**Design Documents for News Stream**

**Group May13-31**

**Group Members: Jamison Bradley, Lance Staley, Charles Litfin**

**Advisor: Srikanta Tirthupura**

**Client: IBM Rochester**

**Client Contact: Paul Bye**

**Hardware Specifications**

Our project will have no hardware based component to it, so there will not be hardware specifications for it.

**Interface Specifications**

The User Interface should be clean, clear, concise, and easy to use. On the left side of the browser will be a table called categories which will list the different category of news topics. When a category is selected articles from that category will be display in the news feed (which will be discussed later). The news feed will be in the center of the page and will display the four most recent articles based on the time posted. There will be a next and previous buttons in the news feed to allow the user to see the next four oldest or the next four newest articles. In the upper right corner will be a search bar which will allow the user to search by keywords and then display the relevant articles in the news feed. There will be a login link in the upper left corner which will redirect you to the login page.

There will be a second page which will be a simple login page. It will ask for your email address and password, and then have both a submit and a back button. Both of this option will redirect the user to the main homepage after clicking.

**System Analysis**

Key Word Analyzer:

We currently plan to scan the articles using java and the Infosphere streams to process the input for this function using the build in capabilities of streams and modify them with our own java code as needed.. The Key Word Analyzer will receive its input from the Data Parsing sub-function.

Data Parsing:

The streams application will pull the text from the articles and allow the parsing of the text from the articles.

Database:

The database will take their input from the processed information (Received from the Key Word Analyzer sub-function) from our streams application. It will then store this data in the Database and will sent queries back to the user using JDBC.

Story Matching:

The Story Matching sub-function will receive its input from the Database. It will then send the information to the user interface to be displayed to the user.

**Use Cases**

* + 1. Article Selection  
       Actor: User

Main Success Story:

a. The User clicks on one of the four article links.  
 b. The article is opened in a new tab and displayed.

Alternate Success Story:

a. The User clicks on the next button in the news feed.

b. The news feeds displays four more articles which are more recent.

c. The user clicks on one of the four article links

d. The article opens in a new tab and the article is displayed.

Alternate Success Story:

a. The User clicks on the previous button in the news feed.

b. The news feeds displays four more articles which are more older.

c. The user clicks on one of the four article links

d. The article opens in a new tab and the article is displayed.

Exception #1 – The article does not display in the new tab.

a. The tab displays a 404 error (page not found error).

2. Category Selection

Actor: User

Main Success Story:

a. The User selects a category link from the list of categories.

b. Articles from the selected category are display in the news feed by most recent articles first.

Exception #1 – No articles fit in the selected category.

a. The news feed displays no articles.

b. A pop-up tells the user that no articles match their search.

3. Search for Articles

Actor: User

Main Success Story:

a. The user enters a keyword or words in the search bar.

b. Articles which are relevant to the keyword are displayed in the news feed.

Exception #1 – No articles are relevant to the keywords or words.

a. The news feed display no articles to the user.

b. A pop-up tells the user that no articles match their search.

4. Login

Actor: User

Main Success Story:

a. The User click on the login link from the main page.

b. The User is redirected to the login page.

c. The User enters their email and password in the corresponding fields.

d. The User click submits.

e. The User is login successfully and redirected back to the home page.

Alternate Success Story:

a. The User click on the login link from the main page.

b. The User is redirected to the login page.

c. The User clicks the back button and is not login in and is redirected to the home page.

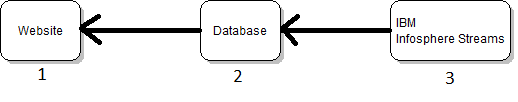
Exception #1- User email or password is incorrect.

a. The User enters their email and password into the corresponding fields.

b. The User password or email is incorrect and login fails.

c. The User is prompted with a pop-up telling them their password or email is incorrect and to try login again.

**Module Design/System Architecture**



#1 Website

This will be our front end that the user will interact with. It will get the stories that are to be displayed and the information about the stories such as what category the users are in from the database.

#2 Database

The database will store all of the information about the articles that we are displaying on the website. It will be populated with the information from Infosphere Streams.

