CS5604 (Information Retrieval) F2020
Front-end (FE) Team Project

Yusheng Cao Reza Mazloom Makanjuola Ogunleye

December 17, 2020

Instructed by Professor Edward A. Fox
Assisted by Satvik Chekuri and the
Digital Library Research Laboratory

Virginia Polytechnic Institute and State University
Blacksburg, VA 24061

Abstract

With the demand and abundance of information increasing over the last two decades, generations of computer scientists are trying to improve the whole process of information searching, retrieval, and storage. With the diversification of the information sources, users' demand for various requirements of the data has also changed drastically both in terms of usability and performance. Due to the growth of the source material and requirements, correctly sorting, filtering, and storing has given rise to many new challenges in the field. With the help of all four other teams on this project, we are developing an information retrieval, analysis, and storage system to retrieve data from Virginia Tech's Electronic Thesis and Dissertation (ETD), Twitter, and Web Page archives. We seek to provide an appropriate data research and management tool to the users to access specific data. The system will also give certain users the authority to manage and add more data to the system. This project's deliverable will be combined with four others to produce a system usable by Virginia Tech's library system to manage, maintain, and analyze these archives. This report attempts to introduce the system components and design decisions regarding how it has been planned and implemented.

Our team has developed a front end web interface that is able to search, retrieve, and manage three important content collection types: ETDs, tweets, and web pages. The interface incorporates a simple hierarchical user permission system, providing different levels of access to its users. In order to facilitate the workflow with other teams, we have containerized this system and made it available on the Virginia Tech cloud server. The system also makes use of a dynamic workflow system using a KnowledgeGraph and Apache Airflow, providing high levels of functional extensibility to the system. This allows curators and researchers to use containerised services for crawling, pre-processing, parsing, and indexing their custom corpora and collections that are available to them in the system.

Contents

List of Figures

List of Tables

Acknowledgements

We would like to express our gratitude towards Dr. Edward A. Fox for his mentorship, in addition to the help from the members of the Digital Library Research Laboratory at Virginia Tech that helped us bring this project to fruition. We would also like to give our thanks to all other participating teams in the CS5604 Fall 2020 class – INT, ETD, TWT, and WP – as well as the Fall 2019 class's FEK team that provided the foundation we worked on. We appreciate the two agencies supporting this project:

- Institute of Museum and Library Services, through grant LG-37-19-0078-19
- National Science Foundation, through grants CMMI-1638207 and OAC-1835660

Finally we also appreciate all of the external help and feedback we received from the Virginia Tech CS department, Library, IT staff, and the inter-university researchers that helped us along the way.

1 Overview

The Electronic Thesis and Dissertation (ETD), tweet, and webpage information retrieval and analysis system is a web application that will provide full-text and metadata search of Virginia Tech's ETD, Twitter posts, and webpage archives. Aside from the basic search, result display, ranking and curation of content collections, used by researchers, the system will also provide an internal logging system for further analysis.

Our team is in charge of developing a usable web-based front-end for all potential users, creating an opportunity to interact with the exploration and analysis of three datasets: the archived ETD dataset, tweets from Twitter, and indexed web pages. The major task for our team is using Flask and other GUI applications to visualize the data provided from the three content teams in addition to providing client-side filtering. The basic UI would be similar to the general search engine interface. The engine requires the user to choose which database they want to access and to then input their search term(s). The UI would then show the resulting data based on the feedback from the analysis and indexing back-end developed by other teams.

The FE Team will make use of the previous FEK team's project as a foundation, update the ETD processing, and add new features to support the tweets and web pages. In addition to the base functionalities, we also support multi-access user login as well as faceted search, and provide functionality accordingly (Figure 1).

During the IR1 (Interim Report 1) period, we have thoroughly analyzed all the code left from the previous project. We researched all the tools and APIs we are going to work on, and we used some sample data from other teams to create a prototype.

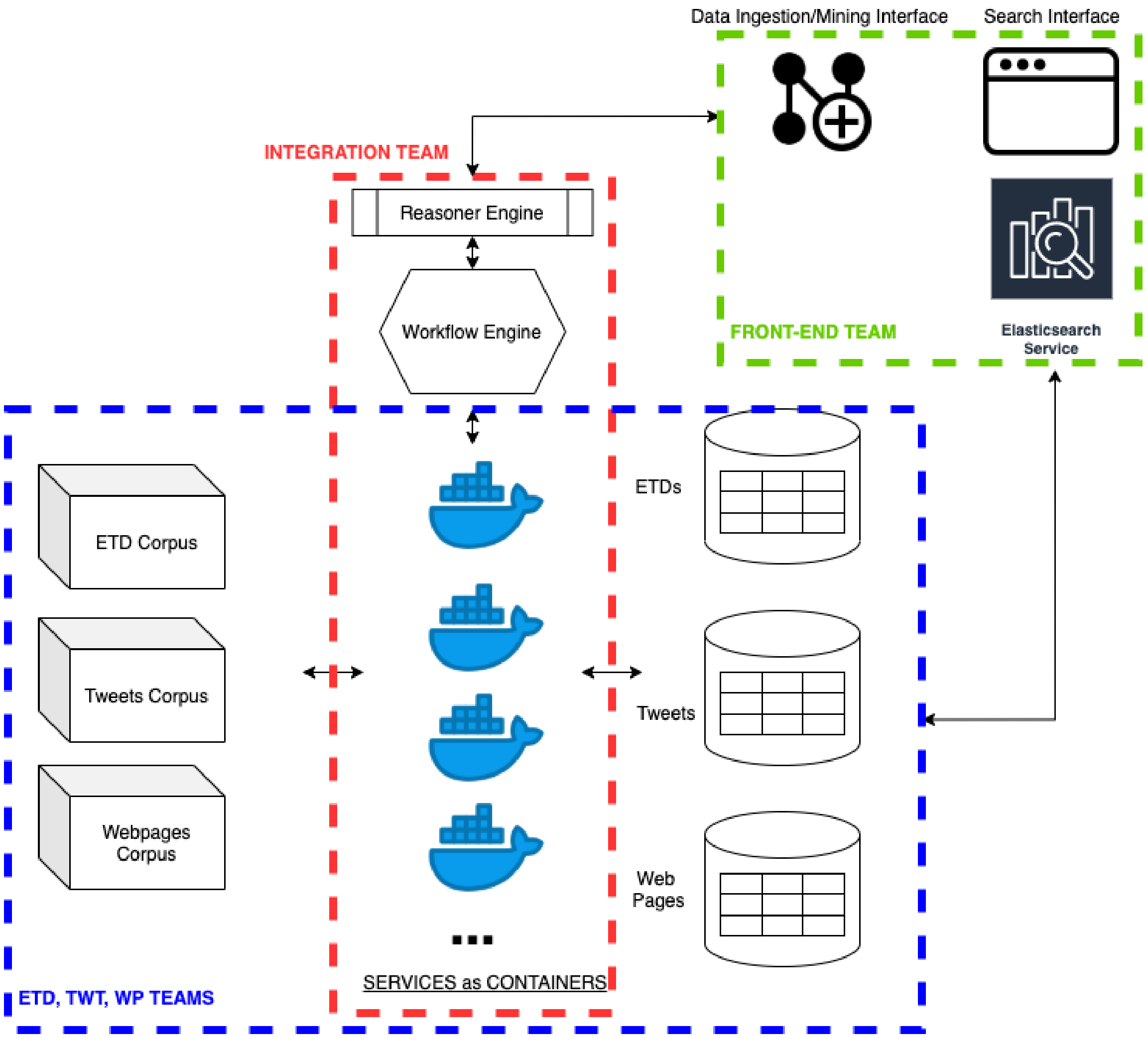
By the end of the IR2 period, we had built a working front-end prototype which is able to search metadata. By analyzing and modifying the given sample data from the TWT and ETD teams, we populated and used a local Elasticsearch server to prototype the entire workflow. Most of the data was clean but some modifications were necessary such as adding indices for widely used fields, and some formating problems that required fixing before our local Elasticsearch server accepted the data. Then we used the existing Reactivesearch UI tools to create a working Elasticsearch front-end that uses the modified sample ETD and Twitter data.

During IR3, we transported our local workflow to a container-based cluster. The first step was to create a Docker image for the project and connect the current project to the class Elasticsearch server. We improved the result UI for the tweets and incorporated a new page for the web page data. We made some improvements to the search filter, to be further improved based on the content teams' feedback. We linked the UI login mechanism to the same cluster back-end.

In the final phase, we improved the login mechanism and added a number of internal admin management features. On the user management page, any admin can view and modify a user's information, and on the index management page, an admin or curator can view and edit access limitations to individual Elasticsearch indices, and grant special privileges or set limitations for each individual user.

We have added and improved curation features. In certain pages, we assigned access limits to different users, where some functions will only be accessible by certain users. For instance, there are some curation specific tasks that can only be accessed when the system detects the user is logged in as a curator.

We have also added KnowledgeGraph functionalities. A curator services page with dynamic input capabilities has been created. The system has been improved to connect to the Ceph file system to allow file downloads/uploads. Airflow connections have also been integrated to run the content teams' services and to display/provide results.



Project High-level Overview

Figure 1: High level view of all of the components from the Information Retrieval Project from the Fall2020 CS5604. This shows the connection between the front end and the analysis back-end which provides APIs to Elasticsearch and the KnowledgeGraph after the raw corpus of each one is analyzed. This image was designed by the class moderators.

1.1 Project Introduction

The full-text search project includes a web application that provides metadata and full-text search for over 30,000 ETDs, about 5 billion tweets, and around a few million web pages.

1.2 Management

We are managing this project through weekly meetings between members, leveraging Ally.io and the use of Objectives and Key Results (OKRs) as well as in and out of class discussions. The IR1 milestone is defined as a stage for us to create basic familiarity with the tools and the available resources such as the existing software/tools from the 2019 iteration of the class. IR2 is defined as the first point where a working prototype of the software is available as a proof of concept. This stage's results will be a building block upon which all developers, this group and the rest of the class, will be able to improve functionality and UI design and performance. IR3 is considered as the step before finalizing the product which comes just before the Final Report requiring the removal of any discovered problem from all the previous milestones making it ready for deployment in the real world.

1.3 Problems and Challenges

There are a number of challenges in this project. One is to learn how the previous project is set up and how it works given what information is available from its developers. This includes reports, repositories, and helpers of those teams (i.e., Professor, TA, etc.). Furthermore, we plan to use our team's innovative power to add and build new features based on what is already available. Further challenges are learning how to use the relevant tools such as Flask to develop UI, build queries for Elasticsearch APIs, visualizing using Kibana, using KnowledgeGraphs, dividing user authority based on the type of the users, and so on. Ultimately, connecting all of the different bits and pieces of this system with all the curation, analysis, and integration teams and containerizing them will be one of the most important challenges to overcome.

The other challenging section of our work was on creating and initializing and customizing the code. This was mainly done to allow high level reusability and further development using our guidelines. Since React programming has a component-based structure, its code has to be compiled in order to provide a standard HTML page parsable by any browser's Document Object Model (DOM). Hence, component HTML properties are slightly different such as "class" being called "className" in React due to "class" being a reserved name for Object Oriented modules. The module installation, debugging, testing, and building processes are also different and require their own dependencies (explained in Section 6.5.1). Additionally, other structures such as Bootstrap (responsive components) and customizability should also be linked to the current components so developers can make use of them during the UI development.

Finally, the overall design needs to possess a high level of dynamic configuration points that can be changed easily such as: links to external servers, databases, upload/download points, persistent data volumes, etc. These will allow for better control when the different parts come together in containers and sections such as KnowledgeGraph APIs, or new Elasticsearch servers/indices are introduced. The REACT programming approach helped reduce many of these dynamic programming challenges while introducing its own set of problems such as inter-component communication and a different design paradigm.

To overcome these challenges many different tools and workflows were considered and the most suitable were selected. Containerization with Docker is used to isolate all platform related issues and simplify initial setup using pre-configured Docker images. Elasticsearch is used as the general multipurpose indexing and searching tool for the datasets and their metadata while KnowledgeGraphs are used to search, select, and execute dynamic workflows.

1.4 Improvements

There will be multiple major aspects of the 2019 version of the system that has been worked on and more that needs to be worked on. First we will be expanding the existing user system and creating accessibility to visuals and APIs for the planned three types of users. In addition, we created new pages for certain users to manage existing website resources and other users. Next is updating the existing UI and transforming the support for the tobacco corpus into support for the Twitter and web-page contents to search and query. Additionally, two new segments have to be developed for data curators/analysts in addition to linking into KnowledgeGraphs to increase the dynamic capabilities of data processing.

In the 2019 version, many of the external dependencies and data connection points in the code were not accessible on a high level (e.g., environment variables) which can cause issues when setting up a multi-component system using Docker which is our final product. These components include software such as Kibana. There were problems from the 2019 version due to hard coded addresses, websites/server URLs, the user database, and the Elasticsearch server/indices. Resolving these will be essential when building self-deploying images and when using a Docker cluster for setting up the final deliverables of the project. This will also allow higher level container management software like Docker, Docker-compose, Kubernetes, or any other platform to modify and manage the external connections through high-level controls. The exact setting will be further discussed in the Developer Manual (Section 6).

Additionally an entire new section was added for a new group of users called curators. This allows them to run services introduced by the system developers and potentially other curators. Achieving this required entire new systems such as user level/permissions and Airflow services. Each was added in combination with their corresponding APIs and UIs that were designed in-house. This provides more flexibility to creators who need more than just search permissions on the mostly static interfaces which are usually available.

