FrontPage

Software Requirements Document

Matthew Bindewald
Reid Kolaczek
Justin Lukas
Elliott Miller

Change History

Version	Date	Summary	Author
0.1	2/22/2018	Initial Draft	FrontPage Team
0.2	2/22/2018	Introduction, Requirements Update	Elliott Miller
0.3	2/25/2018	Project Description Update	Matthew Bindewald
0.4	2/25/2018	Competitive Analysis Update	Justin Lukas
0.5	2/26/2018	Construction of Diagrams	Reid Kolaczek
1.0	2/28/2018	Draft Changes	FrontPage Team
1.1	3/1/2018	Implementation Details and Further Information added	FrontPage Team
1.2	4/2/2018	Addition of Activity, Use Case, and Sequence Diagrams	FrontPage Team

Table of Contents

Introduction

1. Purpose

FrontPage is a web application that provides a centralized location for users to access trending news. The application also supports searching for specific topics across a wide variety of social media platforms and news outlets and condenses the results based on popularity. This document outlines the requirement specifications for the implementation of FrontPage.
As there is not currently one application that offers trending news from both social media sites and new sources in such a simplified format, FrontPage aims to become a centralized platform for users to access top news stories with ease, without having to visit and scroll through multiple websites and applications to access the most popular posts.

2. Scope

Since sixty-seven percent of adults get their news from social media, and over one quarter of this demographic get news from one site only, FrontPage offers a unique experience that allows users to track top news stories from a centralized location (Pew Research Center). Our mission addresses “reinventing news” and the way that consumers track trending stories.
Front page will pull data from both news networks and social media sites in order to provide the most integrated, content-rich experience.

3. Goals

FrontPage seeks to be a channel through which users can stay up to date on current news by accumulating top stories around the United States through the convenience of a single, user-friendly environment.

4. Key Definitions

API - an **application programming interface** (**API**) is a set of subroutine definitions, protocols, and tools for building application software.

Metadata - a set of data that describes and gives information about other data.

JSON - **JSON** (JavaScript Object Notation) is a lightweight data-interchange format.

Script - A **script** or **scripting** language is a **computer** language with a series of commands within a file that is capable of being executed without being compiled.

Database - a structured set of data held in a computer, especially one that is accessible in various ways.

Project Description

1. Project Motivation

The main goal of this project is to provide an overall picture of what the hottest news trends are across the entire internet. Social media platforms are some of the most common ways we get news, but every social media platform has different communities and different types of people who use them, spreading different types of news. We want to create a news source that gives a more general view, exposing people to news that they might not otherwise be aware of if they stuck to their normal social networks. Just as important as what FrontPage is is what FrontPage is not. FrontPage is not a social network, users don't share stories or interact with other reads. FrontPage is not biased - it evaluates trends created by the millions of users that use today's most common social networks. There are no journalists that choose what stories are published and what aren't, it's all automatic.

2. Product Features

FrontPage is a website. When users load the website they'll be presented with a list of the current most popular trends, and stories from various news sources and social media websites related to those trends. These trends will be updated every few minutes so that users will always have the latest news. Each trend is determined by analyzing the most popular posts from various media sites such as YouTube, Twitter, Google News, and Reddit. Once the trends are determined, we reach out to those individual sites and grab stories related to each trend. For example, if a new Tesla car was just announced, it may appear on our FrontPage. The display of the trend may display popular tweets about the announcement, a news article about it, or maybe a YouTube video of someone test driving the car. Our website will only initially display the top 5 or 10 trends, but users will be able to search for other trends by utilizing the search bar on our site.

Functional Requirements

1. The application will determine most popular news stories and articles to display to the user
2. The application will allow users to search for specific topics and display resulting posts from various media platforms
3. The application will filter news stories and articles based on the user's preferences
4. The application will provide a summary of each post for users to view

5. The application will pull data from multiple news and social media sources through the use of APIs

Non-Functional Requirements

1. Trending news will be determined by pulling data from various social media platforms and identifying the most prevalent topics
2. Top stories will be updated in semi-real time based on a predefined refresh delay, no longer than five minutes
3. Clicking on a post will redirect users to the appropriate source
4. The application will list metadata regarding the popularity of each post, including how many views or reposts
5. Addition of searching by keyword and filtering out by keyword as well as select location and language.

Competitive Analysis

Functional Requirements

Application	The application will determine most popular news stories and articles to display to the user	The application will allow users to search for specific topics and display resulting posts from various media platforms	The application will provide a summary of each post for users to view
FrontPage	full	full	full
Google News	full	full	full
Apple News	full	full	full
Reddit	full	full	full
Twitter	partial	full	partial
flipboard	full	full	full

Most functional requirements are currently being met by all competitive applications. That is to be expected since our application is designed to build off the design of many competitive applications. There is already a large group of applications designed to provide news as well as serve as social media platforms currently in the market. They vary with UI, origins of content, and user interaction. However, most of the premise is similar to the purpose of FrontPage.

However, most of what sets our application apart as compared to the others in the marketplace comes from the non-functional requirements.

