Design Documents for News Streams

Group May13-31

Group Members: Jamison Bradley, Lance Staley, Charles Litfin

Advisor: Srikanta Tirthupura

Client: IBM Rochester

Client Contact: Paul Bye

Executive Summary of the Project

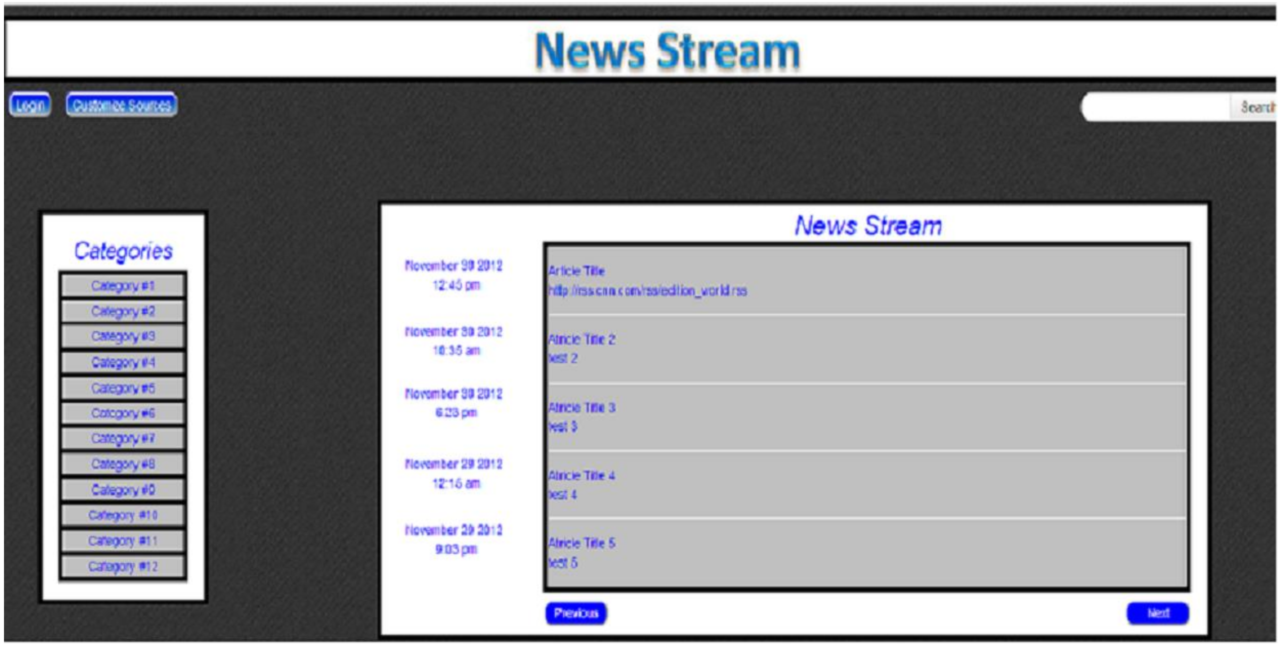
The goal of this project is to deliver a web-based application, which will allow the user to access different news feeds from a range of news providers. There will be a java backend, which will gather the articles from various sources, categorize these articles, and detect and aggregate duplicate articles. These articles will be stored in a database with the user's information and will communicate to the webpage the necessary information to be displayed.

