Project Plan for News Stream

Group May13-31
Group Members: Jamison Bradley, Lance Staley, Charles Litfin
Advisor: Srikanta Tirthupura
Client: IBM Rochester
Client Contact: Paul Bye

Problem Statement

In today's world there is a plethora of information that is available on the internet for users to consume. The problem is that this information is spread out over several different websites rather than being streamlined and presented to the user conveniently in one location on the web. This diffusion of data over many different sites forces users to access several different sites a day to pick out the news stories the user wants to read, which takes up valuable time that could have been spent doing other things, such as reading the stories rather than searching for the articles.

Another issue that arises from using multiple news sources is whenever there is a major news story, the story will be covered by almost all of the different websites the user uses, and the websites will all provide the user with more or less the same information about that event. This is not an ideal situation for most users, it would be better if the stories could be grouped together and presented to the user so that the user knows all the articles discuss the same story and can choose to read just one from the source that the user desires to read from.

Scope

The scope of this project is to deliver a web-based application which will allow the user to access different news feeds from a range of news providers. The webpage should also all the user to view articles based on categories and to remove duplicate articles.

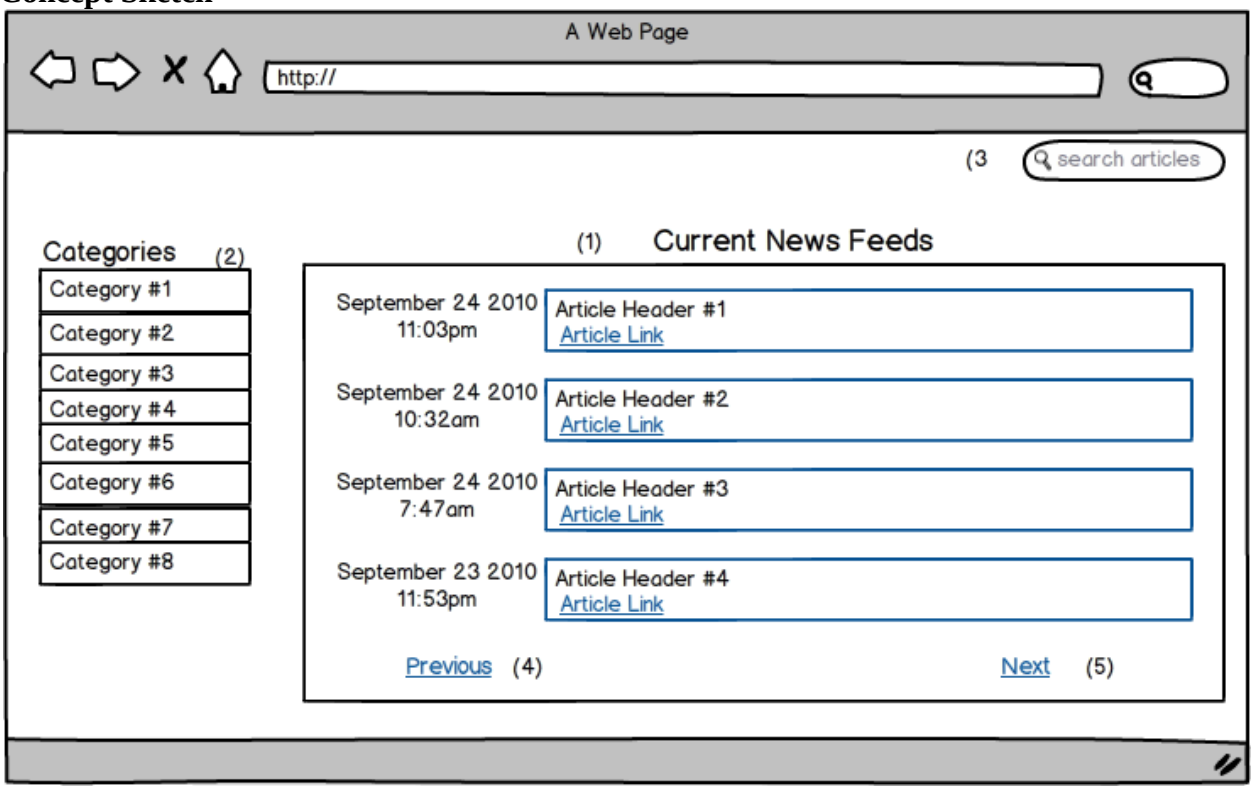
Goals

During the first semester of the project our group would like to have all documentation and design done, as well as, have an understanding of both the IBM Infosphere Streams software and have gained experience with web design. We would then hope to start implementation over the Christmas break.

