Debias

The Agile Alliance

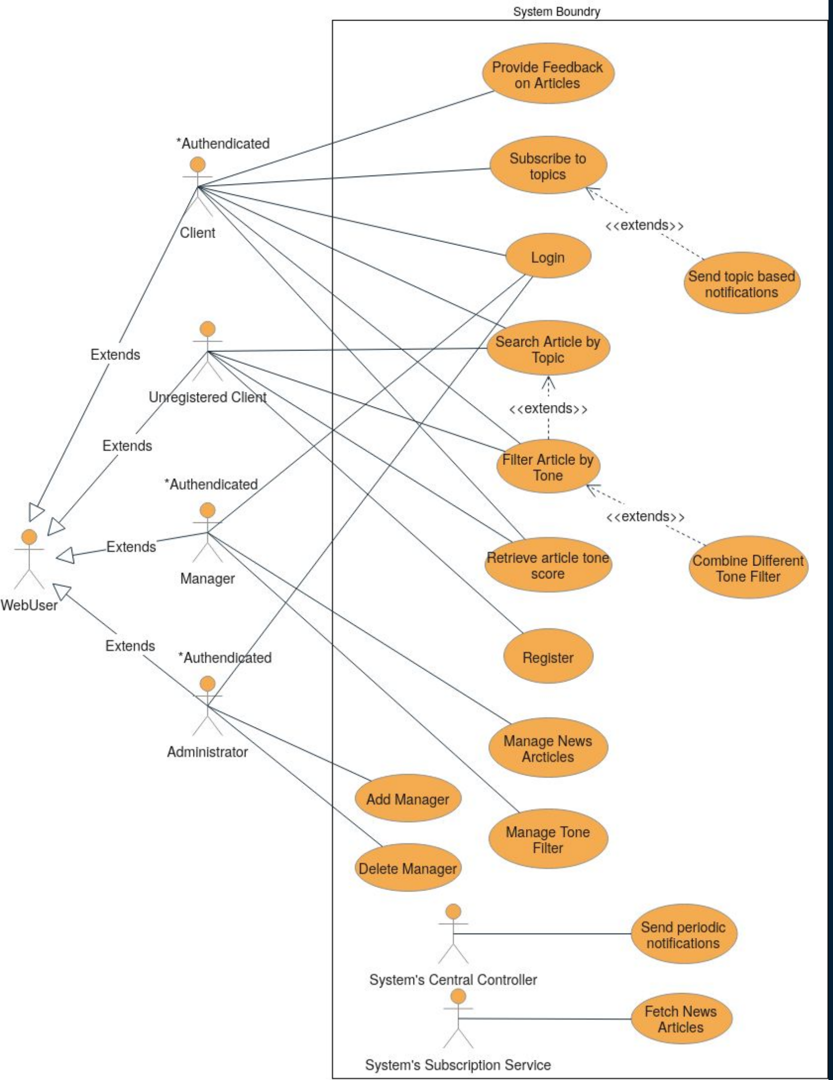
Members

Siddharth Bhardwaj
Michael Brog
Ruizhe Liu
Arshdeep Saini
Hengchao Xiang

Presentation Structure

What is Debias?

- News aggregation tool.
- Aims to solve the difficulty in finding articles with different viewpoints.
- Empower the Customer to obtain balanced perspective on news topics, which may be riddled with biased rhetoric.

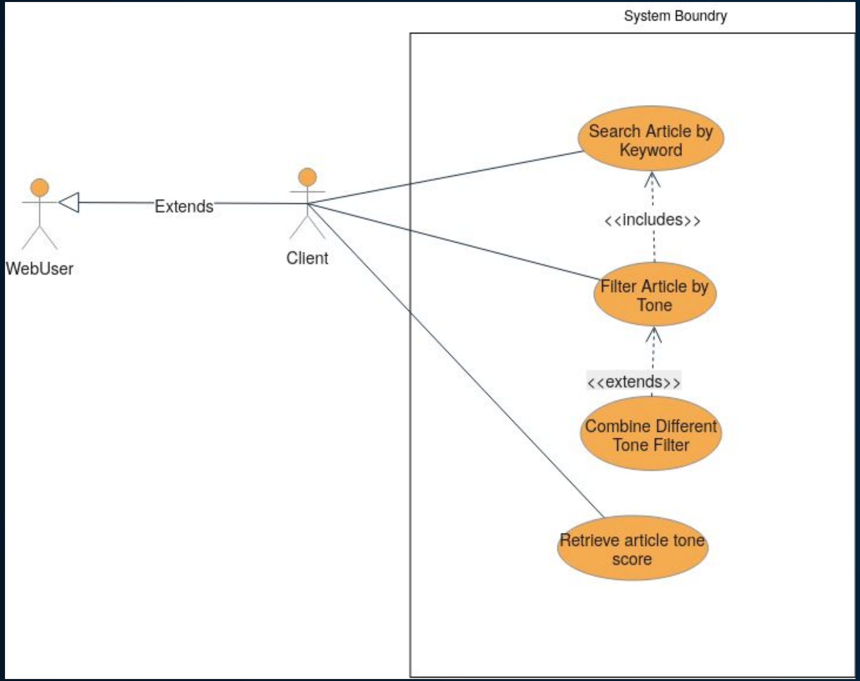


Use Case Model of Debias

Based on set of all use cases specifying the complete functionality of the system.

