NORTH SOUTH UNIVERSITY

DEPARTMENT OF ELECTRICAL & COMPUTER ENGINEERING

CSE299 Junior Design

Project Report
Section: 5
Group: 7

Project Title: RE-SEARCH

Date of Submission: April 18, 2022
Submitted To: Meem Tasfia Zaman

Group Members:

Name	Student Id
Nazmul Hasan	1911742042
Samya Sunibir Das	1911563642
Mahir Afser Pavel	1911580642

Table of Contents

Table of Figures

RE-SEARCH

A cluster based search engine

1. Abstract

In this data-driven technological era, there is an abundance of data readily accessible to all with the help of search engines. But users, especially researchers and such may face hurdle to get specific results for their need with the overwhelming amount of data available. This is where RE-SEARCH, a cluster-based search application comes in to play as its main purpose is to let its users to search in their custom-made clusters of links with the type of files the users are looking for. Re-search is designed to make searching very specific for its users. Users will be able to set their own clusters of links and will be able to define search methods based on textual file types such as: pdf, txt, doc etc. Users then will be able to search for keywords within the clusters to get intended results. The software serves to make researching easier for its users as other conventional search engines can often provide overwhelming number of results or data that can make the research very cumbersome.

2. Introduction

Re-search is an application that enables user to search within website of their selection. Typical Google search comb through the entire internet. This can be useful for most cases but for certain case like research, data manipulation restricted search can be useful. A sample search result for keyword ‘nsu’ shows search result like this:

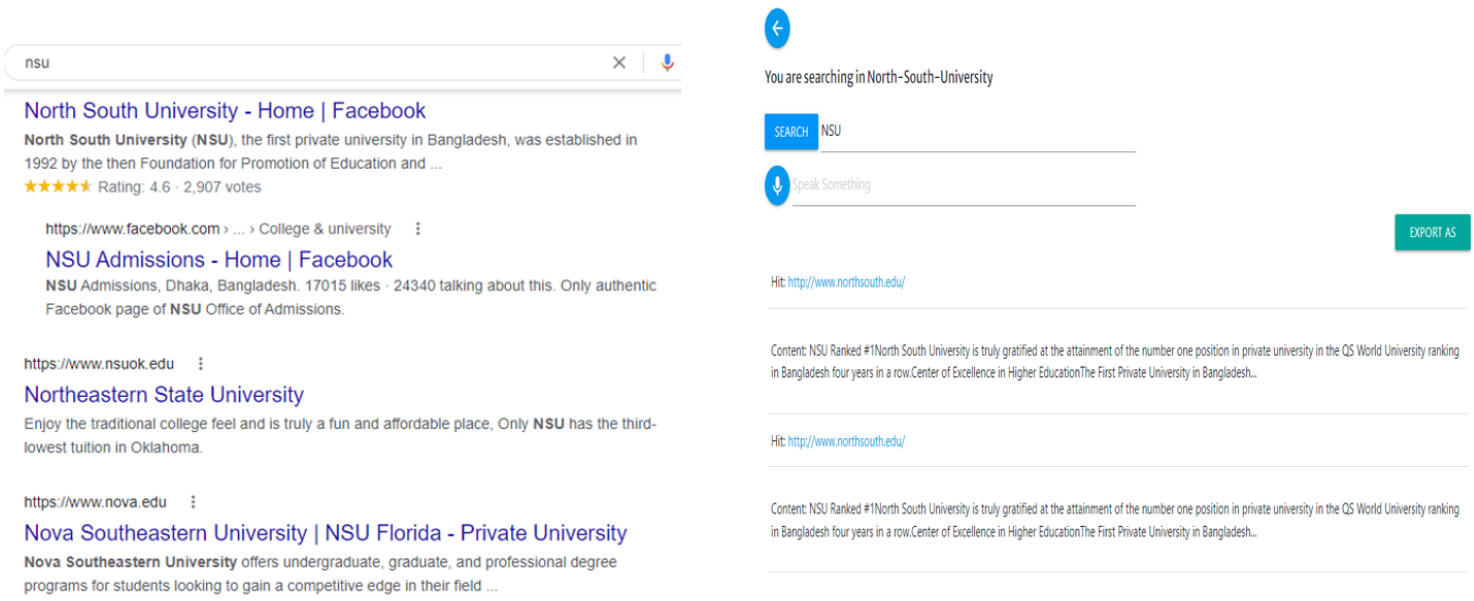

Figure 1: A sample search result in Google vs NSU

This is how most popular search engines show more data than the needed one. Sometimes it takes more time to find out accurate information. Here comes our RE-SEARCH web application to help its users to find accurate results in a more efficient way. RE-SEARCH crawls non-html contents from webpages, pdf, doc, ppt, and txt files from given urls following Data Crawling ethics, and index scrapped data in such a way so that an accurate search result can be shown and takes less time. A conventional search engine cannot provide search results based on the depths users set, whereas RE-SEARCH will provide that. Conventional search engines may also not have any option of keeping search results structured for future usage whereas RE-SEARCH will provide that in the form of search clusters.

3. Background / Literature review

RE-SEARCH aims to solve the problem of data redundancy in search results. In the background section, we will answer two important questions that dissects into the current state of research in this application field and the future problems that we might need to tackle upon deployment.

What are the similar applications or research works conducted in the past?

There are no similar applications currently providing active service like our application would be. No current popular search engines provide such flexibility to the users as RE-SEARCH would. The research works that were conducted involving clustering and search engines, were based around clustering of the obtained search results using various clustering algorithms, none had the similar traits, such as the users being able to provide their own parameters to search, and perform search in those clusters as RE-SEARCH would have. There was a research work conducted in 2011, *CONQUIRO*, a cluster-based information retrieval search engine that used Machine Learning algorithms which proposed a solution to the problem of information management since as the research work would go beyond just a ranked list of documents like Google. Still, this did not come to much fruition or developed further. Our app would be using the market leading search engine, Elasticsearch.

