’ Click ‘About Us’ Click ‘Historic Old Town’ Click ‘Regional Partners’ Click ‘Workshops’ Click ‘Book a Tour’ Click ‘Partner with Us’ Click ‘Tourism Niagara’ Click ‘Twitter icon’ Click ‘Instagram icon’ Click ‘Facebook icon’ Click ‘indeed icon’ Click ‘Contact Us’	No data	Directs to the correct corresponding page	As Expected	Pass
Case4	Timeline functionality	Go to Timeline page Scroll down feature to set exact year - 11,000years ago - 1500s - 1764 - 1790 - 1791 - 1812 - 1815 - 1831 Click white dot for a specific year (display info) Re-click white dot for same year (hide info) Changes to filter timeline viewing years Ranges from 0 - 2010 Calendar feature	No data	Timeline features work	As Expected	Pass
Case5	Sub-timeline functionality	Go to sub-timeline page	No data	Sub-timeline features work	As Expected	Pass

		Click in the middle of two years to open sub-timeline for that time period Hover over a card to flip it and see description Stop hovering card to flip back Click outside the sub-timeline to close it Calendar feature				

4.4 Performance and Stress Testing

Stress testing is a non-functional software test. The goal of stress testing is to test an application while it is stressed. A stressed application is when it is working at its maximum limit. Find the maximum limit of our application and test our program to get as close to or reach this limit.

1) database can hold up to 1000 records. Load all these records into our database and make sure no crashes occur.
2) Capacity of users on the website will need to be checked. Having 250 concurrent users on the website is max. Test to make sure 250 concurrent users will not crash the website.

ID	Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/ Fail
Case1	Test max records for database	Create 1000 records Load records into database	1000	No crashes	As Expected	Pass
Case2	Test max concurrent users on the website	Create 250 users All signed in at the same time All users on the website at the same time	250	No crashes	As Expected	Pass

Performance testing is a non-functional software test where the application will be tested on its stability, speed, scalability, and responsiveness during a given workload. Note that finding bugs is not the goal for performance testing. We are looking for long loading times, bottlenecking, slow response time, and poor scalability. For bottlenecking issues a user's CPU, memory, internet, OS, disk space can all play roles in why they are experiencing issues.

1) Checking max concurrent users before software crashes
2) Check response time for a certain number of users ensuring it does not cause the application to be slow
3) Checking the execution time for our database when 1000 records are read/written simultaneously
4) Check application with poor wifi connection
5) Check CPU and memory usage of application and database server during high concurrent usage of application.

ID	Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass /Fail
Case1	max concurrent users before software crashes	Create 250+ users All signed in at the same time All users on the website at the same time	250+	No bottlenecking No long loading times No bottlenecking No slow response time No poor scalability	As Expected	Pass
Case2	response time for a certain number of users	Create 250 users All signed in at the same time All users on the website at the same time	250	No bottlenecking No long loading times No bottlenecking No slow response time No poor scalability	As Expected	Pass
Case3	Checking the execution time for our database when 1000 records are read/written simultaneously	Create 1000 records Load records into database	1000	No bottlenecking No long loading times No bottlenecking No slow response time No poor scalability	As Expected	Pass

Case4	Check application with poor wifi connection	Use slow internet speeds	No data	No bottlenecking No long loading times No bottlenecking No slow response time No poor scalability	As Expected	Pass
Case5	Check CPU and memory usage of application and database server during high concurrent usage of application.	Create 250 users All signed in at the same time All users on the website at the same time	No data	No bottlenecking No long loading times No bottlenecking No slow response time No poor scalability	As Expected	Pass

4.5 User Acceptance Testing

User acceptance testing is the last phase of the software testing process. Conduct your tests on the software by its intended audience. Determine if all requirements of your contract for the application are met. This testing ensures your product is finished, effective in its goal, or does not have faults/bugs.

The test cases will be based on user stories created during the earlier stages.

