Debias

The Agile Alliance

Members

Siddharth Bhardwaj
Michael Brog
Ruizhe Liu
Arshdeep Saini
Hengchao Xiang

Presentation Structure

Introduction

What is Debias?
Enhancement Proposal.
Stakeholders.
Use case & Sequence diagrams.

01

System Architecture v0.2

Implementation Alternatives.
Concrete architecture overview.
Modifications made to system and subsystems.

02

Testing and Concurrency Control

Overview of systems self-healing testing strategy, and implemented system concurrency control.

03

04

Quality attributes

Effects of the new enhancement, and division of team responsibilities.

05

Functional Prototype

Illustration of functional prototype, along with URL to service.

06

Lessons learned

Lessons our team learnt during this project phase.

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.

Enhancement Proposal for Debias

What is it?

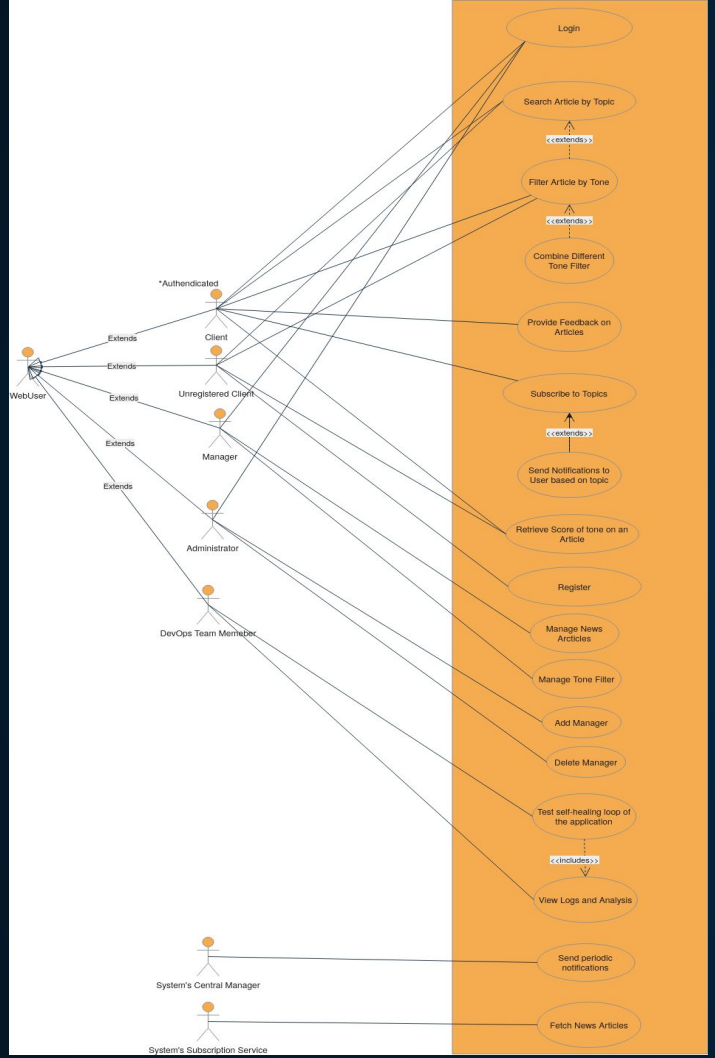
- Log basic information about daily operations, including request, exception, and some completion flags of the pipe process.
- Allow debugging system log information when the system is running into error state, using analysis of log messages provided by our system.
- Use the log analysis result to improve the fault tolerance of the system, and establish a certain degree of self healing loop.
- Feature is accessible on live website, under the “Logs and Analysis” view- primarily intended for a DevOps team members

Stakeholders:

- Professor.Litou and Team
- Our team
- DevOps team member(s)
- All users of our service

Note: As seen in subsequent sequence diagram below, we implemented this feature in the form of the “logging and analysis service.”