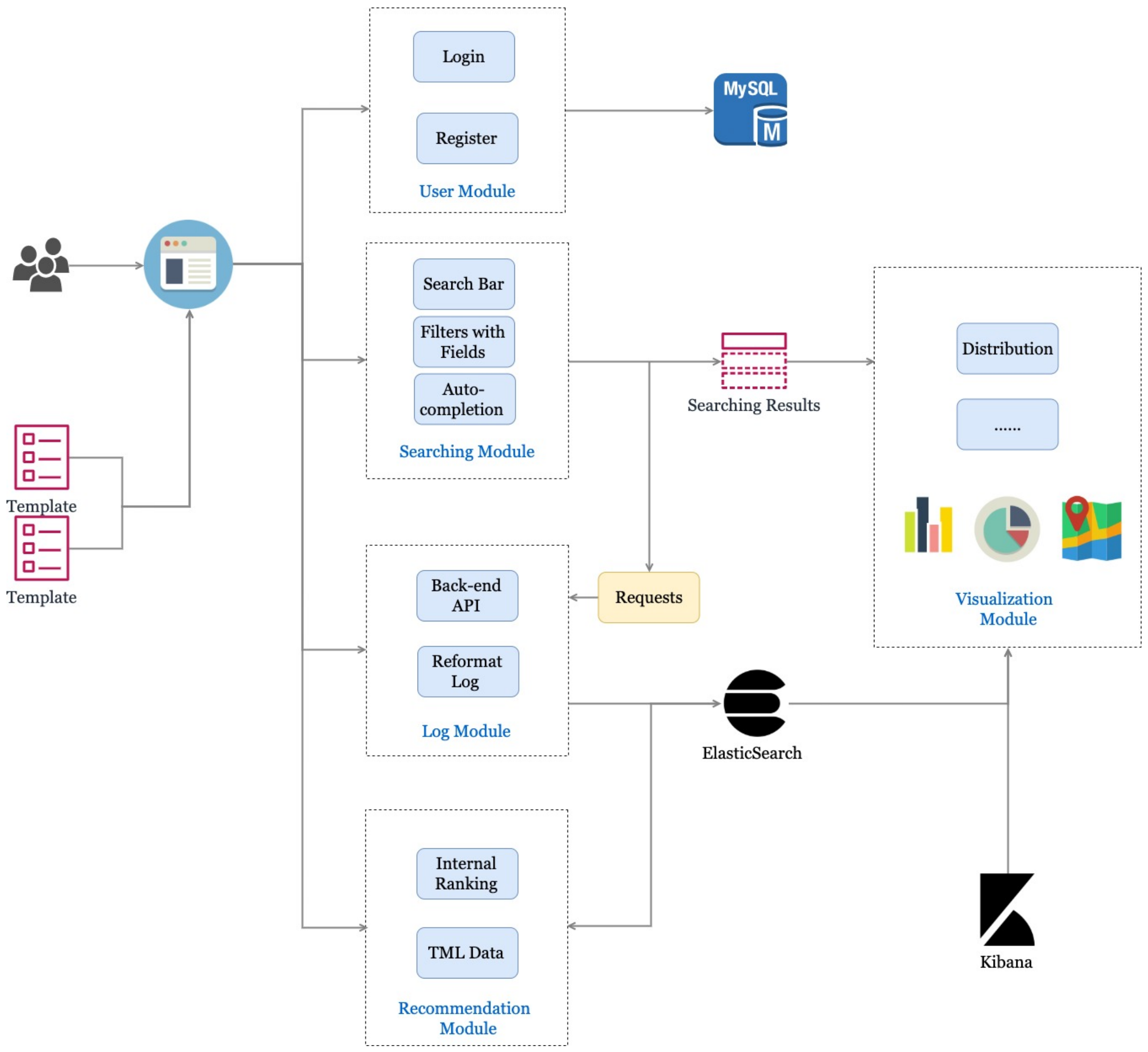
2 Literature Review

2.1 Base Design 2019

We decided on using a similar project design to the 2019 iteration of this project (see Figure 2) so it could be used a base for designing the interface. This would reduce the time and effort spent on setting up the infrastructure and tool sets. Since the project has similar objectives and structure to the current project, many of the foundational designs can be reused. There exists a repository that we used as one of the major points for the initial setup [16]. However, there are some deliverables that would require us to extend last year's work. They include but are not limited to:

1. An interface that allows researchers and curators to manage the collections (adding, updating, adding value) or doing research with the collections (developing, testing, experimenting with new techniques).
2. An interface to the Reasoner and Knowledge Graph for invoking Airflow for running workflows for curation/research.

Moreover, a number of reports from said year as well as other years with slightly different designs and objectives are also available with more information regarding the design and usage of the tools and structures [17].



FE Team 2019 Design

Figure 2: The connection between the front and back end, and the connections to Elasticsearch, from 2019. Adapted from [17].

2.2 Information Retrieval System

Information retrieval is finding materials, referred to in the literature as documents, of text and other types of content, that satisfies an information need from within large collections (stored on computers) [14]. Traditionally, this was limited to librarians and professional clerks, however, with the advancements in technology and the exponential increase in data availability, retrieval of information has graduated into an everyday task for most people. ChengXiang Zhai and Sean

Massung describe the relevance and importance of information retrieval this way: “There is an urgent need for developing intelligent text retrieval systems to help people get access to the needed relevant information quickly and accurately, leading to the recent growth of the web search industry” [22].

2.3 Collection of ETDs

ETDs are electronic versions of theses and dissertations. They enable the incorporation of graphics, animations, sounds, videos, and other forms of multimedia. They are especially valuable for reasons that include:

- **Availability:** ETDs make theses and dissertations available and accessible to the world wide web.
- **Enhancing Library functions:** ETDs help enhance the storage and archiving functions of libraries and make it much easier and more cost effective. Libraries do not need to worry as much about additional physical storage since the documents are already in electronic form.

VT has been a world leader in the electronic theses and dissertations initiatives for more than 20 years. On January 1, 1997, VT was the first university to require electronic submission of ETDs [20].

The VT ETD system allows graduate students to submit, review, and publish their theses and dissertations online. The system can be accessed through this link https://guides.lib.vt.edu/find/byformat/etds. Listing 6 in Section A.1 shows a sample metadata record representing one ETD in JSON format.

When working with corpora such as ETDs, and as mentioned in [4], there are many challenges that have to be addressed such as format, storage, correctness, conflicts, and many more that both the front-end team and each of the respective teams are facing. Many designs within the system are used to tackle or avoid such problems. The most prominent include the formatting and correctness of ETDs which in some cases needed validation, in contrast to tweets and web pages. Please refer to the “Fall 2020 ETD” team’s report for more information.

2.4 Collection of Tweets

A tweet is a message sent on Twitter. It is a form of modern independent media or self-media, where users can publish text and pictures to a potentially interested audience. In order to send a tweet, the user has to create an account on Twitter. Any tweet posted on Twitter has the potential to be viewed by every Twitter user or non-Twitter user if the user didn’t set up any limits. Most users usually set up filters to only receive tweets from their interested Twitter feeds from other users.

In May of 2020, Kyle Vincent and Emma Meno used a collection of tweet data to build a KnowledgeGraph for subject matter experts (SMEs), statisticians, and developers. This enabled them to generate workflows for their projects involving Twitter data and its advanced analysis [15].

Tweets are particularly interesting for textual research purposes and Natural Language Processing because of the quality of data it possesses. Twitter restricts users to 240 characters per tweet.

This restriction pushes users to use their best words in a tweet that passes their messages most effectively. This in turn increases the overall word value used in tweets.

We have a Tweet Collection Management team (TWT) who are working on ingesting 5 billion tweets. This includes cleaning the tweets, analyzing the metadata, extracting key information, classifying tweets into categories, and finally indexing the tweets into Elasticsearch so they can be searched [18]. A sample metadata record representing a tweet in JSON format can be found in Listing 8 in Section A.2.

In an attempt to understand the data, we observed three types of metadata sources for location reference in tweets. The first one is in the tweet location in the JSON file. It contains an exact location with long/lat coordinates or four pairs of lat/long coordinates that define a "bounding box" for the possible location. The second is the mentioned location which is the location contained in the tweet message. The final one is profile location which mentions the account-level coordinates for locations of interest [3]. We will be able to search all of the tweets based on each of the three location data points. However, this can become challenging since not all tweets contain all three instances of location data. It is unclear how to handle such cases, when all or none of these instances are available for searching with Elasticsearch. This is one of the problems to be discussed with the TWT team, including about a consistent format in the representation of tweets.

2.5 Elasticsearch

Elasticsearch is a distributed open-source search and analysis engine suitable for all data types, including text, numbers, geospatial, structured, and unstructured data. Elasticsearch was developed based on Apache Lucene and was first released in 2010 by Elasticsearch N.V. (now Elastic). Elasticsearch is known for its simple REST-style API, distributed features, speed, and scalability [9]. It is the core component of the Elastic Stack; Elastic Stack is a set of open-source tools for data collection, enrichment, storage, analysis, and visualization. People usually call the Elastic Stack ELK Stack (referring to Elasticsearch, Logstash, and Kibana). At present, the Elastic Stack includes a series of rich, lightweight data collection agents, collectively called Beats, which can be used to send data to Elasticsearch [11]. All teams working on this project will be using Elasticsearch for its many functions to manage and use our colossal corpora.

2.6 React and ReactiveSearch

React is a newly popularized library for JavaScript which simplifies web UI development by introducing reusable components [7]. React uses the JSX language which works with JavaScript but not with the current browsers directly. The approach introduced by React allows for a syntax similar to classical programming languages like Java and C++ with encapsulation, inheritance, internal variables and functions, while making use of the dynamic nature of the JavaScript language on the web.

React uses immutable properties (props) from when the component is instantiated, and states, to manage and update the components and their dependencies. It is designed such that information flows from the parent elements to the children, creating a mostly strict information flow. The tool provides a great platform for creating reusable components with high flexibility. However, due to the nature of information flow in React, it has a steep initial learning curve for more experienced web designers. That being said, its advantages outweigh its disadvantages, and it can more easily be used to create and manage pages/elements that are both highly dynamic in addition to being responsive to user interactions.

ReactiveSearch is an Elasticsearch UI components library for React and React Native. It has 25+ components consisting of lists, ranges, search UIs, result displays and a way to bring any existing UI component into the library [2]. We have discussed the ones that we used in Section 6.5.1.

2.7 Docker

Developing apps today requires more than just writing code. Multiple languages, frameworks, architectures, and discontinuous interfaces between tools for each lifecycle stage creates enormous complexity. Docker simplifies and accelerates workflows, while giving developers the freedom to innovate with their choice of tools, application stacks, and deployment environments for each project [5].

3 Requirements

3.1 Introduction

The goal of the FE team is to create a web-based front-end interface, which provides all of the listed functions below. All of the results should be displayed in a fashion that is easy to access and understand with a well formatted layout.

3.2 Functionalities

Table 1 defines the list of functions and their planned status in this project.

3.3 Performance

The load on the web server container can be alleviated by adding a load balancing layer to the containers (e.g., using Kubernetes). This will ensure that the application would scale as more users start to use it. The most common functionalities of the system do not have any shared writing tasks, hence parallelization conflicts would rarely occur.

3.4 User Support

We are planning to introduce a ticketing system that allows the users to submit problems with the system which should greatly assist in solving problems.

Table 1: This is a tentative table for all current and potential functionality of the front-end system. When there are two check marks specified, the code was improved to add more functionality and/or be more efficient. Each IR represents an interim report date specified by the project managers. The *Stretch* goals are tasks that will be attempted if time allows.

Function List	2019	IR1	IR2	IR3	IR4 (Final)	Stretch
Tobacco search (excluded from this project)						
ETD search	✓	✓				
Link to original content (Elasticsearch)	✓	✓				
Content filtering by author, date, location, etc.		✓				
Login and sign up pages		✓				
Recommendations for searching	✓		✓			
Clean and update developer environment			✓			
Real-time data aggregation for ETD filters			✓			
Combined Reactivesearch, Bootstrap and custom styles			✓			
Tutorial for setting up the development environment			✓			
Tweet search			✓			
Log system to record user activities				✓		
Web-page search				✓		
Multiple user access support				✓		
Docker support				✓		
Admin management				✓		
Team introduction and descriptions	✓				✓	
Project concept overview page					✓	
Curator user support					✓	
Curator service interaction					✓	
User support ticket system						✓
Visualization using Kibana API	✓					✓

3.4.1 User Type

The system categorises the users into three types:

1. **General users** are able to search any type of content available in the system.
2. **Researchers** or curators are allowed to manage their collections and do research with them in addition to the general user functionality.
3. **Administrators** have all the functionality mentioned before with the addition of managing users.

Furthermore, the previous three user types can have extra privileges. If the user has **VT affiliation**, each individual will in addition to public indices have access to indices available to members of the VT community. These restrictions will mainly be addressed by the data provider or by the VT library/legal staff.

3.4.2 User Manual

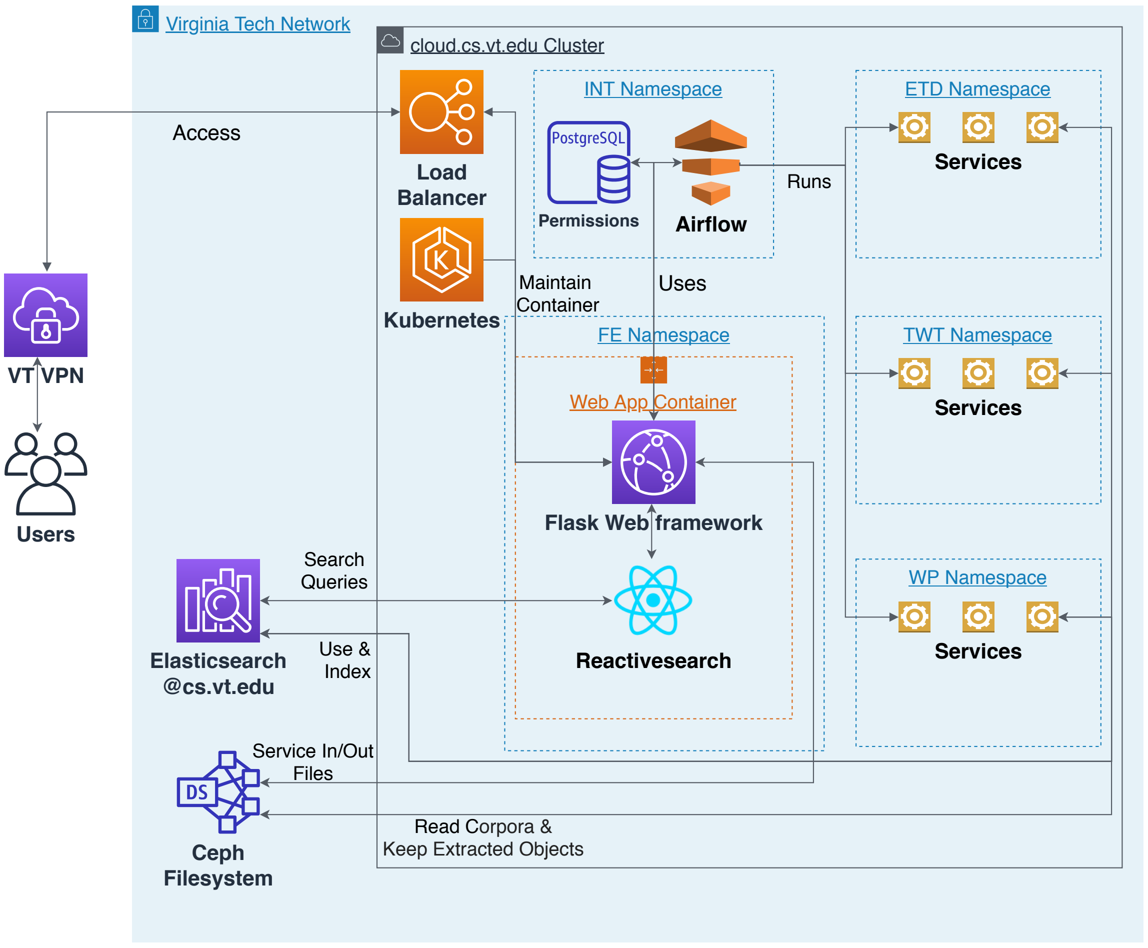
The user manual explains all of the functions and pages we have developed so far and is available in Section 5.