What are the problems that we will have to tackle?

As the users will be able to provide their own links to crawl to get search results, there is the concern that arises with spamming. Users will not be able to spam and misuse our application as we would implement scheduled crawling, which means, users' crawling requests shall be executed only in fixed intervals set in our system. There are also websites that explicitly do not allow to scrape their webpages, as in such cases the users should always be aware of such policies. The users should also adhere to data crawling ethics, as repeated requests of the web pages can slow down the webserver. Websites may contain materials with copyright claims. Making copies of those copyright materials is considered as an illegal activity. So, web crawling should be done at a disruptive rate by maintaining website's terms and conditions so that it does not slow down the website and hamper its regular activities.

4. Aims & Goals

- User friendly interface on the frontend which would boost up RE-SEARCH and make searching easier and faster.
- The users are in charge here, and set their own parameters and search for their desired content. No redundant results which might waste the user's time.
- One of the primary goals of our project is to provide a safe, secured search experience. We do not use or make profit by selling any of the user's data secretively or deceptively, and only require just their e-mail IDs and a password to register or to login via their other social accounts.
- RE-SEARCH makes searching accurate and easy for its users with no strings attached. Users can find whatever they are looking for with ease and its even enjoyable to use.
- The web application RE-SEARCH would also aim to fulfill other quality attributes such as:
  Availability: This software will be available to use at all times for its users as the data would be indexed in a 3rd party service, Elasticsearch and would also provide search functionalities. Temporary unavailability may happen for scheduled maintenance.
  Correctness: This software will provide accurate results based on the links the users provide in clusters and the keywords they will use when they need to search.
  Reusability: The users will be able to reuse and search many times on the clusters that they create.

5. Application of the project in real life

As stated before, RE-SEARCH is made to best serve the researchers in various sectors such as academia, private research companies or non-profit organizations. This will also help the teachers, the students as well, as it would be a very useful application in finding specific topics of interests for them inside the search parameters they set, for example finding very niche research papers from an abundance of other results which may not be helpful for them. It would also be incredibly efficient in collecting data as this search application lets the users to crawl their own links that they provide and scrape data within the given depth for an individual link. The users of this application are going to enjoy keeping their data, content they would search for in the form of search clusters, this will provide better ease of use and convenience. As this application provides searching for different types of files to search through, the users will be able to extract text and search within those as well. The potential of this web application would be very high and the app would be very scalable in future to improve with further development and cater to a massive amount of userbase.

6. Tools needed (software)

- **Programming/Markup Languages:** Django (Python framework for the backend), HTML, CSS, and JavaScript (for the frontend)

  Django was the framework that glued everything together in the project. Django was used to created models which are tables in the database. Django views were made used of to render http requests from the frontend. Django Rest Framework was used to build the APIs. HTML and CSS were the main two technologies that were used to build the frontend. Javascript was used to implement voice searching functionality.

- **Libraries (Python):** Textract (For extracting data), Django-allauth (For user authentication), Beautifulsoup (for scraping and crawling)

  Powerful Python libraries like Beautifulsoup, Textract will be used to build our crawler which will crawl the links based on the inputs and the input parameters the users set.

- **Database:** Sqlite3

  Django's default Sqlite3 database is used in the project to store relevant data/models.

- **Search Engine:** Elasticsearch.

  Elasticsearch is the core of the project. It was used to provide searching functionalities inside the clusters. Elasticsearch was also used to store the scraped content in Elasticsearch indexes.
- **External APIs/Services:** Paypal API, Gmail API.

  Paypal API helped facilitating donation options to support the developers of this project. Gmail API was used to enable easy social login feature for the users to use this app.

- **IDE/Text Editor:** Visual Studio Code

  VS Code was the only code editor that was used in the development of this project.

7. Tentative Schematic Diagram

The following is the UML use case diagram for the software RE-SEARCH. This shows the different types of users the software will have and the features the users shall have access to. Different types of users shall have multiple common features among them. The Client user here refers to our core user-base, our customers, to whom we will cater our software to. They will be able to authenticate their own accounts, able to customize their profiles, make search clusters by providing links to search in. Then, they will be able to do full text search in their clusters. The admin user will be in control of the webapp, will be able to maintain the database powered by the default SQLite database of Django in the backend. Will also be able to add new plugins for scraping. Meanwhile the System will handle the scraping, indexing scraped data to Elasticsearch, and handle authentication.