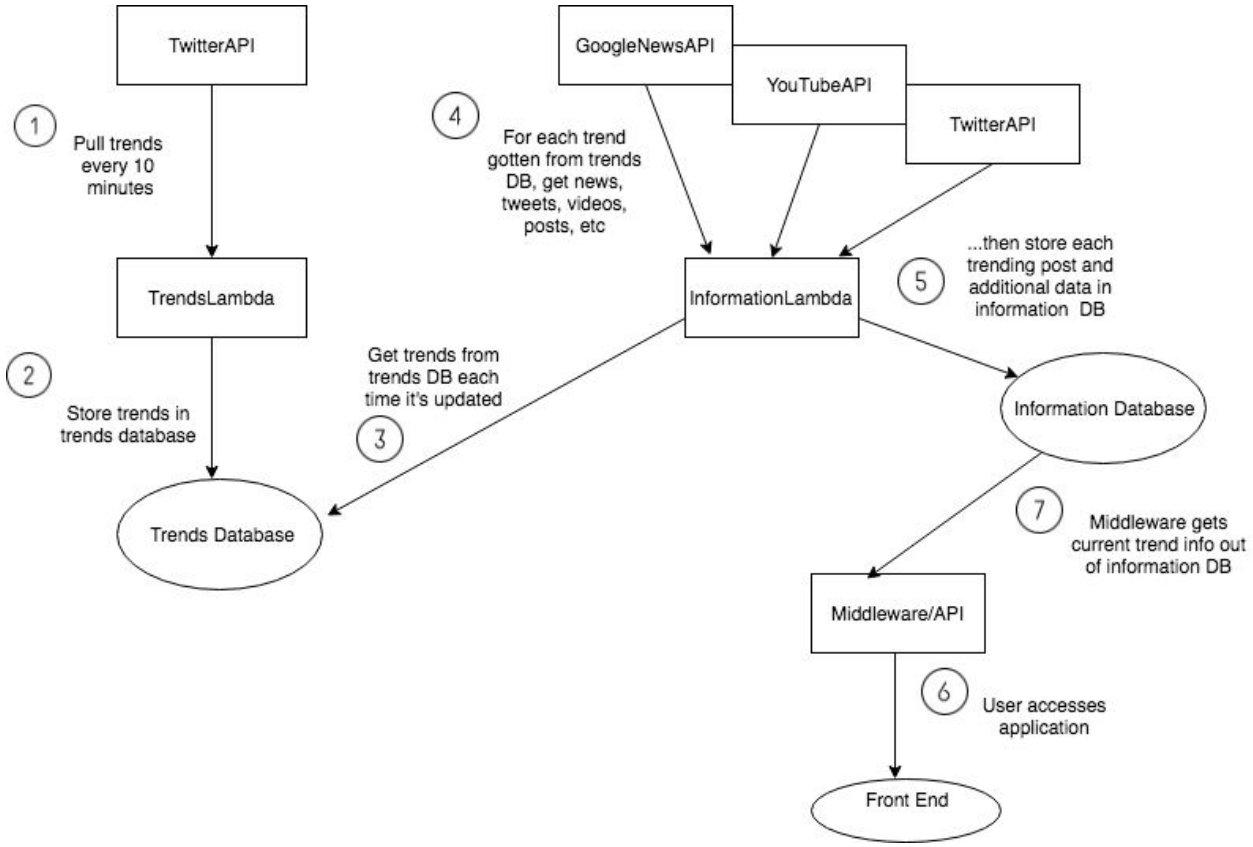
Non-Functional Requirements

Application	Trending news will be determined by pulling data from various social media platforms and identifying the most prevalent topics	Top stories will be updated in semi-real time based on a predefined refresh delay, no longer than five minutes	Clicking on a post will redirect users to the appropriate source	The application will list metadata regarding the popularity of each post, including how many views or reposts
FrontPage	full	full	full	full
Google News	partial	partial	full	No coverage
Apple News	partial	partial	full	No coverage
Reddit	No coverage	partial	partial	partial
Twitter	No coverage	partial	partial	partial
flipboard	partial	partial	partial	No coverage

Where FrontPage differentiates itself from the competition comes from non-functional requirements of the application. As the table shows, there is not any applications currently in the market that can provide all the features that FrontPage would be able to provide. Most notably, applications are separated into two main categories, applications that allow users to post content, and applications that are geared towards publishing news stories without user interaction. FrontPage falls into the latter, but with a competitive advantage because it takes the most popular posts from many different platforms and combines them in one place. Google and Apple News have relatively similar means of operating, however their posts come from news focused articles, can also have sponsored stories, and are not cross platform. FrontPage is able to differentiate itself from the competition by combining posts from major social media and news platforms. Unlike Facebook, Twitter, YouTube, and others that only have posts from users on their networks, our application is able to combine the top posts from many platforms in one location.