Hardware Specifications

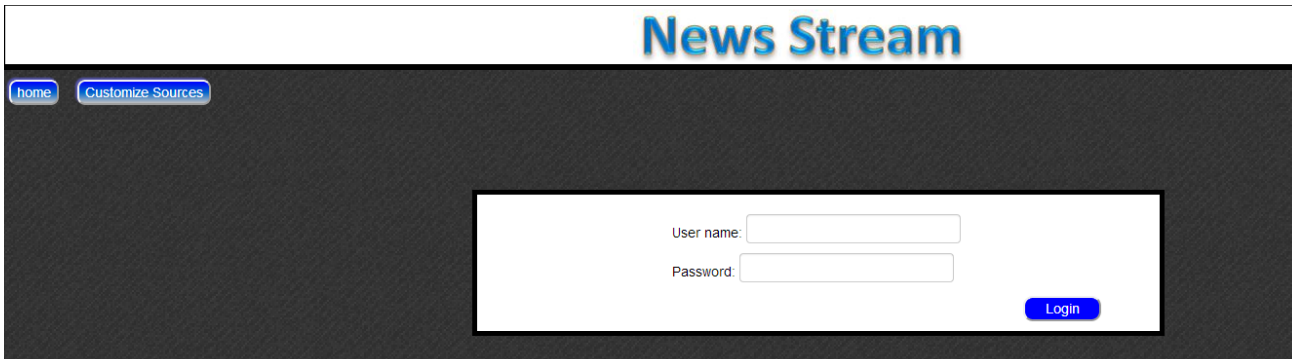
Our project will have no hardware based component to it, so there will not be hardware specifications for it.

Interface Specifications

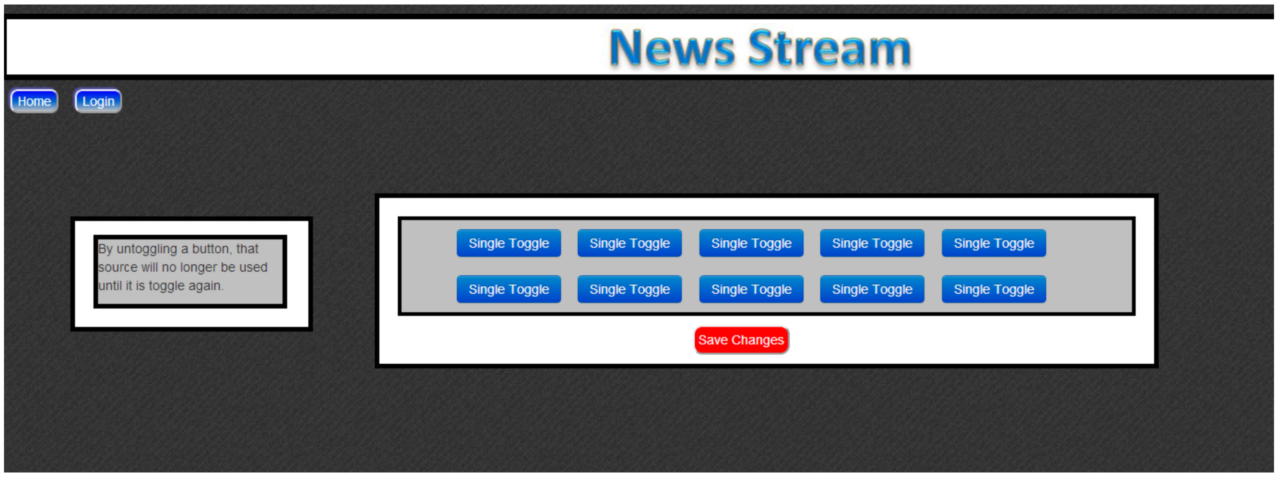
The User Interface should be clean, clear, concise, and easy to use. On the left side of the browser will be a table called categories which will list the different category of news topics. When a category is selected articles from that category will be display in the news feed (which will be discussed 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 button and a back button.



There will a customize sources page which has buttons to redirect to the other pages in the upper left. Also the is a main box in the middle which contains toggle buttons with the name of each source on them. When toggled a source is used, when it is not toggled is will not be used. There is also a save button used to save changes to which buttons have been toggled.



By untoggling a button, that source will no longer be used until it is toggle again.

System Analysis

<u>Key Word Analyzer:</u>

Our current plan to scan the articles using text analytics written in Java code.The Key Word Analyzer will receive its input from the Data Parsing sub-function.

<u>Data Parsing:</u>

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

<u>Database:</u>

The database will take their input from the processed information from our streams application. It will then store this data in the Database and will sent queries back to the user using JDBC.