4 Design

This section discusses the design decisions of the system both on the high structural and lower database levels.

4.1 Application Structure

The overall structure of this system is very complex due to its numerous external dependencies. It starts from Kubernetes which manages the front end container all the way down to the PostgreSQL database and Ceph file systems. Figure 3 shows the complex relations between the components.



Application Structure

Figure 3: High level view of the system, including most of the direct and indirect external dependencies that support and drive the system. Please note that everything under the cloud cluster umbrella is a container.

4.2 Docker

Docker is a container-based tool that separates the processes from the host system. It is well known for its ease of use and compatibility functionality in software development and testing. The prototypes and final system will be available as Docker images on the project repository [21]. We will provide documentation on how to initialize and use the system which would work on any machine when provided with an analysis back-end. See Section 6.6 for more details.

4.3 Front-end

4.3.1 Homepage

The home page will provide all of the links to all of the functions this system provides. Here is the current link to the home page: http://[2001:468:c80:6102:1:7015:4f8b:c659]:3000/. However, this link could change from time to time, as the front-end container is reconfigured.

4.3.2 User Login and Register

The system will allow any user to register as a general user of the system. Different types of user will be given different types of authority to use this system. Admin users also can use this system to manage all of the users and track their activities.

4.3.3 Curator page

As we mentioned above, different types of users have different levels of access to the system. Here we describe a set of users called curators or researchers. They are capable of access beyond general users. They have the ability to manage the collections (including adding, updating, and adding values). They can also perform exploratory research with the collections (e.g., developing, testing, and experimenting with new techniques). They are able to use and potentially create services that can be run on datasets and collections directly to achieve different research goals. The open-ended nature of these services provide a vast basis for such users to explore and experiment.

This system is built such that once a user logs in, it detects the user type and then provides access based on their respective privileges. This implies that only a curator would be allowed to see curator features, that a general user cannot see. Currently, curators can see all public indices and ones that they have access to or manage privately. The system can be extended such that they can choose which index they want to search on, or can define entirely new indices.

We discuss this page and its functionality in further detail in the user manual.

4.3.4 Administration page

The full admin page will be accessible only to an admin user. They can access this page by going through the link from the home page after they log in. The system will check their credentials and grant the right to modify the user and index permissions. On the admin page, the user will able to access two lists:

- User list: This list will show all the users who are registered in the system. The admin will be able to modify an individual user's information and role in the "Modify User" page, which can be accessed from this list.
- Index List: This list will show all the available indices accessible by this system. Admins will be able to modify each individual's index access permissions and can grant a user access

privilege to an index in the “Modify Index” page, which they can access from this list.

The curator will able to access the admin page with limited access to its index modification functionality. Currently, the curator will only be able to access the index list and edit the indexes about which he or she has ownership.

4.3.5 Modify User page

The admin will access this page through the admin page’s user list. This page will pull the requested individual user’s information and let the admin change it. The user’s username, email, and user role are the attributes that can be modified on this page. Also, the admin can delete this user on this page.

4.3.6 Modify Index page

The admin or the curator who has the ownership of an index will access this page through the admin page’s index list. The page will pull the requested Elasticsearch index’s access information from the server and let the admin modify it. The admin can change the index’s public access status and grant an individual user special privilege on this index, such as editing or owning the index. The user’s role will change automatically when the privilege is granted and/or removed.

4.3.7 Searching Page

This page gets search keywords and commands from the users, then retrieves data from the analysis back-end and presents the data in a summarized format which can be expanded for a better view. Additionally, the system also provides filtering functionality for the users to perform on the data.

4.3.8 React and NodeJs

Reactivesearch [2] is an Elasticsearch UI component library for React and React Native. It has more than 25 components consisting of lists, ranges, search UIs, result displays and ways to bring any existing UI component into the library. NodeJs [19] is a free open source server environment which runs on various platforms consisting of many modules for web-based development which uses JavaScript.

4.4 Back-end

4.4.1 Flask

Flask is a lightweight web application framework in Python for developing websites quickly and easily. It has the ability to scale up to complex project applications and it is one of the most popular Python web application frameworks [12].

4.4.2 MySQL

This project uses MySQL, a light weight relational database system, to manage its users and their permissions [8]. There are currently three levels of access planned as mentioned in Section 3.4.1. Currently the database only contains three tables within a database named after the team acronym FE:

- Table 'user'

  This table contains all of the information about an individual user. For now, all information can only be modified by admin users. Admins can modify this table through the individual Edit User Page[5.5] with the link in the User list in the Admin page[5.5]. The user can be promoted to an admin if another admin promotes them through this Edit User Page[5.5]. The user will be automatically promoted as curator if they are granted special privileges to an Elasticsearch index.

  1. *user_type*: Type of user privilege
  2. *username*: Name of the user
  3. *email*: Email address attached to the username for password retrieval
  4. *register_time*: Time when the user registered at the website
  5. *password*: Encrypted password of the user
  6. *vt*: User has VT affiliation or not

- Table 'IndexTable'

  This table contains all the Elasticsearch server's public UI reading access settings. Since we don't want to modify the Elasticsearch server status, we use this table to limit the user access privilege. All of the indices are set as private by default until an admin makes a change. Only the admin can modify this table through the index manage page[5.6].

  1. *idIndextable*: Auto increment ID for tracking the number of records
  2. *Iid*: Name of the Elasticsearch index
  3. *Raccess*: Public reading access status of the index

- Table 'PermissionTable'

  This table contains all personal privileges and limitations for an individual to access the Elasticsearch indices. The user can pass the limit set by the IndexTable conditioned on them having a record in this table. The admin can also ban a user by adding a specific record in this table regardless of whether the index is public or not.

  1. *idPermissiontable*: Auto increment ID for tracking the number of records
  2. *Iid*: Name of the Elasticsearch index
  3. *Uid*: Username for an individual
  4. *permission*: The permission details of the individual relative to the index

4.5 Dynamic Configurations and Environment Variables

Many internal configurations are used throughout the back end and front end code. However, dynamic behavior will mainly be necessary for external links to programs (e.g., Kibana), data (Elasticsearch), services (Airflow), the main URL, etc. These will allow high level access during runtime, providing flexibility for the use of different stages such as development, testing, and deployment. Some have already been implemented such as the *REACT_APP_BASE_URI* and *REACT_APP_ELASTICSEARCH_URI* environment variables which allow the main URL of the website and the Elasticsearch server URL to be changed during deployment. These will be essential when setting up a stand-alone installation such as container-based usage with Docker and/or Kubernetes. We are using environment variables to avoid localized variables in the settings. Please refer to the end of Section 6.6 for more details on the environment variables.

4.6 Milestones

- **The Interim Report 1 (IR1)**: This milestone is defined to create basic familiarity with the tools and the available resources such as the existing web pages from the 2019 iteration of the class.
- **The Interim Report 2 (IR2)**: IR2 is defined as the first point where a working prototype of the software is available as a proof of concept. This stage's results will be a building block upon which all developers will be able to improve functionality and UI design and performance. At this stage, our working prototype should be able to search with each of this year's content types using metadata (ETDs, tweets, web pages).
- **The Interim Report 3 (IR3)**: IR3 is considered as the step before finalizing the product which comes just before the Final Report which should have removed any potential problem from all of the previous milestones making it ready for deployment in the real world. Also at this stage, our working system should be able to search with each of this year's types of content, using text content (ETDs, tweets, web pages). It should also have an Interface to the Reasoner and Knowledge Graph for invoking Airflow for running workflows for curation/research [10].
- **Final Report**: This is the final report on the project where all of the components need to be delivered as a final package to the project coordinators.
- **Stretch**: These are some goals that are great additions to the system and are not part of the requirements. However, due to their numerous benefits, they will be implemented when extra time is found at the end of the project development lifecycle.

  You can see a tentative further breakdown of our milestones in Table 1 including the milestones and the deliverable dates.

4.7 Future work

Here are some features that were not added to the system due to time/design constraints, but are good choices for new features. Many of these ideas were researched but did not make the final product due to constraints.

- **Use the Flask back end as a proxy for Reactivesearch requests:** Currently the Reactivesearch components send a request directly to the Elasticsearch VM with the required credentials. However, this is not ideal since in addition to security risks to the Elasticsearch VM, the requests could also be targeted. This could cause further issues when the client is on an external network compared to Elasticsearch, causing the requests to fail. Despite our best efforts, we were not able to find a good enough solution.

- **Using a secure production web-server as a back end for the Flask base:** Flask is not recommended as a standalone production web server for large scale applications. However, it supports many such widely used servers. This will become critical when the system is deployed in a large scale production version.

- **Using HTTPS and using a proxy for external services to avoid CORS problems:** The system currently uses HTTP for all of its pages. However, this does not provide enough protection for things such as sensitive traffic (e.g., login, private data) and data storage (e.g., passwords and permission tables). We have mainly focused on creating a functional prototype and it can be easily secured by an expert while keeping all the functionality.

- **Allow searching through multiple indices and group indices based on structure:** Currently, our search pages work mainly on the content team's primary indices. However, when new indices are added it will be fairly trivial to group them based on structure and allow uses to switch between them based on the already implemented permissions when searching.

- **Service registration page:** Currently our system can make use of the services registered through the INT team's API. However, it would be ideal if a UI could be built that connects to the API for registering these services, thus reducing the technical know-how for doing so.

- **View and edit services/goals for admin and/or curator users visually with a graphical UI:** The workflow graph that is comprised of services and goals could be dynamically visualized by the system. This could be beneficial to the system both visually and functionally, allowing super users access to make efficiency and usability tweaks directly to the nodes/edges. The details needed to make such tweaks could be unknown to the users when not properly visualized. A number of examples of these could be duplication of nodes, multiple nodes withing a workflow that are preceded by a more generalized step or a chain of nodes that could be replaced by one saving overhead time.

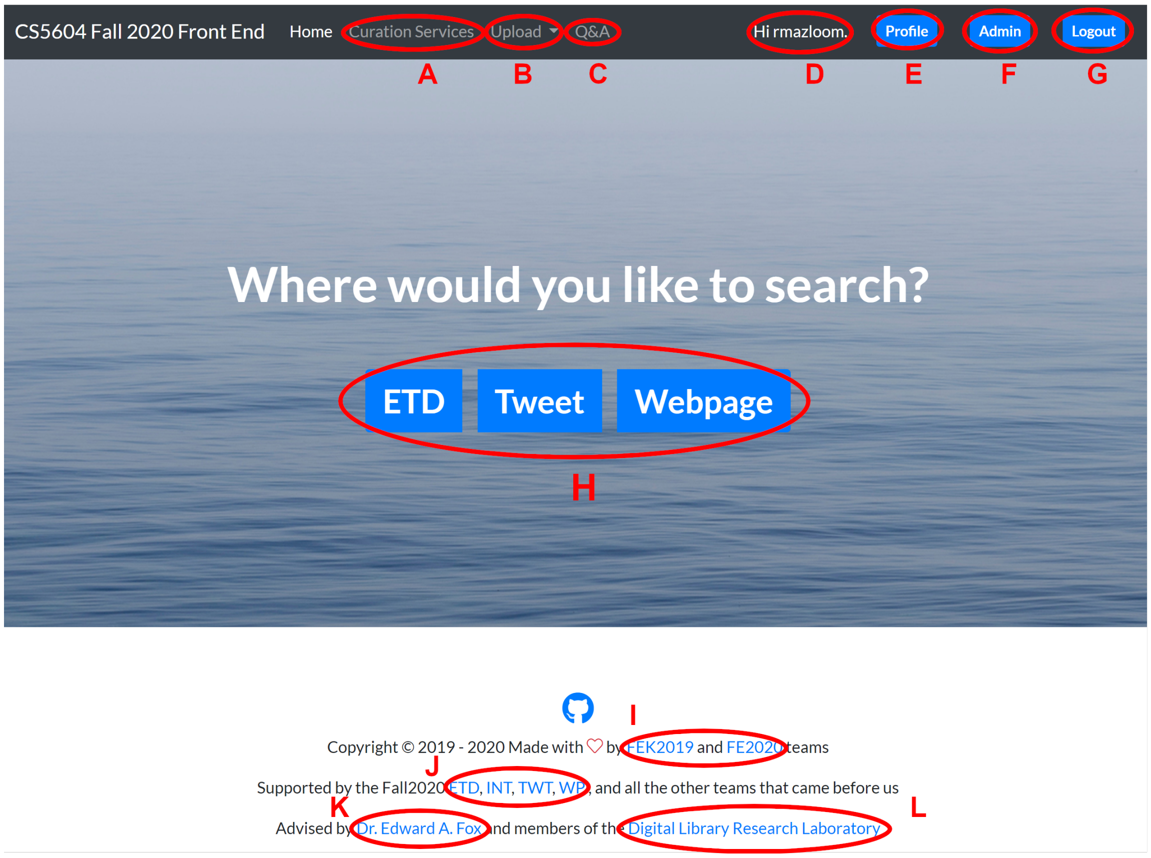
- **Create standards for service/index/goal usage and design:** As the system and the idea behind it grows, it will require better documentation and coordination between its components and users for increasing productivity. This could be achieved by introducing standards and baselines for the different components. This would streamline many of the processes, and focus creativity of development and research in the right direction.

5 User Manual

This section introduces all available pages for the users and their functionality using screen shots.

5.1 Understanding the Homepage

Figure 4 shows the homepage of the website when a user with the highest access privilege account is signed in (this is done to show all available features in one place). When not signed in, the "Sign In" button will bring up a minimal login that allows you to log in to the system (see Section 5.2 for details). If you do not have an account, the "Register" button redirects you to a page to create a general user account to access the system (see Section 5.4 for details). The "Curation Services" link (denoted by A) will take you to the page designed for running Airflow services. See Section 5.10 for details. A and B are only accessible to a user with at least curator privileges. The upload link (B) will allow a curator to edit indices while the "Q&A" link (C) will direct you to some useful links, and commonly asked questions with their answers. E and G will let you access your profile and logout from the system. The "Admin" (F) link will redirect you to the administration dashboard (see Sections 5.5 and 5.6). The search buttons (H) redirect you to the ETD, Twitter, and Webpage search pages, respectively (discussed in Sections 5.7, 5.8, and 5.9). Finally, links at the bottom of the page redirect users to the developer repositories and group pages (I-L).