At the end of the second semester we want to have a working web application which displays new feeds, provides categorization of articles and removes duplicates articles.

Lastly we would want to have enough time to add additional functionality to our web application, such as customization of sources, time permitting.

Concept Sketch



System Description

Our system will gather streams of articles from several different sources on the internet to be displayed for the user of our site. The user can either accept the default sources that we choose to support, or the user can login to our website and customize the sources that are used, by unselecting sources, we support that the user does not want to receive articles from. Our team is planning on using Facebook logins as the means of login for our site. This will make it easier to maintain the security of our websites by not having to handle sensitive user data like passwords and email addresses.

Our team have chosen the sites that are planned on being supported (Can be viewed below), baring technical difficulty in working with these sources. Our team will pull data from these sites that already categorize their data. This will allow our team to know, based on where the story comes from on a site, what category an article belongs to.

After an article is gather the article will be analyzed and compared to the stories in the database. If our team determines that the article is the same as another article from a different source, only one article will be displayed together on the website. This will save the user of the hassle of seeing the same story over and over again from different sources.

Below are the sources that our team are planning on supporting on our website initially. We decided to go with a wide range of sources and hit several different countries, so that there is

a wide range of articles and opinions for our users. Our team did decide to add one local website for Iowans by including the Des Moines Register.

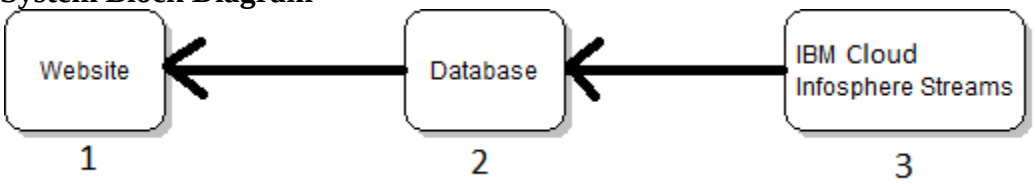
Planned Sources
#1 CNN (US)
#2 New York Times (US)
#3 Des Moines Register (Iowa)
#4 The Guardian (UK)
#5 BBC (UK)
#6 Toronto Sun (Canada)
#7 The Australian (Australia)
#8 Al Jazeera English (Qatar)
#9 New Zealand Herald (New Zealand)
#10 The Local (Sweden)

Interface Description

The user interface will be a website. There will be a main news box region (1) which shows the most recent articles, their titles and links to the articles. It will also show the time that the article was posted. There is a next (5) and previous (4) button which will show more articles or previously seen articles by chronological order.

There will be a menu on the left side (2) which contains categories. If a category is selected only articles from that category will be display in the main box(1). There is a search bar(3) in the upper right corner which will allow the user to search by article title and display relevant results in the main box(1).

System Block Diagram



1) The website will use the database to populate the user interfaces with news stories for the user to read. The website will also store data about user preferences, so that we, the developer can tailor the experience to individual users.

2) The developers will have a database that will store the information about the article that will be presented on our website. This database will be populated by the backend and will be used by the web server to populate the website with stories.

3) The backend will be used to gather the stories from news websites. This will be done by using Infosphere Streams on an IBM cloud server. Our team will gather the data from the news websites and carryout the analysis of that information on this server and then populate the database with that information.

Operating Environment

The operating system will be on any system that has an internet access and can use a web browser. Therefore our software will work on Windows, Linux, iOS, and mobile devices. Testing of our software will be done mostly on a Windows machine, the interface, which is the website, will be tested primarily on Firefox and Google Chrome web browsers.

Functional Requirements

A). Infosphere Backend

1. Obtaining Articles

The backend should be able to pull articles off of the internet and add them to the database. The backend will obtain whole article contents plus links back to the original source.

2. Scanning

The backend should be continuously scanning the preselected sources for new articles to be added to the database. It should have a turnaround time of less than a hour.

3. Sorting

The backend should be able to sort articles into a number of categories based on the category the articles are designated into on their source website. These categories will include such topics as sports and politics.

B). Database

1.) Article Information

Each article should be stored with the information that is specified by the Article Table in the database.

2.) Categories

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

3.) Aggregation

The database should be able to correctly identify which articles are duplicates and which are unique. The duplicate ones will be marked as such and grouped with other articles containing identical content.

C.) User Interface

1. User Access

The user should be able to use a pre-existing Facebook profile to login. This will help lower the amount of content that must be saved per account and avoid opening new security risks that a new login system would create.