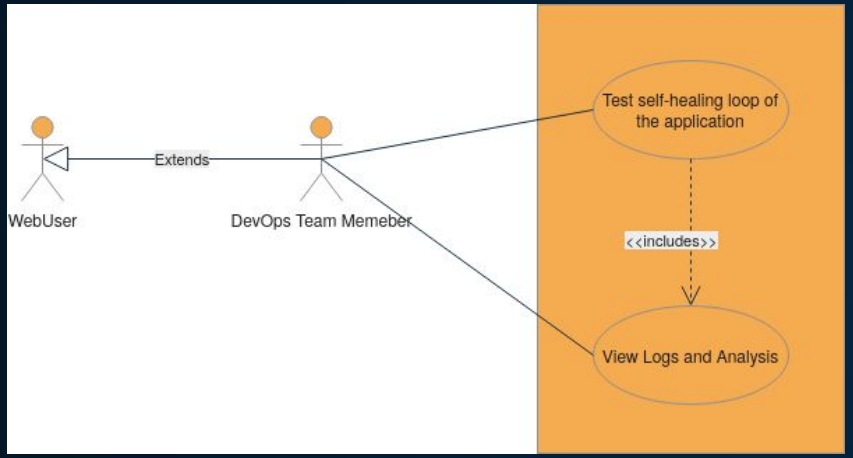
Use Case Model of Debias



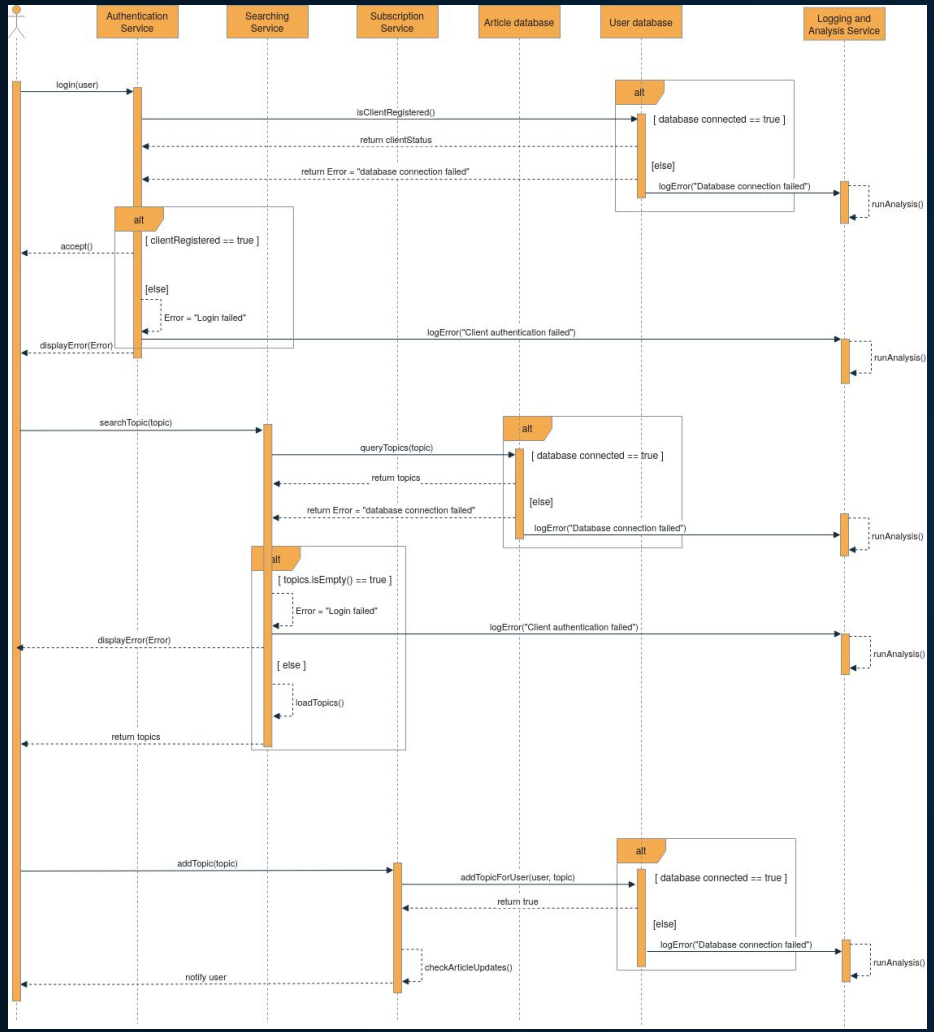
Based on set of all use cases specifying the complete functionality of the system.