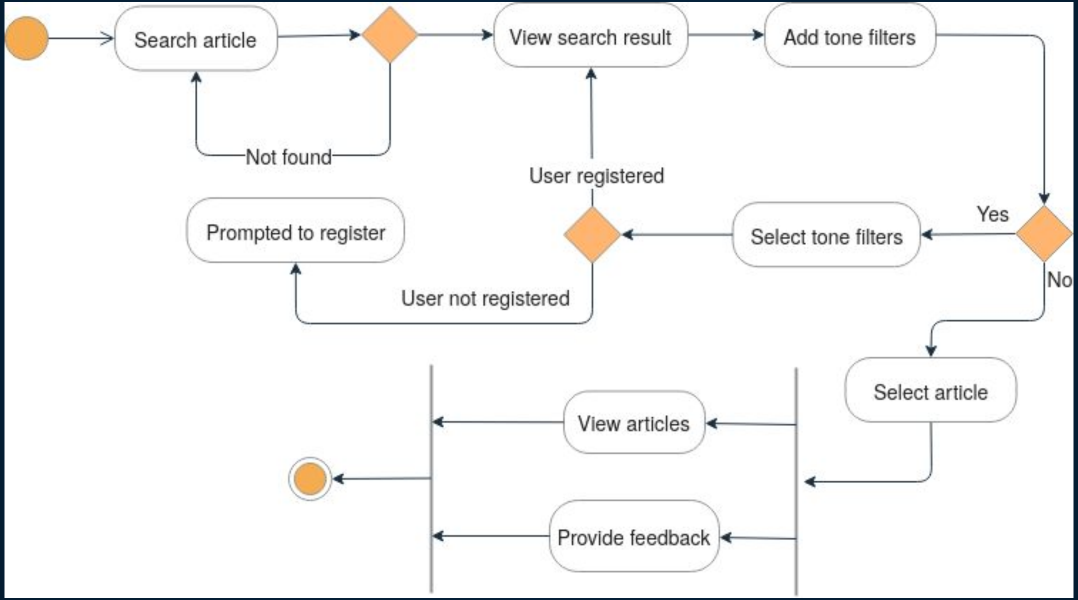
Detailed information on requirement elicitation and use case modeling can be found from: https://viloil.github.io/EECS4314-Team-Project/

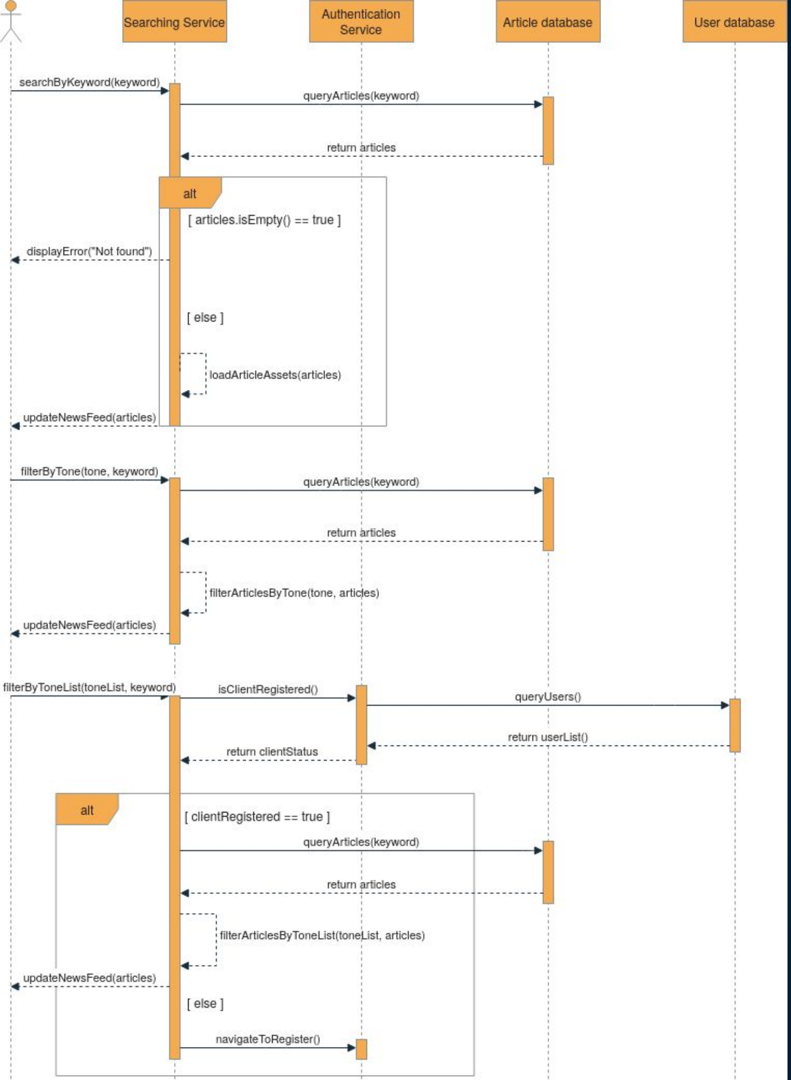
Use Case Model implemented in Debias's Functional Prototype