Copyright © 2019 - 2020 Made with ♡ by FEK2019 and FE2020 teams

Supported by the Fall2020 ETD, INT, TWT, WP, and all the other teams that came before us

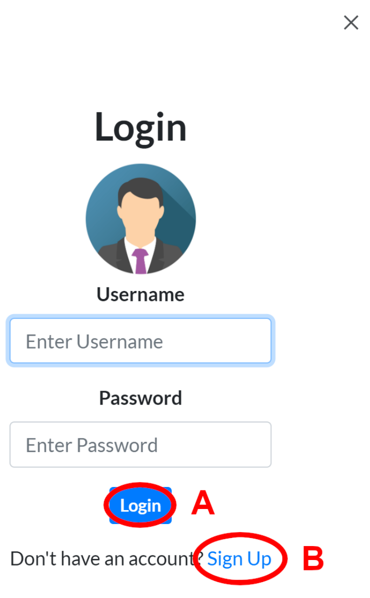
Advised by Dr. Edward A. Fox and members of the Digital Library Research Laboratory

Homepage Screen Shot

Figure 4: Screen shot of the current homepage screen with all of its functional elements.

5.2 Minimal Login

The page in Figure 5 is displayed when a user requests to login or attempts to access the search page and has not logged in. This page will show as a hovering frame over your page allowing you to login using your username and password (A). It also provides a link to register for a new account in case you do not have one (B).

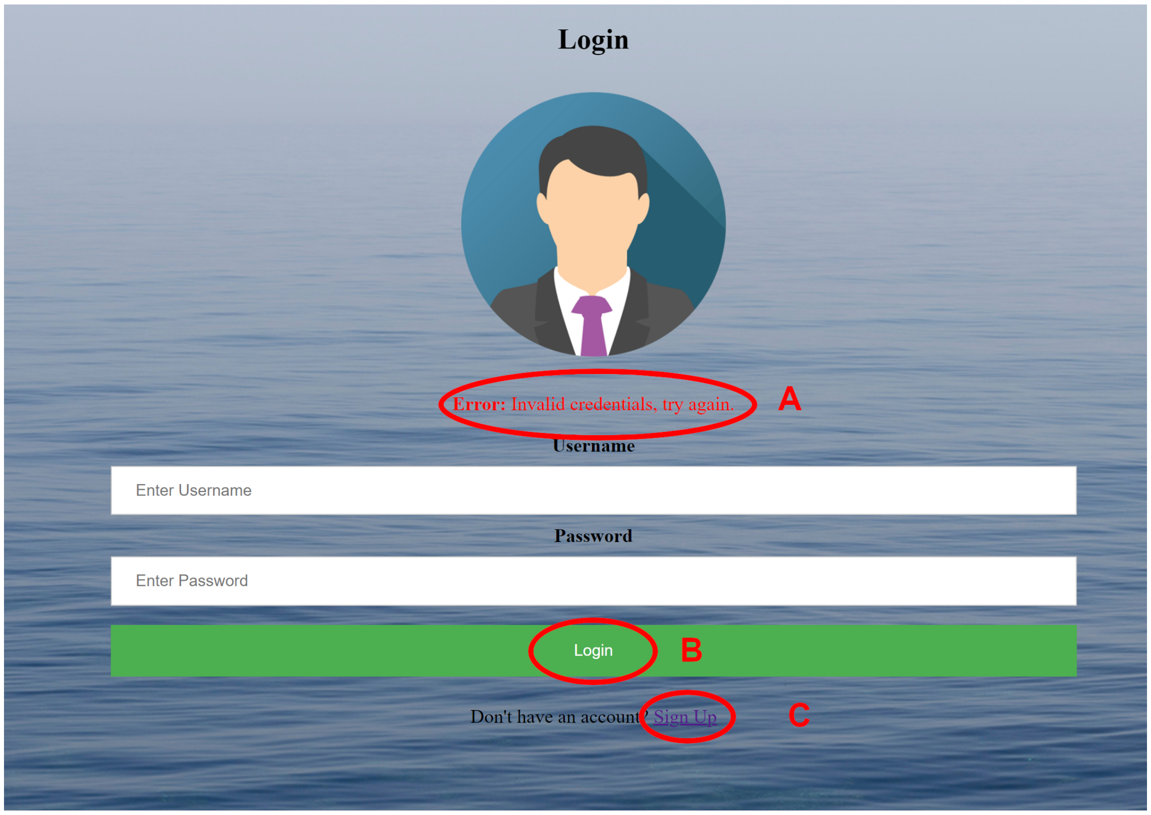


Minimal Login Screen Shot

Figure 5: Screen shot of the minimal login screen with all of its functional elements.

5.3 Main Login

The main login page shown in Figure 6 is the login page and is mainly used for cases of credential errors on the system. This is shown as a warning in red below the avatar (A). Otherwise, this page works exactly like the “Minimal Login” page described in Section 5.2 with login (B) functionality and a registry navigation link (C).

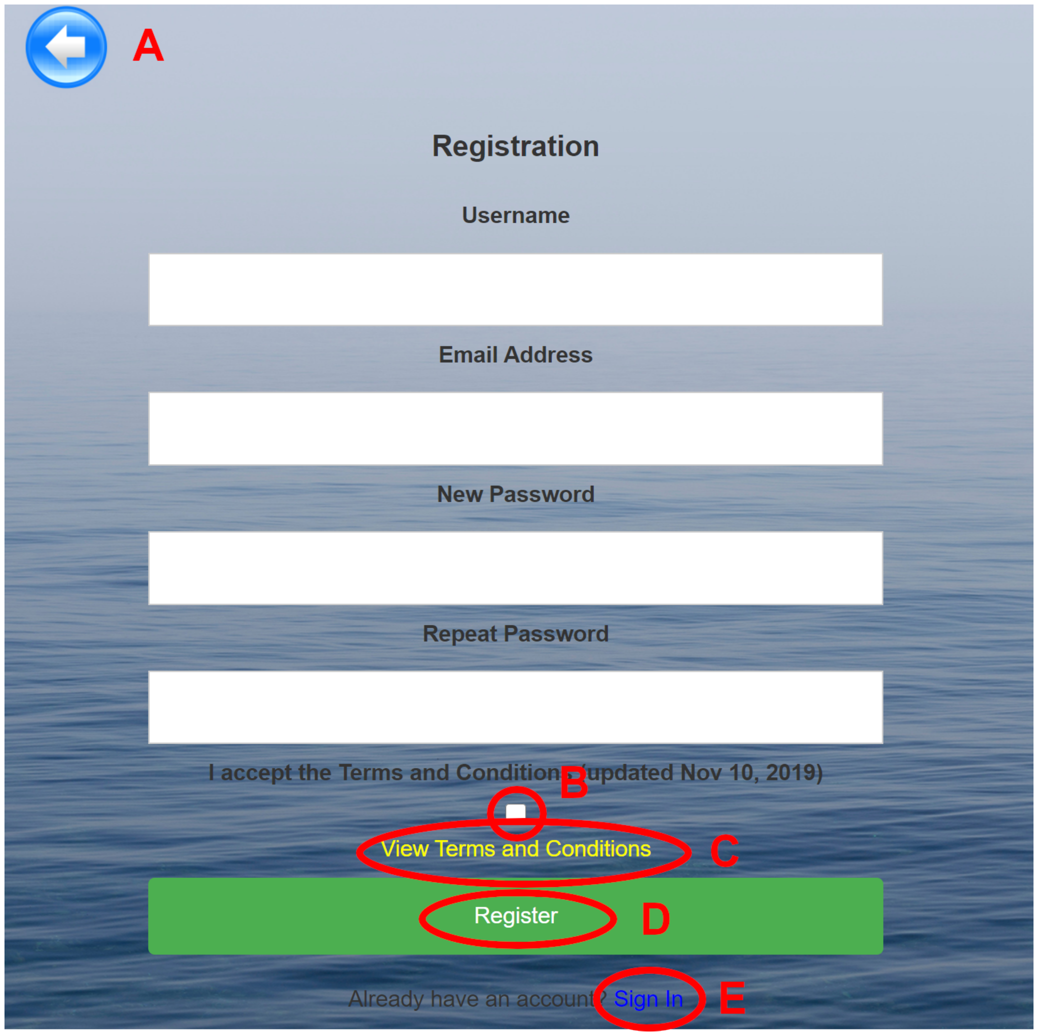

Login
Error: Invalid credentials, try again.
A
Username
Enter Username
Password
Enter Password
Login
B
Don't have an account? Sign Up
C

Login Screen Shot

Figure 6: Screen shot of the main login screen with all of its functional elements.

5.4 Register as a User

The registration page can be accessed through the homepage or any of the login pages. You can go back to the homepage with the back button on the top left (A). Registration requires a unique username, a valid email address, password, and accepting the fair use terms (B) to create a general user (D). If you already have a username you can use the “Sign In” link at the bottom (E) to login instead. The public version of this page registers the new user as a general user. The admin can access a page with a similar layout from the admin page, but the new user created there will automatically have admin user privileges.

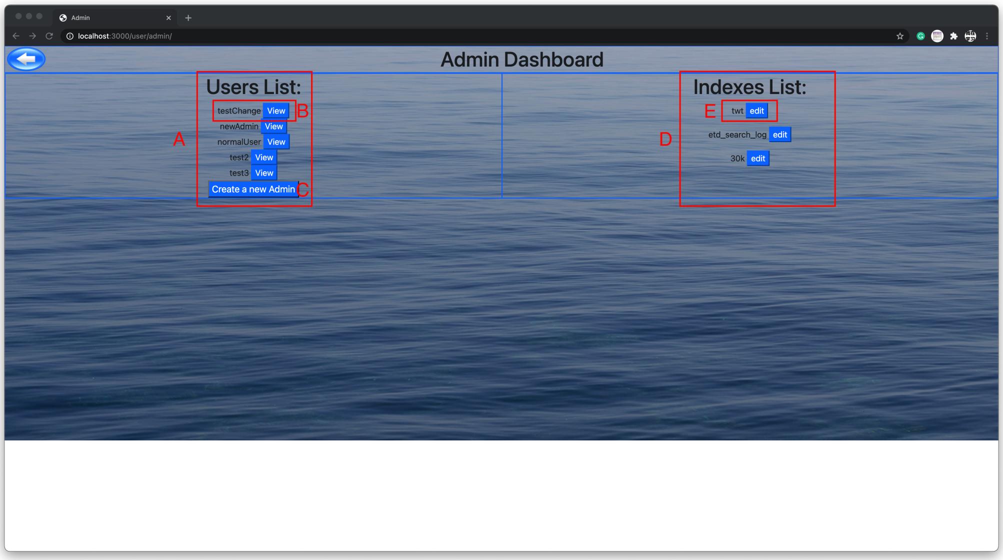

A
Registration
Username
Email Address
New Password
Repeat Password
I accept the Terms and Conditions (updated Nov 10, 2019)
B
View Terms and Conditions
C
Register
D
Already have an account? Sign In
E

Register/Signup Screen Shot

Figure 7: Screen shot of the register screen with all of its functional elements.

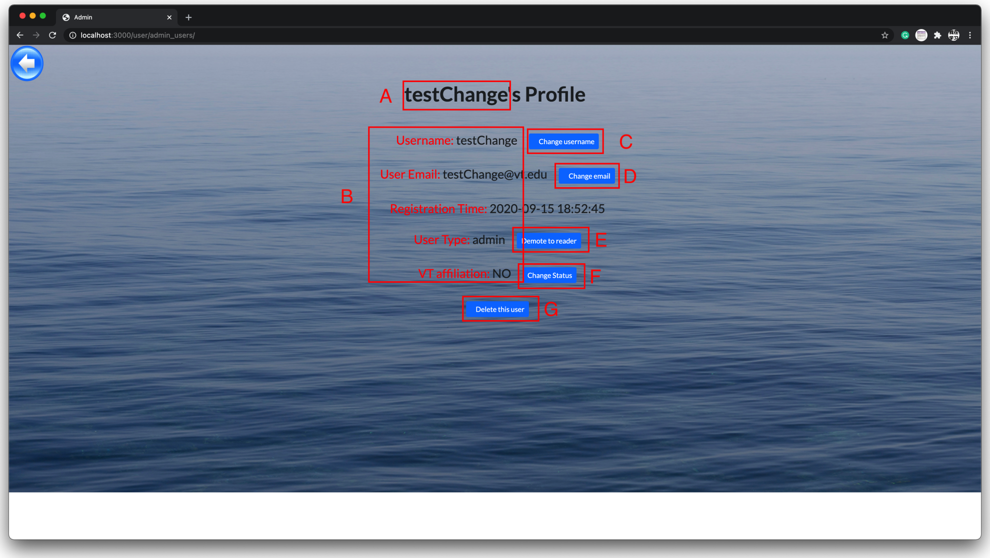
5.5 User Management

The admin page will only be available to admin users . This page[8] can be accessed through the homepage after login in as an admin user or the curator. In this page, all that the existing user will be shown is a table which includes their username, called Users List (A). This list will be invisible for curators since they don't have the right to view or edit it. The current admin user is able to view all available information about an individual user in a new page by clicking the button (B) next to the username in the admin page. In that new page [5.5], the current admin user is able to view the user's (A) information such as username, password, created time, and the user role (B) for this individual user. Additionally, a user's username, password, and role can be modified in this page as well (C D E). Switching a user's role will change the user's access permissions within the system. Admins can delete users from the system as well (G). Below the admin page [8], the admins are able to access a special register page. The new user created by this page will be an admin by default (C).



Admin Page

Figure 8: Screen shot of the admin screen with all of its functional elements.



testChange's Profile

Username: testChange

User Email: testChange@vt.edu

Registration Time: 2020-09-15 18:52:45

User Type: admin

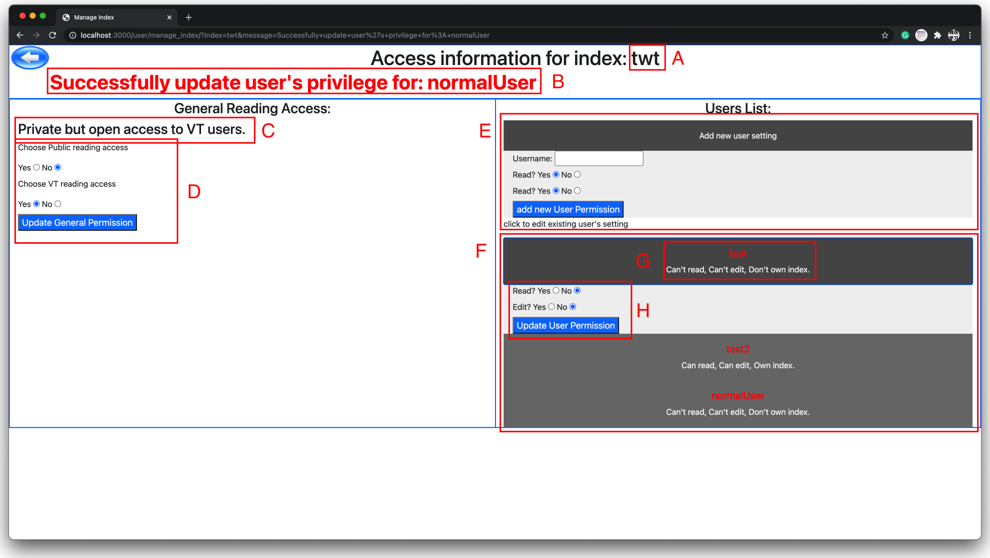
VT affiliation: NO

Edit User Page

Figure 9: Screen shot of the edit individual user screen with all of its functional elements.