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

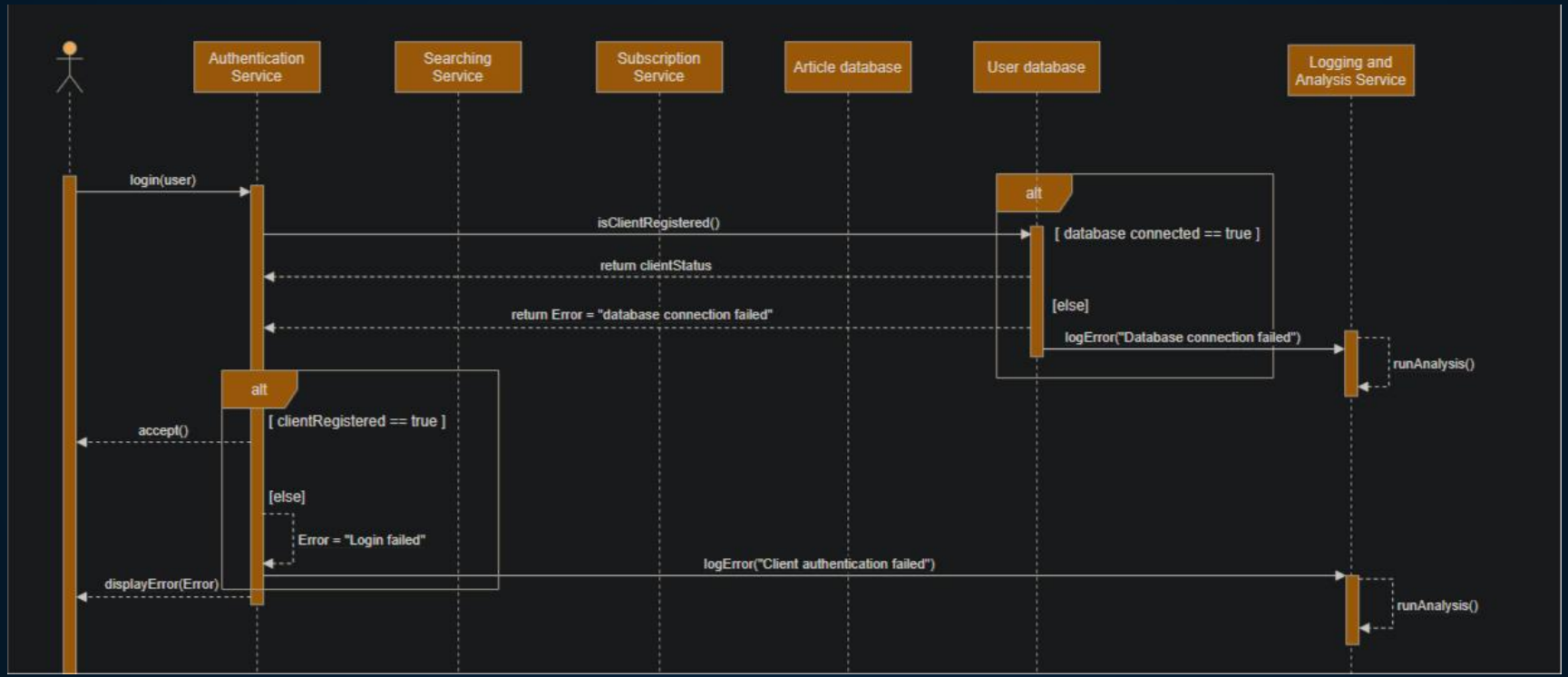


Through viewing logs and performing analysis, a DevOps team-member will be able to test the self-healing functionality of Debias. (Note: Fault was injected ahead of time, for the purposes of this assignment, so DevOps team-member can simply view that log data collected and perform analysis). Viewing logs use-case will be achieved through implementation- adding an additional interface to the website, which is accessible by the DevOps team member. Also, perform analysis use-case will be achieved through this same interface accessible by the DevOps team-member, where an explanation of each error/anomaly logged is briefly explained.