System Boundry
WebUser
Extends
Client
Search Article by Keyword
<<includes>>
Filter Article by Tone
<<extends>>
Combine Different Tone Filter
Retrieve article tone score

Use case where a client uses keywords to search for articles (which are aggregated and presented to the client). Thereafter, the client can filter the presented articles by tone- combining different tone filters. Furthermore, can view the tone score associated with each article, displayed to the client by the system.

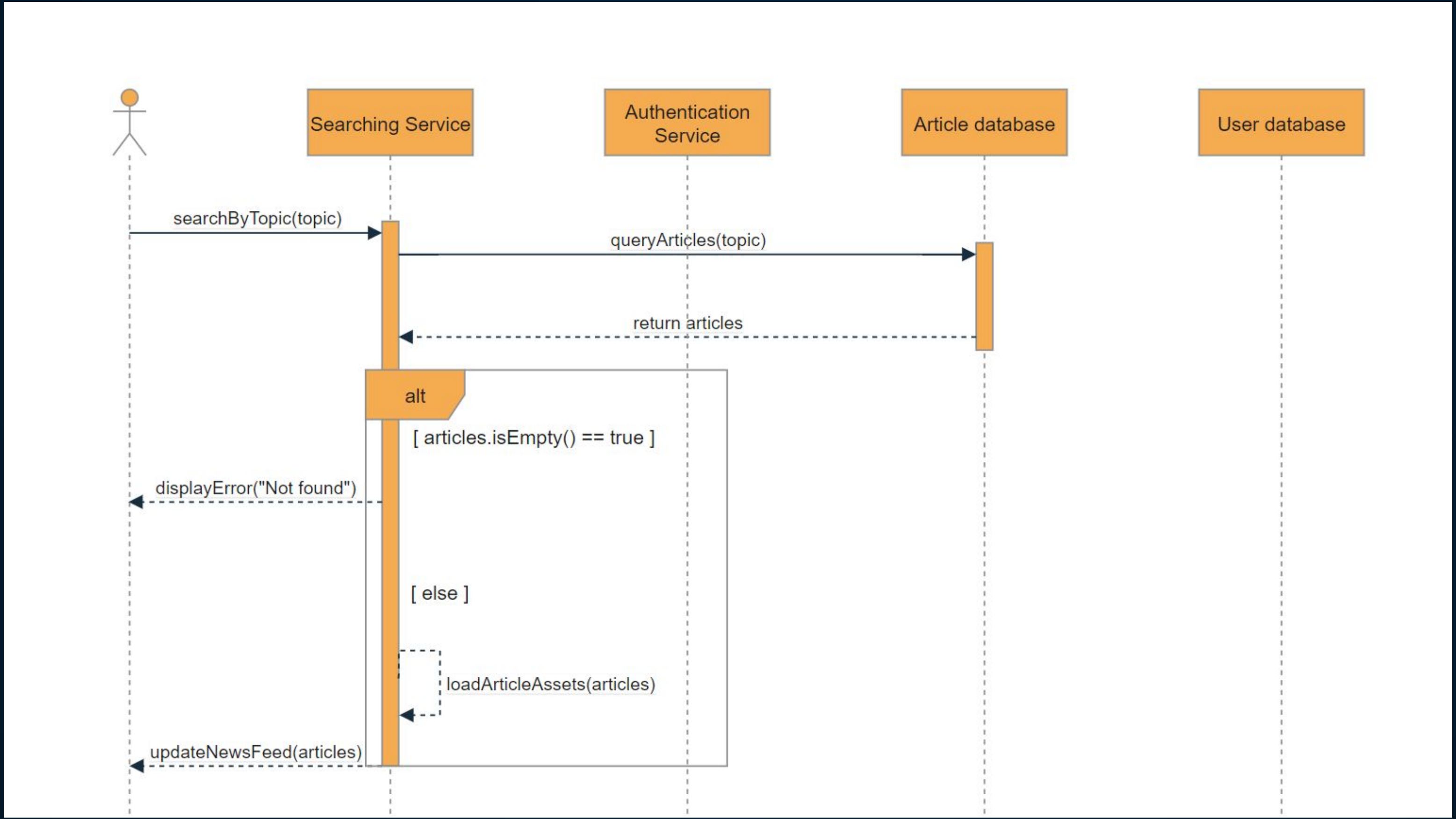
Activity Diagram of the Use Case implemented for our MVP