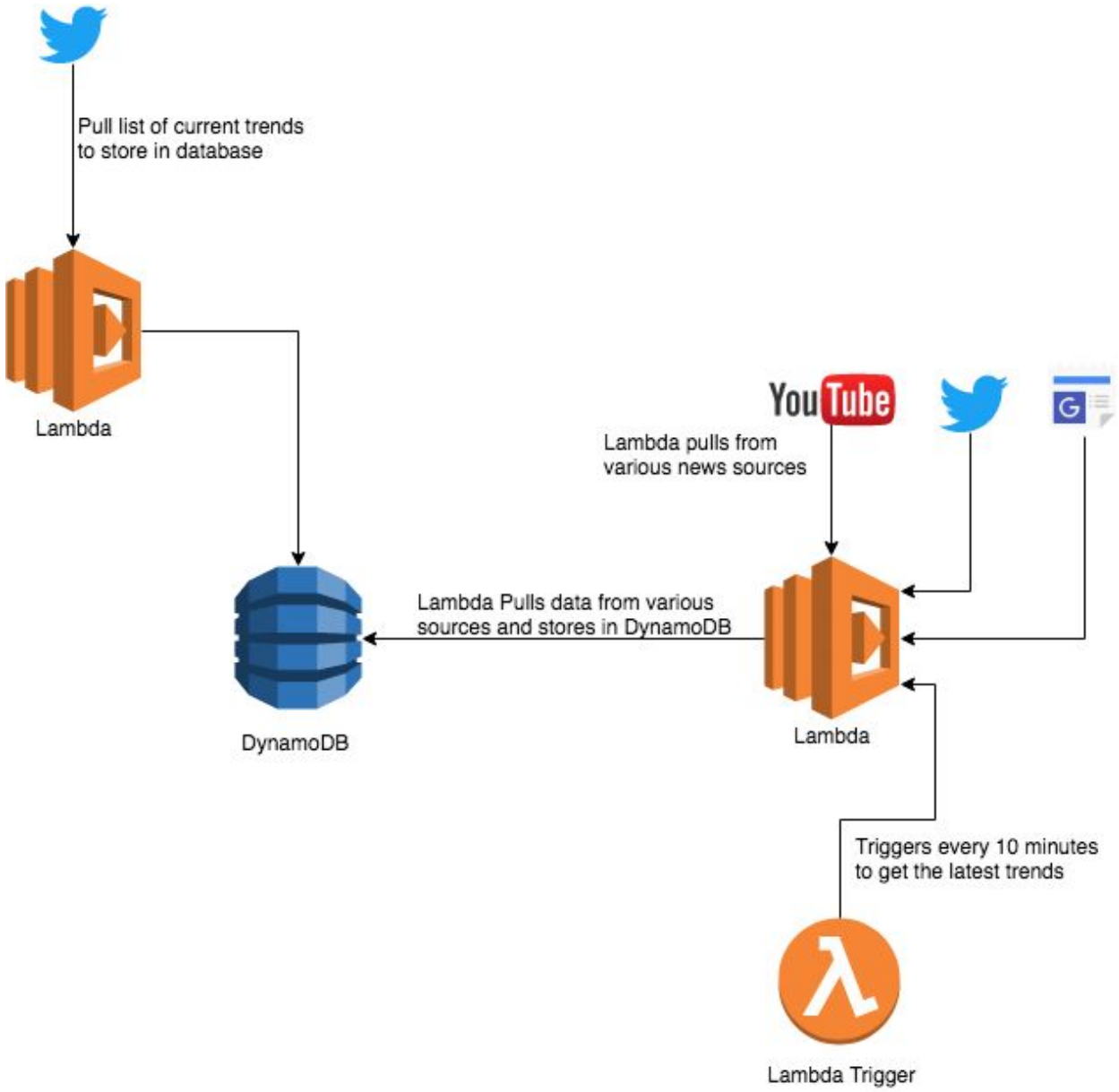
UML Diagrams

1. Overview Diagrams



a. General project structure

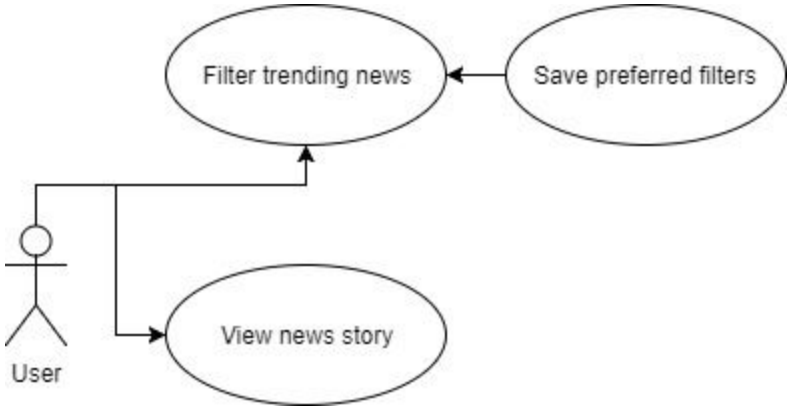
This diagram depicts the comprehensive architecture of our project. Each step is indicated by a number and explanation for how and when the different actions are taken. The general behavior is as follow: Top trends are retrieved from Twitter by Trends Lambda, stored in the Trends Database, then pulled by the Information Lambda and used to identify which posts from each media source should be stored in the Information Database. Those posts are then updated in almost real time (every 10 minutes) to be pulled by the middleware and displayed to the user.



b. Database structure overview

This diagram reiterates the process explained in the previous diagram and gives a better visualization of the interactions between the databases.

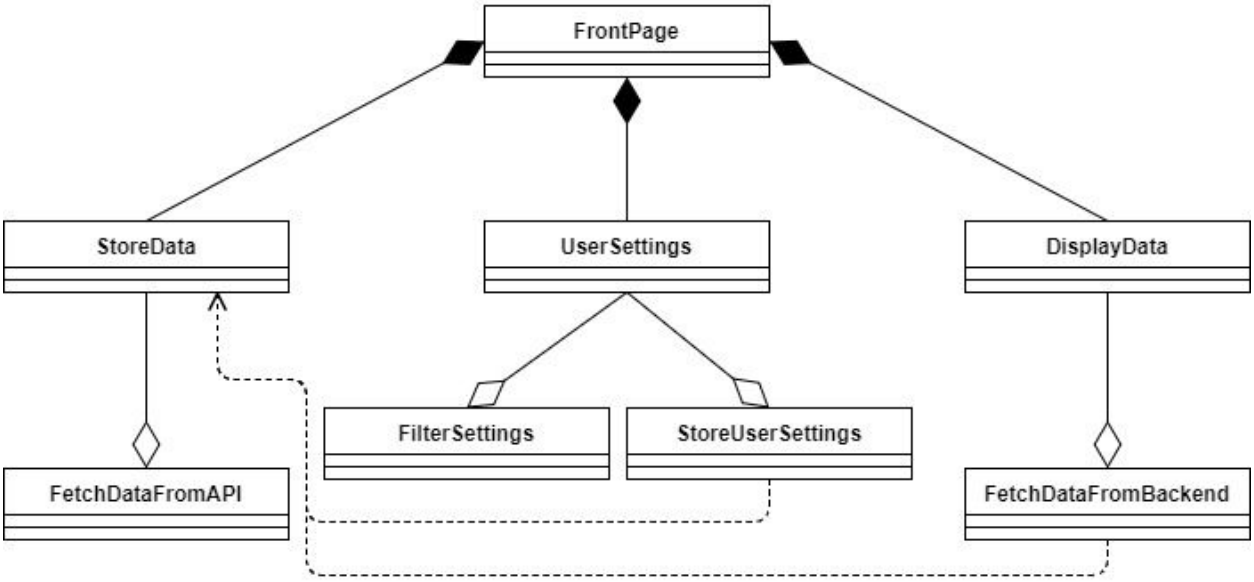
2. Use Case Diagrams



a. Simple use case scenario

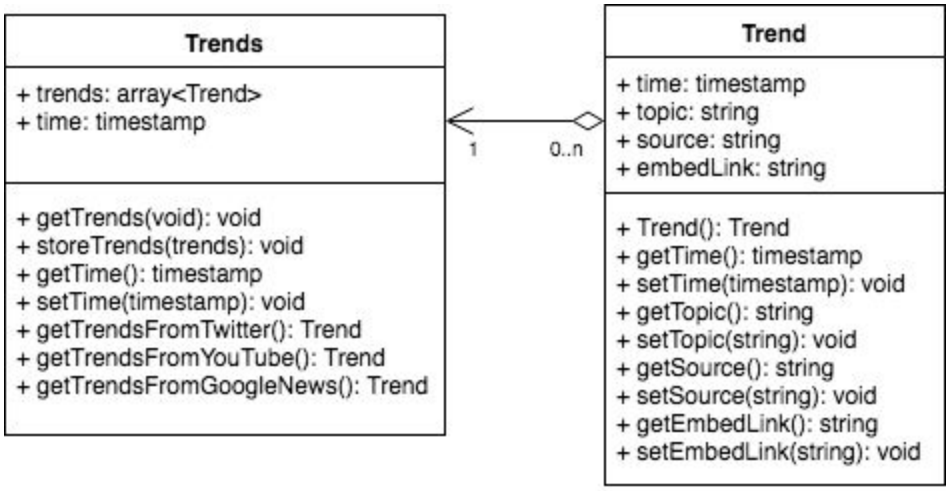
The use case for front page is fairly simple. We will have two main views, one where a user can view the trending news story, and the other one will allow the user to filter the trending news. Finally, there will be an option to save these filters to the user's account. Some example filters would be narrowing results by source, time it was posted, media type (video, photo, text), and category.