<u>Article Aggregation:</u>

The Article Aggregation 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 message on the page will say that no results were found.

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 message on the page will say that no articles were found.

4. Login

Actor: User

Main Success Story:

a. The user clicks 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 clicks submit.

Alternate Success Story:

a. The user clicks on the login link from the main page.

b. The user is redirected to the login page.

c. The user clicks the back button in the web browser and is not logged in and is redirected to the home page.

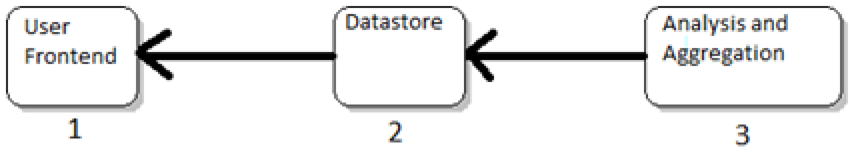
Exception #1- User email or password is incorrect.

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 that their password or email is incorrect and to try 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 category and date that will help correctly display the stories the user desires. The front end will retrieve the stories from the database.

Frontend Modules

A. News Display - This will be the module that displays the news stories for the user to view on the website.

B. Sources - This module will provide the user with options about what sources are used on the website. The user will be able to select and deselect default sources and add new ones that aren't part of the default.

C. Login - In order to customize content on the website the user will have to create an account and log in. This module will handle the account creation and login process.

D. Database Connection - This module is used to connect to and retrieve information from the database. This module consists of java servlet pages(.jsp) which connects to the database using java and jdbc and make query to the database based on which jsp page is called. The other frontend Modules with then make AJAX get requesting using javascript to this jsp pages to retrieve the results of the query.

#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 the java backend.

Tables in Database:
Article Table

Field	Type	Description
Article ID	Int	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.
StoryID	Int	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 Story table.
URL	String	The url of the article.
Source	String	The name of the news source that the article is from.
Date	Long	The date of publication for the article.
Title	String	The title that was given to the article by the news source.
Article Keywords	String[]	The key words that were found in the article.
Category	String	The category that the news source placed this article in on its website.
Quotations	String[]	Quotations that were found in the article.

Story Table

Field	Type	Description
Story ID	Int	See description in Article Table.
Category	String	The category that the story falls under on our website.
Date	Long	The date that the particular story occurred on, it will be an average of all the articles that match to this story, and will be the date we use on our website.
Title	String	Will be the title of the story that we display on our website.