Tables in Database

Article Table

Article ID - The id that is associated with the article, a unique id will be given to every article. This will be the primary key of the Article Table.

Story ID - The id that is associated with the story the article is talking about, if several articles are about the same story they will have the same id. This will be the primary key in the other table.

URL- The url of the article.

Source - The name of the news source that the article is from.

Story Table

Story ID - See description in Article Table.

Category - The category that the story falls under.

Key Words - The key words that are associated with the story.

Date - The date that the particular story occurred on.

#3 Infosphere

This is the IBM software that we are going to use in our project. It will read in the stories from the websites that we are going to support, and it will allow us to provide the system with the information needed to process the articles. After the article is processed it will store relevant information in the database for the website to use.

Sub Modules

A. Data Parsing - The data parsing sub module will contain the code that will be used by Infosphere Streams to parse the websites that it reads in. Primarily the code will revolve around how to get the text of the story off of the web page, so that it can be analyzed for key words.

B. Key Word Analyzer - This sub module will be in charge of providing key word analysis of news articles.

C. Story Matching - This sub module will determine if the article that is currently being processes is the same story as any of the other articles that have already been processed. The key word analysis provided by the Key Word Analyzer will play a major part in finding matches.

D. Database Communication - The Database Communication sub module will be used by the other modules to both write and read from the database.

**Input Output**

**System**

Input: A link to a website that comes from one of the websites that we are supporting.

Output: The article will be categorized and grouped with articles from other sources about the same story on our website.

**Infosphere Streams Module**

Input: Link to website.

Output: Relevant information about the article and it will have been grouped with other stories in the database if they discuss the same story.

**Infosphere Streams - Data Parsing**

Input: The source code of the website being analyzed.

Output: The text that makes up the article and other relevant information needed in the analysis process.

**Infosphere Streams - Key Word Analyzer**

Input: The text of a news article.

Output: A list of key words associated with that article.

**Infosphere Streams - Story Matching**

Input: Information, such as key words and date of the article, that would be useful in analyzing the article.

Output: The story from that database that the current article being analyzed matches with, or nothing if it is a new story not present in the database.

**Infosphere Streams - Database**

Input: Data that needs to be stored in the database, or commands to retrieve information from the database.

Output: Confirmation that the data was successfully stored in the database if storing, and the information requested if a retrieval of information was requested.

**Database Module**

Input: Data is sent to be stored.

Output: Data is requested and returned.

**Website Module**

Input: Articles from the database are retrieved by the website.

Output: A user interface for the user to interact with and explore the articles we are presenting.

**Functional Decomposition and Test Plan**

**Overall Function:** The overall function of our project is to take in news articles from several different news sites and then aggregate that data together on one website.

**Infosphere Streams - Key Word Analyzer**

Removal of Common Words: This function will remove the 1000 most common words in the English language from a string of text. The words being removed should never be considered a key word since they are so commonly used in English. The function will return a list of words in the article that aren't common.

Test Plan Procedure: This function will be tested by unit tests. A couple of articles will be tested and if any of the words the function returns are on the list of common words it will have failed the test.

Test Plan Interpretation: If the unit test is successful we will be able to feel confident that common words are being removed correctly.

Key Word Detection: This function will take a list of non-common words and determine which ones in the list are likely key words.

Test Plan Procedure: We will unit test this function by inputting list of words that we have designed, and already know what key words should be detected.

Test Plan Interpretation: If the function is picking out the correct words that we believe our algorithm should pick out we will consider the test to be successful.

**Infosphere Streams - Data Parsing**

Parser: This function will parse the source code of web sites that our project is pulling articles from to get the desired information like the text of the article. Each website will have a slightly different format so this function will rely on plug-ins to know how to parse the different website's source code.

Test Plan Procedure: We will make test case for each site we are going to support, were we pick out the information it should gather by hand and use unit tests to make sure it picks out the same data.

Test Plan Interpretation: If the function is correctly picking out the data for our test cases we will consider the test a success.