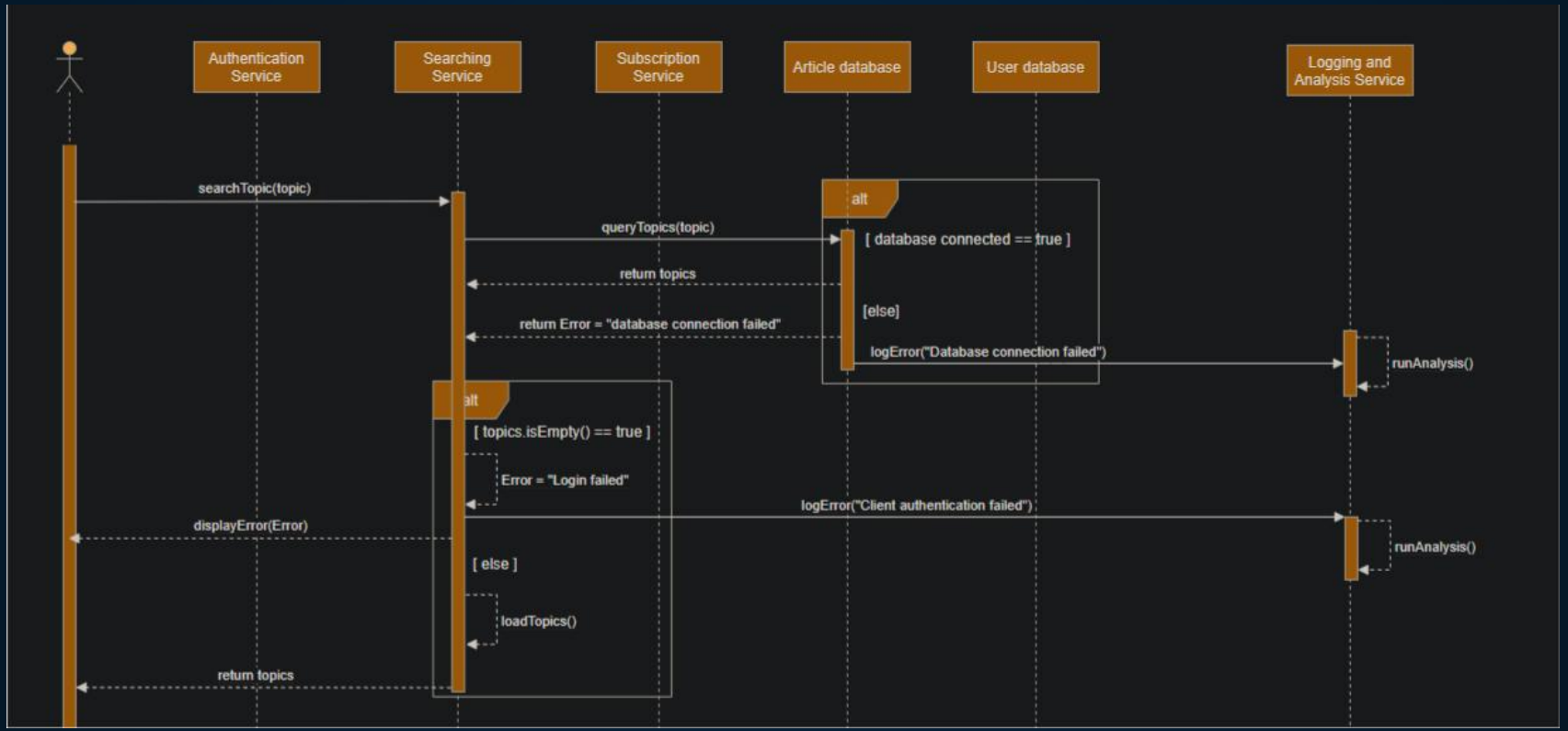


(Updated) Sequence diagram of the Use Case implemented and for our MVP

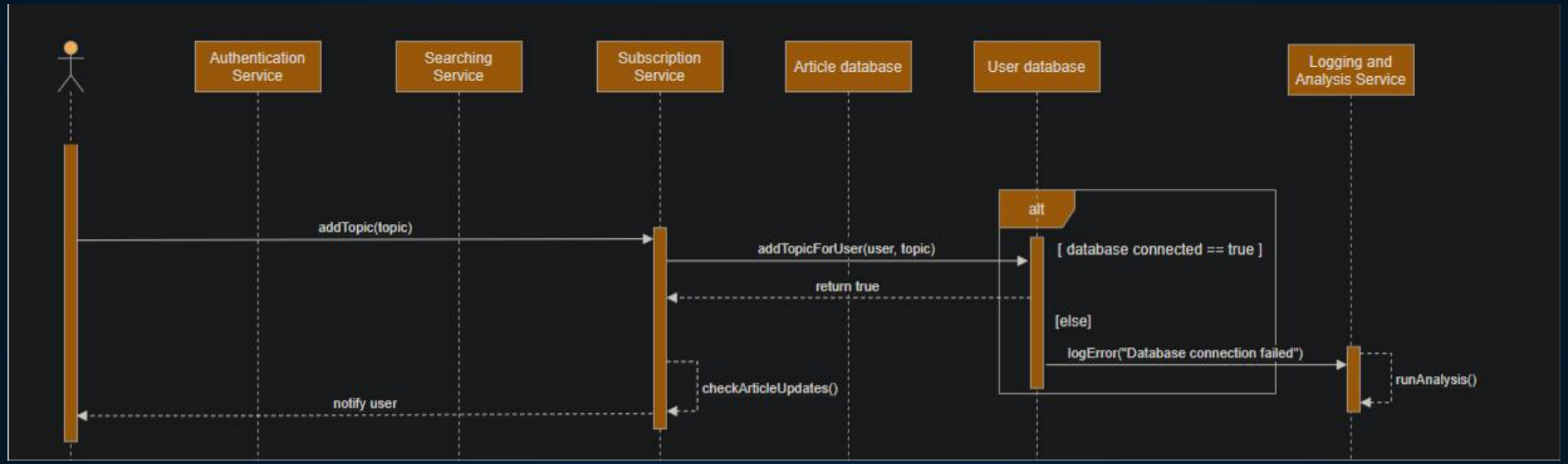
Sequence diagram of the Use Case implemented and for our MVP