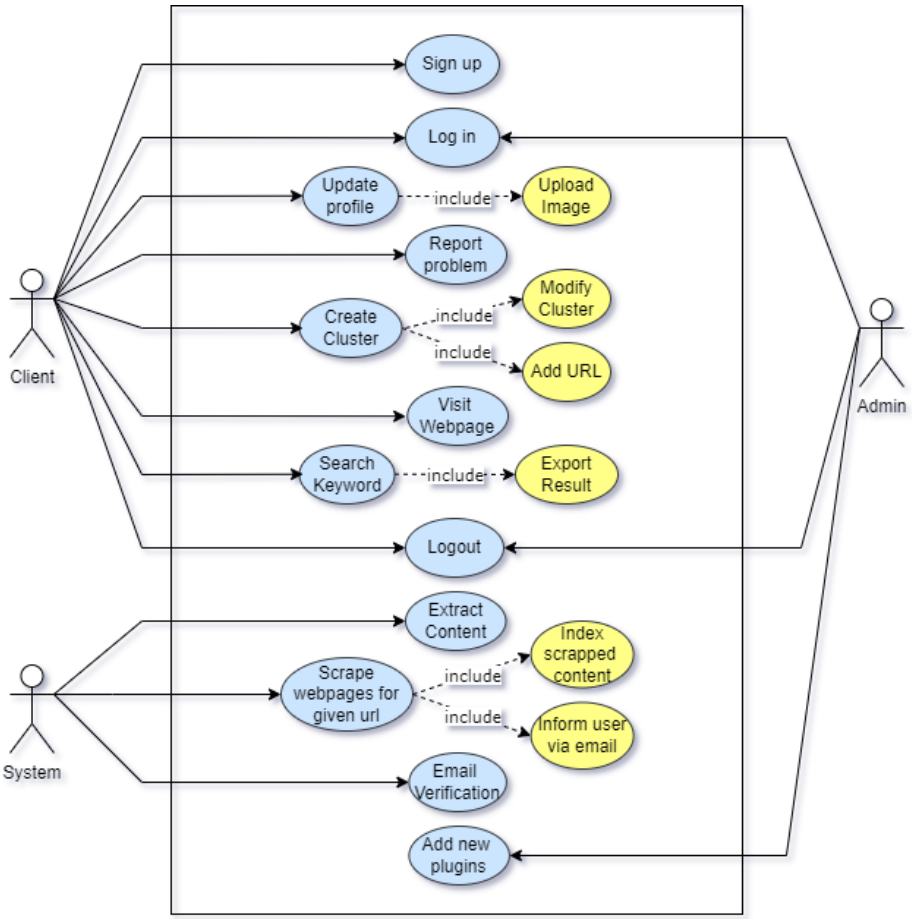

Figure 2: Use-case diagram for RE-SEARCH

8. Projected Plan

We start working for our project by designing the UI/UX for Login, Signup, Logout, then progressively work through our project by adhering to the flowchart given below, and we finish the project work by Testing, Debugging. The start, and end processes are colored in Green while the other steps are colored in Blue in the flowchart.

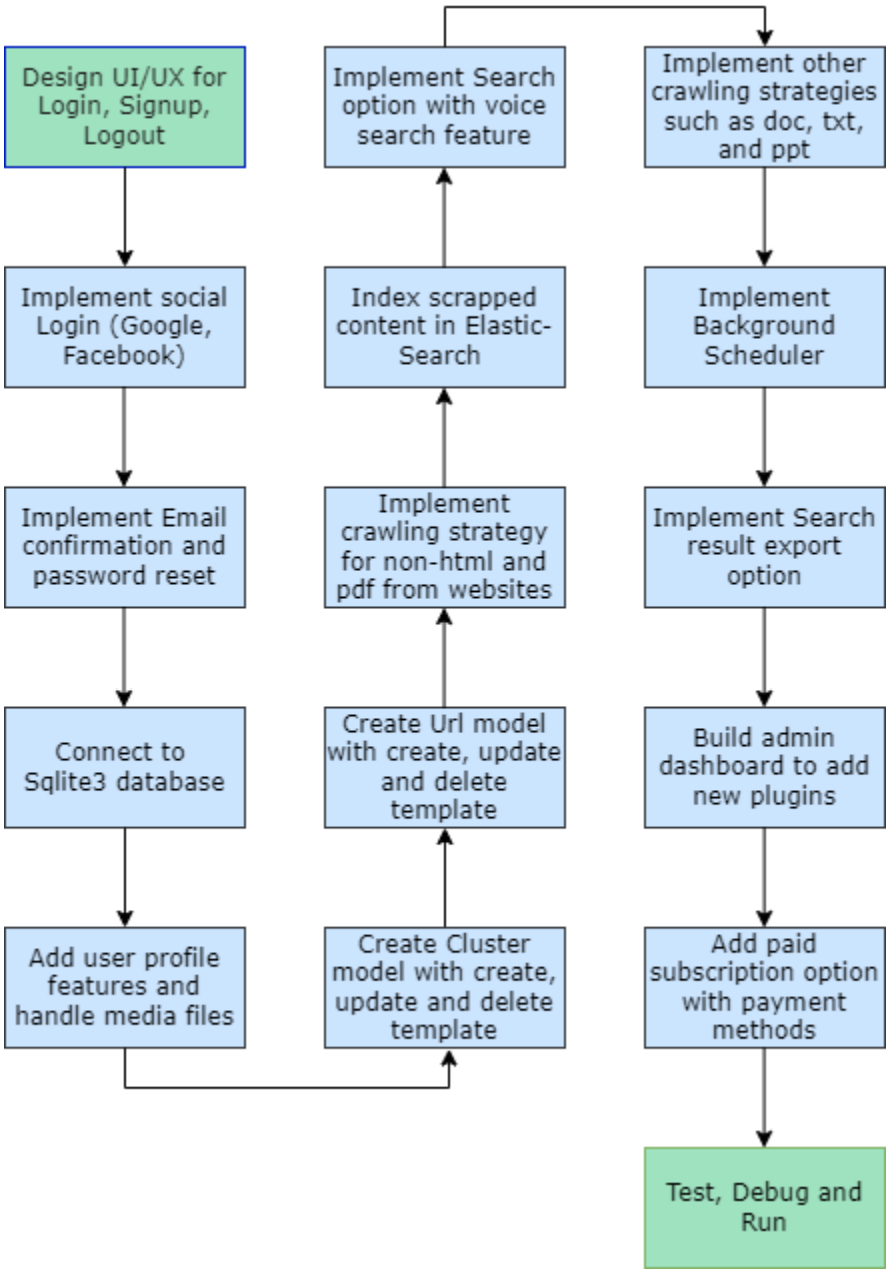

Figure 3: Project Plan

9. Projected Timeline

The following is the Gantt chart of the project. We have set up a five-week timeline for the project. We will start the project by designing UI for Signup, login, logout and user profile modifications. The complex part of the project is covered in week 2 and 3. The key feature (searching) is developed in week 4. Then in week 5, the UI is improved and tested the project for deployment, and necessary APIs are built.

