Search'M – Search and Map

"Search and view news that is important to you and see how, where, and why that news is important to others around the globe"

Test Plan

Krista Bacungan
December 16, 2010

Table of Contents

Introduction

This document is the Test Plan for Search'M and verifies that the application works as required and validates that the correct functionality will be delivered. Search'M is a globalized news search engine, stressing the importance of understanding the trends in news around the world. Search'M provides users with an RSS feed aggregator, search engine, and different visualizations for search results. The goal of this application is to promote globalization and supply users with a way to search and view news that is important to them and see how, where, and why that news is important to others around the globe.

Testing Strategy

Objectives

The objective of this test plan is to state the bug tracking process and the testing features that need to be implemented in the source code in order to improve the integrity of this application.

Structure

Search'M has in-line source code tests during compile-time and run-time for each module of the application. This ensures that the each level of the program architecture is monitored. An elaborate logging system is in place to help with debugging run-time issues. With these test mechanisms in place, all of the functional specifications can be tested and confirms the usability of the application for demo purpose.

In-Line Source Code Tests

User Interface

Compile-Time Tests
- Check for appropriate libraries

Run-Time Tests
- Check that images and other resources are available
- Check that Google Earth is found on disk and available to be used
- Confirm that search results are being displayed

Application Controller

Compile-Time Tests
- Check for appropriate libraries

Run-Time Tests
- Check that preferences can be loaded
- Check status of application and database
- Confirm that communication between User Interface and the Backend are responding
- Check if system is connected to internet

News Indexer

Compile-Time Tests
- Check for appropriate libraries

Run-Time Tests
- Check that newly added RSS Feeds are valid
- Check that RSS Feeds that are being monitored are available
- Check that individual articles are accessible
- Catch and respond to article I/O errors
- Catch and respond to article parsing errors
    - Article metadata extraction errors
    - Finding the article content body div in HTML errors
    - Geotagging errors
    - Extracting images and video errors

Search Engine

Compile-Time Tests
- Check for appropriate libraries

Run-Time Tests
- Catch and respond to invalid search parameters
- Catch and respond to XML and KML streaming I/O errors
- Catch and fix invalid KML

Settings Manager

Compile-Time Tests
- Check that Preference file is found on disk

Run-Time Tests
- Catch and respond to invalid settings input
- Catch and respond to preference file I/O errors
- Catch and respond to any logging errors

Database Interface Module

Compile-Time Tests
- Check for MySQL JDBC connection and libraries

Run-Time Tests
- Check state of database and reset database if needed
- Check size of database and monitor to see if it is stable
- Check that database is accessible
- Catch and respond to database errors
- Confirm that changes in the database are being processed

Logging System

The logging system gets created in the Application Controller and logs out to disk in the Search'M folder. There is a separate log file for each run of the application and is named with the date and time to distinguish the files. The logging system being used is the Java java.util.logging.Logger package. Logging is done in every single source file of the application to ensure that nothing is being overlooked. The logs include both errors and success messages to provide a detailed run through of what has been done.

Logger Settings

- Severe: All errors and severe messages. Gets written to the log file and is displayed in the console.
- Fine: Information that is not an error but used for debugging (ie. RSS feeds being monitored, search requests, database connectivity, ect.)

- Finest: Information that is repeated several times (ie. detailed information about individual articles being added to the system, detailed search results, parsing details, ect.)

Recovery

Adding Code

As code is added to the baseline, the described tests and logging system supplies the knowledge to determine if the new code has negatively affected the original code. If there is a case that the code tree is broken and cannot be fixed, the Google Code repository that is established provides the ability to revert to working versions.

Program Errors

Compile-time errors are written to the log, written to the console, and the User will be alerted through Java Swing message windows. The following are a few examples of what is established to help the user solve any errors.

- Google Earth Not Found Error – User is provided with a pop up window stating that Google Earth is not found on disk, the link of where to download Google Earth, and a file chooser option for the user to give Search'M the file path to Google Earth on disk if it is already installed.
- Invalid RSS Feed Error – User is informed through the Search'M user interface by highlighting the RSS Feed input box and a message stating that the RSS Feed is invalid and to try again.
- Invalid Setting's Input Error – When the user clicks on the save button to change the settings, they are informed through the Search'M user interface by highlighting the invalid input box and a message stating that the input is invalid and to try again.
- Preference File Not Found – User receives a popup message stating that default preferences are being used and the user is pointed towards the setting tab to change the preferences if they prefer.

Bug Tracking

Search'M is using the bug tracking feature in Google Earth and can be found at http://code.google.com/p/searchnmap/issues/list

Bug Requirements

- Determine if bug has already been written up
- Indicate the steps to reproduce the bug – write enough details so that it can later be duplicated; exclude unnecessary steps
- Actual results – be specific on findings.
- Expected results – how the product should behave based on the specified or implied requirements.
- Choose the correct impact level (ie. Fatal, Serious, Minor, Task, ect.)