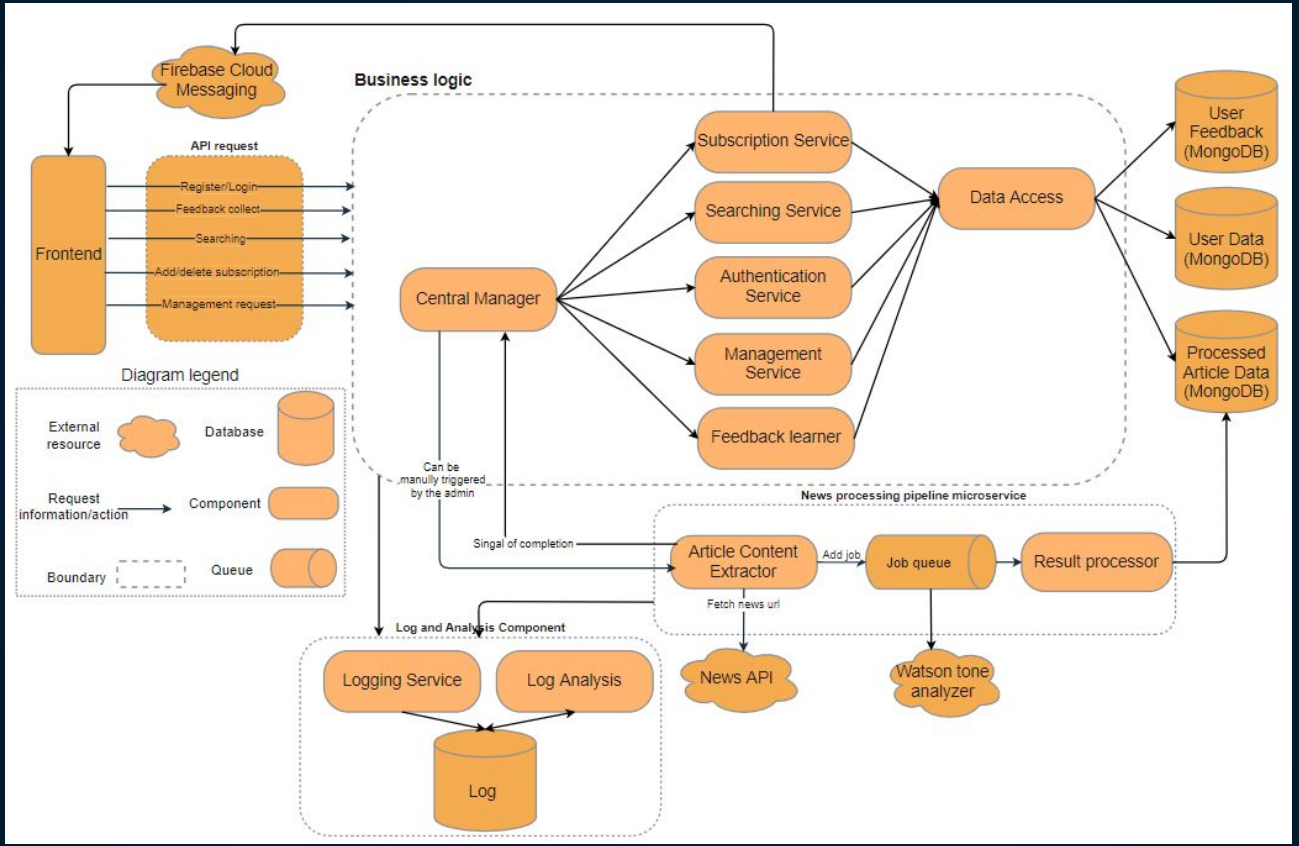
Sequence diagram of the Use Case implemented and for our MVP



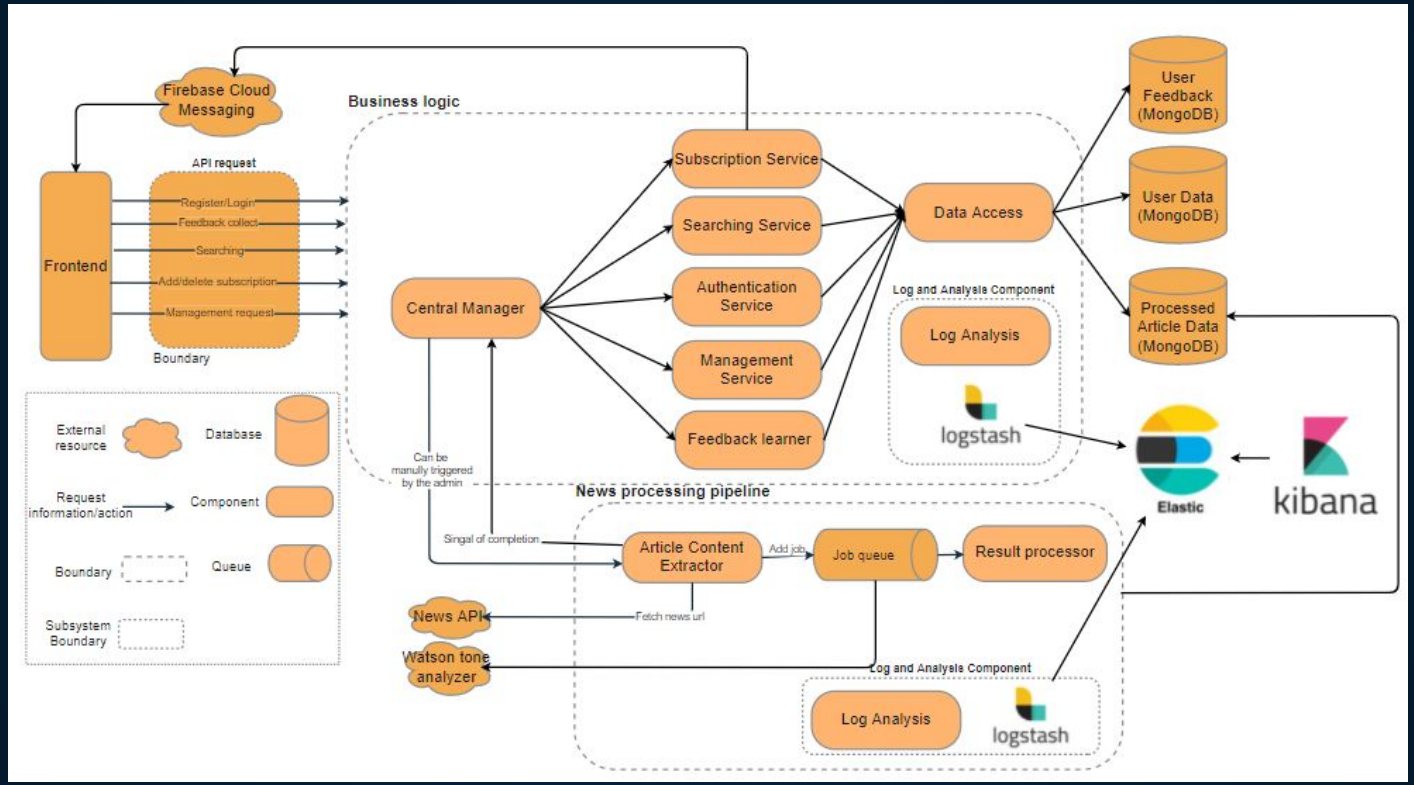
Sequence diagram of the Use Case implemented and for our MVP