**Infosphere Streams - Database**

Retrieve Information: This function will retrieve information from the database to be used by other functions in the Infosphere Streams module.

Test Plan Procedure: We will write a series of unit tests to retrieve information from the database, and test to make sure the correct things were retrieved.

Test Plan Interpretation: If the unit tests all pass we will feel confident that the function is working correctly.

Store Information: This function will store information in the database.

Test Plan Procedure: We will combine the testing of this function with the Retrieve Information function. We will create some unit tests were we store some information then immediately retrieve it.

Test Plan Interpretation: If the unit tests that we include this function in are all passing we will consider the test to be a success.

**Infosphere Streams - Story Matching**

Match Story: This function will look at the stories that are already stored in the database and see if any of them are the same as the story that is currently being analyzed by the Infosphere Streams module. If the article is similar it will return information about the story it is similar too, if it isn't similar it will return that the article is a new story.

Test Plan Procedure: This will be more of an art than a science, since our algorithm will likely not be capable of working for every single story. So we will look at how stories are getting grouped and continue to tweak the algorithm to make it the best we can.

Test Plan Interpretation: If the majority of the articles are getting grouped correctly, and we feel we can't improve the algorithm any more we will consider the function to be complete.

*Sources*

#1 CNN (US)

#2 New York Times (US)

#3 Des Moines Register (Iowa)

#4 The Guardian (UK)

#5 BBC (UK)

#6 Toronto Sun (Canada)

#7 The Australian (Aussie)

#8 Al Jazeera English (Mid East)

#9 New Zealand Herald (Kiwi)

#10 The Local (Sweden)

**Project Measures**

Our goal is to create a program to categorize and aggregate news articles using IBM Infosphere streams software. The table below shows our metrics.

|  |  |  |
| --- | --- | --- |
| Goal | Question(s) | Metric(s) |
| Producing our software in a reasonable amount of time. | How long will it take to produce? | Until three weeks before the end of CprE 492 |
|  | What is a reasonable timetable for this type of project? | Weeks the developers think the project will take to develop |
| Producing a product with a long sustainability. | How long is a long sustainability? | As long as the product is still useful and/or profitable to IBM |
|  | The cost to sustain the product? | The cost of any additional programming needed to fix discovered bugs |
| Have reasonable accuracy in article aggregation and category grouping | What is a reasonable aggregation accuracy | 80% of all articles or more should be correctly deleted or retained |
|  | What is a reasonable category grouping accuracy? | It will also be 80% |
| Have a reasonable response time between article creation and addition to the database | What is a reasonable response time? | One hour |

**Functional Requirements**

**Infosphere Backend**

**Obtaining Articles**

The backend should be able to pull articles off of the Internet and add them to the database.

**Scanning**

The backend should be continuously scanning the preselected sources for new articles to be added to the database.

**Sorting**

The backend should be able to accurately sort articles into a number of preselected categories based on keywords and content found within them.

**Database**

**Article Information**

Each article should be stored with the following information: category it belongs to, title, and source of original publication.

**Categories**

Each category has multiple articles assigned to it, but each article can belong to no more than one category.

**Aggregation**

The database should be able to correctly identify which articles are redundant and which are unique.

**User Interface**

**User Access**

They should be able to use a pre-existing Facebook profile to login if they so desire.

**Article Access**

The user will be able to access articles through links listed under a number of fixed categories.

**Customization**

The user should be able to some degree be able to change the categories into which their news is sorted.

**Design Constraints**

**Backend**

The backend will be using Infosphere Streams which utilizes Java.

**Database**

The database will use MySQL which is open source.

**Operating System**

Any system capable of running a web browser.

**Web-based**

The system will be a browser-based web application.

**Non-Functional Requirements**

**Security**

**Username**

A user account is associated with a single username and password

**Modification**

The user is able to modify which websites they are getting their feed stories from.

**Performance Requirements**

**Speed**

The system should be able to obtain a news article quickly after it is posted to an rss feed.

**Database**

The system should be able to store links for several days before it begins deleting older links to make way for newer news.

**Interface**

The user-interface should respond quickly to user input in order to facilitate faster use overall.