3. Class Diagrams



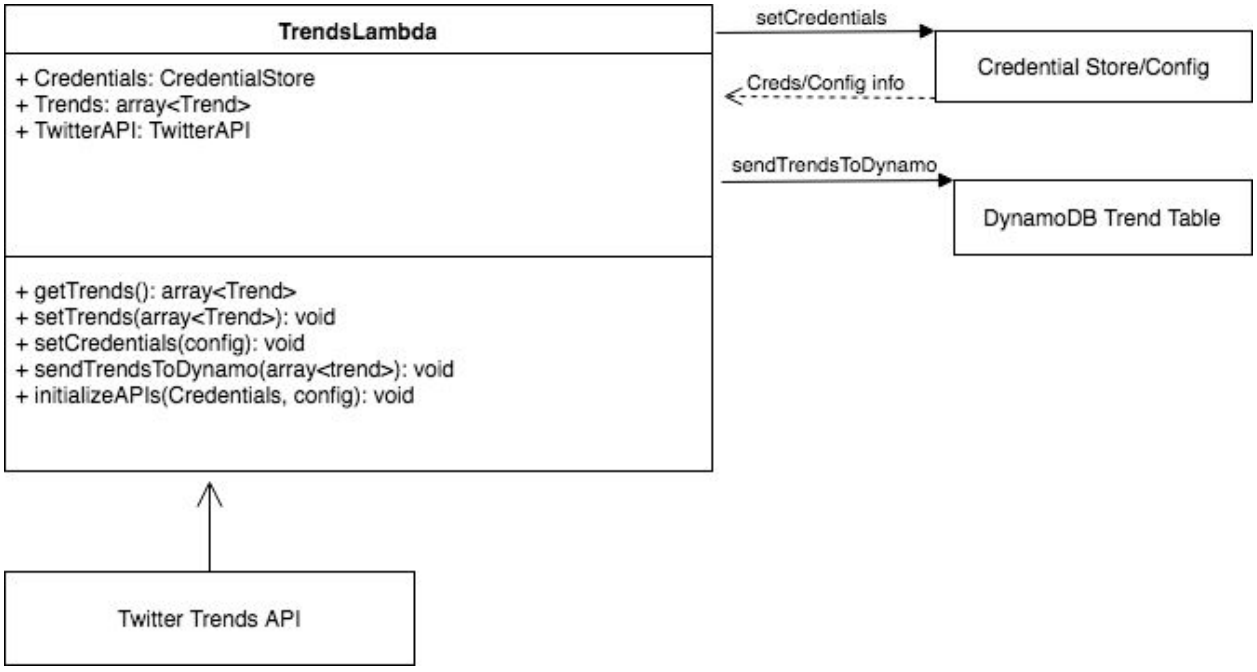
a. Simplified class overview

At a high level, our system design attempts to follow the principle of high cohesion and low coupling. Separating everything into modules, as well as utilizing inheritance where necessary, our design will be modular and adaptable to new requirements. The three main classes each handle a part of the application. First we will store the data in the database where it can be retrieved from the displaydata class. Furthermore, we will also utilize the storedata class to handle storing user settings in the database. This class needs to be as abstract as possible to allow for multiple types of data to be stored in the database.



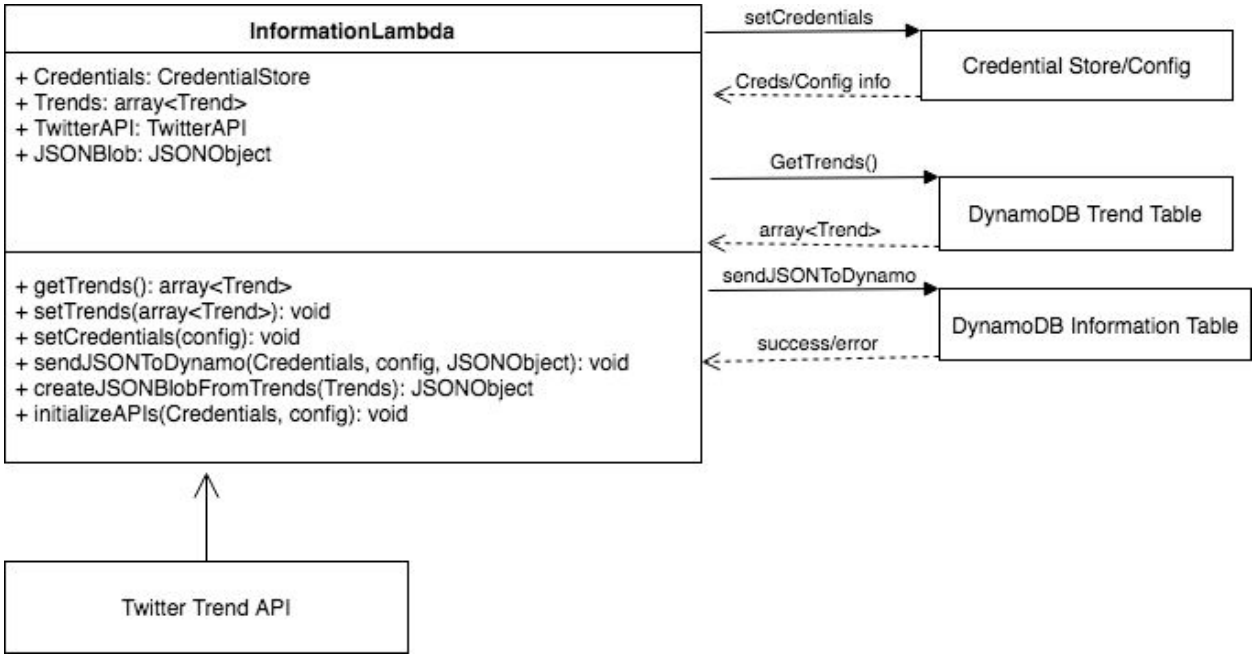
b. Trends Class Diagram

This is the diagram depicting the class on which our project is based. The trends class is how we will store both the top trending topics initially retrieved from Twitter as well as the posts pulled from various platforms and their associated metadata.