Figure 4: Gantt chart for RE-SEARCH

10. Design and Implementation

Design

The design part of the project was completely conducted by following the structure of Django framework. The main project directory being Googleauth folder, where the important plugins and definitions of most of tools used were defined and added in the settings.py file. The urls.py file contained the necessary urls to be added to the project.

The main workings of the project were divided into two separate, independent apps. The app base_app contains most of the functionalities of the project which are separate python files themselves. The export.py file would be the file where the function for exporting search results as csv files are defined. Models.py would contain necessary database models which are Cluster, Url, these are both classes that were injected into the Database as Tables. Schedule.py, updater.py would handle background scheduling for scraping. Scrape.py would contain the crawler for the project. The views.py file is the most crucial here as it would contain the class-based definitions for all the functionalities available for Clusters, and Url models. The same file also would contain the most important search function, and the function for handling payments.

The API related works are done in the API app, with own views and serializers.

The templates folder would contain other subfolders which would have all of the necessary html files for the frontend.

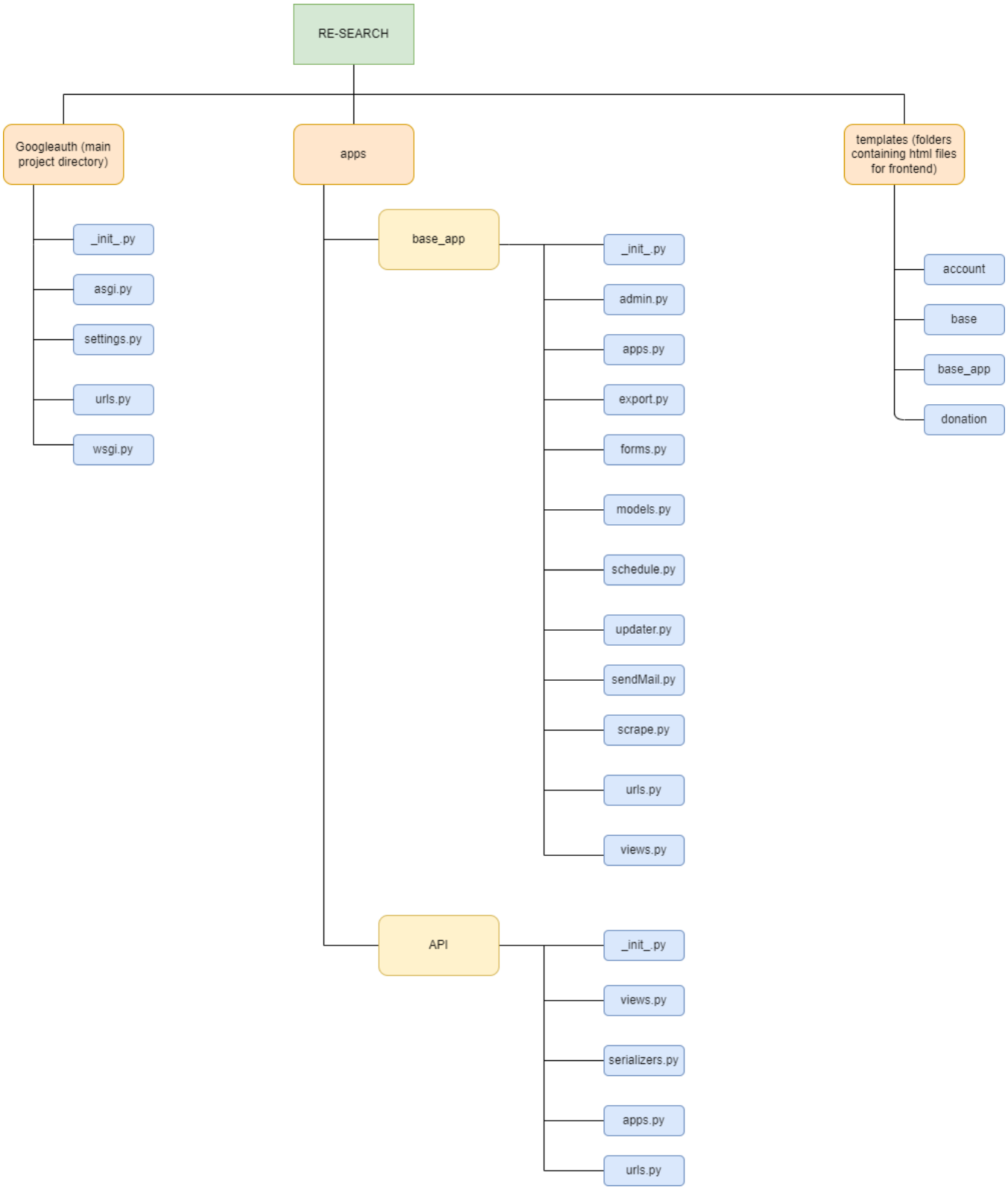

Figure 5: Design and the structure of RE-SEARCH

Implementation

The implementation adhered to the design and the programming was conducted accordingly. The functions, folders, classes, files all were named in accordance of what they would do, and are self-explanatory. Structured accordingly

The googleauth folder is the main project directory where all of the installed apps' definitions are defined in settings.py, and urls.py has all the necessary urls to be added. Django Allauth and Gmail API were added and helped implementing the all of the authentication related parts.