5.6 Index Management

The index management page [5.6] will only be available to the admin users, or curators who have some index ownership. This function shares the same page with the user management of the Admin page [8]. You can view all of the existing Elasticsearch indices in a table called "Indices List" (D). The admin user will be able to add or edit permissions for any index. If an admin or curator wants to change permissions, they can access the individual index management page by clicking the edit button next to the index name in the indices list (E). Curators who don't have the index's ownership will not able to access this feature. In the index management page [5.6] there are two major parts. The first part shows the general reading access of this Elasticsearch index. This limits the reading access for the general users who don't have a special role such as curator or admin. The VT reading access will allow all users who have VT affiliation to access the index even when the index is private. Users who access this page will able to view the current index access status (C). The current user can update the general reading access by selecting from two options, and click on the update button to change the current status (D). The second part of this page is the "User List" which identifies all users who were granted special privileges to access this index. Admins can view each user's privileges (G), and use the update button below the username to update its privilege details (H). Additionally, admin users can add a user to the list by using the blank form which contains a text box where the desired username can be entered (E). Privileges can be assigned to this user using the right portion of this page.



localhost:3000/user/manage_index/?index=twt&message=Successfully+update+user%27s+privilege+for%3A+normalUser

Access information for index: twt

Successfully update user's privilege for: normalUser

General Reading Access:

Private but open access to VT users.

Choose Public reading access

Yes [✓ No]

Choose VT reading access

[✓ Yes] No

Users List:

Add new user setting

Username:

Read? [✓ Yes] No

Read? [✓ Yes] No

click to edit existing user's setting

test

Can't read, Can't edit, Don't own index.

Read? Yes [✓ No]

Edit? Yes [✓ No]

test2

Can read, Can edit, Own index.

normalUser

Can't read, Can't edit, Don't own index.

Edit index Page

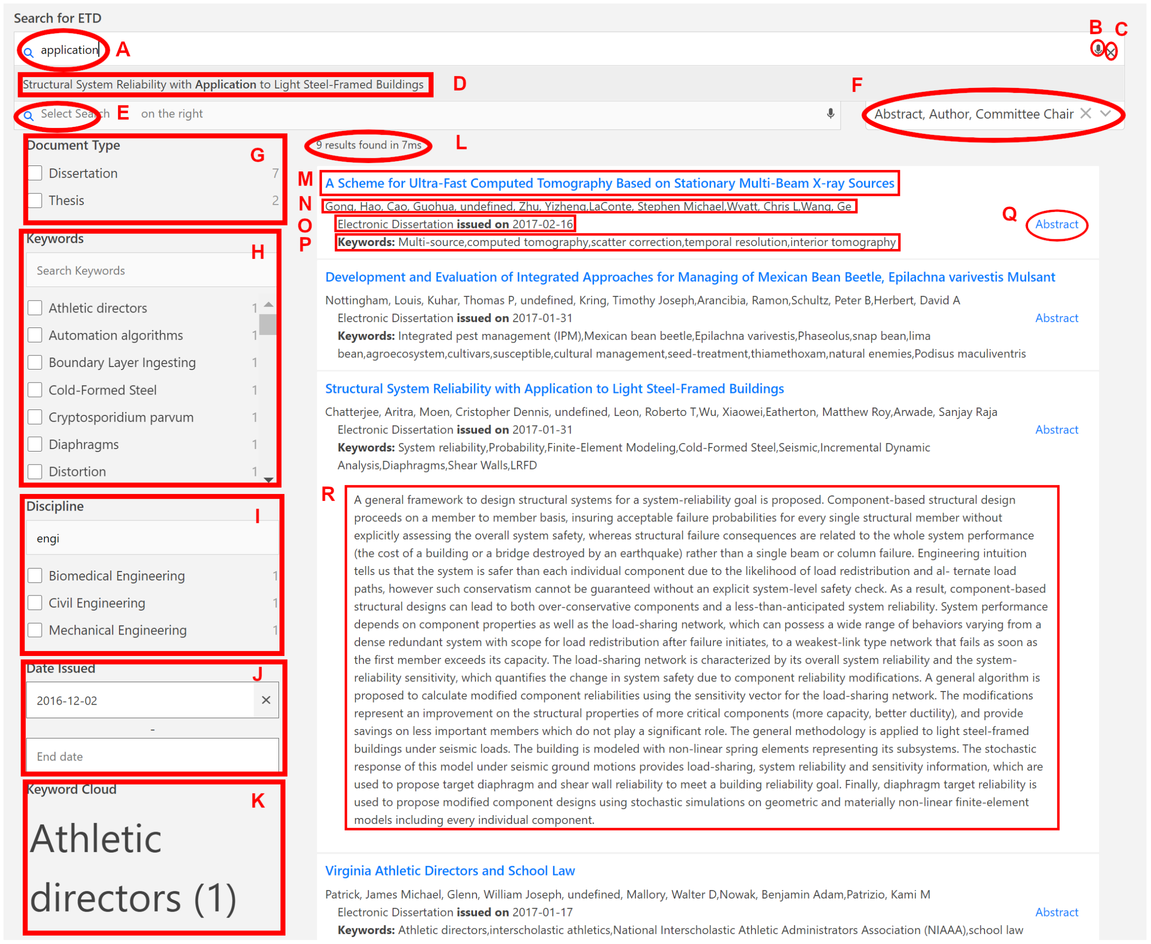
Figure 10: Screen shot of the edit individual index screen with all of its functional elements.

5.7 ETD search

Figure 11 shows a screen shot of the ETD search page. The main search bar (A) can be used to search within all fields (which will be highlighted) other than ones that have category specific filters on the left hand side (G-K). Helper buttons on the right of the main search bar can be used to search using voice [1] (B) and clear the search contents (C), respectively. The second search bar (E) allows for more tailored searches based on a selection of categories. You can selected multiple fields to search on within the drop-down on the right side of the page (F). A suggestion box will appear for each of the search boxes based on their respective filters (similar to D). The two categorical entries on the left (G-I and K) are populated dynamically including both the filtering option and their counts. Filter (I) allows you to enter the year range for when the document was issued. The available options in (H and I) can be searched using the search boxes on top of each section. The already used filters are listed as items on the top of the results next to L and can be removed individually or fully using their respective buttons. The number of results and Elasticsearch server turnaround time are shown below the filtering items (L). For each result the title (M), embedded with a link to the ETD, the list of authors (N), type and issuing time (O) and keywords (P), is displayed. The abstract can also be shown/hidden (Q) as an extension to the result header (R).

Future improvement: The interface can be further customized to meet the needs of the users and subject matter experts, and based on their feedback.

[1]This feature is based on Javascript and should work with most modern browsers.



Search for ETD

application

Structural System Reliability with **Application** to Light Steel-Framed Buildings

Select Search on the right

Abstract, Author, Committee Chair

9 results found in 7ms

Document Type	
Dissertation	7
Thesis	2

Keywords

Search Keywords

Athletic directors	1
Automation algorithms	1
Boundary Layer Ingesting	1
Cold-Formed Steel	1
Cryptosporidium parvum	1
Diaphragms	1
Distortion	1

Discipline

engi

Biomedical Engineering	1
Civil Engineering	1
Mechanical Engineering	1

Date Issued

2016-12-02

-

End date

Keyword Cloud

Athletic directors (1)

A Scheme for Ultra-Fast Computed Tomography Based on Stationary Multi-Beam X-ray Sources

Gong, Hao, Cao, Guohua, undefined, Zhu, Yizheng,LaConte, Stephen Michael,Wyatt, Chris L,Wang, Ge

Electronic Dissertation **issued on** 2017-02-16

Keywords: Multi-source,computed tomography,scatter correction,temporal resolution,interior tomography

Abstract

Development and Evaluation of Integrated Approaches for Managing of Mexican Bean Beetle, Epilachna varivestis Mulsant

Nottingham, Louis, Kuhar, Thomas P, undefined, Kring, Timothy Joseph,Arancibia, Ramon,Schultz, Peter B,Herbert, David A

Electronic Dissertation **issued on** 2017-01-31

Keywords: Integrated pest management (IPM),Mexican bean beetle,Epilachna varivestis,Phaseolus,snap bean,lima bean,agroecosystem,cultivars,susceptible,cultural management,seed-treatment,thiamethoxam,natural enemies,Podisus maculiventris

Abstract

Structural System Reliability with Application to Light Steel-Framed Buildings

Chatterjee, Aritra, Moen, Cristopher Dennis, undefined, Leon, Roberto T,Wu, Xiaowei,Eatherton, Matthew Roy,Arwade, Sanjay Raja

Electronic Dissertation **issued on** 2017-01-31

Keywords: System reliability,Probability,Finite-Element Modeling,Cold-Formed Steel,Seismic,Incremental Dynamic Analysis,Diaphragms,Shear Walls,LRFD

Abstract

A general framework to design structural systems for a system-reliability goal is proposed. Component-based structural design proceeds on a member to member basis, insuring acceptable failure probabilities for every single structural member without explicitly assessing the overall system safety, whereas structural failure consequences are related to the whole system performance (the cost of a building or a bridge destroyed by an earthquake) rather than a single beam or column failure. Engineering intuition tells us that the system is safer than each individual component due to the likelihood of load redistribution and al- ternate load paths, however such conservatism cannot be guaranteed without an explicit system-level safety check. As a result, component-based structural designs can lead to both over-conservative components and a less-than-anticipated system reliability. System performance depends on component properties as well as the load-sharing network, which can possess a wide range of behaviors varying from a dense redundant system with scope for load redistribution after failure initiates, to a weakest-link type network that fails as soon as the first member exceeds its capacity. The load-sharing network is characterized by its overall system reliability and the system-reliability sensitivity, which quantifies the change in system safety due to component reliability modifications. A general algorithm is proposed to calculate modified component reliabilities using the sensitivity vector for the load-sharing network. The modifications represent an improvement on the structural properties of more critical components (more capacity, better ductility), and provide savings on less important members which do not play a significant role. The general methodology is applied to light steel-framed buildings under seismic loads. The building is modeled with non-linear spring elements representing its subsystems. The stochastic response of this model under seismic ground motions provides load-sharing, system reliability and sensitivity information, which are used to propose target diaphragm and shear wall reliability to meet a building reliability goal. Finally, diaphragm target reliability is used to propose modified component designs using stochastic simulations on geometric and materially non-linear finite-element models including every individual component.

Virginia Athletic Directors and School Law

Patrick, James Michael, Glenn, William Joseph, undefined, Mallory, Walter D,Nowak, Benjamin Adam,Patrizio, Kami M

Electronic Dissertation **issued on** 2017-01-17

Keywords: Athletic directors,interscholastic athletics,National Interscholastic Athletic Administrators Association (NIAAA),school law

Abstract

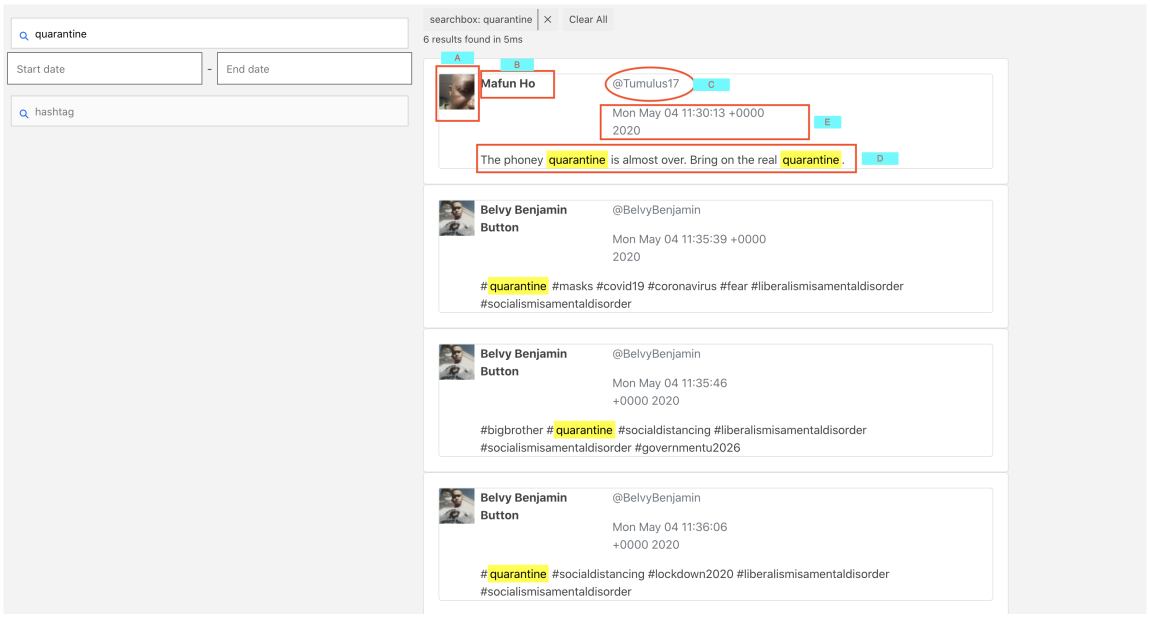
ETD Search Screen Shot

Figure 11: Screen shot of the current ETD search screen with all of its functional elements.

5.8 Tweet search

The tweet search page, shown in Figure 12, is very similar to the ETD search page mentioned in the previous section. From the abundant amount of metadata, the username (B), post text (D), user picture (A), user ID (C), and the tweet's posting date (E) are displayed. Currently we have a UI with the basic display resembling Twitter's style.

Future improvement: We will adjust the main search bar to search in all of the indices in the Elasticsearch server. Additionally, similar to the ETD search, we will give the user an option to choose the data indices they want to search. The secondary data indices (not searchable, like date and location) will be treated as a filter option on the side as filters.



quarantine

Start date - End date

hashtag

searchbox: quarantine × Clear All

6 results found in 5ms

Mafun Ho @Tumulus17

Mon May 04 11:30:13 +0000 2020

The phoney quarantine is almost over. Bring on the real quarantine.