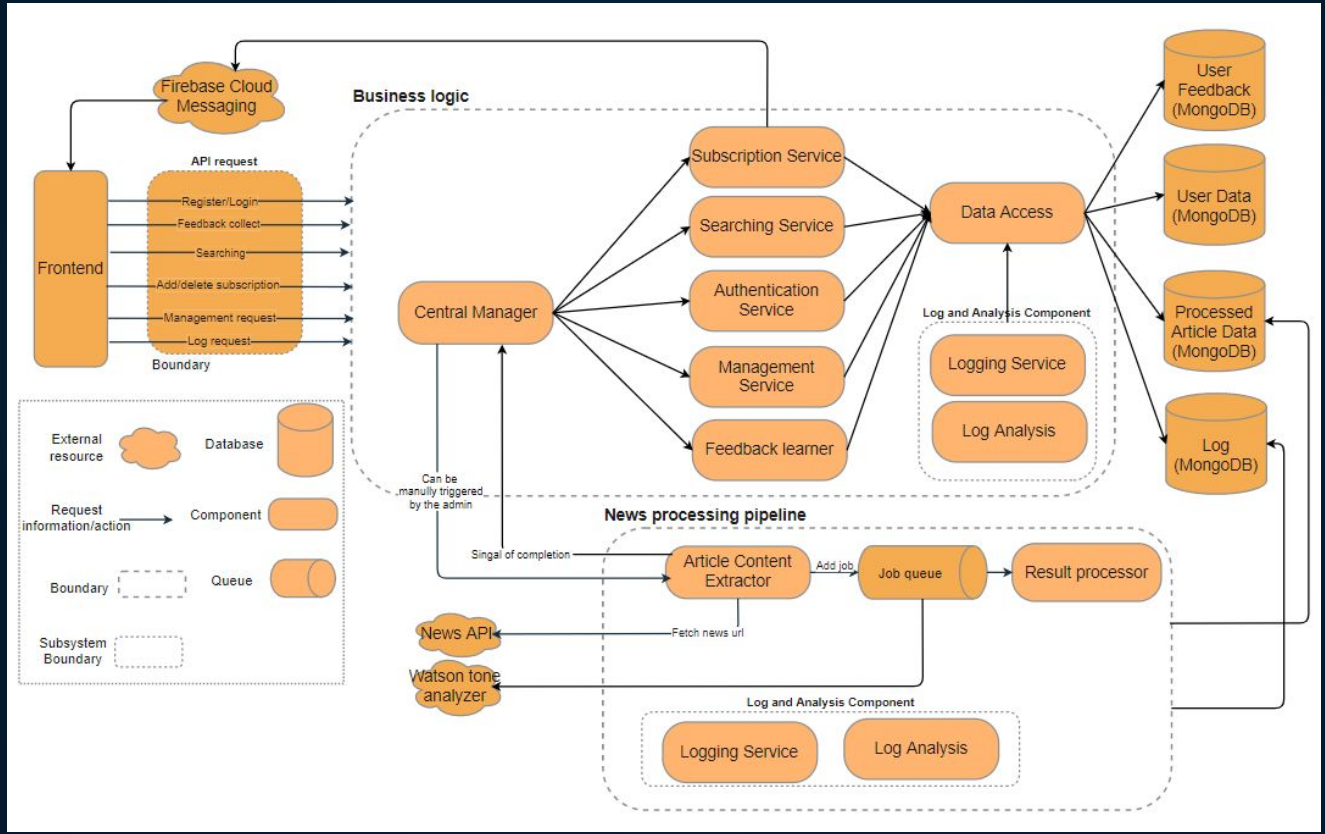
Conceptual Architecture of Enhancement



Implementation Alternatives: ELK



Alternatives: Utilize existing component



Comparison: ELK

Pros

- Provides a mature process for aggregating logs
  - Logstash: retrieving and outputting.
  - Elasticsearch: collecting, analyzing, and storing data
  - Kibana: visualization
- Good scalability, no-schema log storage, and mature concercy control.
- No additional cost for development of front-end log UI

Cons

- ELK needs three parts to be deployed and maintained separately, it will increase the burden of maintenance and more possible failure points
- Deploying logstash separately on each app server will consume more computation resources, risk of reducing performance
- Still need extra effort to build our own log analysis component in order to realize application-specific log analysis

Comparison: Utilize current component

Pros

- No need to deploy more services
- Time cost and complexity of this solution are relatively lower
- Remains the possibility of evolution

Cons

- Need to develop frontend UI for log visualization
- Need to handle the concurrency issue ourselves
- More communication between the backend and the database.

Testing and Concurrency Control

Testing

- Make sure the existing features work as expected after the enhancement
- Test logging and analysis feature