The base_app has most of the functionalities of the project. As explained in the design part, the models.py contains the necessary classes to be injected in database. Similarly, the Url model was defined as well. These two are crucial entities which lets the users to create and search within clusters and add, remove urls. The views.py file was used to define all of the relevant functionalities related to Url and Cluster models. Class-based views were opted for these implementations as it was convenient and more usable. Classes related to deleting, updating, creating, details of a particular cluster were defined in the classes called ClusterDelete, ClusterUpdate, ClusterCreate, ClusterDetails. Similar classes were made for the functionalities related to Url model as well. Search function is defined in the views.py as well. This was designed in the form of function-based views as a class-based view was not working to implement search. Search function made calls to the corresponding elasticsearch indexes and returned relevant search results for the input keywords. The scraping.py file had the crawler and the indexing function to index scraped content to Elasticsearch. The level_crawler, crawl functions were the two necessary functions to handle crawling. The level_crawler function handled crawling at next level, including the types of files it would scrape. The crawl function handled depth crawling. The scraping and crawling part was implemented with the help of beautifulsoup, textract. The elastic_indexer indexed all the scraped data to particular elasticsearch indexes for particular clusters. The base_app folder also has schedule and updater.py which let background scheduling with an interval set for 5 minutes.

The API app was built in similar manner, had the necessary API views related to Cluster and Url models, ClusterAPI, URL_API are created as class-based views. Their corresponding serializers are also defined in serializers.py. APIs related to registering and user info are also defined in serializers.py and views.py. Django Rest framework was used to implement all of these.

Finally, all of the views were rendered with the help of the html files located in the templates folder. For every particular view, their corresponding html files were created and implemented. All of the account authentication html files can be found in the account file. Html and CSS were used. Voice search function was implemented by Javascript in the html file for cluster_search.

11. Result and Evaluation

After implementing all of the necessities following the design by using all the required tools, the project, RE-SEARCH was worked on and concluded week by week and step by step.

All of the authentications: social login, login, sign-up, sign out, resetting passwords, adding secondary e-mails, verifying e-mail IDs were handled using Django Allauth. The corresponding html files were designed and implemented according with the help of relevant tools. The following is a snapshot of the UI of the login page.

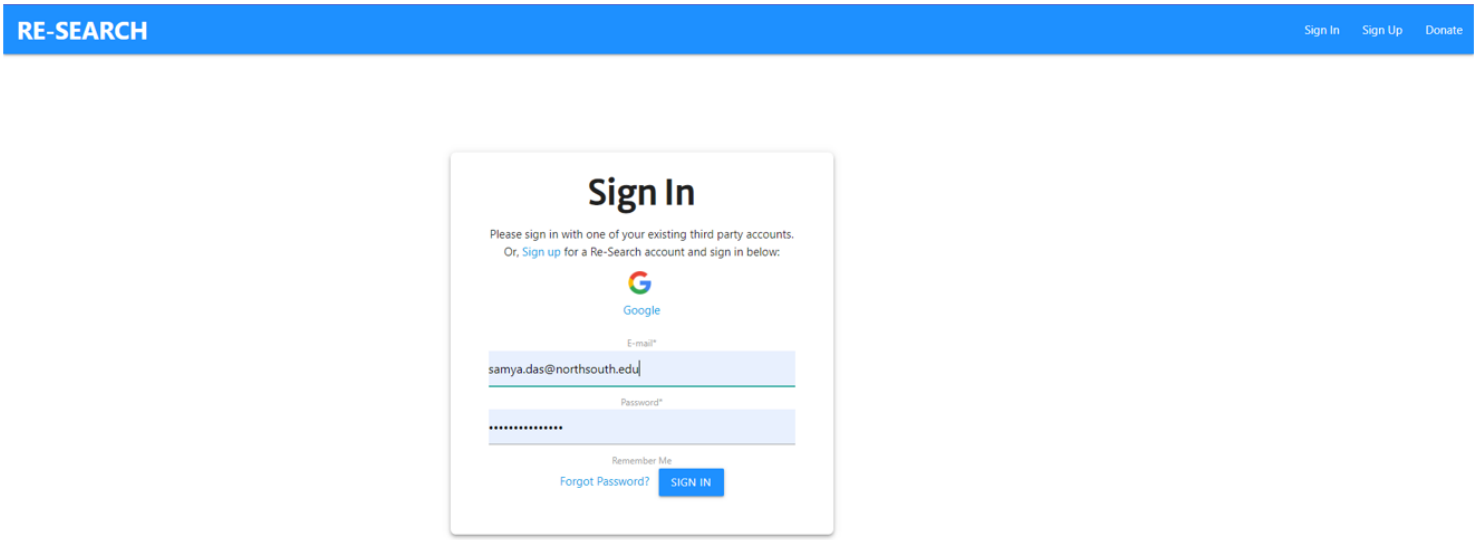

Figure 6: UI of the Login Page

After a user successfully has logged in to the system, whether by logging in or registering for the first time, the user is able to successfully create “search clusters” with a title and the description for a particular cluster initially. Then the user can create more clusters if they want to. The cluster view page is successfully created with relevant views and html files. There are other options such as search, view, edit, delete are kept on the UI, and they both successfully render those requests with the help of the relevant views and corresponding html pages. The cluster view page has pagination support, and will have multiple pages of clusters if the total amount of clusters exceeds for a particular page. The saved clusters will be stored into the database for particular users in the form of Cluster table, which was a model as we have implemented.