2. Article Access

The user will be able to access articles through links listed under a number of fixed categories. These articles will be presented as the article title in text along with a link back to the original web source of the article.

3. Customization

The user will be able to remove sources that our supported that the user does not want to see articles from. Also the user should be allowed to add any previously removed sources. If additional time is available at the end of development there will be plans to allow the user to add their own sources.

Non-Functional Requirements

1). Security

A).Username

A user account is associated with a single username and password, as well any modifications the user has made to their sources.

B). Modification

The user is able to modify which websites the articles are being displayed from.

2). Performance Requirements

A). Speed

The system should be able to obtain a news article quickly after the article is posted to an RSS feed. The current requirement will be adding an article within one hour of the article being added to an RSS feed.

B.) Interface

The user interface should respond quickly to user input in order to facilitate faster use overall. The user interface should also be user friendly in order to make news consumption easier.

Market and Literature survey

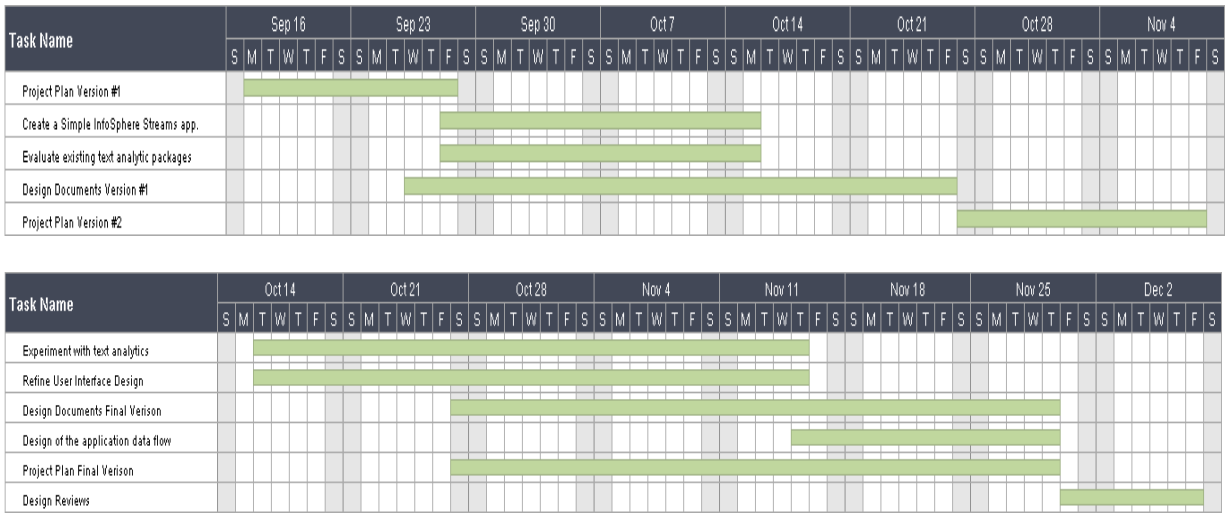
There are a number of similar news aggregators currently on the market. Our research leads our team to believe our system will be the only one currently using IBM's Infosphere streams as part of its system. Also the majority of other news aggregators do not allow the user to customize the sources from which the articles are gathered from. This provides our system with a major difference to other systems, which should give us a distinctive edge over current systems. Also our user interface should be clear, simpler, and more concise then those currently available.

Deliverable

Our final deliverable will be a website that will aggregate news stories from several different sources. Supporting this website will be a database that contains the information needed to provide the articles to the user. Our team will also implement a backend using IBM

InfoSphere Streams and Java that will be used to populate the database with content for the website. This content will be categorize and checked for duplicate articles.

Work Plan



Risks and Mitigations

Risk	*Probability of Occurrence*	*Criticality (0-100)*	*Risk Factor (Prob. Of Occurrence x Criticality)*	*Mitigation strategy*
The IBM Streams software has not been used by any of our members and could be difficult to learn and implement	.50	80	40	Hold a meeting with our IBM contact about streams as well as prototyping small streams applications for experience

Lack of experience with web development	.40	60	24	Research web development as well as prototype some web development
Development of the article comparison algorithm is more difficult than expected	.30	75	22.5	Research to see if there are any third party algorithms that could be used. Build a prototype of the algorithm to test early on. Add extra time to the schedule.
Usability in many different web browsers and operating systems	.40	50	20	Research and develop for a number of browsers
We are unable to get streams to work with our java back and web frontend.	.20	50	10	Research alternate design options as well as using streams with external code