Testing and Concurrency Control

Concurrency Control

- Flask handles multiple user's requests
- 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

Effects of the enhancement

Improvements

- Strengthen fault tolerance and reliability
- Easier debugging and testing
- Front-end UI for DevOps makes fault detection easier
- Improved system maintainability

Effects of the enhancement

Potential risks

- Higher database usage (Read/Write operations)
- Impact on backend performance
- Higher dependence on concurrency
- Increase in code base complexity

Division of responsibilities

Roles

- Cloud Engineer
- Backend Developer
- Full Stack Developer
- Front End Developer

Lessons learned

During our implementation architecture phase, we:

- Originally planned for the product to be monolithic (3 tier architecture)
- Found that individual subsystems scale better independently
- Found that some subsystems scale better as microservices
- Found that we can deploy our application better using containers

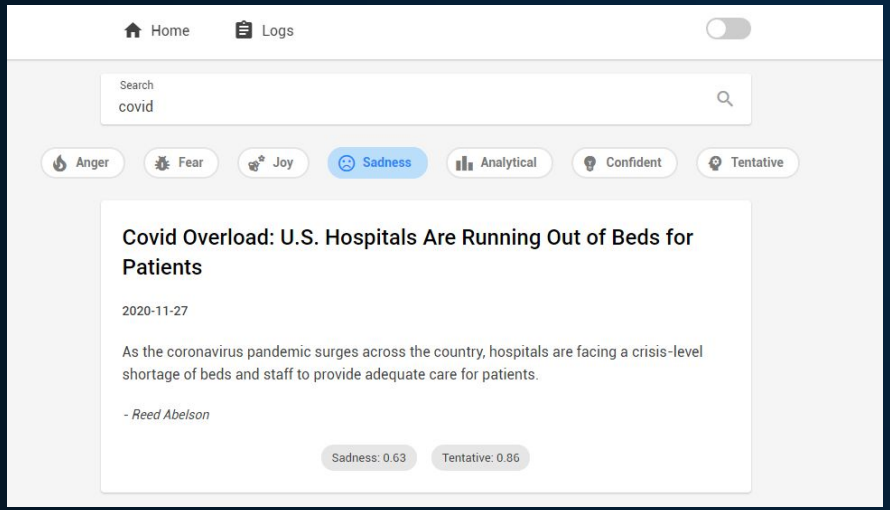
During our enhancement phase, we learned:

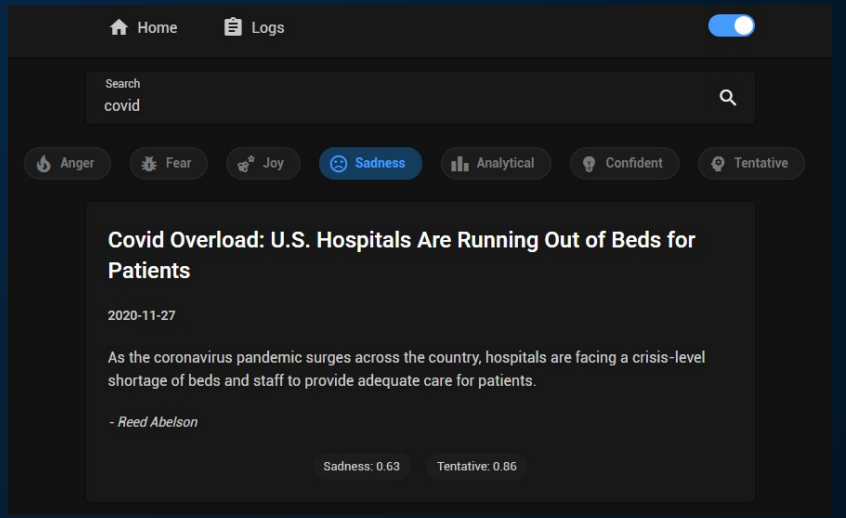
- using microservices allows for less downtime, and less painful maintenance
- microservices allow for more sophisticated testing
- additional work is required during setup to ensure proper communication between microservices