5. Requirements

5.1 Hardware Requirements

Include the minimal specifications required to run applications without issues. This will include the processor speed, memory and disk space. Ensure your hardware exceeds these requirements for adequate performance.

5.2 Environmental Requirements

Different browsers such as Google Chrome, Microsoft Edge, Safari, and firefox. All of these browsers will be used and tested on different operating systems such as Windows, macOS, Linux, IOS, and Android.

6. Test Schedules

This section will describe the schedule in which each testing phase will take place. Below is a table detailing such schedule:

Date	Phase	Members involved
April 5 - April 8	Unit Testing	Development team, Testing team
April 9 - April 12	Component Testing	Development team, Testing team
April 12 - April 14	System Integration testing	Development team, Testing team
April 14 - April 18	User Acceptance Testing	Development team, Testing team

7. Control Procedures

Reporting problems

Document any incidents or problems encountered during the testing process.

Changes made

Document the modifications made to the software to fix any problems.

8. Features to be tested

The database will be tested for its ability to store and retrieve system generated information such as number of visitors, cookies, and etc. Furthermore, the database will also be tested for its ability to store and retrieve information generated by the admin/user such as any edits created by the admin, register requests, login requests, contacts requests, information on events and workshops, and etc.

The web-page component will be tested on its ability to direct users to intended page destinations, showcase up-coming events and workshops along with correctly showing any information associated, displaying relevant information from the search function, and correctly displaying any necessary actions permitted for each user. Furthermore, the web-page component will need to have the ability to perform on multiple devices and screen resolutions adequately.

Out of all the components listed above, this component will by far contain the most features, and so, the testing phase on this component will be very feature-driven. Here, the timeline-component will be tested on features such as the scrolling feature, content-display feature, sub-timeline feature, filter feature, search feature, photocard feature, image-scroll feature, calendar feature, device detection, and edit/add content feature for admins. In addition, the timeline will also be tested on its accessibility features such as text-to-speech, speak-to-text, a color-blind feature, and signifiers.

9. Risks and assumptions

This section underscores the risks and assumptions regarding the testing plan and schedules.

9.1 Risks

Some risks abouts the schedule include delays of the testing process, as some bugs may take longer to fix than others. To address this, the team has agreed on the strategy of continuous iteration, where if a testing phase gets done before its scheduled end date, then the team will move onto the next phase immediately on the same date. This will allow for more overhead near the end of the schedule.

Failure to identify complex functionalities and time required to develop those functionalities. Having other group members help with the functionality when it becomes too complex or time begins to run low. Speaking up and accepting help is a must to ensure the whole team can continue to move along without being held back.

9.2 Assumptions

This testing plan outline is created with assumptions in mind. Few of those assumptions include:

1. The web-application works consistently across all platforms.
2. Defects are reported through a chain of command.

3. There is no downtime for the environment during testing due to either updates or maintenance.
4. The database is treated as a black-box for most of the test-cases
5. The exploratory testing level will be carried out on the first day.
6. Production data is readily available at the start of each test.

10. Tools

This section will describe the various tools used in the process of testing. The numerous tools used include: JUnit, JMH (java microbenchmarking harness), MonkeyBox, and OpenTelemetry.

JUnit will be used in the process of automating test scripts. This tool will prove to be useful and valuable because of its ability to cover a numerous amount of test cases sequentially, which will allow the development and testing team to allocate precious time elsewhere.

JMH will be used in the performance testing of the system. This tool will allow the team to measure and quantify the speed and performance of the web-application, and can prove to be especially useful when in use during the phase of stress testing the system.

MonkeyBox will be used in the process of testing the web-application on a mobile device. This tool will help introduce random inputs for the system to deal with, and will help uncover hidden bugs in the system.

OpenTelemetry will be used both as a debugging tool, and also as a preventive maintenance tool as it keeps track of logs, traces, spans, and metrics. This tool will prove useful at the production level since it will log every activity across all users and allow the team to debug via statistical results.