Krista Bacungan

Search'M – Functional Specifications and Use Cases

Search'M is an application that allows users to view the news that is important to them, the way they want to view it, and as see how the news has affected the rest of the world. Users specify RSS feed URLs from various news providers that they want to monitor. The articles from each feed are automatically processed for keywords, locations, categories, images, related articles' URLs, and a description, and then are stored in the system's database. The user may search specifically or generally by keyword, location, date, category, or a combination of those options. The results will then be displayed in a search result list in the user interface and if the user wishes, he/she can view the results in Google Earth by clicking on a specified icon. For each result with geographic data, a placemark is displayed with its corresponding information. When a placemark is hovered over, lines that represent the related article path are displayed to show the relationship between that item and other items on the globe (similar keywords, categories, dates, ect.).

All articles can be viewed fully in a browser by clicking on the article URL. Users can modify what RSS feeds are being monitored and have to the option to clear the system. This system can be used whether or not connected to the Internet. If offline, new articles cannot be processed but all other features remain. Once the system is connected to the internet again Search'M will start it's RSS feed ingest again.

Core Requirements

- Parse RSS feeds
  - Retrieve RSS stream from RSS feed URLs
  - Parse through HTML of each article, process text, and store discovered keywords, locations, dates, and categories to be associated with article
- Search
  - By keywords, dates, categories, and locations, or a combination of filters
- Display results
  - In a result list (designed like commonly used search engine results)
  - In Google Earth using KML
    - Each result is a placemark pin
    - Connecting lines between pins show relationships between the articles
- Manage Search'M

    - Add, delete, or modify RSS feed URLs
    - Clear system from all data

List of Functions:

- Process articles
    - Determine what sections of the HTML is the article content (TagSoup API)
    - Process article text for keywords, locations, dates, categories
    - Pick out embedded images and create thumbnails
- Retrieve articles from RSS Feed
    - Stream in articles one at a time and read HTML (ROME API)
    - Automatically pull in articles as they are updated to the RSS feed
- Create results
    - Determine relationships between articles. This is not the same as related articles which are URLs found in the article. Instead this subfunction must connect articles together based on common dates, locations, keywords, or categories
    - Create KML to show results in Google Earth (JDOM, API)
- Store and retrieve information from database
    - Create and design database (Java Database Connectivity, JDBC)
    - Store the article content, description, thumbnail of embedded images, keywords, categories, URLs of related articles, dates, locations, and URL of the article
    - Create efficient queries to get results for searches

Use Cases:

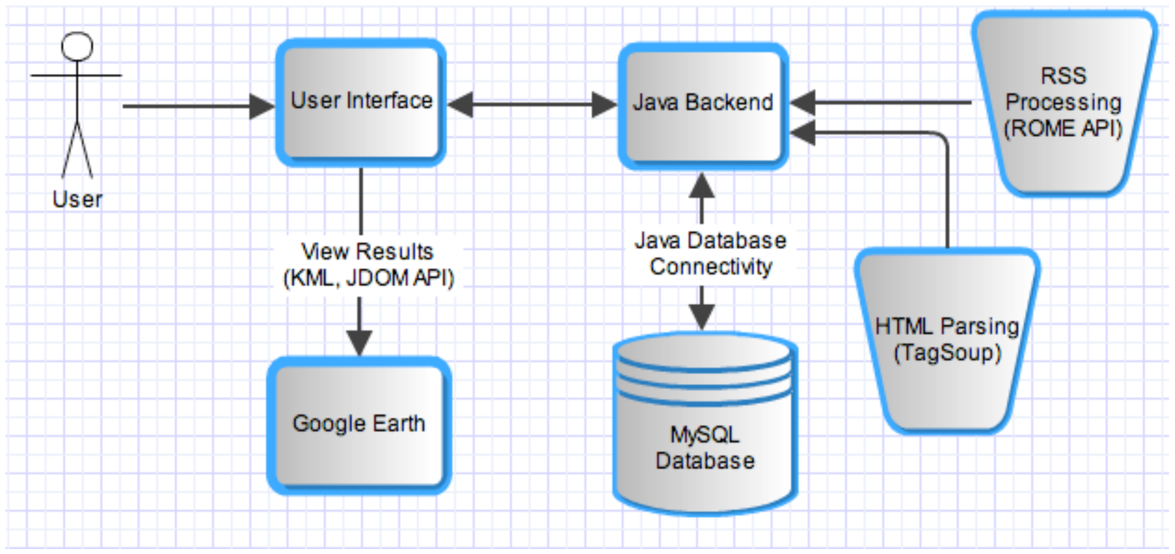
- General Users
    - Select RSS feeds to monitor
    - Search the system, filtering by keyword, category, date, location, or a combination of filters
    - View results in result list
        - View results in Google Earth (Click on icon)
            - View related article paths
    - View full articles (Click on URL)
    - Clear RSS feeds and database
    - Add, delete, or update RSS feed URLs

- Use system on or off the internet

Data Structure:

This application will be developed in Java and will interface with the ROME API for reading in RSS feeds, TagSoup for parsing HTML, and JDOM for creating the KML for Google Earth. Data will be stored in a MySQL database and use JDBC to communicate between the database and the Java application. There is a location hierarchy tree that will be used for determine the primary location of an article and to relate that article to fellow articles. The front end will be done in Java as a stand-alone desktop application and will open the primary internet browser when viewing article's original file. When the user wants to view the results in Google Earth, the KML will be opened in a new Google Earth browser or in the current browser if one exists.

Diagram:


User
User Interface
Java Backend
RSS Processing (ROME API)
View Results (KML, JDOM API)
Java Database Connectivity
HTML Parsing (TagSoup)
Google Earth
MySQL Database

Algorithms:

Keyword generation is very complex and I am still in the process of finding an appropriate algorithm. Word frequency combined with a don't-include-list of non-keyword words (such as conjunctions and common nouns) could be a possible solution. As described above, the location hierarchy tree will be used to find locations in the text. Determining the primary location of an article is a similar problem to keyword generation and will most likely use the location that is most frequent. Lastly, determining which articles are related to each can be solved by correctly designing the database to allow similarities between articles to be determined at the time of the search.