Functional Prototype

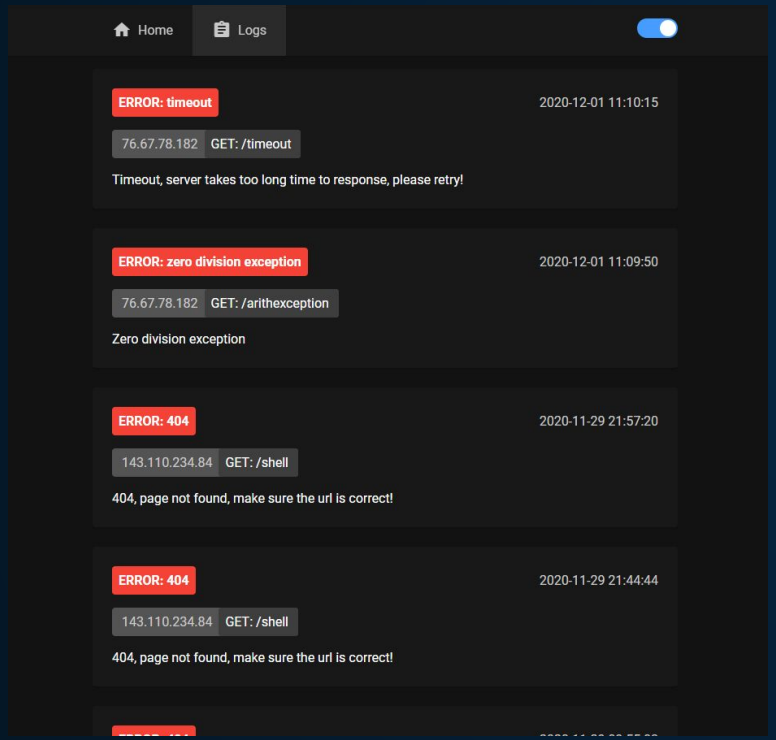
Debias your news

- Search for popular news articles by topic or title using the search feature
- Filter out popular news articles by one or more tones
- Try it out: http://18.206.238.196:5000



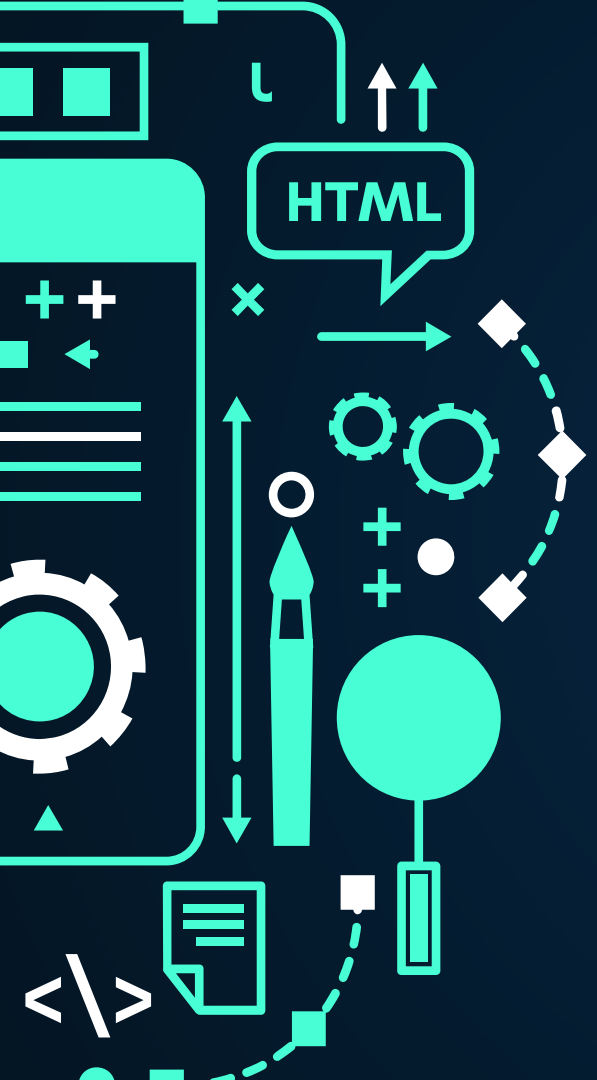


Functional Prototype



Live demo link:

http://18.206.238.196:5000/logs



THANK YOU!

Feel free to ask any questions.

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