**Sweet News**

**Addressing the Modern Culture of “Fake News”**

Stephen Weber, William Rosman,

Sonika Rajan, Vishak Lakshman Sanjeevi Kani Murugesh

**Table of Contents**

**Abstract**………………………………………………………………………………………….**2**

**Introduction**……………………………………………………………………………………..**3**

**Requirements Analysis**.………………………………………………………………………**4**

**Technical Requirements and Specifications**……………………………………………...**6**

***Programming languages***…………………………………………………………......**6**

***Reused algorithms and programs***………………………………………………….**6**

***Tools and environments***……………………………………………………………...**7**

***Database system***……………………………………………………………………....**7**

**Design Specific Documentation**…………………………………………………………….**9**

***Software architecture****………………………………………………………………....****9***

***Sequence diagrams****………………………………………………………………….****10***

***Class diagram****………………………………………………………………………...****13***

***Component diagram****….……………………………………………………………...****13***

***Deployment diagram****….……………………………………………………………..****14***

**Implementation**……………………………………………………………………………….**14**

**Performance and Result Analysis**………………………………………………..……….**15**

**Abstract**

In today’s age, with the prominence of social applications such as Facebook, Twitter, and Instagram, news stories travel fast, whether they are true or not. This has lead to a large influx of news stories being published that offer questionable information at best and from sometimes unknown sources. Much of the general public are unable to decipher the good news from the bad as they are putting their trust in the outlet that is provided the stories to them. With that being said, many applications that currently exist in the marketplace feed off this and push stories out to the masses, often without vetting their sources and verifying the validity of their statements. Many of these stories appear in those intrusive advertisements that one may see at the bottom of a webpage, however they are becoming increasingly more abundant thanks to the social media outlets mentioned above. To address this, there should be an application that provides curated news stories based on only the topics that interest that particular user.

This paper presents a solution to the abundance of deceptive news stories by demonstrating the development, implementation, and testing of the Sweet News web application. In this application the user has the ability to, through creating an account, select only the topics that they are most interested in. Then, they are shown only those news stories that are relevant to them and only from credible news outlets.

**Introduction**

The idea for Sweet News came from the abundant amount of misinformation that is circulating via the various social media mediums and news outlets. While sites like Facebook and Twitter can provide a soap-box like platform for which it’s users can share information, they can also provide a means for its users to share very factually incorrect information. That’s because this information is rarely, if at all, vetted for accuracy which can lead to a very misinformed public. This was evident during the 2016 election season as both candidates had false news stories circulating about them.

Users are interested in what’s going on around the world but are now having trouble trusting the sources from which they receive their articles. Therefore, this project focused on allowing the users to select exactly what they want to see and then provide them with news articles. These articles would be provided by some of the most trusted sources (AP, ABC, New York Times, etc) and therefore offer a wide selection of fact-check news stories.

In order to differentiate from news that a source like Google News can provide, this application needed to provide the user with specific functionality. Therefore, we explored adding a feature in which additional sources can be added to the application, which in turn can be approved by an administrator. This allows the application to constantly evolve past the limitations of the API that is delivering the news articles/sources.

This report will cover the design, implementation, challenges, lessons learned and conclusions of the project. The user documentation submitted alongside of this report will provide detailed instructions of how to use the user interface of the web application.