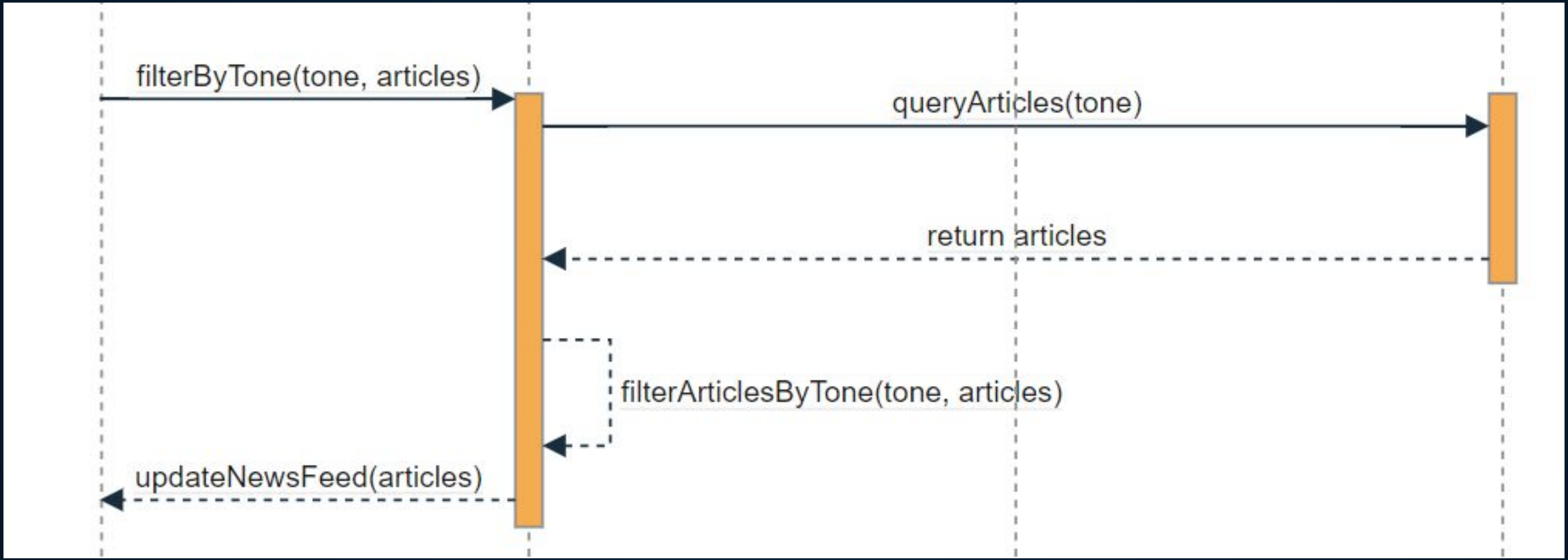


Sequence diagram of the Use Case implemented and for our MVP

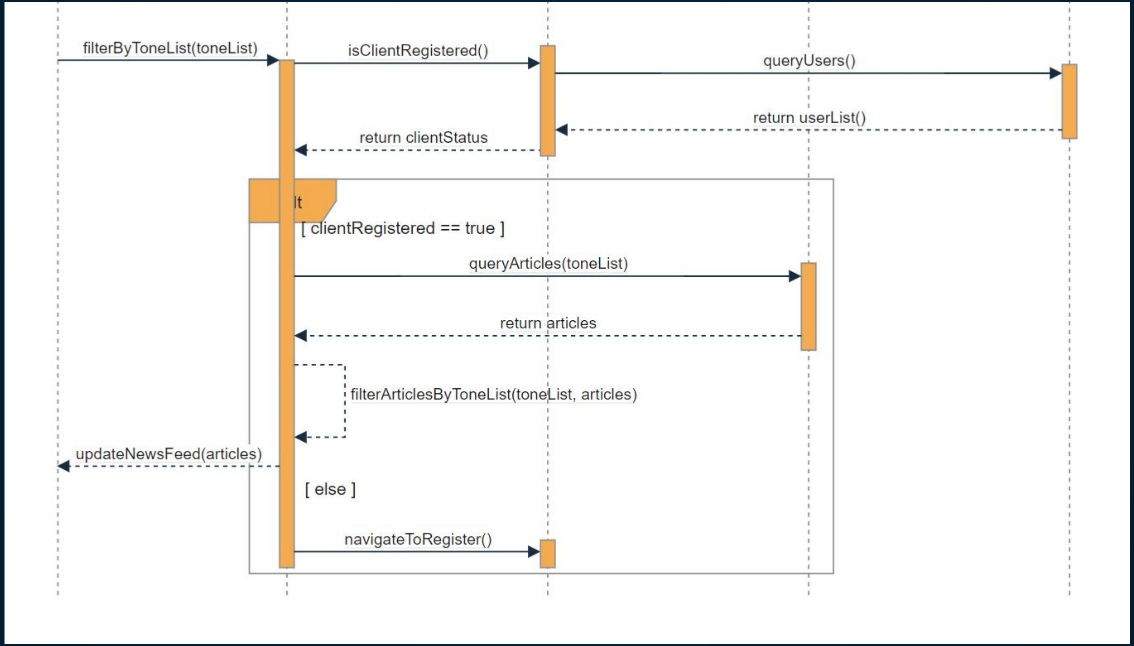
Sequence diagram of the Use Case implemented and for our MVP