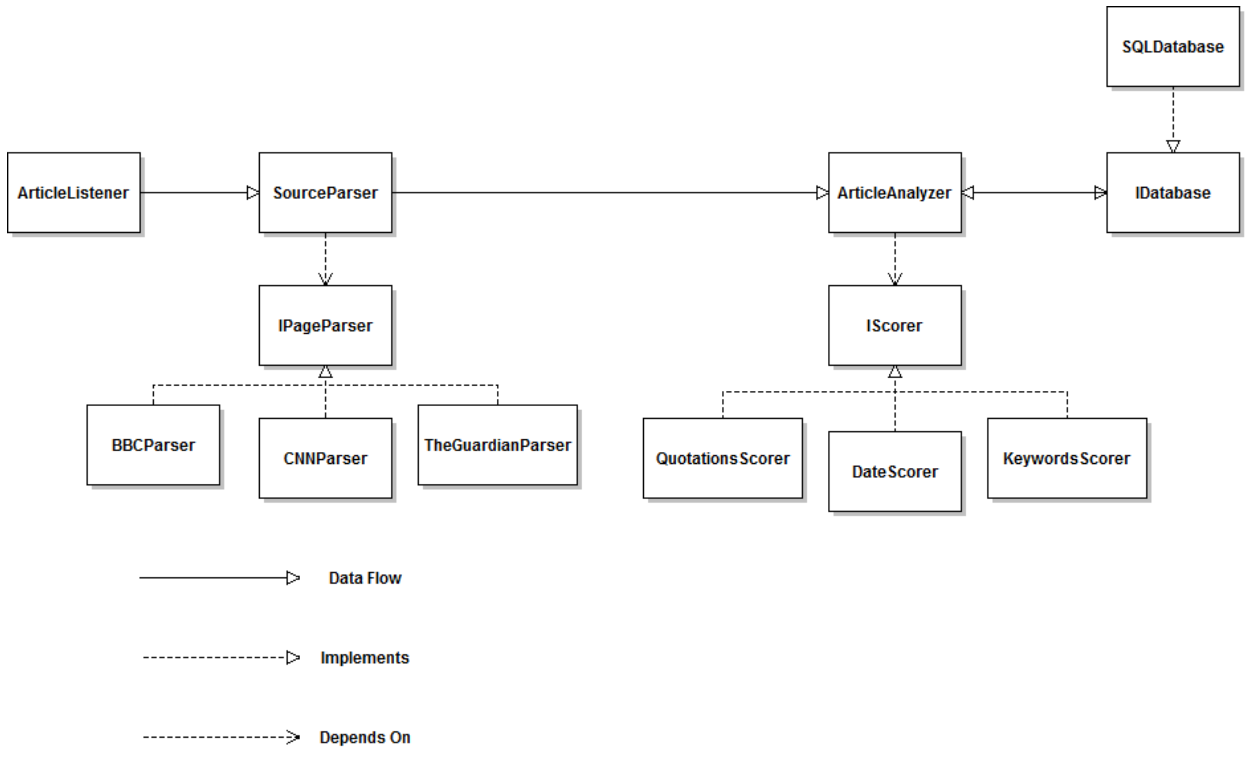
Users Table

Field	Type	Description
User Name	String	Username of the person using our service.
Source Exceptions	String[]	Sources they do not wish to view.
Password	String	Hashed version of user's password.

#3 Java Backend

Our backend will be running on an Iowa State server. The backend will constantly monitor the websites we are supporting and bring in new articles for analysis when they are posted to the websites.

Backend Modules and Data Flow



Backend Modules

A. Article Listener - This module will be in charge of finding new articles as they are posted and bringing them into the system for analysis. This will be achieved by having this module constantly monitoring a RSS feed.

B. Source Parser - This module will be in charge of determining relevant information about the articles that will be needed for story aggregation. It uses implementations of the IParser interface which parse web pages to collected the needed information.

Information Gathered
#1 The keywords in the title
#2 Date of publication
#3 The keywords in the article
#4 Location (if applicable, not all sites provide this information in the meta data)
#5 Category that the article was placed in on the website it comes from
#6 Quotations from the article if there are any

C. Article Analyzer - This module will take the information gathered about the article and try to match it to one of the stories in the database, and if it can't find a story it matches with it will make a new story entry for the article in the database. The matching will be done using IScorer objects.

D. IScorer - This is the interface is used by the Article Analyzer module to determine the matches between a given article and story in the database. It stipulates that the class implementing it must have a method called score that takes in two Article objects and returns a double value between 0 and 100, with 0 meaning the worst match possible between the two articles and 100 meaning a perfect match between the two articles.

E. Keywords Scorer - The keyword scorer will look at the keywords of two articles and compare how well they match up. If the two articles share no keywords the comparison will be given a score of 0, if they share all the same keywords the comparison will be given a score of 100. For a number of matching keywords in between the score will be calculated by using the formula below.

Formula = 100 * (number of keywords with matches / total unique keywords between articles)

F. Date Scorer - This scorer will score two articles based on how close the date they were published is to each other. The maximum amount of time that they can be separated by can be changed at anytime in the preferences file of the project. A 0 score is given to two articles that were not published within the allowed time frame and a score of 100 is given to two articles published at exactly the same time. Formula used to calculate the score is given below.