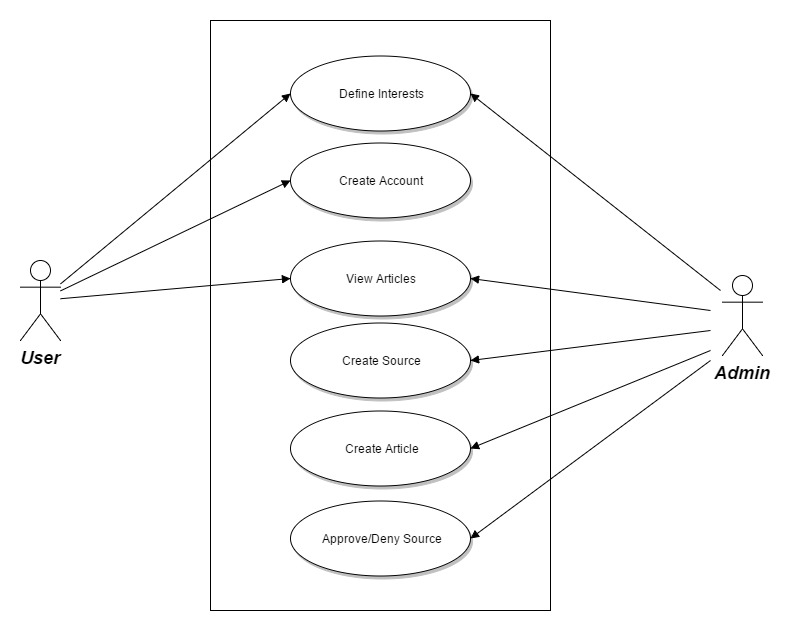
It is important to note that this project was initially designed to incorporate a Twitter scraping API that would present us with stories that users would like to read based on their tweet history. Through the design process the group found it more practical to instead allow the users to select the categories that interest them the most. It is sometimes difficult to say whether or not a user who tweets about something is actually interested in the story or if they are just currently have a short-lived opinion on the topic.

**Requirements Analysis**

Designing an application that would be used by people who are used to gathering their news from certain locations had it’s difficulties. We needed to be able to provide the user with accurate and factual information however still provide the ability for the platform to evolve with the addition of new sources. Providing this kind of dynamic nature would increase the chances of them returning to the site, as new news sources could be constantly added.

Figure 1 dictates the use-case for this project and demonstrates what the user is able to do. The user can, first and foremost, create an account which would only require the typical account information from them (email address and password). Once the account is created, they can then define their own interests based on the categories provided by the news sources via the API. These categories would be used to generate and display relevant news articles to the user.

Administrators, the other actors in the use-case, are a type of user. They have their own account, interests and can view articles. The main difference between an admin and a user is that they can add new sources to the site. Once a source is added, the admin can link specific articles that can then be added to the database, which in turn allows the articles to be viewable to all users. In addition, the admins can either approve or deny sources that have been submitted by other administrators. This system of checks and balances allows admins to submit sources and articles, but prevents them from submitting non-factual articles from deceptive sources.



**Figure 1 -** Use case depicting the Admin and User and the corresponding abilities the web application provides them.

***UI Design***

From an aesthetic standpoint, there needed to be a design language that kept the user on the page. A design that would be cluttered and/or cumbersome would not be an efficient way of attracting users regardless of what kind of service the application provided. For that reason a material design was selected that allowed for a very minimalistic viewing experience. The components of each page are evenly laid out in columns and rows, making it very easy for the user to navigate throughout the application. The design was made to be fully responsive so that users could access the applications from any device and still experience the same functionality. As an additional way to keep users on the site and interest them in creating an account, a row of three sample stories was added to the homepage. This row shows up-to-date articles and demonstrate what the user would have access to upon creating an account.

***Methodology***

For this project, the Agile method of Software development was employed. This methodology was selected as there were multiple iterations of designs stemming back from conception. The original idea was scrapped due to the subjective nature of the project goal and therefore iterations of development occurred as we discovered new use cases and user requirements. An example one of these iterations is the inclusion of an administrator. This idea was included after we found that users would want the ability to add sources that are not provided by the API.

The group members also reused code from various other assignments, examples, and projects to help the implementation process. This allowed for easier development as the developers had an idea of what the code did and how to apply it to this particular application. The reuse of code also helped with refactoring as the developers were able to use the code that was the most efficient and eliminate the code that was unnecessary. They saw the limitation of past implementations with that code and were able to make an informed decision of how to manipulate it to their specific need.

**Technical Requirements and Specification*s***

***Programming languages***

For achieving the most collaborative environment possible, the back end of the application is written in Java, with the Tomcat library supporting servlets, and JDBC for database communication. This decision was based on knowledge that Java was the language that was most familiar to everyone involved in the project. The Front end of the application is written in JSP. The chosen database environment is MySQL.