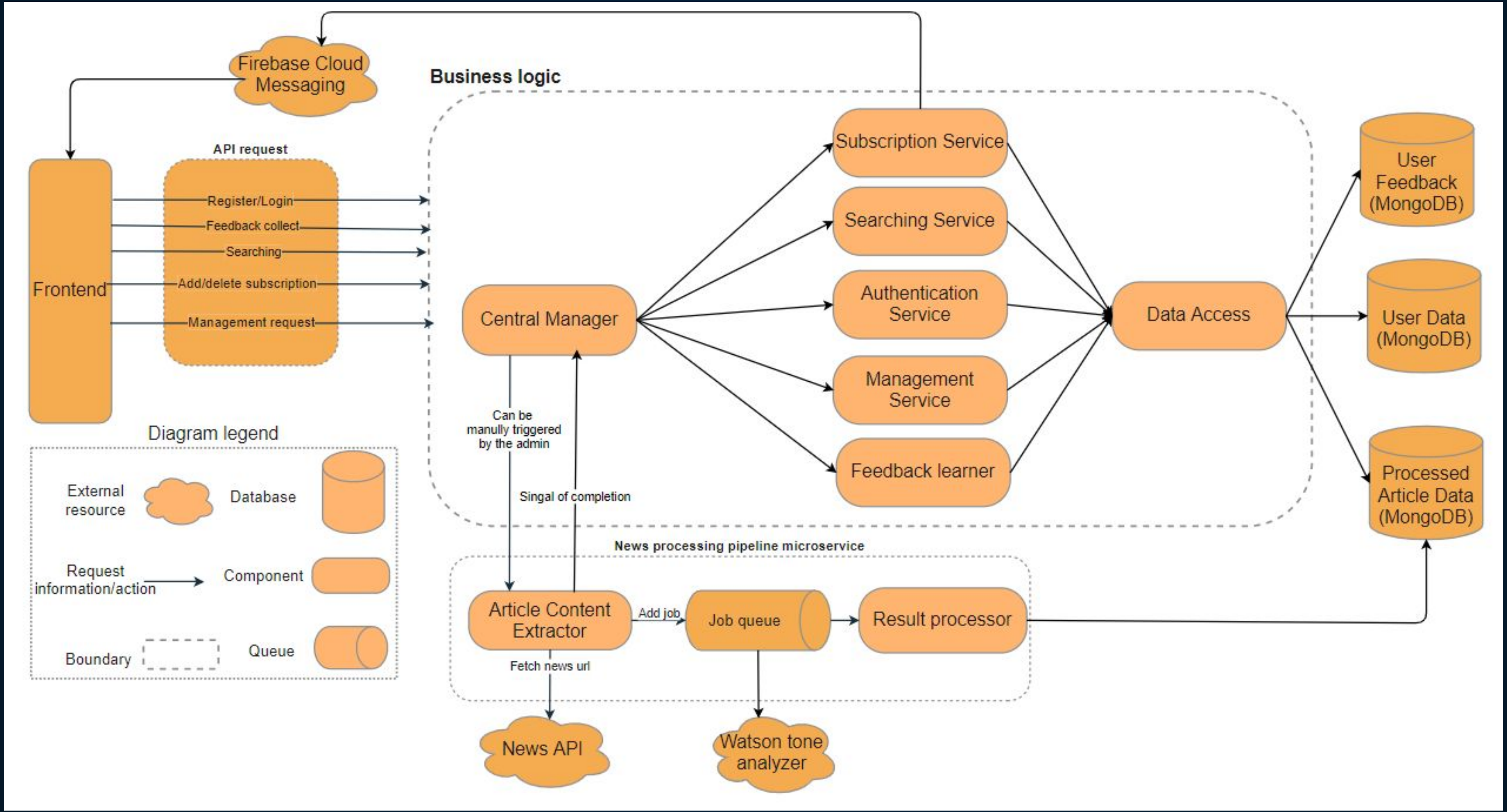
Sequence diagram of the Use Case implemented and for our MVP


filterByTone(tone, articles)
queryArticles(tone)
return articles
filterArticlesByTone(tone, articles)
updateNewsFeed(articles)