Formula = 100 * ((max time difference - actual time difference) / max time difference)

G. Quotations Scorer - This scores two articles based on how similar the quotations in the two articles are to each other. It uses two algorithms, that are then weighted to determine the score. The weighting of the two algorithms can be changed as need in the preferences file. The first algorithm used is Edit Distance, which is a measure of how many insertions, deletions, and replacements are needed to transform one array of values to be equal to another. The second algorithm used is Longest Common Substring, which returns the longest string of elements that appear in both arrays. The way these are used to determine the score is a bit harder than the other two, so no formula will be provided, but if two articles have exactly the same quotes a score of 100 would be given and if the two articles have quotes that share no words in common it would be given a score of 0.

H. IDatabase - Allows communication with a database to take place so that the state of the system can be persistent.

Input Output

System
Input: RSS feed of the websites that we are supporting.
Output: A website containing several different news stories, with articles about the same story aggregated together.

Java Backend Module
Input: Link to article from a website that we support.
Output: Entry in the database about what story that article belongs to.

Java Backend - Article Listener
Input: Multiple RSS feeds from the web sites that we are supporting.
Output: The source code of the webpage whenever it detects that a new article has been posted on one of the sites.

Java Backend - Source Parser
Input: Source code of the web page containing a news article from one of the sites that we are supporting.
Output: Several different pieces of information, for more information on what data it will collect look at the module description, that will be useful in determining if the article is about the same story as other articles from different sites.

Java Backend - Article Analyzer
Input: Information about the article that is obtained by the Article Analyzer module.
Output: The story from that database that the current article being analyzed matches with, or a null output if there isn't a story in the database that it matches to.

Java Backend - 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 or data is requested to be retrieved.
Output: Data is input is stored or data requested is 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.

Java Backend - Article Listener
RSS Feed Monitoring: This function will monitor an RSS feed to determine whenever a news site that we are supporting has posted a new article. Whenever it detects that a new article has been published it will get the source code of the page that the article is published on.

Test Plan Procedure: After we get this written will look at the RSS feed to see what articles are on it and then we will make sure that this function is detected all of them and gathering the source code.

Test Plan Interpretation: If the function is gathering information from all of the links on the RSS feed it will be considered a successful test.

Java Backend - Source Parser
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 plugins 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, we're 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.

Java Backend - Article Analyzer
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.

Java Backend - 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 test 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 where 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.

Sources

#1 CNN (US)
#2 The Guardian (UK)
#3 BBC (UK)

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
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	The cost of any additional

	product?	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 aggregated.
	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

A). Java 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 various pieces of relevant information as laid out in the article table.

2. Scanning

The backend should be continuously scanning the rss feeds for new articles to be added to the database.

3. Sorting

The backend should be able to sort articles into stories based on date, location, and article content. These stories will be specific events on which multiple articles are reporting on.

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.) Stories

Articles belong to stories which are created when the first example of a given story's member articles is committed to the database. Stories groups of articles that all deal with the same event or subject matter, during the same time frame.

C.) <u>User Interface</u>

1. User Access

The user should be able to use a profile to login. Their preferences and excluded sources will be recorded along with their account information.

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 are supported which 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). <u>Security</u>

A).Username

A user account is associated with a single username and password, as well any modifications the user has made to their sources. Their passwords will be hashed in order to avoid compromising user security.

2). <u>Performance Requirements</u>

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 three hours of the article being added to an RSS feed.

B.) Categorization and Aggregation

The articles should be properly placed into stories at least 80% of the time. This is tested through individual user testing.

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

Standards

Our team shall always follow the same coding guidelines throughout the project. At the start of every Java, HTML, CSS, or other file should be a comment describing what the file is for and does. All functions will have comments one line above the function declaration declaring what the function does and how the function works. There shall be two blank lines between the end of one function and the start of another function. The “{“ after a function will be on the same line as the function with one space between the function and the bracket. The closing bracket should line up with the starting position of the declaration with the corresponding opening bracket. All if, for, and while loop statements should be indented one tab from where function declaration starts. All other code embedded inside a loop should be indented one tab from the loop declaration starting position. There should be one space on each side of any operators (=,-,+ ,*, etc.). An example is: x = y + 5. All testing should be documented on what was done and the corresponding results of the tests.