Belvy Benjamin Button @BelvyBenjamin

Mon May 04 11:35:39 +0000 2020

#quarantine #masks #covid19 #coronavirus #fear #liberalismisamentaldisorder #socialismisamentaldisorder

Belvy Benjamin Button @BelvyBenjamin

Mon May 04 11:35:46 +0000 2020

#bigbrother #quarantine #socialdistancing #liberalismisamentaldisorder #socialismisamentaldisorder #governmentu2026

Belvy Benjamin Button @BelvyBenjamin

Mon May 04 11:36:06 +0000 2020

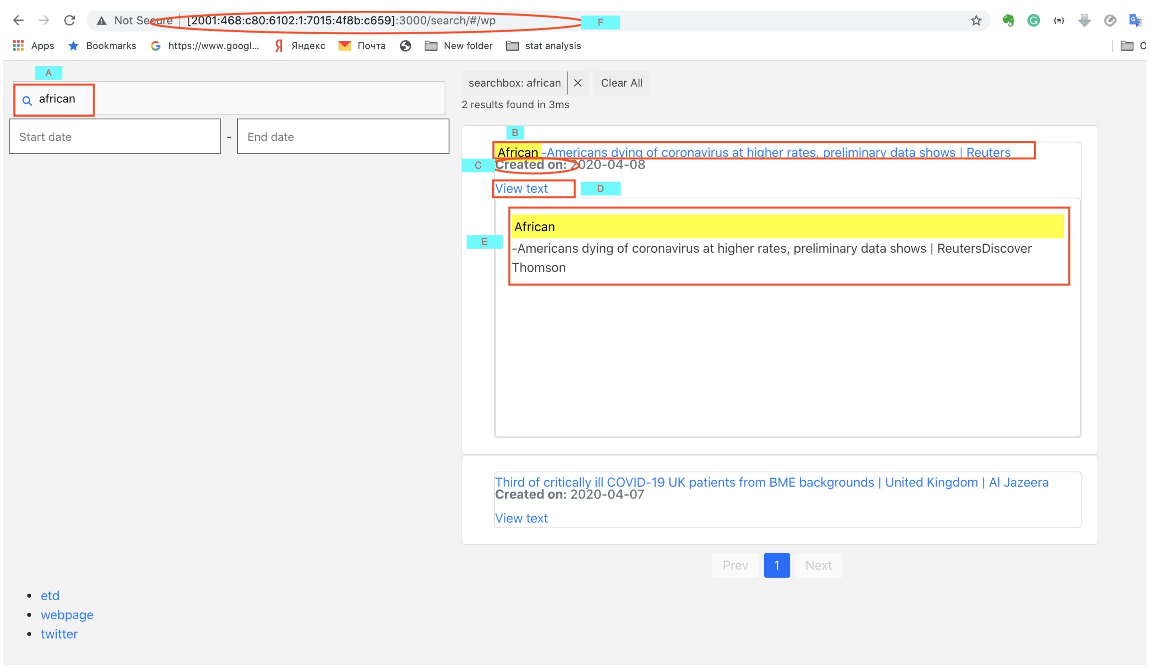
#quarantine #socialdistancing #lockdown2020 #liberalismisamentaldisorder #socialismisamentaldisorder

Updated Tweet Search Screen Shot

Figure 12: Screen shot of the current tweet search screen with all of its functional elements.

5.9 Webpage Search

The Webpage search user interface shows many similarities with the Tweet and ETD search pages. One new addition is the URL (F), showing the containerized front-end UI. The container now exists on the Virginia Tech Cloud server. The link denoted (F) is the current link to access webpage search. Please note the user must be on campus or on the VT network (using VPN) to be able to access this link. Starting with (A), we have the search bar similar to ETD. We are currently searching three fields on the Elasticsearch index: the title (B), the text (E) and the date. In Figure 13, the word "African" was queried and all of the titles and texts that have the word are returned. The "view text" button (D) expands section (E) containing the full text of the webpage which is scrollable.


Not Secure | [2001:468:c80:6102:1:7015:4f8b:c659]:3000/search/#/wp
F
Apps
Bookmarks
https://www.googl...
Яндекс
Почта
New folder
stat analysis
A
african
Start date
End date
searchbox: african
Clear All
2 results found in 3ms
B
African -Americans dying of coronavirus at higher rates, preliminary data shows | Reuters
C
Created on: 2020-04-08
View text
D
E
African
-Americans dying of coronavirus at higher rates, preliminary data shows | ReutersDiscover Thomson
Third of critically ill COVID-19 UK patients from BME backgrounds | United Kingdom | Al Jazeera
Created on: 2020-04-07
View text
Prev
1
Next
etd
webpage
twitter

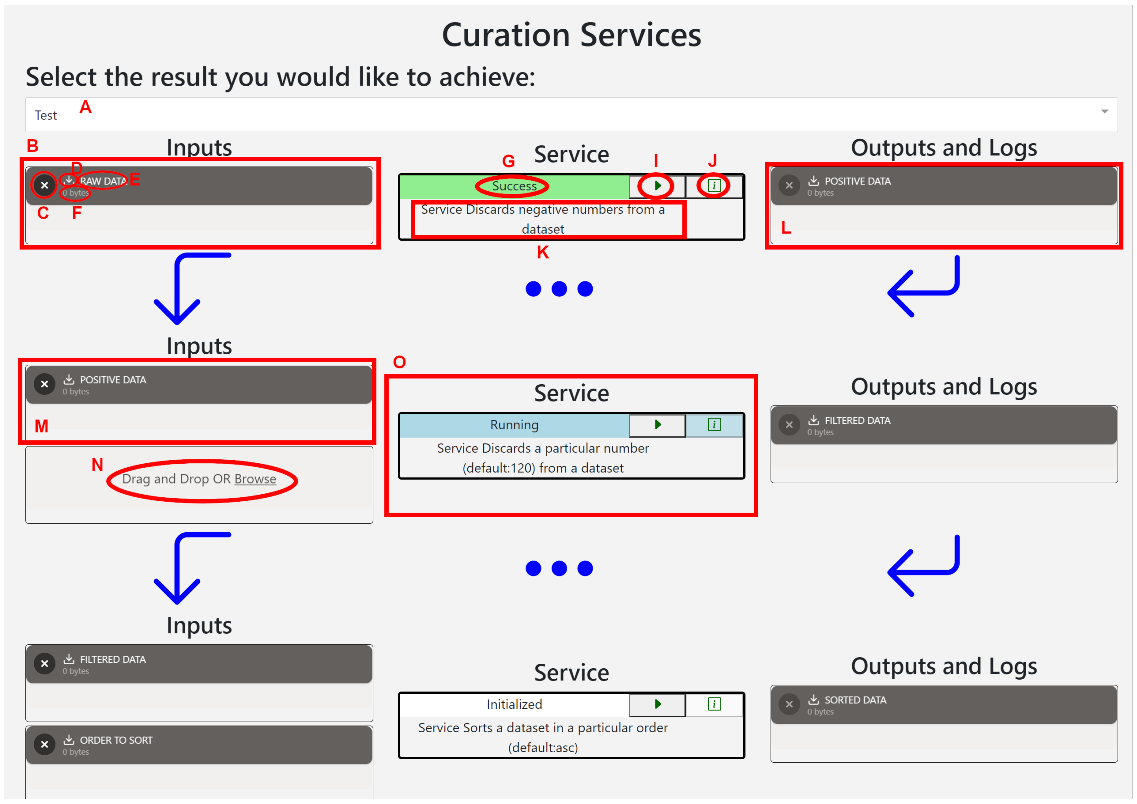
Webpage Search Screen Shot

Figure 13: Screen shot of the current Webpage search screen with all of its functional elements.

5.10 User Manual for KnowledeGraph

Figure 14 shows a screen shot of the "Curation Services" page. This example page allows users with curator privileges or higher to run Airflow services. First, the users will select from a list what they will like to achieve as the final result (A). The designated workflow and its services will be displayed. This particular example of a selected workflow has three services. Each service has an input and an output block (e.g., B and L, respectively). For each input/output block you download the current file (D); see the file description (E) and the size of the current file (F). Files are uploaded/replaced by drag-and-drop of the file on the input boxes (B, M, and N), or browsing the client machine (N) for a file after removing (C) the current file. Note that the output for the first service (L) is automatically used as input to the second service (M). The user can download and check the output, and make changes and re-upload the file if they deem it necessary.

The middle block controls the service. The status (G) and the service description (K) are shown in addition to the ability to (re)start the service (I) and look at the logs for the last run of the service (J). The service status bar will change based on its current status, which is updated every 10 seconds while the service is running (e.g., G and O).



Service Discards negative numbers from a dataset

Service Discards a particular number (default:120) from a dataset

Service Sorts a dataset in a particular order (default:asc)

Curation Services Screen Shot

Figure 14: Screen shot of the current Curation Services screen with all of its functional elements.

5.11 Curator User Manual

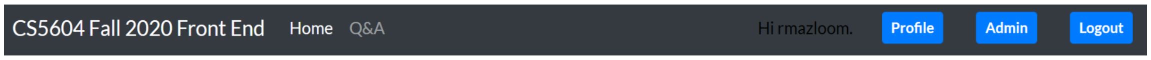
As described in Section 3.4.1, every user has a user type ID in the database, and information that denotes who the user is and what access levels the user has. We currently have three access levels.

- General user
- Curator
- Admin

Everyone at the time of sign up is assigned the default “user” category. A user can read and search on all publicly available indices. If the user is affiliated with VT, they can also read VT tied indices. They do not have write or owner access to these indices by default. A curator has access levels above the general users. They can read, write, and own indices. They can also invite other curators to edit an index. An admin has the highest level of access. They can promote a user from general user to being a curator or admin. They can manage the assignment and re-assignment of indices to curators as well.

The system tries to detect the user-type of the user by connecting to system's database using Flask. Once the user-type has been detected, the system then shows the user specific pages based on permission levels.

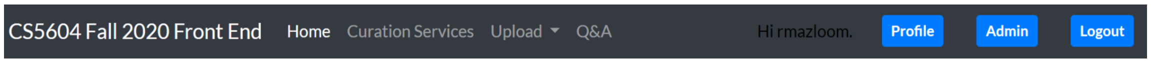
Figure 15 is a page that a general user would see.



General User Screen Shot

Figure 15: Screen shot of the User Interface functionality that a general user sees.

However if a user is detected to be a curator, they would see the page in Figure 16. Curation Services and Upload (A and B) of Figure 4 are additional services that a curator can access.



Curator page Screen Shot

Figure 16: Screen shot of the additional User Interface functionality that a user who is a Curator or Administrator can access.

6 Developer Manual

This section is a useful manual intended for future developers of the system. It describes the details on how to run the system and the major links required for managing the system.

6.1 File Inventory

Figure 17 shows a tree structure of files and folders that are essential for the system to perform. The sections most important to the developers has been annotated.

```
team-fe-repo/
|__ airflow/ ........ Store local Airflow testing directory structure
|__ data/ ........ Store local Elasticsearch sample metadata
|   |__ etd_metadata_sample_v2
|   |   |__ add_data.py ........ Add data to Elasticsearch index
|   |   |__ create_mapping.py ........ Elasticsearch mapping creator (source ETD team)
|   |   |__ etd*.json ........ Final version of the ETD metadata sample
|   |__ airflow*.json ........ Airflow API's sample outputs
|   |__ etd_metadata.json ........ Sample ETD metadata
|   |__ etd_metadata_indexed.json ........ Sample indexed ETD metadata
|   |__ README.md ........ Readme file for adding metadata to Elasticsearch
|   |__ twt_sample_indexed.json ........ Sample indexed TWT metadata
|   |__ wp_sample_indexed.json ........ Sample indexed WP metadata
|__ reactivesearch/ ........ Store code to build Elasticsearch UI
|   |__ public/
|   |__ src/ ........ REACT components of search pages.These require compilation before use
|   |   |__ index.js ........ REACT main default wrapper page
|   |   |__ App.js ........ REACT application main app page
|   |   |__ curator.js ........ Curator services front-end REACT components
|   |   |__ index_etd.js ........ ETD search REACT UI components
|   |   |__ index_twitter.js ........ Twitter search UI components
|   |   |__ index_wp.js ........ Web page search UI components
|   |__ static/
|   |__ templates/
|   |__ README.md
|__ routes/ ........ Record the webpage design and flow logic
|   |__ auth.py
|   |__ dbconnect.py ........ Database connection hanfler
|   |__ knowledgegraph.py ........ Curator services back-end API
|   |__ log.py
|   |__ login.py ........ Login and registration handlers
|   |__ search.py ........ Backend file connecting the search pages to the homepage
|__ static/ ........ Store constant file for the other UI component
|   |__ build/
|   |__ css/
|   |__ fonts/
|   |__ images/
|   |__ js/
|   |__ scss/
|   |__ visualizations/
|   |__ prepros-6.config
|__ templates/ ........ Folder containing all non-REACT pages (e.g. homepage)
|__ tests/
|   |__ test_app.py
|   |__ test_log.py
|   |__ test_login.py
|__ .dockerignore
|__ .gitignore
|__ app.py
|__ docker-compose.yml ........ File used to run Docker image will the required configuration
|__ Dockerfile ........ DockerFile used to create Docker images
|__ flask_monitoringdashboard.db
|__ init_db.sql ........ MySQL database initialization script
|__ README.md ........ Main readme for the project
|__ requirements.txt ........ Python dependencies
```

Figure 17: Tree showing the files/folders in the repository with some annotations on what each one is used for.

6.2 Setting up the Database

- You can download MySQL for your operating system using the links below:
  - Ubuntu
  - CentOS
  - MacOS
  - Windows
- You can download and install Elasticsearch by following the tutorial at https://www.elastic.co/guide/en/elasticsearch/reference/current/install-elasticsearch.html.

6.3 Setting up, Building, and Deploying Reactivesearch Components

In order to run and build the Reactivesearch components of the website, the developers needs to have NodeJs installed. Afterwards, the Reactivesearch project directory should be visited and the following code can be run.

Listing 1: compiling the Reactivesearch components into DOM components

```
npm run build
npm start
```

The first command builds the app for production to the build **folder**. It correctly bundles React in production mode and optimizes the build for the best performance. The build is compressed and the filenames include the hashes. Now your app is ready to be deployed!

The second command then can be used to run the app in the development mode which can be viewed at http://localhost:3000 in the browser.

6.4 Setting up Elasticsearch Locally

Once you have Elasticsearch and MySQL running on your computer, modify your Elasticsearch config file: `elasticsearch.yml` – this can be found in the installation folder. Normally, the path should be `/usr/local/etc/elasticsearch/elasticsearch.yml`

Add the following settings in this file.