Sequence diagram of the Use Case implemented and for our MVP



Concrete System Architecture



Conceptual vs. Concrete

- A new data access component is added as a subsystem to handle the low level connection with the underlying database and expose generic CRUD interface to other subsystems.

- Article content database is removed to reduce the overhead of database storing/reading.

- Isolate the News Processing Pipeline as an independent microservice.

Implementation Alternatives

MongoDB vs. DynamoDB

- Limited query flexibility if using DynamoDB
- More data type supported by nature through MongoDB
- Not heavily rely on AWS stack

FCM vs. Python+Redis+Websocket

- FCM provides functionality of pub/sub based on the topic
- Complex Implementation
- Not “reinventing the wheel”

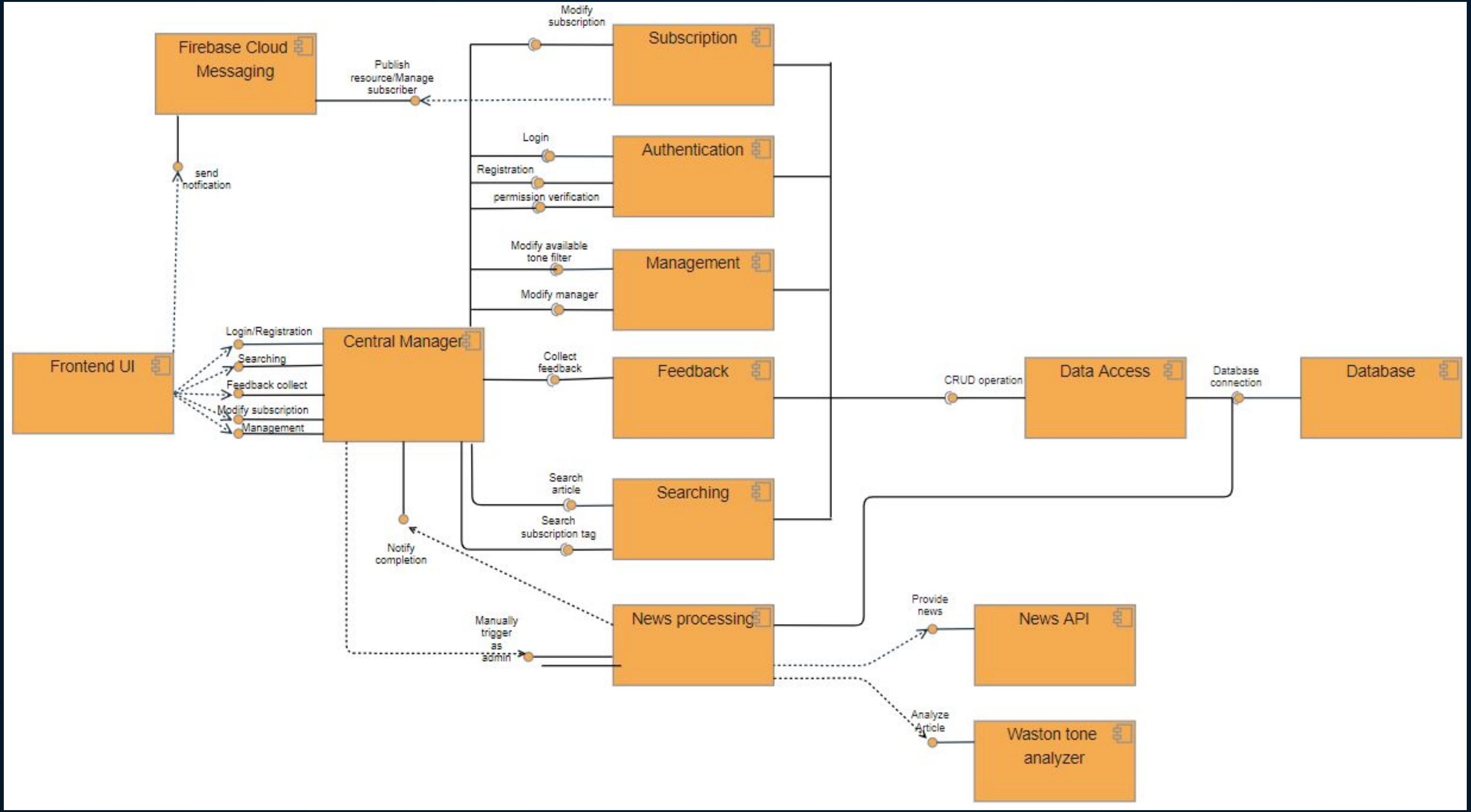
Python vs. Other Languages

- Simplicity
- Needs of data scripting

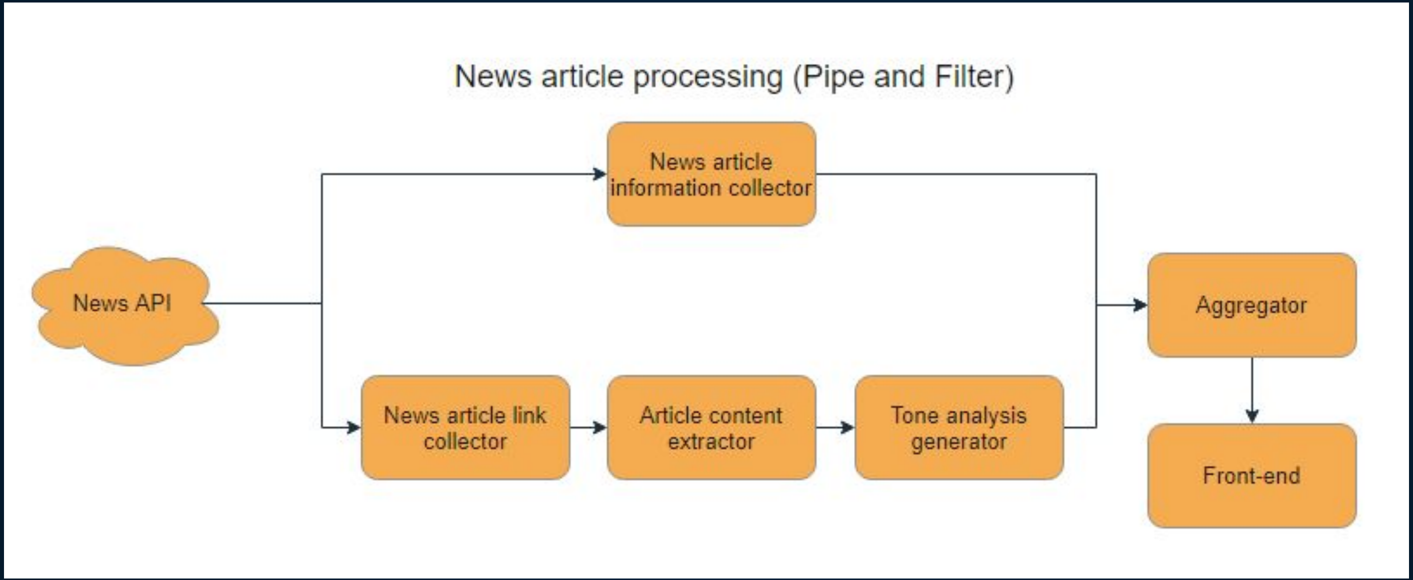
Watson vs NLTK

- Watson provides more robust solution using AI, however cost is per API call and could inflate as product grows
- NLTK (Natural Language Toolkit) while a much cheaper solution, would need to have a dataset created to train a model and would require a create deal of dev work