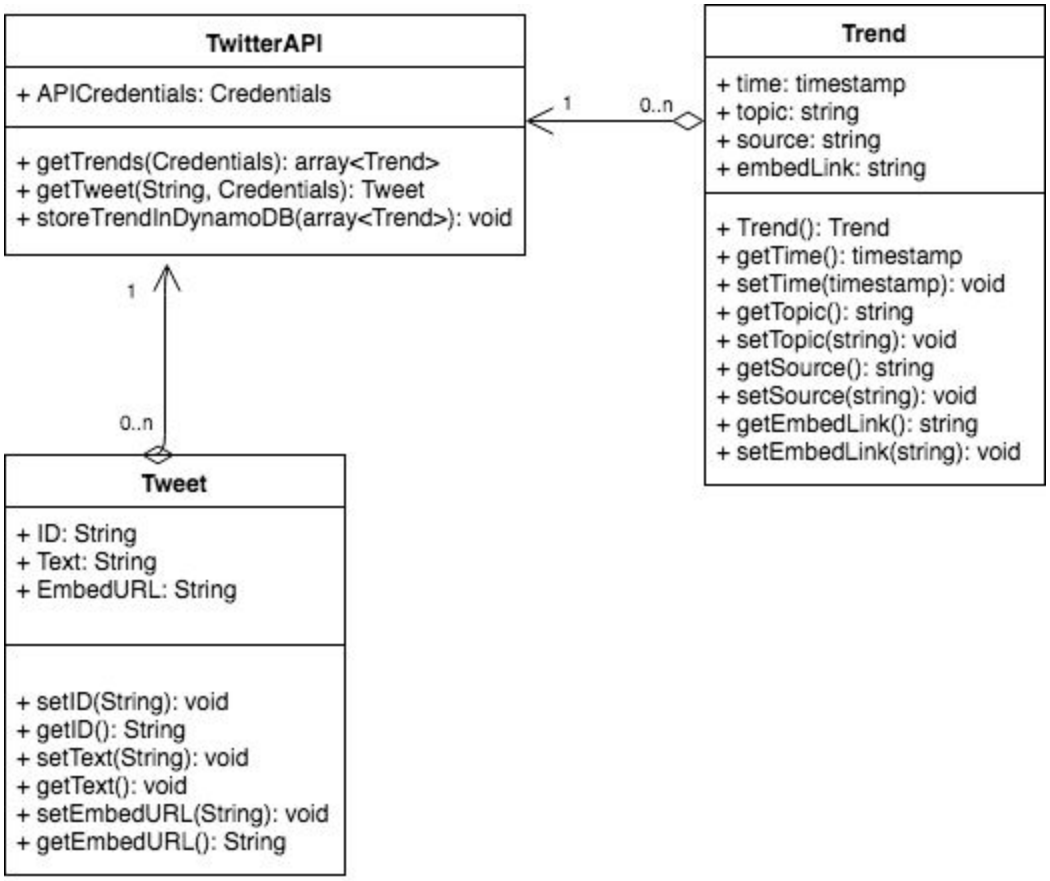
c. Trends Lambda Class Diagram

The trends lambda class describes lambda that is triggered every 10 minutes to pull trending topics from Twitter in order to establish the topics that will determine the posts pulled from other sources.



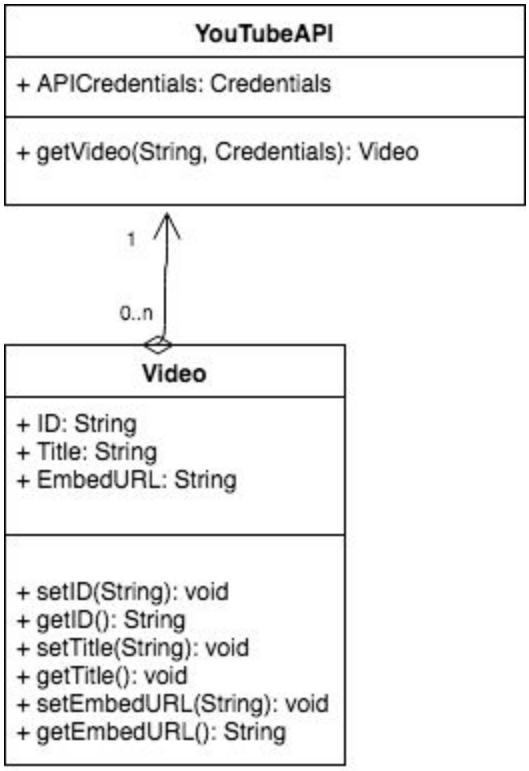
d. Information Lambda Class Diagram

The information lambda executes every time the trends database is updated with the top trends by the trends lambda. This class uses the trend class object to pull top posts from Twitter, Youtube, and Google News containing the trend “keywords” and stores them in the information database, to be accessed for later use.



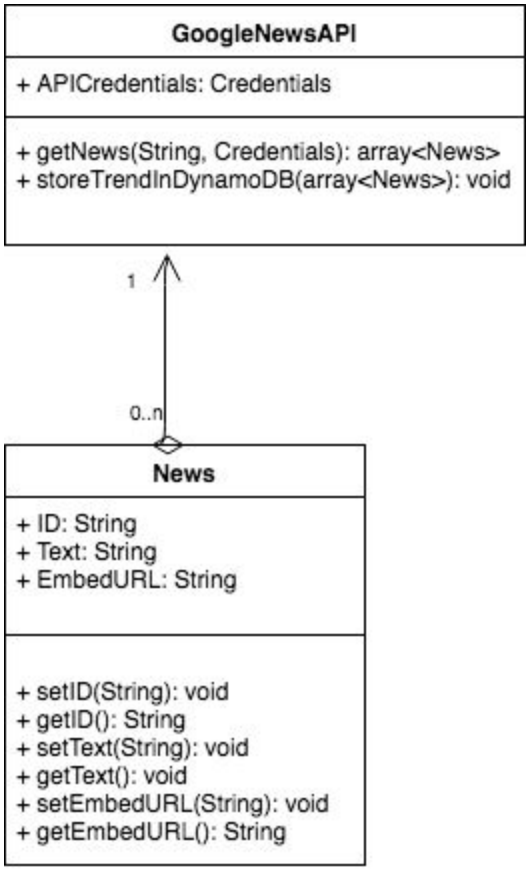
e. Twitter API Class Diagram

This class serves two purposes: the first is to pull current trending topics from Twitter using the Twitter API every 10 minutes (handled by the trends lambda), and the second is to then retrieve the top tweets for each of these topics (prompted by information lambda).