***Reused algorithms and programs***

To load the initial news sources, we used the newsapi.org API, this API also allows us to refresh the articles available from the news sources it supports on demand. The responses from the API calls for news sources and articles is returned in JSON format, which the application uses the JSON library to parse through, and create Database Objects, which are used to update the database with current articles, and articles relevant to the user are returned to the JSP page for display.

The news sources in the API are given a category, which is tied back to the user’s preference choices in determining which articles to display to the user. When a news source is manually added and approved by the website administrators, the articles for that news source are not automatically added to the database, but must be manually added by other administrators. The administrators adding articles from a manually added approved source must provide all relevant information before the article can be displayed to users who share an interest in that topic.

Data structures in this application are simple Lists, JSONArrays, and JSONObjects. JSON data structures are more efficient than DOM, or other XML structures, which is why an API that returned data in JSON format was sought.

Java with JSP was chosen as the optimal language choice for this application because of the efficiency that code can be reused, as well as the simplicity of implementing JSON functionality and the support of inheritance. Because Servlets are very cross functional, a single servlet can be used to provide a wide array of functionality depending on what information is passed to it.

Exception handling proved to be challenging. Because of the amount of SQL in the application, a plan for acknowledging and responding to exceptions was needed. Tomcat provides built in logging functionality, which we make use of when an exception is thrown. In addition, the number of objects that need to be reset, closed, or orphanized upon exception had to be taken into account. We also needed to take into account which methods needed to handle exceptions, versus which methods could simply throw the exceptions up. During testing, it was discovered that handling exceptions was best done in a public method that served as an entry point into the class, to close any class variables, and then throw the exception back up to the calling class.

***Tools and environments***

Eclipse IDE is used for developing both frontend and backend components. The J2EE platform provides the necessary environment and testing tools to code and incorporate the changes regarding the different modules. The compiler version of JDK used for development is javac 1.8.0\_112.

Apache Tomcat server is used for hosting the web application while MySQL workbench provides the platform for designing the schema and establish the relationship between the tables. GitHub is used as a team management tool which allowed our two groups of backend and frontend developers to have synchronized codebase for development. Google Chrome is used as browser for viewing the web application for testing purposes.

***Database system***

MySQL is used as the platform for Database design and implementation. Figure 2 shows the EERD for our project. The relational schema has the following tables :

1. UserLogin (Username , Password)
2. User (UserId , FirstName, LastName, Email, DOB)
3. Category (CategoryId, CategoryDescrition)
4. User\_Preference ( UserId FOREIGN KEY, CategoryId FOREIGN KEY)
5. Source (SourceId, Name, Description, Url, Category FOREIGN KEY, Country, Language, UrlLogo, ApprovalStatus)
6. Article (Author, Description, Title, Url, UrlToImage, PublishTime, SourceId FOREIGN KEY)