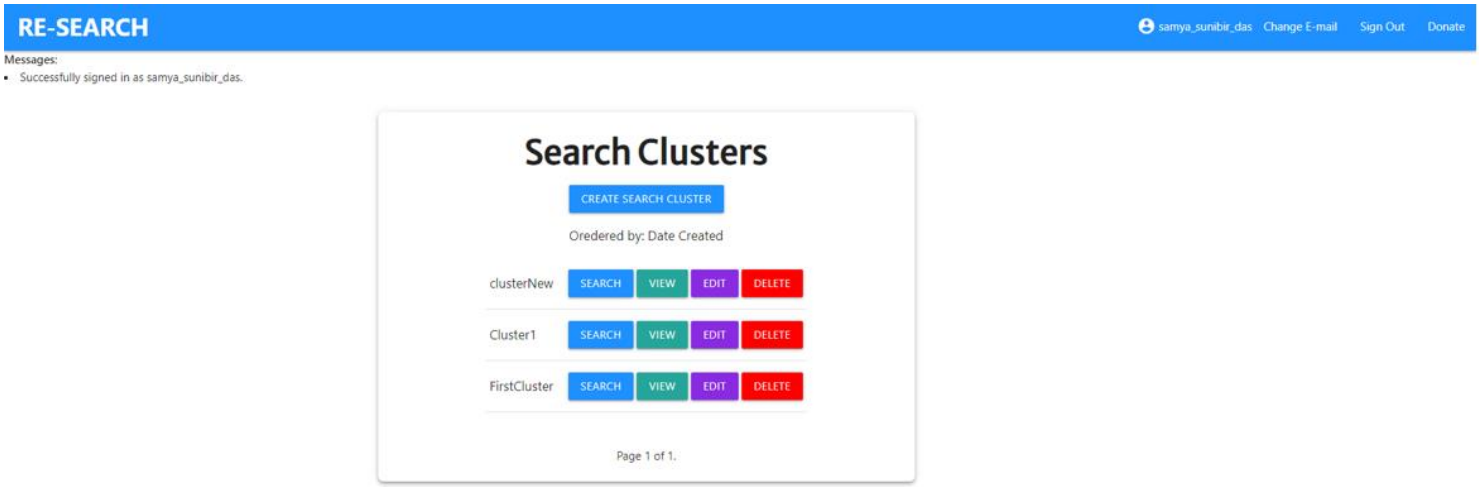

Figure 7: Cluster view page for an individual user

The user will be able to add a URL, define the depth of crawl, and the type of data they would like to scrape/crawl. This option is found on the view option for a particular cluster. The URLs and their relevant info are stored in the DB. The scraping triggers upon a set interval after an URL is added. The depth logic, crawling strategies are all defined in the scrape.py file.

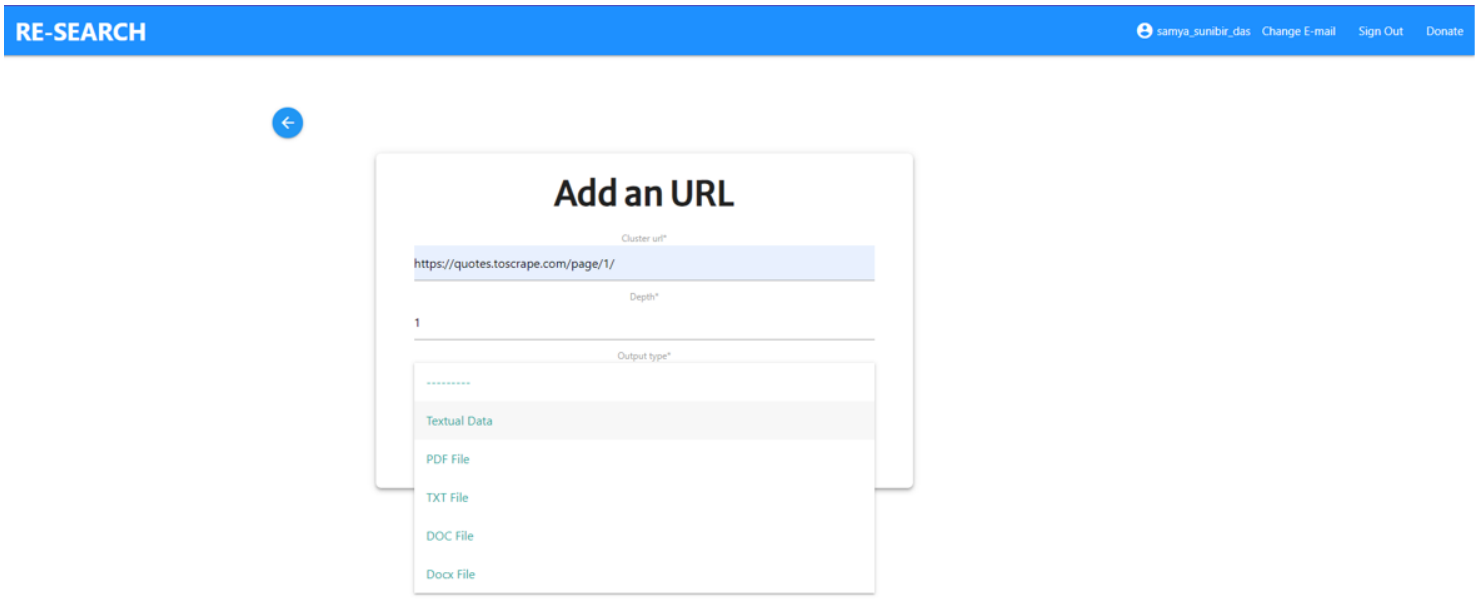

Figure 8: Adding URLs and setting output types in a Cluster

Users can now add multiple URLs inside a cluster and set their parameters accordingly. All of the scraped content is saved in individual Elasticsearch indexes for individual clusters.