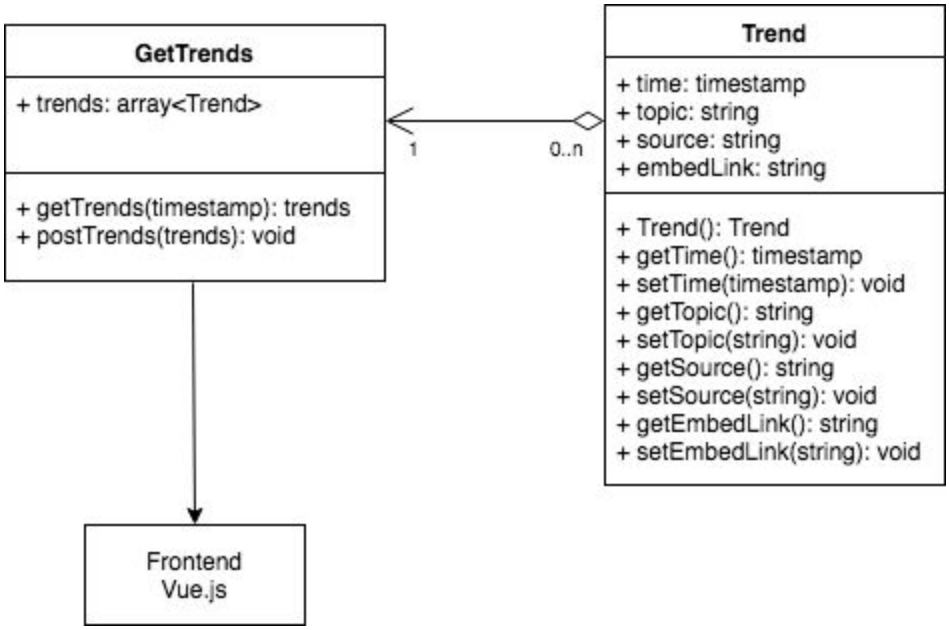
f. YouTube API Class Diagram

This class uses YouTube API's to pull the most popular videos on YouTube for each of the trending topics.



g. Google News API Class Diagram

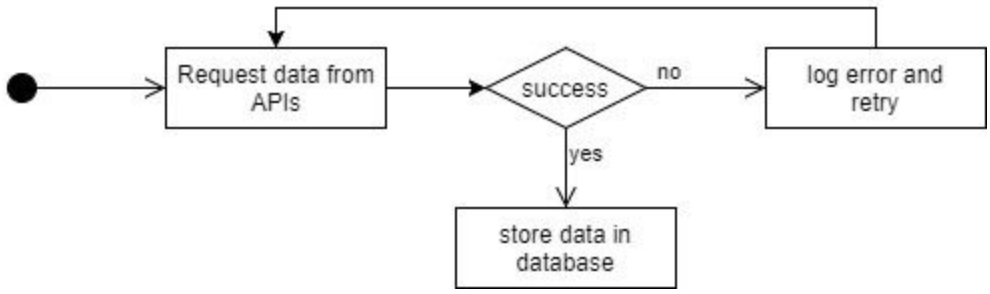
This class manages retrieving the top posts from Google News using Google News API for each trending topic.



h. Middleware Class Diagram

This class handles the communication between the front end of the application and the database in which the already retrieved posts are stored. When a user opens Frontpage, the middleware class will pull media from the information database and send them to be displayed within the application.

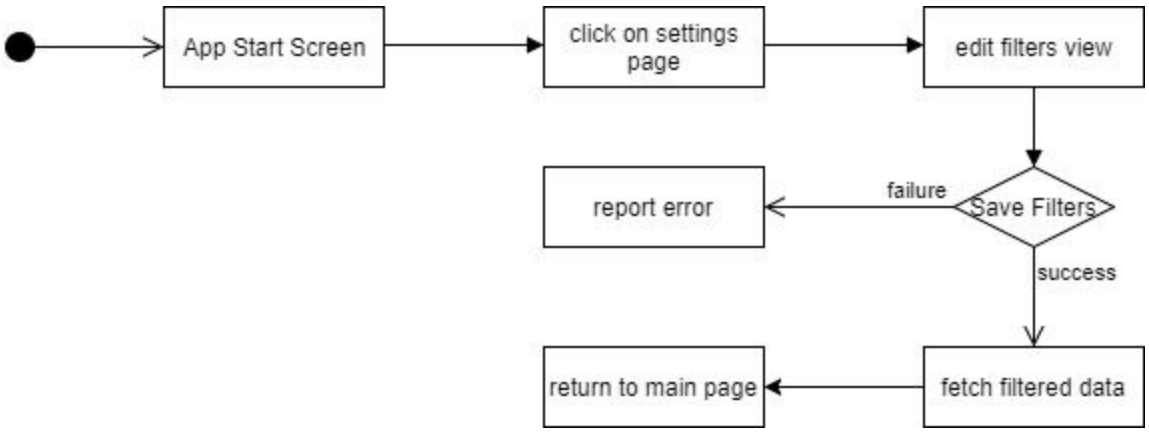
4. Activity Diagrams



a. Activity diagram for fetching data from Youtube, Twitter, etc.

