Kelli Bragg
LS590 – Java Programming Project Proposal
March 10, 2011

Name of Project: "Get That Blog Post"

Purpose of Project: Given a list of index terms, users can search a specified set of twenty blog postings from my account at WordPress (for the UA-SLIS Spring 2011 LS 590 Metadata class) to find the one or ones that include a specific index term. Creating an interface that allows a user to search a select set of files for a select set of terms, and then returning desired items, demonstrates my knowledge of using the Java language for proper data storage and retrieval.

Input/Data Protocols: Program input is achieved through selecting a term from a pre-defined list using either a command-line or a graphic user interface (GUI) interface, depending on the actual final interface. Once an index term has been searched and the record found, they will be given the choice – again through either a command-line interface or a GUI interface – of whether they would like to continue searching for other terms. Default values will be handled with proper exceptions.

As the author of the blogs, I assign tags based on my intention for the content; users will be told that results indicate my personal perspective of each post's "of-ness". The user will understand that the relationship of input-output is not weighted, since WordPress tags are not weighted, but merely indicate "of-ness" of a single blog posting. Results will be ordered by file number (blog1, blog 5, blog 20) since that's the order in which they'll be searched.

Blog Postings: To create a searchable database, twenty blog posting will be chosen, each one placed into its own individual text file. The tags that I assigned to each blog post will become the index terms, and will be preceded with"#" to match the index file terms (unless the '#' delimiter may turn out to have conflicts with normal usage). Because the file(s) retrieved need to be easily read, white space is included within the text; utilizing white-space delimiters in the program should ensure that scanning for matching text strings will not stop when "void space" is encountered. Below is a sample of how each blog post will appear once saved as a .txt file:

Image record vs. Work Record

Posted on March 8, 2011 by KellboBraggins

I'm not familiar with VRA Core 4, but looks like there are 2 records in VRA for the image I was viewing. While there aren't 2 records (work record vs. image record) for each image, I'm still trying to figure out the differences and what their purposes are (I just stumbled across them). Now I'm wondering if whether the Group Project should reflect a similar duality?

Wow, the more I delve into the "title" element, the more I see the complexity of element assignage.

Here's the image I was viewing while trying to get familiar with different forms of title to see all the parameters associated with "title" in different schemas. They have different refid #'s. (if you change the # in the url of the link below to 28, that one only has 1 form listed: the work record)

Nice, France: Topographic view, cityscape [lantern slide]

Posted in Group 2, LS566 | Tagged #Image_record, #Title_element, #VRA_core, #Work_record

? → perhaps also include date: #March?

Index Terms: Index terms will be placed into their own text file for easy addition/subtraction for accessing added/deleted sets of blog postings. To utilize the GUI, the GUI will display the components of the *Index Terms* file, and a hash tag (#) will be attached either to the beginning or end of every search term to ensure that it's only scanning my author-assigned tags and not every instance of that term within the entire text of a blog. For example, if I'm looking for the tag "image", I don't want a blog posting that may contain the word "image" that happens to fall outside of discussion of metadata (ex: I don't want it to bring up the blog post about Google Image maker because my post with that as a heading is actually tagged with the index terms: *labels, indexes, indices,* and *author-generated*).

Below is a sample of how the indexTerm.txt file might appear:

> The tags below are those that have been assigned to the postings I retrieved from my WordPress account for LS 590 Metadata (http://wp.slis.ua.edu/maccall-spring2011-ls566-01-7/):
>
> #Indexes, #Digital_image, #Assignage, #Cultural_contet, #Metadata, (#March ?)

Further exploration: One possibility that I would like to examine is conducting a feasibility study exploring the usage of multiple index terms in a single search to see if such scoping would enhance the user's experience. Also, it is unknown at this point whether or not the date will be included in the index file. If user would want to look for all blogs posted in March that contain a certain index term, I would need to include the month within the index.txt file, and, like the index terms themselves, would require inclusion of '#' at the front to allow for string comparison (again, unless the '#' turns out to have conflicts).

Output Protocols: Data will be in a textual format using the keyboard & monitor. There will be a dropdown list of index terms and a "yes/no" button to let the user decide if they want to search. To search, the user will choose an index term from the dropdown list. Since each blog is a different length and you thus cannot use a program saying "look at line X for this term", the term will be put into a string-search method that scans every line of text in each unique blog.txt file until it reaches the lines containing the index terms (located at the end of the file). There will be a '#' automatically added onto the searching term because the index terms are set apart from the rest of the text by utilizing '#' at the beginning of each index term.

Items can be printed if desired using default "File Print" setting common to all machines. The index term chosen will be repeated at the top of each results/set of results to ensure user is looking at what they wanted to see (in case they accidentally chose the wrong index term from the list). Once blog files have been found and read /printed out user will be asked if they would like to perform another search (depends on interface: command-line vs. GUI: see flow chart).