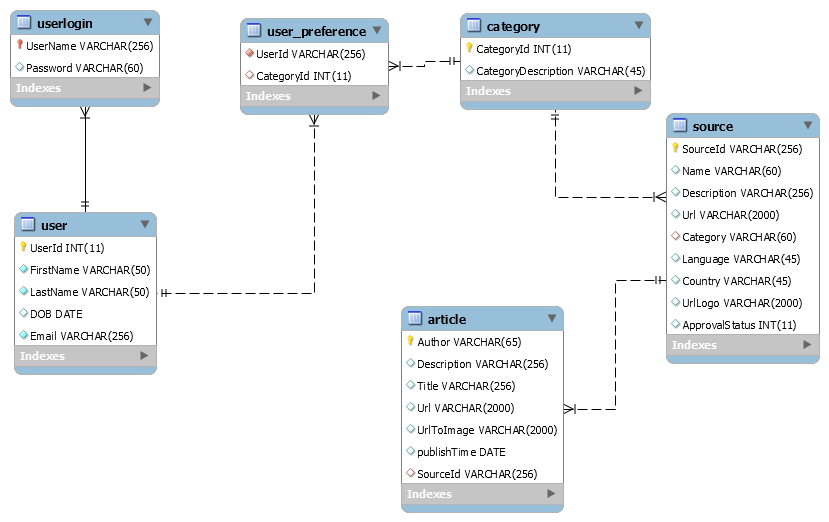
****

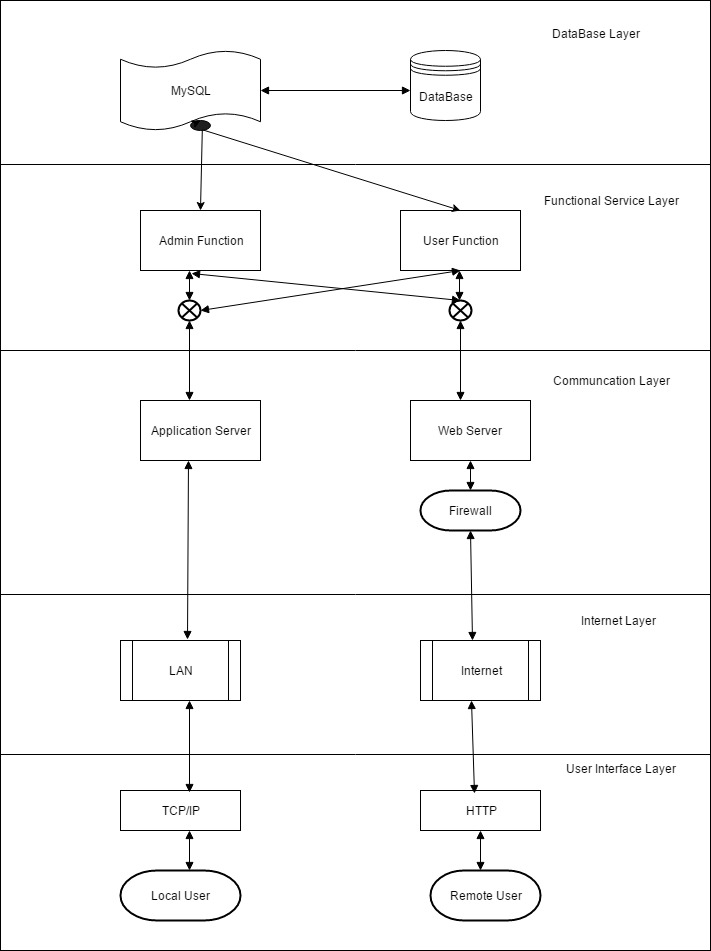
Figure 2 - Entity-Relationship Diagram for Sweet News Database

The relations between the tables can be given as follows :

|  |  |  |
| --- | --- | --- |
| Table 1 | Table 2 | Cardinality |
| UserLogin | User | One to One |
| User | User\_Preference | One to Many |
| User\_Prefernce | SourceId | Many to Many |
| Source | Article | One to Many |