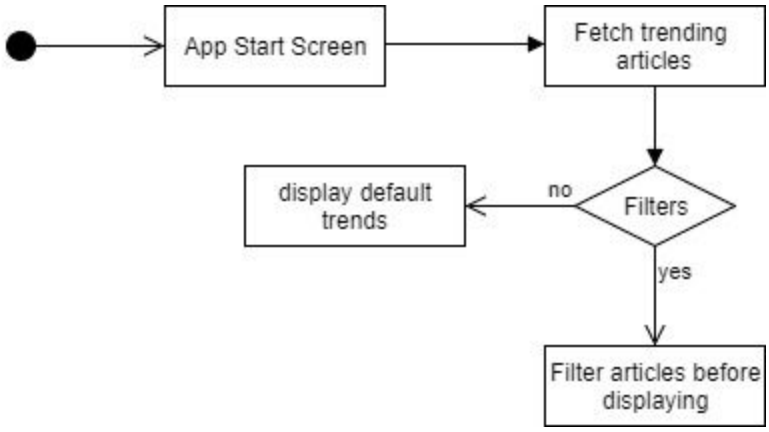
In order to find the trending news, our team will utilize the APIs from various sources. If we are successful in gathering the data, we will store it, otherwise retry the call and log the error. In

order to achieve the ability to filter by the date it was trending, links to this data will be stored in the database. If a user wants to see trends from previous days, this will allow for that.



b. Activity diagram for allowing the user to set filters

While some users will be fine with seeing the global trends, other users may want to filter it down to a specific topic, time, or other parameter. This activity diagram will allow the user to set these filters, and handle any errors that may occur. These filters will persist between sessions, so on success of saving the filters, they will be stored in the database.

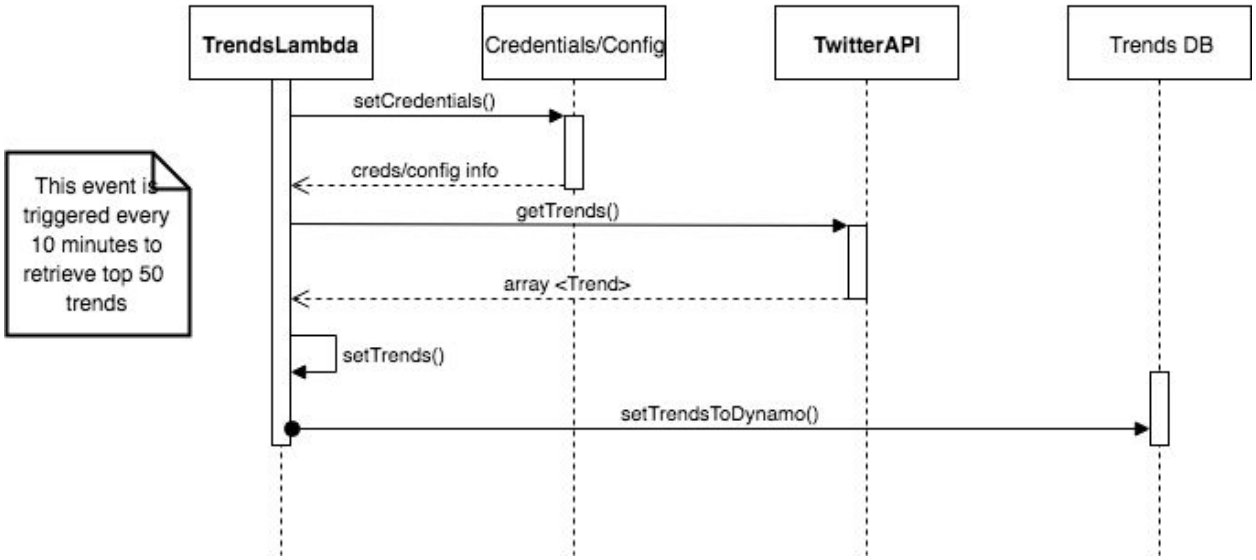


c. Activity diagram for displaying default trends, or filter preferences

If a user does not have any filters saved, the application will default to showing the trending news with no filters applied. However, if the user does have filters saved in the database, the application will default to showing the user a filtered view along with a notification that filtering is currently enabled.