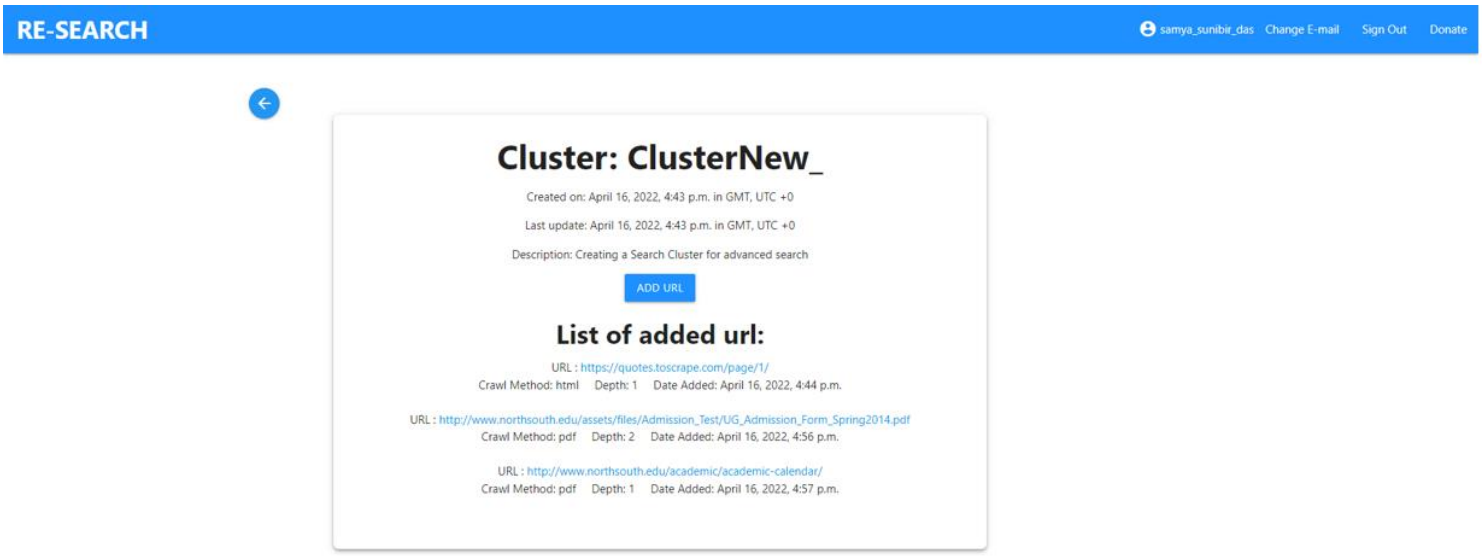

Figure 9: A cluster with multiple links and types of outputs set by the user

After scraped content is successfully stored in an index, users can perform searching inside those indexes for the data they scraped. The search function works perfectly and returns accurate hit links for a search query and the desired content as well. The users will be able to export the obtained search results as a CSV file. The CSV file will contain the links as of now, but can be easily implemented including or just the content itself by little adjustment. There is voice searching implemented as well inside the clusters, and users shall be able to conveniently just speak at their microphones and perform fast, accurate searching with voice commands. This was implemented using a simple Javascript function.