Interactions between subsystems

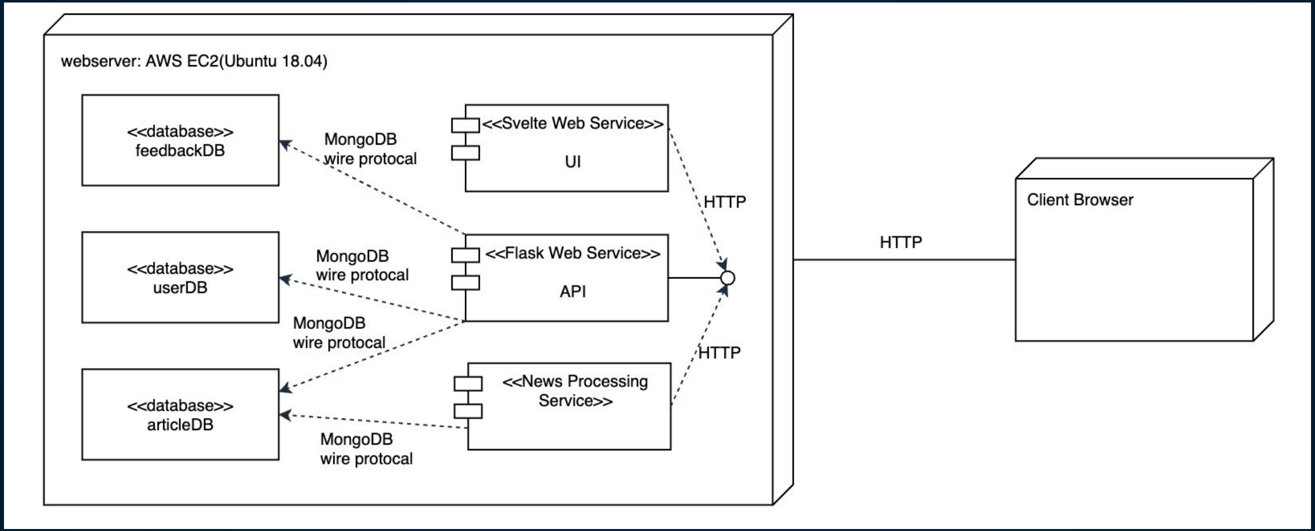


News Processing Pipeline


News article processing (Pipe and Filter)
News API
News article information collector
News article link collector
Article content extractor
Tone analysis generator
Aggregator
Front-end

Deployment and Concurrency Control

Deployment



Deployment and Concurrency Control

Concurrency control

- Flask handles multiple user's requests
- Queue is utilized to process the collected news articles
- Concurrency in MongoDB

Quality Requirements and Debias?

Performance	Limit api requests (throughput) to 100 requests/hour for each client
Integrity	User authentication and session management for user data integrity
Privacy	Use local storage for search history
Availability	- Continuous Integration/ Continuous Development (CI/CD) - Limit service downtime by 1 hour per every 1,1000 hours of continuous
Portability	Progressive Web App support (PWA) -> Android, iOS and desktop installable (Lighthouse)
Accessibility	Monitor Lighthouse benchmark score

Lessons learned

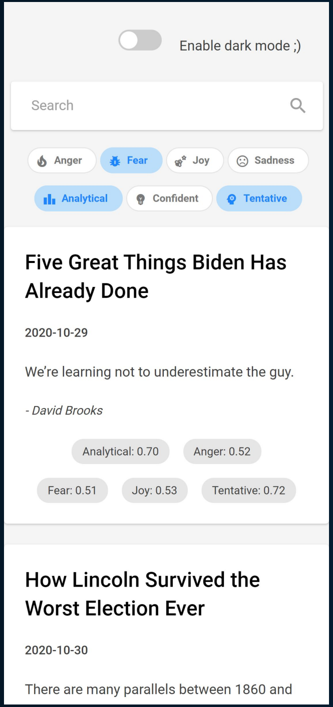
During our conceptual architecture phase, we had:

- planned for it to be monolithic (3 tier architecture)
- avoided considering complex communication between different subsystems
- focused on horizontal scaling of each tier

During our implementation phase (concrete architecture), we learned:

- that individual subsystems scale better independently
- that some subsystems scale better as microservices
- that we can deploy our application better using containers

Functional Prototype



Enable dark mode ;)

Search

Anger Fear Joy Sadness

Analytical Confident Tentative

Five Great Things Biden Has Already Done

2020-10-29

We're learning not to underestimate the guy.

- David Brooks

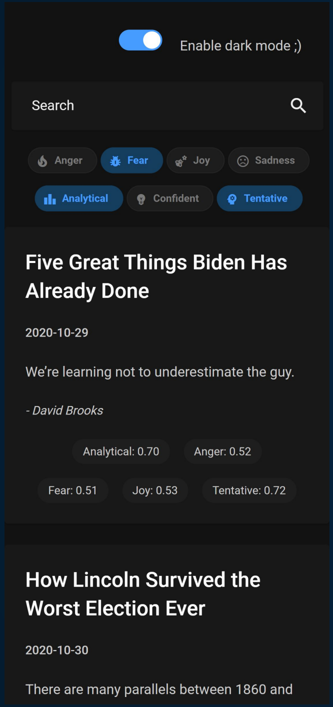
Analytical: 0.70 Anger: 0.52

Fear: 0.51 Joy: 0.53 Tentative: 0.72

How Lincoln Survived the Worst Election Ever

2020-10-30

There are many parallels between 1860 and



Enable dark mode ;)

Search

Anger Fear Joy Sadness

Analytical Confident Tentative

Five Great Things Biden Has Already Done

2020-10-29

We're learning not to underestimate the guy.

- David Brooks

Analytical: 0.70 Anger: 0.52

Fear: 0.51 Joy: 0.53 Tentative: 0.72

How Lincoln Survived the Worst Election Ever

2020-10-30

There are many parallels between 1860 and

THANK YOU!

Live Prototype Link: http://3.85.125.119:5000

Feel free to ask any questions.

https://viloil.github.io/EECS4314-Team-Project/