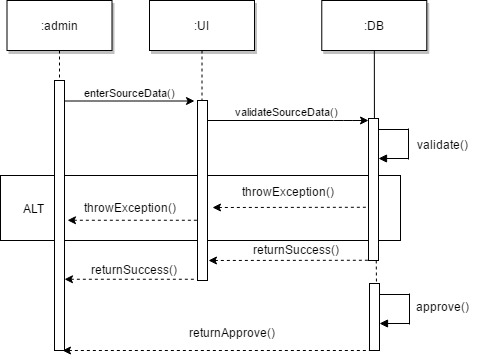
**Design**

***Software System architecture***

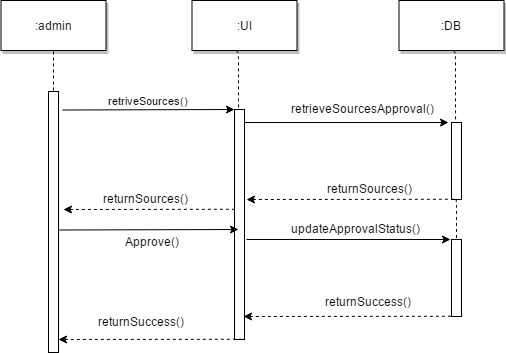
******

***Sequence diagrams***

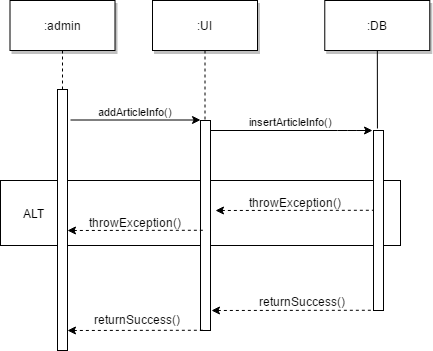
.1 Creating new source

******

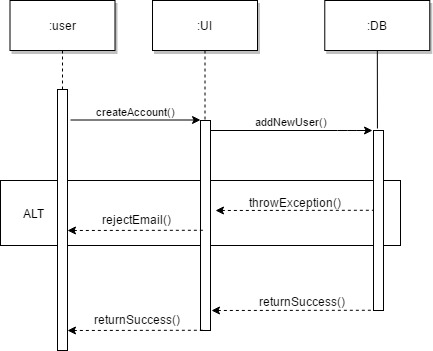
2. Source Approval/Denial



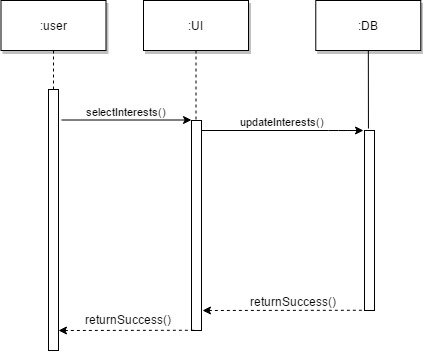
3.Creating a new Article

******

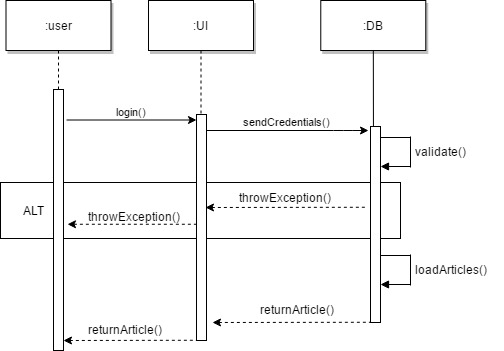
4.Creating a new account

******

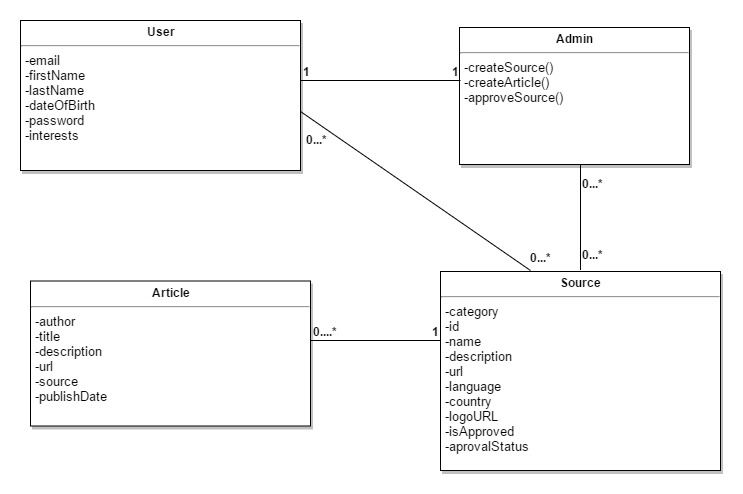
5.User interest update

******

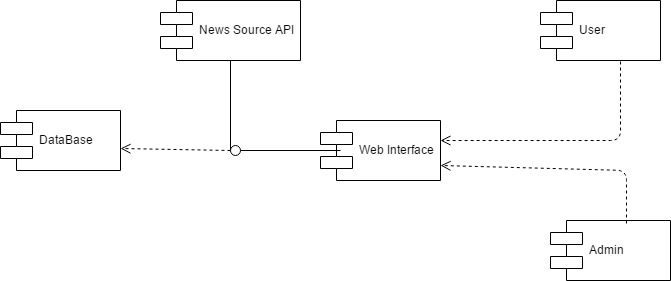
6. Login and Article Loading

******

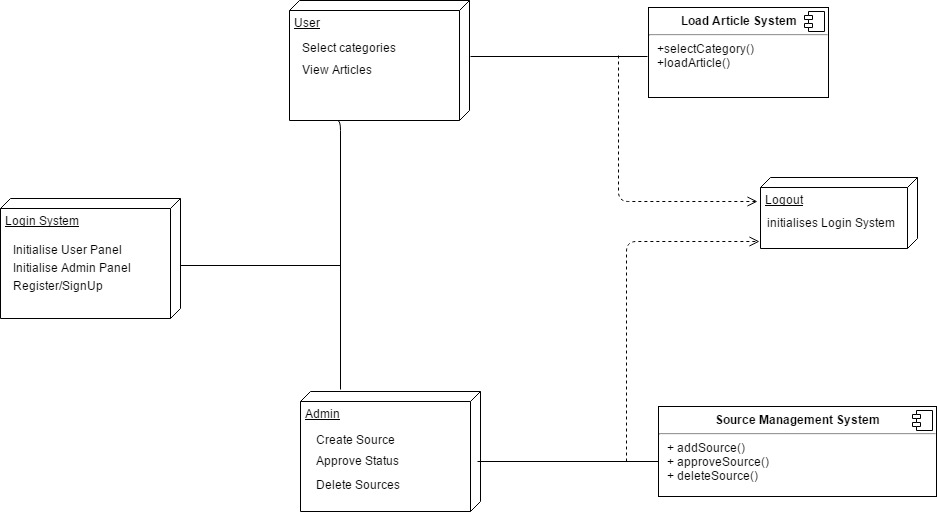
***Class diagram***



***Component Diagram***



***Deployment diagram***



**Implementation**

Once the design goals were established, the group members were paired off in teams and broke the project up into two sections. One team worked solely on the backend coding and the other on frontend design and incorporation of the dynamic functionality provided by the team assembling the backend.

The two teams worked separately taking turns writing code as the other instructed the coder what the next course of action was. This was especially helpful on the front end as designing the user interface can be tricky to those who are not well versed in HTML5 or CSS. At the same time, the backend developers benefited significantly from the pairing up as one wrote application code and the other incorporated the SQL implementation. This strategy allowed the group to not only accomplish the task but also learn from one another throughout the course of the project.

The group used a wireframe that was created very early on in the design process to implement the application. Although that the developers went with a third-party template for a base CSS (cited at the bottom of all of the web pages), the wireframe was instrumental in the reformatting the pre-existing template to meet the specific needs of the user.

One common theme of the implementation process was the agile principle of reusing code. All of the developers had experience working on other projects and were able to use that experience and bits of code to implement features that were needed for Sweet News.