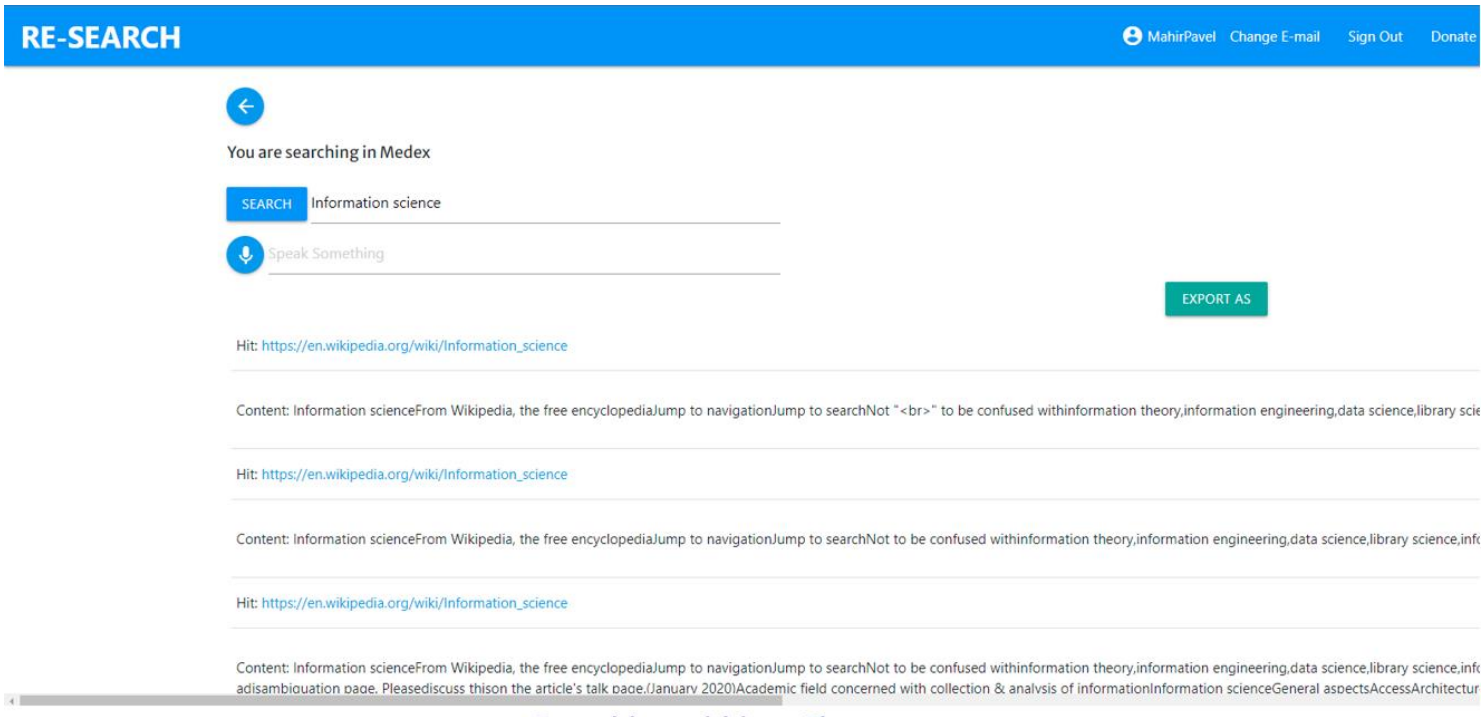

Figure 10: Searching within a Cluster

As evaluated, all of the main functionalities of the project are working seamlessly. Now, there is an option to donate has also been implemented powered by the Paypal API and the users may donate if they wish to support.

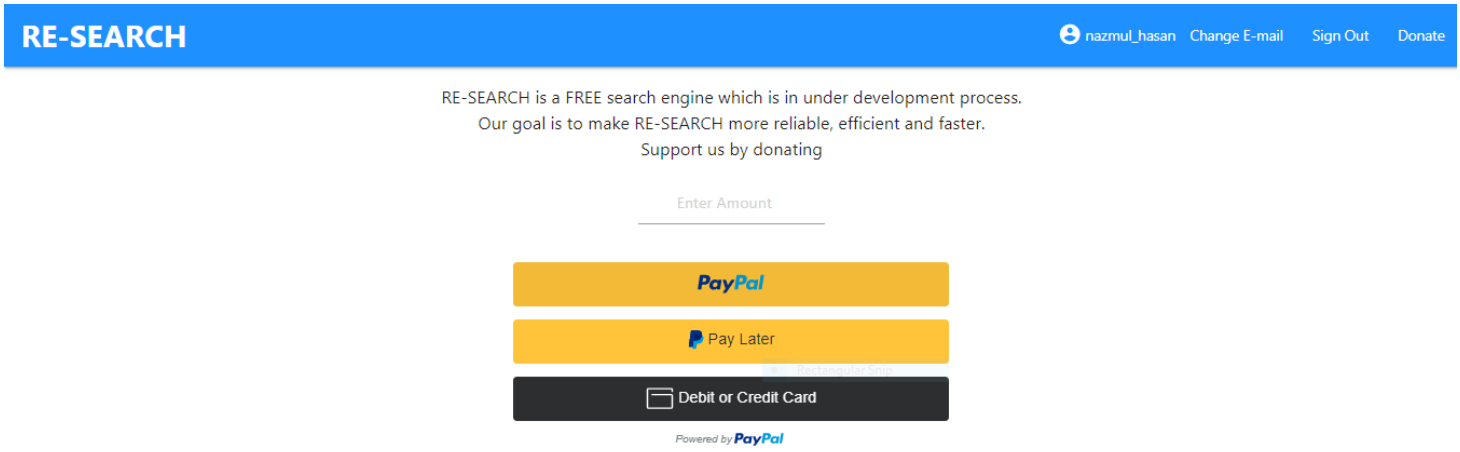

Figure 11: Donations portal with Paypal Gateway

12. Future Work

Development of API

The development of the necessary APIs is finished. The APIs that are successfully built so far are: RegisterAPI, ClusterAPI, UrlAPI, UserAPI. These APIs would provide for registering users, making clusters, adding URLs even outside of the web app. The APIs will work as a middle man between the default database and other platforms. These would be crucial in future to provide cross-platform support for other operating systems such as Android, and iOS.

Plans for deployment

The plan is to make this software open for all to use and in order to do that, deployment should be the next step. And as mentioned before, we will provide multi-platform support for Web, Android.

13. Conclusion

The goal of the project is to build a more user friendly search engine that provides a safer, more secured, and smoother searching experience. That is why RE-SEARCH is designed to make searching very specific for its users. It saves the valuable time for its users. Features like easy to use and attractive user interface, voice searching, wildcard searching, less populated search result and exporting search results make RE-SEARCH different from other search engines available in web.

14. **References**

[1] IBM, "Use-case diagrams," 04 03 2021. [Online]. Available: https://www.ibm.com/docs/en/rational-soft-arch/9.6.1?topic=diagrams-use-case.

[2] D. A. N. Amol Rajmane, "Cluster Based Web Search," *INTERNATIONAL JOURNAL OF ADVANCED RESEARCH IN COMPUTER SCIENCE AND ELECTRONICS ENGINE ERING*, pp. 36-40, 2012.

[3] Y. M. D. MariaVargas-Vera, "CONQUIRO: A cluster-based meta-search engine," 2011.

[4] J. WILLIAMS, "Data Crawling Ethics and Best Practices," [Online]. Available: https://www.promptcloud.com/blog/data-crawling-and-extraction-ethics/?fbclid=IwAR1U815roNgcK-WrU9hfWunrrZCW4oN6cPeFjnGUjupcrPSiIki9m2vdK0w.

[5] R. Abueg, "Elasticsearch: What It Is, How It Works, And What It's Used For," 7 3 2020. [Online]. Available: https://www.knowi.com/blog/what-is-elastic-search/.