5. Sequence Diagrams

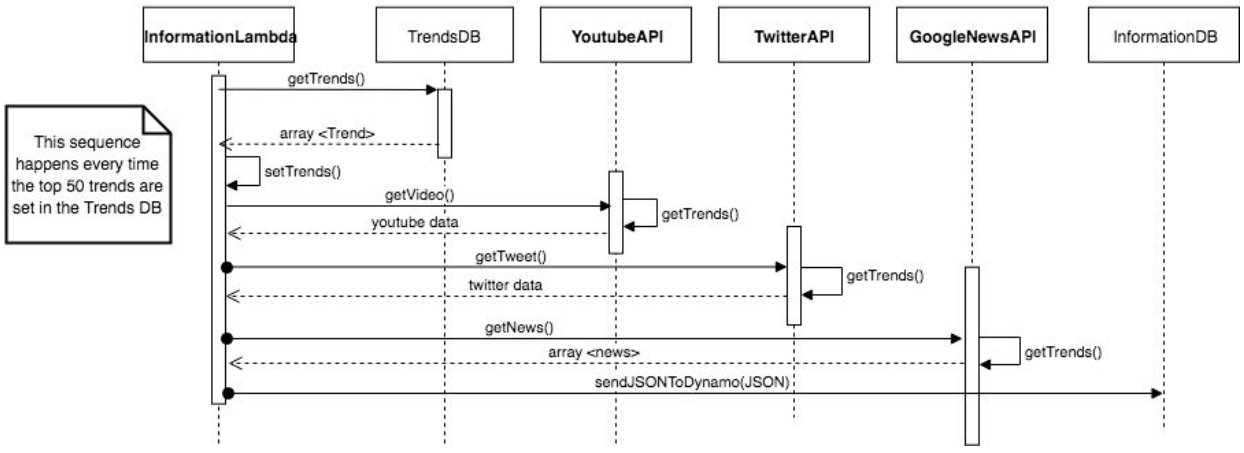
1. Fetching Top 50 Trending Topics



a. Trends Lambda Sequence Diagram

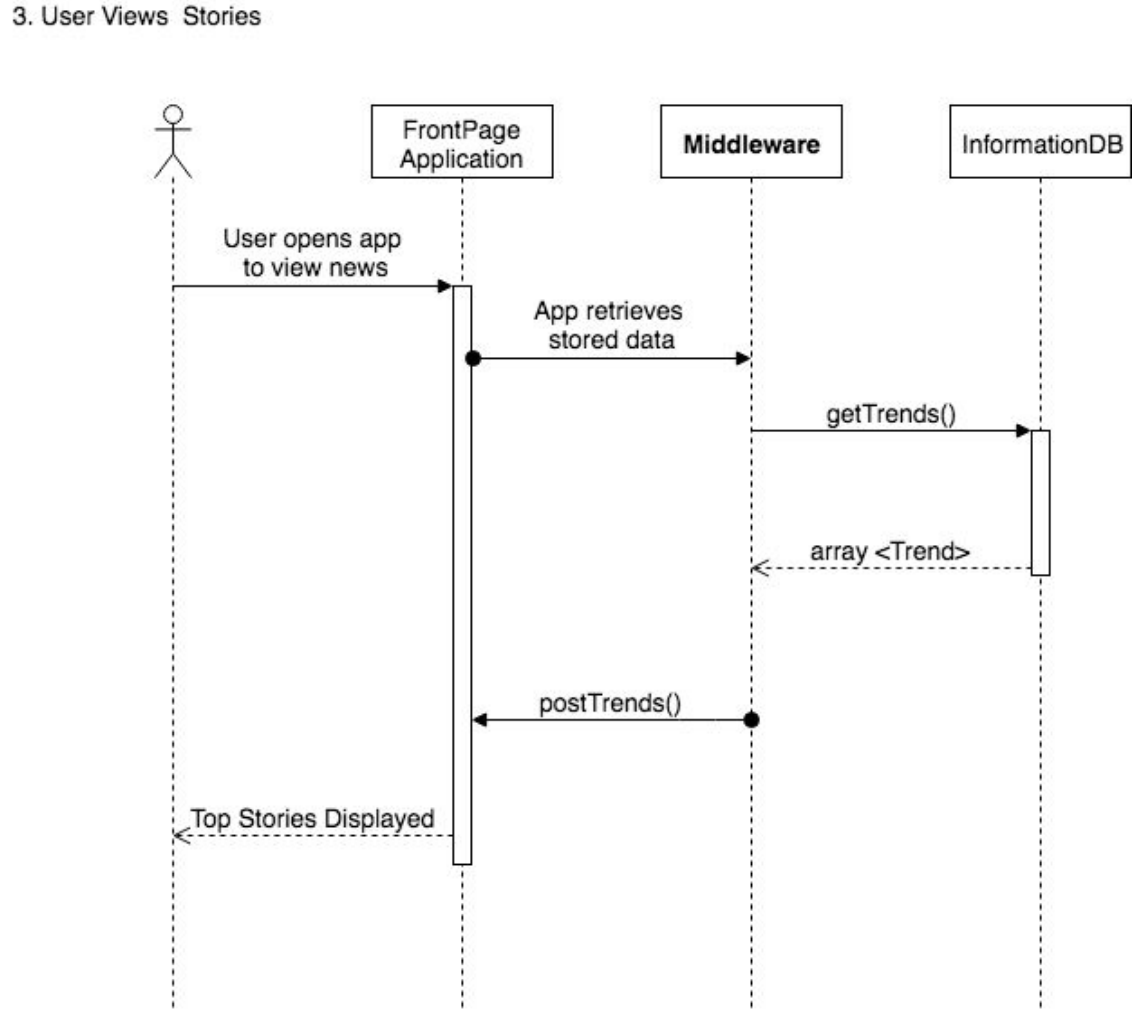
This sequence diagram demonstrates the behavior of the Trends Lambda class, triggered every 10 minutes, which is to pull the trending topics from Twitter to be stored as Trend objects in the Trends DB.

2. Retrieving Top Posts based on Trending Topics



b. Information Lambda Sequence Diagram

When the Trends DB is updated, the Information Lambda class retrieves the Trends stored in it and pulls top tweets, videos, and posts from each platform based on these popular trends, storing them as Trend objects in the Information DB.



c. Middleware Sequence Diagram

This diagram illustrates how the middleware class retrieves the top posts stored in the Information DB to be displayed to the user on the Frontpage application.

Implementation

1. General Information

We plan on hosting a lot of our services on Amazon Web Services. AWS provides many useful tools for accessing some of the APIs we're using (such as collecting data from Twitter streams). AWS is also really easy to work with when you have multiple people on a team, as anyone on our team will be able to easily remote in to any of the servers and databases once they're set up.

2. Data Collection and Storage

Because we'll be pulling data from multiple APIs we need to find a solution that allows us to easily get data and store it in a format where we can do near real time analysis. We currently plan on updating our data every 10 minutes. One thing that we agreed upon is that in the module for each API we would preprocess the data into a universal format, so no matter whether the initial data was from Reddit or Twitter it'd end up stored in the same format in the same database to reduce the complexity of our data analysis. Below are some of the APIs we plan on accessing and some considerations for how we will transform the data.

Reddit: The information we'll pull from Reddit will be the top 100 posts from the /r/news subreddit. We'll take the titles of these posts and store them in the database to be analyzed. We'll use an AWS EC2 instance to host the script that does this.

Twitter: Twitter is probably the most challenging API for us. We're currently planning on accessing the Twitter streaming API to get a constant load of tweets. We'll use an AWS Kinesis stream to access this API. We'll collect tweets constantly for 10 minutes and then take that data, find keywords in those tweets, and stick those keywords in the database for analysis. While we're collecting the tweets to be batch processed we'll stick all of the tweet contents into AWS DynamoDB. We're choosing DynamoDB because it can handle the high bandwidth that we'll need with all of the incoming tweets.

YouTube: YouTube has well-documented API that has been very useful for us in the implementation of our project. We will use the YouTube API to pull the "Most Popular" chart which lists the current trending videos on YouTube. From there we can access video metadata such as view count, ratings, tags, and more. YouTube also provides a search API, that we will use for the situation when users search particular top news stories.

3. Data Analysis

Once trending posts have been pulled, their metadata must be analyzed to decide the ranking for trend analysis. The user base for certain applications could improperly skew the trends, i.e. a Facebook post could be more popular than a tweet, but since Facebook has a larger user base, they must be weighted differently to determine which post is garnering more total views. Our application cannot update in real-time since we will be monitoring trends, a post cannot be trending until a certain amount of time has passed in order to analyze trend level accurately. Therefore we will be constantly pulling data, populating a database, and analyzing the metadata of posts to actively monitor the popularity of posts to ensure our application only updates with trending data.

Every 10 minutes we'll analyze the data put in the collection database from the data collection phase. Once the data is analyzed and our top trends are found we'll put those trends into a database along with social media posts about those trends that we will query the APIs we're using once again to get.

4. Website

Our frontend should be simple. Our web page will display the top 10 or so trends from our trending database. Each post will have the title of the trend along with several stories from YouTube, Twitter, etc. embedded in it. Users will be able to click on these stories to view them on the original websites if they so choose. At the top of the web page there will be a search bar. Users can enter the name of a trend and if the same trend is found in our trending database we'll display just those trends.