**Performance and Result Analysis**

When the application was finally up and running, we began to notice things that could be corrected. For one, there was a good amount of code that could be optimized and taking the time to go back and refactor said code helped improve our speed and memory consumption. There were also numerous things that were not evident in the design process that became abundantly more clear during testing. An example of this was the way in which the administrator is defined.

Another main lesson learned when designing and implementing this application is that the vetting process for news stories is a tough task. News sources that may not be completely reliable, and hence do not belong in this application, can be included in most APIs, and therefore the human eye was important in deciding which news sources should be included.

There is more to explore in when it comes to providing news to users as the reach of negative rhetoric continues to spread throughout the internet. However, being able to filter the “fake news” goes a long way in tackling this issue. In that respect, the implementation of Sweet News is a success. Being able to provide curated and fact-checked articles goes a long way in reinstating user confidence.

**Lessons Learned**

The weekend before the project was due we experienced a massive git issue, which resulted in significant loss of data. While this caused panic, emotional outbursts, and lots of bad language, we pulled together and got the project back up to the best of our ability with the very limited time we had available. We feel that this experience, however disheartening it was, is the most valuable experience that could have happened. While hopefully the application is in a state that can be run and demonstrated, the experience of the stress and the following coming together to achieve a common goal was by far the most rewarding thing that any of us has experienced in the academic world thus far.