Listing 2: Modifying Elasticsearch config file

```
http.cors.enabled: true
http.cors.allow-credentials: true
http.cors.allow-origin: /https?:\/\/(localhost)?(127.0.0.1)
    ?(0.0.0.0)?(:[0-9]+)?/
http.cors.allow-headers: X-Requested-With, X-Auth-Token, Content-
    Type, Content-Length, Authorization, Access-Control-Allow-
    Headers, Accept%
```

The third configuration here allows any browser on the localhost to directly communicate with Elasticsearch. For the VM version of Elasticsearch this needs to allow the container of the front end to have the same privilege. This can be limited to different requests, for instance PUT, GET, DELETE, POST, etc. You can then start Elasticsearch by using the following command `elasticsearch`

6.4.1 Creating simple Elasticsearch indexes with JSON data

A JSON (JavaScript Object Notation) file is a lightweight data-interchange file format. It is easy for humans to read and write, as well as easy for machines to parse and generate.

JSON is built on two structures:

- A collection of name/value pairs. In various languages, this is realized as an object, record, struct, dictionary, hash table, keyed list, or associative array.
- An ordered list of values. In most languages, this is realized as an array, vector, list, or sequence [6].

The ETD sample data in Section A.1 is an example of data in JSON format.

NDJSON (Newline delimited JSON) is a convenient format for storing or streaming structured data that may be processed one record at a time. It works well with Unix-style text processing tools and shell pipelines. It's also a flexible format for passing messages between cooperating processes [13].

Elasticsearch accepts and works with the newline delimited JSON file format. Therefore, it is necessary to convert our JSON sample dataset to the NDJSON file format.

First remove all of the newlines and have each record be on a new line.

Next, above each line add a header with a counting ID.

`{ "index": { "_index": "test", "_type": "type1", "_id": "1"} }`

The following script can then be used to automate it, given each record is in one row, the first and last brackets ([ ]) are removed, and the ending commas are removed after each record.

Listing 3: Automating the JSON to NDJSON conversion

```
c=0;while read p; do \
echo "{ \"index\" : { \"_id\" : \"$c\" } }" >> indexed.json; \
echo "$p" >> indexed.json; \
c=$(($c + 1)); echo $c;done<cleaned.json

# Creating an Index in Elasticsearch
curl -X PUT \
  http://localhost:9200/twts\
  -H 'Content-Type: application/json'\
  -d '{
    "settings" : {
        "index" : {
            "number_of_shards" : 3,
            "number_of_replicas" : 2
        }
    }
}'

# Adding Test Dataset to Elasticsearch
curl -s -H "Content-Type: application/x-ndjson" \
-XPOST http://localhost:9200/30k/\_bulk \
--data-binary @indexed.json; echo

curl -s -H "Content-Type: application/x-ndjson" -XPOST http://localhost:9200/twts/\
--data-binary @clean-sfm-coronavirus-sample-geo.json; echo

# To make sure it went through
curl -s -H "Content-Type: application/x-ndjson" \
-XGET http://localhost:9200/30k/\_count?pretty -d \
'{
  "query": {
    "match_all": {}
  }
}'
# To view indices and documents in your elasticsearch server
visit this link: http://localhost:9200/\_cat/indices

# Delete all documents
curl -s -H "Content-Type: application/x-ndjson" \
-XPOST http://localhost:9200/30k/\_delete\_by\_query?pretty -d \
'{
  "query": {
    "match_all": {}
  }
}'
```

Alternatively, you can use methods similar to the Python scripts in the repository found in *data/etd_metadata_sample_v*2/ ∗ *.py* to add and create mappings using Python.

6.5 Workflow

6.5.1 Elasticsearch Dataflow in ReactiveSearch

There are several UI components in Reactivesearch that are responsible for making search and display of results work on the website. Here we give a brief overview of some of the highly useful elements we utilized. Also Figure 11 is an image that shows them in action [1].

6.5.1.1 DataSearch

DataSearch displays a search input box. It supports autosuggestions, highlighting of results, and querying against more than one field via properties (*component A in Figure 11*).

6.5.1.2 MultiList

MultiList creates a multiple selection based on the list UI component that is connected to a field. It is similar to a SingleList except it can support multiple item selections. SingleList creates a single selection based list UI component that is connected to a database field (D).

6.5.1.3 ResultList

ReactiveList creates a data-driven result list UI component. This list can reactively update itself based on changes in other components or changes in the database itself (O).

6.5.1.4 SingleRange

SingleRange creates a numeric range selector UI component that is connected to a field [1].

6.6 Docker Image Creation and Deployment

To run the web application as a container you need to use the pre-built image from the 2020 front-end repository [21]. This section will cover how to build and eventually deploy the image for production. Having Docker installed, and the Docker Demon, is a prerequisite to build an image.

6.6.1 Preparations

First you need to prepare an empty directory and pull the front end team's repository [21] `git clone https://git.cs.vt.edu/cs-5604-fall-2020/fe/team-fe-repo/` and navigate to the cloned repository's directory `cd team-fe-repo`.

Afterwards, you can use the included DockerFile (shown in Listing 4) to build the Docker image using the command `docker build . -t TAGNAME`. Tagname is the name that Docker saves the image under. You can assign multiple tags to a Docker image using `docker tag existing_tag new_tag`.

Listing 4: Web application DockerFile used to create the application Docker image.

```
FROM node:latest

ENV LANG=C.UTF-8 LC_ALL=C.UTF-8
ENV PATH /opt/conda/bin:$PATH

RUN wget --quiet \
https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh \
    -O ~/miniconda.sh && \
    /bin/bash ~/miniconda.sh -b -p /opt/conda && \
    rm ~/miniconda.sh && \
    /opt/conda/bin/conda clean -tipsy && \
    ln -s /opt/conda/etc/profile.d/conda.sh /etc/profile.d/conda.sh && \
    echo ". /opt/conda/etc/profile.d/conda.sh" >> ~/.bashrc && \
    echo "conda activate base" >> ~/.bashrc && \
    find /opt/conda/ -follow -type f -name '*.a' -delete && \
    find /opt/conda/ -follow -type f -name '*.js.map' -delete && \
    /opt/conda/bin/conda clean -afy

COPY . /app

WORKDIR /app/reactivesearch

RUN /opt/conda/bin/conda init bash && \
        . /root/.bashrc && \
        conda create -n env python=3.6 && \
        conda activate env && \
        pip install -r /app/requirements.txt && \
        npm install

WORKDIR /app

EXPOSE 3000

CMD . /root/.bashrc && \
        cd /app/reactivesearch && \
        npm run build && \
        cd /app && \
        conda activate env && \
        env && \
        python app.py
```

We use the latest NodeJs container as a base (Listing 4 line 1), install Miniconda3 using the first RUN command, create a Conda environment and install the required Python and NodeJS packages using the second RUN, and eventually reserve building the React components and running the web application using the CMD command for when the image is started as a container.

6.7 Container Deployment

Using the built image, either from the repository or the previous section, a container can now be run. To do so, you need to configure several virtual environments for the container.

- REACT_APP_BASE_URI   web application URI to be accessible

REACT_APP_ELASTICSEARCH_URI	elasticsearch_URI
REACT_APP_ES_SECRET	Elasticsearch credentials username:password
FLASK_DB	database_type (mysql/postgres)
FLASK_DB_STRING	Format: hostname;port;username;password;database_name
REACT_APP_ES_MAIN_ETD	elasticsearch_etd_index_name
REACT_APP_ES_MAIN_TWT	elasticsearch_twitter_index_name
REACT_APP_ES_MAIN_WP	elasticsearch_webpage_index_name

Your Docker run command will look similar to Listing 5 below.

Listing 5: Docker run command sample with environment variables.

```
docker run -d -rm \
-e REACT_APP_BASE_URI=http://0.0.0.0:3000 \
-e REACT_APP_ELASTICSEARCH_URI=http://localhost:9200 \
-e REACT_APP_ES_SECRET=username:password \
-e FLASK_DB=mysql \
-e FLASK_DB_STRING='localhost;3306;username;password;db' \
-e REACT_APP_ES_MAIN_ETD=fe-etd \
-e REACT_APP_ES_MAIN_TWT=fe-twt \
-e REACT_APP_ES_MAIN_WP=fe-wp \
team-fe-repo:latest --name team-fe-webapp -p 3000:3000
```

When the container is run successfully you can access the page at your designated base URI (default: http://0.0.0.0:3000).

A Sample Data

A.1 ETD Sample Record

Listing 6: Example of a raw ETD metadata record

```
[{
  "contributor-author": "Bailey, Justin Mark",
  "contributor-committeechair": "O'Brien, Walter F",
  "contributor-committeemember": [
   "Dancey, Clinton L",
   "Lowe, Kevin T",
   "Wicks, Alfred L",
   "Ng, Wing Fai"
  ],
  "contributor-department": "Mechanical Engineering",
  "date-accessioned": "2017-01-06T13:34:06Z",
  "date-available": "2017-01-06T13:34:06Z",
  "date-issued": "2017-01-05",
  "degree-discipline": "Mechanical Engineering",
  "degree-grantor": "Virginia Polytechnic Institute and State University",
  "degree-level": "doctoral",
  "degree-name": "PHD",
  "description-abstract": "Future commercial transport aircraft will feature more
      aerodynamic architectures to accommodate stringent design goals for higher
      fuel efficiency, reduced cruise and taxi NOx emissions, ...",
  "description-degree": "PHD",
  "description-provenance": [
   {
    "description-provenance-summary": [
     "Made available in DSpace on 2017-01-06T13:34:06Z (GMT). No. of bitstreams1
         Bailey_JM_D_2017.pdf9128042 bytes, checksum7438e886322739e17247ed2c907658b0
          (MD5)   Previous issue date2017-01-05"
    ]
   },
   {
    "Author Email": [
     "jmb@vt.edu"
    ]
   },
   {
    "Advisor Email": []
   }
  ],
  "format-medium": "ETD",
  "handle": "73987",
  "identifier-other": "vt_gsexam:9274",
  "identifier-uri": "http://hdl.handle.net/10919/73987",
  "publisher-none": "Virginia Tech",
  "rights-none": "This item is protected by copyright and/or related rights. Some
      uses of this item may be deemed fair ...",
  "subject-none": [
   "Experimental Engine Testing",
   "Distortion",
   "Interaction",
   "Total Pressure",
    "Boundary Layer Ingesting"
  ],
  "title-none": "Full Scale Experimental Transonic Fan Interaction with a Boundary
      Layer Ingesting Total Pressure Distortion",
  "type-none": "Dissertation"
}]
```

Listing 7: Example of a cleaned ETD metadata record (complements to the ETD team)

```
[{
  "contributor_author": "Gong, Hao",
  "contributor_committeechair": "Cao, Guohua",
  "contributor_committeemember": [
    "Zhu, Yizheng",
    "LaConte, Stephen Michael",
    "Wyatt, Chris L",
    "Wang, Ge"
  ],
  "chapters": [
    {"1": [1,3]},
    {"2": [3,5]},
    {"3": [5,7]},
    {"4": [7,9]}
  ],
  "contributor_department": "Biomedical Engineering",
  "date_accessioned": "2017-02-17T09:00:17Z",
  "date_available": "2017-02-17T09:00:17Z",
  "date_issued": "2017-02-16",
  "degree_discipline": "Biomedical Engineering",
  "degree_grantor": "Virginia Polytechnic Institute and State University",
  "degree_level": "doctoral",
  "degree_name": "PHD",
  "description_abstract": "The current cardiac computed tomography (CT) technology
      is mainly limited by motion blurring and radiation dose. The conceptual multi-
      source interior CT scheme has provided a ...",
  "description_degree": "PHD",
  "description_provenance": [
    "Author Email: haog1@vt.edu",
    "Made available in DSpace on 2017-02-17T09:00:17Z (GMT). No. of bitstreams: 3
        Gong_H_D_2017_support_1.pdf: 150830 bytes, checksum: ..."
  ],
  "format_medium": "ETD",
  "handle": "75054",
  "identifier_other": "vt_gsexam:9743",
  "identifier_uri": "http://hdl.handle.net/10919/75054",
  "publisher": "Virginia Tech",
  "rights": "This item is protected by copyright and/or related rights. Some uses of
       this item may ...",
  "subject": [
    "Multi-source",
    "computed tomography",
    "scatter correction",
    "temporal resolution",
    "interior tomography"
  ],
  "title": "A Scheme for Ultra-Fast Computed Tomography Based on Stationary Multi-
      Beam X-ray Sources",
  "type": "Dissertation",
  "etd_uri": "/mnt/ceph/cme/sample_metadata/75054/Gong_H_D_2017.pdf",
  "figures_folder_uri": "/mnt/ceph/cme/sample_metadata/75054/figures/",
  "figures_total_count": 38,
  "tables_folder_uri": "/mnt/ceph/cme/sample_metadata/75054/tables/",
  "tables_total_count": 0
}]
```