Testing Protocols: Summative testing: users will test-drive the finished program according to a script, then analysis of feedback and errors (were all relevant files found? any 'wrong' files found? no files found? does '#' work?) will help guide necessary code development changes.

Images: Below are the images for: (1) The tentative flow-chart for the "Get That Blog Post" project; (2) the sample webpage site: how the program, including a graphic user interface, might appear.

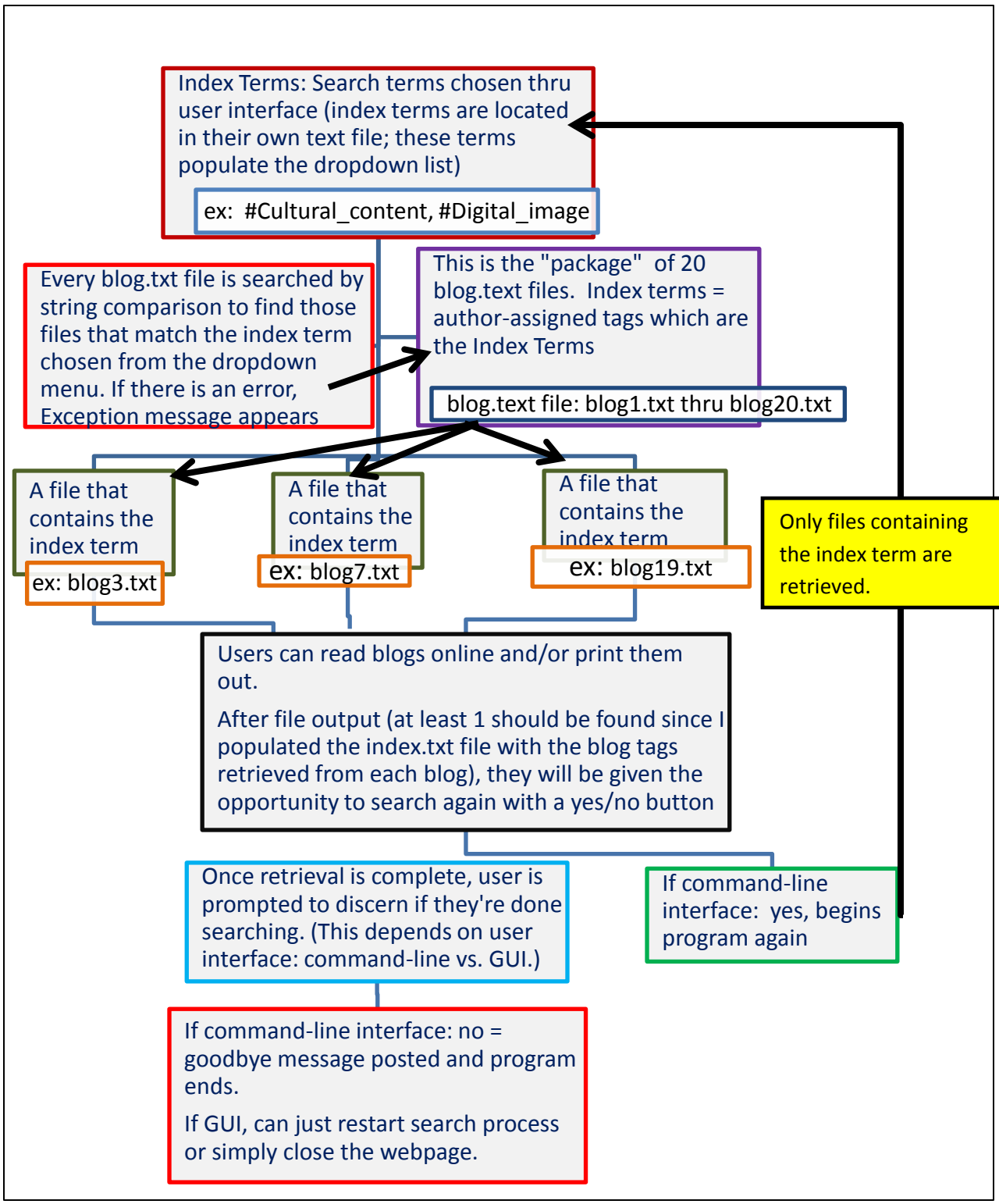

Figure 1: Flow chart for "Get that Blog Post" Project

Figure 2: Sample webpage as it might look for hosting the java package for the "Get that Blog Post" Project. Webpage describes: Purpose; How to Use; shows sample buttons that could act as a Graphic User Interface.