A.2 TWT Sample Record

Listing 8: Example of a tweet metadata record

```
{
  "quote_count": 0,
  "contributors": null,
  "truncated": true,
  "text": "The phoney quarantine is almost over. Bring on the real quarantine. #covid_19 #coronavirus #freebritneyu2026 https://t.co/BJTCp8fI6y",
  "is_quote_status": false,
  "in_reply_to_status_id": null,
  "reply_count": 0,
  "id": 1257271308062208000,
  "favorite_count": 0,
  "entities": {
    "user_mentions": [],
    "symbols": [],
    "hashtags": [
      {
        "indices": [
          68,
          77
        ],
        "text": "covid_19"
      },
      {
        "indices": [
          78,
          90
        ],
        "text": "coronavirus"
      },
      {
        "indices": [
          91,
          103
        ],
        "text": "freebritney"
      }
    ],
    "urls": [
      {
        "url": "https://t.co/BJTCp8fI6y",
        "indices": [
          105,
          128
        ],
        "expanded_url": "https://twitter.com/i/web/status/1257271308062208000",
        "display_url": "twitter.com/i/web/status/1u2026"
      }
    ]
  },
  "retweeted": false,
  "coordinates": {
    "type": "Point",
    "coordinates": [
      -80.25,
      43.55
    ]
  },
  "timestamp_ms": "1588591813470",
  "source": "<a href=\"http://instagram.com\" rel=\"nofollow\">Instagram</a>",
  "in_reply_to_screen_name": null,
  "id_str": "1257271308062208000",
  "retweet_count": 0,
  "in_reply_to_user_id": null,
  "favorited": false,
  "user": {
    "follow_request_sent": null,
    "profile_use_background_image": true,
    "default_profile_image": false,
    "id": 1096083799534960640,
    "default_profile": true,
    "verified": false,
    "profile_image_url_https": "https://pbs.twimg.com/profile_images
        /1251785729518288897/csccyZlo_normal.jpg",
    "profile_sidebar_fill_color": "DDEEF6",
    "profile_text_color": "333333",
    "followers_count": 125,
    "profile_sidebar_border_color": "C0DEED",
    "id_str": "1096083799534960640",
    "profile_background_color": "F5F8FA",
    "listed_count": 0,
    "profile_background_image_url_https": "",
    "utc_offset": null,
    "statuses_count": 1828,
    "description": "The subject who is truly loyal to the Chief Magistrate will
        neither advise nor submit tou00a0arbitrary measures.~~ Junius",
    "friends_count": 428,
    "location": "Dawn-Euphemia, Ontario",
    "profile_link_color": "1DA1F2",
    "profile_image_url": "http://pbs.twimg.com/profile_images/1251785729518288897/
        csccyZlo_normal.jpg",
    "following": null,
    "geo_enabled": true,
    "profile_banner_url": "https://pbs.twimg.com/profile_banners
        /1096083799534960640/1587283898",
    "profile_background_image_url": "",
    "name": "Mafun Ho",
    "lang": null,
    "profile_background_tile": false,
    "favourites_count": 15588,
    "screen_name": "Tumulus17",
    "notifications": null,
    "url": null,
    "created_at": "Thu Feb 14 16:28:36 +0000 2019",
    "contributors_enabled": false,
    "time_zone": null,
    "protected": false,
    "translator_type": "none",
    "is_translator": false
  },
  "geo": {
    "type": "Point",
    "coordinates": [
      43.55,
      -80.25
    ]
  },
  "in_reply_to_user_id_str": null,
  "possibly_sensitive": false,
  "lang": "en",
  "extended_tweet": {
    "display_text_range": [
      0,
      160
    ],
    "entities": {
      "user_mentions": [],
      "symbols": [],
      "hashtags": [
        {
          "indices": [
            68,
            77
          ],
          "text": "covid_19"
        },
        {
          "indices": [
            78,
            90
          ],
          "text": "coronavirus"
        },
        {
          "indices": [
            91,
            103
          ],
          "text": "freebritney"
        },
        {
          "indices": [
            104,
            118
          ],
          "text": "wrayandnephew"
        }
      ],
      "urls": [
        {
          "url": "https://t.co/PF3a1rtMMG",
          "indices": [
            137,
            160
          ],
          "expanded_url": "https://www.instagram.com/p/B_w6qFEnTYF/?igshid=1
              kb2euuf2aszb",
          "display_url": "instagram.com/p/B_w6qFEnTYF/u2026"
        }
      ]
    },
    "full_text": "The phoney quarantine is almost over. Bring on the real quarantine. #covid_19 #coronavirus #freebritney #wrayandnephew @ Guelph, Ontario https://t.co/PF3a1rtMMG"
  },
  "created_at": "Mon May 04 11:30:13 +0000 2020",
  "filter_level": "low",
  "in_reply_to_status_id_str": null,
  "place": {
    "full_name": "Guelph, Ontario",
    "url": "https://api.twitter.com/1.1/geo/id/2740624a2d391c5c.json",
    "country": "Canada",
    "place_type": "city",
    "bounding_box": {
      "type": "Polygon",
      "coordinates": [
        [
          [
            -80.326879,
            43.473802
          ],
          [
            -80.326879,
            43.594596
          ],
          [
            -80.153377,
            43.594596
          ],
          [
            -80.153377,
            43.473802
          ]
        ]
      ]
    },
    "country_code": "CA",
    "attributes": {},
    "id": "2740624a2d391c5c",
    "name": "Guelph"
  }
}
```

A.3 WP Sample Record

Listing 9: Example of a web-page metadata record

```
[{
  "URL": "https://www.theguardian.com/science/2020/apr/07/cancer-research-uk-to-cut-
      funding-for-research-by-44m?CMP=share_btn_tw",
  "webpage_text": "\nCancer charities say coronavirus shortfall will set back
      research | Science | The Guardian\nSkip to main content\nThe Guardian - Back
      to home\nSupport The Guardian\nAvailable for everyone, funded by readers\
      nContribute\nSubscribe\nContribute\nSearch jobs\nSign in\nMy account\nAccount
      overview\nBilling\nProfile\nEmails & marketing\nSettings\nHelp\nComments &
      replies\nSign out\nSearch\nswitch to the \nUS edition\nswitch to the \nUK
      edition\nswitch to the \nAustralia edition\nswitch to the \nInternational
      edition\ncurrent edition: \nUS edition\nnews\nOpinion\nSport\nCulture\
      nLifestyle\nShow\nMore\nnews\nUS news\nElections 2020\nWorld news\nEnvironment\
      nSoccer\nUS politics\nBusiness\nTech\nScience\newsletters\nOpinion\nThe
      Guardian view\nColumnists\nLetters\nOpinion videos\nCartoons\nSport\nSoccer\
      nFL\nTennis\nMLB\nMLS\nBA\nHL\nCulture\nFilm\nBooks\nMusic\nArt & design\nTV &
       radio\nStage\nClassical\nGames\nLifestyle\nFashion\nFood\nRecipes\nLove & sex
      \nHome & garden\nHealth & fitness\nFamily\nTravel\nMoney\nWhat term do you
      want to search?\nSearch with google\nMake a contribution\nSubscribe\nUS
      edition\nswitch to the \nUK edition\nswitch to the \nAustralia edition\nswitch
       to the \nInternational edition\nSearch jobs\nDigital Archive\nGuardian
      Puzzles app\nThe Guardian app\nVideo\nPodcasts\nPictures\nInside the Guardian\
      nGuardian Weekly\nCrosswords\nFacebook\nTwitter\nSearch jobs\nDigital Archive\
      nGuardian Puzzles app\nUS\nElections 2020\nWorld\nEnvironment\nSoccer\nUS
      Politics\nBusiness\nTech\nScience\newsletters\nMore\nCancer research\n ...",
  "title": "Cancer charities say coronavirus shortfall will set back research |
      Cancer research | The Guardian",
  "date": "2020-04-07"}
}]
```

B Tasks and Responsibilities

Task	Description	Start	End	Responsible
1	Getting 2019 code running	IR1	IR1	Reza
1.1	Flask	IR1	IR1	Reza
1.1.1	Setting up required packages/environment/configurations for debugging	IR1	IR1	Reza
1.1.2	Fixing hardcoded database columns	IR1	IR1	Reza
1.2	Database	IR1	IR1	Reza
1.1.1	Reverse engineer database schema from Python code	IR1	IR1	Reza
1.1.2	Create schema and add database initializer to repository	IR1	IR1	Reza
1.3	Debugging and testing login/register (full stack)	IR1	IR1	Reza
2	Reactivesearch	IR2	IR2	Reza
2.1	Fixed and updated broken NodeJS dependencies in package.json	IR2	IR2	Reza
2.2	Fixed Dynaconf connection for Flask/React configuration	IR2	IR2	Reza
2.3	Cleaned old code and removed/fixed all errors in code	IR2	IR2	Reza
2.4	Cleaned old code and removed all errors	IR2	IR2	Reza
2.5	Added Bootstrap capabilities to entire project	IR2	IR2	Reza
2.6	Added use of custome CSS in React components	IR2	IR2	Reza
3	Local Elasticsearch	IR2	IR2	Reza
3.1	Fixed major CORS error stopping external web requests to Elasticsearch	IR2	IR2	Reza
3.2	Cleaned ETD data and imported to Elasticsearch	IR2	IR2	Reza
3.2	Cleaned TWT data and imported to Elasticsearch	IR2	IR2	Yusheng
3.4	Created Elasticsearch data initialization README	IR2	IR2	Reza
4	ETD page	IR2	IR2	Reza
4.1	Redesigned page using sample data	IR2	IR2	Reza
4.2	Added new functionality such as voice search and categorized dynamically populated filters	IR2	IR2	Reza
4.3	Added collapsing abstract	IR2	IR2	Reza
4.4	Added highlighting from Elasticsearch and fixed it's color	IR2	IR2	Reza
4.4	Add new facets based on the ETD teams request	IR4	IR4	Reza
5	Containerized web app	IR3	IR3	Reza
5.1	Created DockerFile and Image	IR3	IR3	Reza
5.2	Colleborated to set up permissions on user table on INT team's PostgreSQL container	IR3	IR3	Reza
5.3	Added PostgreSQL connections to project in addition to MySQL	IR3	IR3	Reza
5.4	Set all external locations to environment variables	IR3	IR3	Reza
5.5	Deployed Container on cloud.cs.vt.edu	IR3	IR3	Reza
5.6	Colleborated to make container available on VT network	IR3	IR3	Reza
6	TWT page	IR2	IR3	Yusheng
6.1	Create new page based on sample data	IR2	IR2	Yusheng
6.2	Add Filter functions for the search	IR2	IR3	Yusheng
6.3	Design search result based on the existing twitter page layout	IR3	IR3	Yusheng
6.4	Add dynamic filters to fit the final Elasticsearch indices	IR3	IR4	Yusheng
7	Internal permissions and management	IR4	IR4	Yusheng
7.1	Upload existing table for user roles	IR4	IR4	Yusheng
7.2	Creating new manage page for Admin users to manage existing users	IR4	IR4	Yusheng
7.3	Link Kibana page for Admin users	IR4	IR4	Yusheng
7.4	Record user's role during login session and share to other pages	IR4	IR4	Yusheng
7.5	Add module filters in each page for different users	IR4	IR4	Yusheng
8	Webpage UI	IR3	IR3	Ola
8.1	Cleaned sample data and converted to njson format needed for Elasticsearch	IR3	IR3	Ola
8.2	Extracted more metadata(title, date) in addition to the sample data	IR3	IR3	Ola
8.3	Created UI page for webpage search	IR3	IR3	Ola
8.4	Added collapsing and scrollable ability for the webpage text	IR3	IR3	Ola
8.5	Added and fixed date search feature	IR3	IR3	Ola
8	Curator Page	IR4	IR4	Ola
8.1	Create UI for curators	IR4	IR4	Ola
8.2	Add selection box for Users to select what index to search	IR4	IR4	Ola
8.3	Manage index content access and permissions based on user's privileges/affiliations	IR4	IR4	Ola
9	KnowledgeGraph and Airflow	IR4	IR4	Reza
9.1	Create web interface with dynamic inputs i.e. textbox/files etc	IR4	IR4	Reza
9.2	Design Airflow job status reporting mechanism	IR4	IR4	Reza
9.3	Connect to Ceph for download/uploading files	IR4	IR4	Reza
9.4	Connect to Airflow to run jobs from the content teams	IR4	IR4	Reza

Table 2: Description and details of major steps for the project with timed milestones and individual responsibility

References

[1] Reactive Search Components overview. https://opensource.appbase.io/reactive-manual/getting-started/componentsindex.html, May 2020. Accessed: 2020-10-08.

[2] Reactive Search README.md. https://github.com/appbaseio/reactivesearch#:~:text=ReactiveSearch%20is%20a%20Elasticsearch%20UI,UI%20component%20into%20the%20library, 2020. Accessed: 2020-11-23.

[3] Tweet geospatial metadata. https://developer.twitter.com/en/docs/tutorials/tweet-geo-metadata#:~:text=Twitter%20Places%20can%20be%20thought, 2020. Accessed: 2020-10-07.

[4] Cayabyab, T. A. C. A review of emerging ETD initiatives, challenges and future developments. *International Journal of Information and Education Technology 5*, 10 (2015), 772.

[5] Cormack, J., Ragunathan, A., and Crone, C. Why Docker? — Docker. https://www.docker.com/why-docker, 2019. Accessed: 2020-10-08.

[6] Crockford, D. Json. *ECMA International* (2012). http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf. Accessed: 2020-12-04.

[7] Domes, S. Everything You Should Know About React: The Basics You Need to Start Building. https://www.freecodecamp.org/news/everything-you-need-to-know-about-react-eaedf53238c4/, Nov 2017. Accessed: 2020-11-01.

[8] DuBois, P. *MySQL*. Addison-Wesley Professional, 2013.

[9] Elastic. ElasticSearch Ranking Evaluation API. https://www.elastic.co/guide/en/elasticsearch/reference/current/search-rank-eval.html, 2019. Accessed: 2020-10-07.

[10] Fox, E. A. Fall 2020 FE Team Description. https://canvas.vt.edu/courses/115585/pages/f2020-teamfe, Aug. 2020. Accessed: 2020-08-31.

[11] Gormley, C., and Tong, Z. *Elasticsearch: the definitive guide: a distributed real-time search and analytics engine.* " O'Reilly Media, Inc.", 2015.

[12] Grinberg, M. *Flask web development: developing web applications with python.* " O'Reilly Media, Inc.", 2018.

[13] Hoeger, T., Dew, C., Pauls, F., and Wilson, J. Newline Delimited JSON Specifications. https://github.com/ndjson/ndjson-spec, Nov 2016. Accessed: 2020-12-04.

[14] Manning, C. D., Raghavan, P., and Schütze, H. *Introduction to Information Retrieval.* Cambridge University Press, USA, 2008.

[15] Meno, E., and Vincent, K. Twitter-Based Knowledge Graph for Researchers. https://vtechworks.lib.vt.edu/handle/10919/98239. Accessed: 2020-10-07.

[16] Powell, E., Liu, H., Huang, R., Sun, Y., and Chao, X. CS5604InfoStorageFEK. https://github.com/BourneXu/CS5604InfoStorageFEK, Dec. 2019. Accessed: 2020-08-31.

[17] Powell, E., Liu, H., Huang, R., Sun, Y., and Chao, X. Front-End Kibana (FEK) CS5604 Fall 2019. Accessed: 2020-09-07.

[18] Pranav, C., Hicks, M., Ikjot, J., Manisha, K., Ujjval, M., Akash, P., and Irith, S. Tweet Collection Management IR1 Report. https://canvas.vt.edu/files/14852530/download?download_frd=1. Accessed: 2020-10-07.

[19] Satheesh, M., D'mello, B. J., and Krol, J. *Web development with MongoDB and NodeJs*. Packt Publishing Ltd, 2015.

[20] Vtechworks. ETDs: Virginia Tech Electronic Theses and Dissertations. https://vtechworks.lib.vt.edu/handle/10919/5534, Sept. 2020. Accessed: 2020-09-14.

[21] Yusheng, C., Mazloom, R., and Makanjuola, O. CS 5604 Fall 2020 - Team FE Repo. https://git.cs.vt.edu/cs-5604-fall-2020/fe/team-fe-repo/, 2020. First Accessed: 2020-09-14.

[22] Zhai, C. Massung, S. *Text data management and analysis: a practical introduction to information retrieval and text mining*. Morgan & Claypool, 2016.