Amazon Kendra NLP Search for ICAI (Version 2)

About Challenge

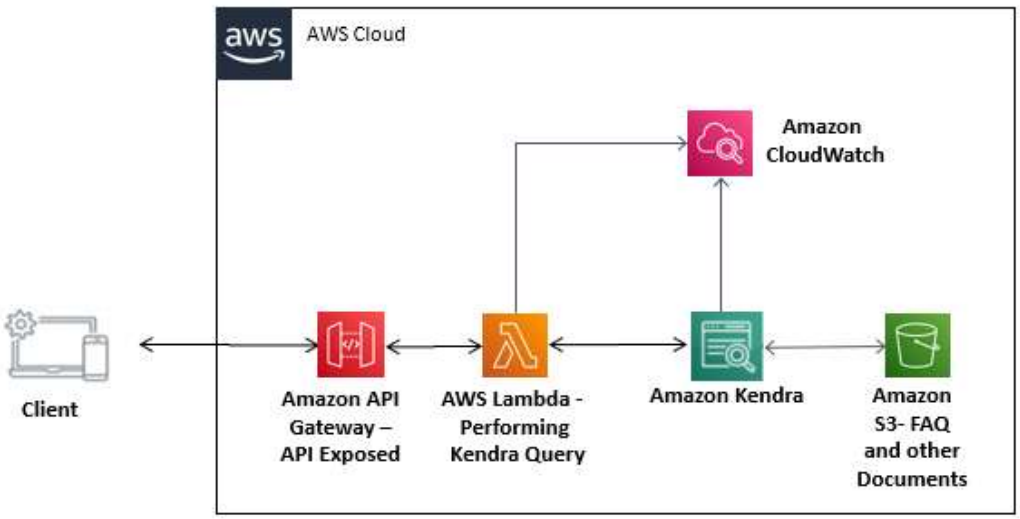
ICAI wanted a Search Engine functionality (Cognitive Searching) for contents of the icai.org on which Search is to be enabled (FAQs/HTML/Word/ppt/pdf). The main challenges faced by the customer in their current environment is that there is no NLP based Search based functionality available. So, the customer desires highly robust and cost-effective solution capable of delivering a quality end user experience regardless of the demands on the platform using best in breed AWS services to achieve this goal. This is where Amazon Kendra comes into picture as it supports cognitive search, i.e. Natural language search and not the traditional keyword search which is offered by services like Elasticsearch.

Key Objective

To retrieve relevant content using semantic search based on the concept of Natural Language processing (NLP). Below are the key objectives:

- ✓ NLP based Cognitive Searching of the contents namely Publications, Journals, Students FAQs, Members FAQs stored in Amazon S3.
- ✓ Enabling user level restriction for each category

Architecture Diagram



Crux of Architecture Diagram-

1. Client is the webpage from where the user types the query like “What is ICAI?” and selects the categories like “Journals, Members”.
2. Two parameters- selected user type, search query is sent to API Gateway.

3. API gateway sends these parameters to Lambda function. Lambda function has the code for Kendra API call “kendra.query” having parameters- index id, search query and user type.
4. In Kendra service, index is setup and an ACL(Access Control List) config file is used which maps user type to corresponding S3 document subset in Kendra index.
5. The search results are then optimized by using the tagging technique which comprises of creating metadata files(having attributes like category and tags) corresponding to every document.
6. AWS Cloudwatch is used to keep a track of error logs.

AWS Services used

✓ Amazon Kendra
✓ AWS S3
✓ Amazon API Gateway
✓ AWS Lambda
✓ Amazon Cloudwatch

Proposed Solution

The approach for ICAI Kendra Search PoC is described below.

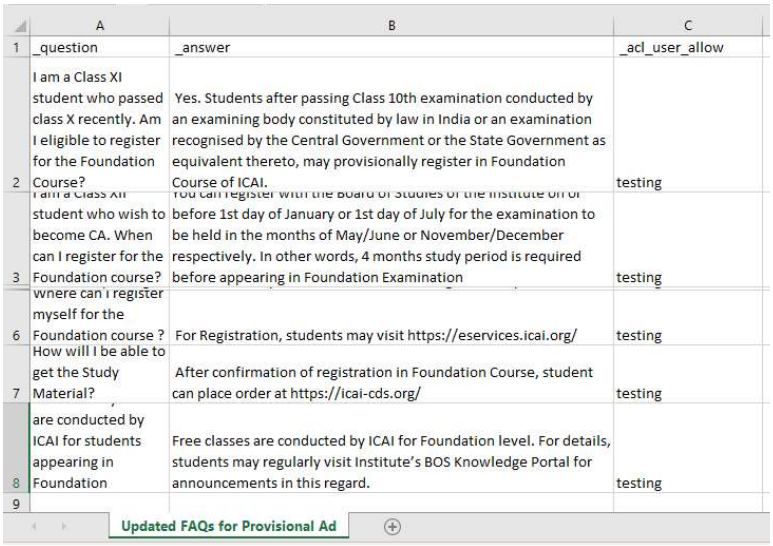
1. ICAI account has an S3 bucket having category-wise documents in its subfolders and category-wise searching is to be performed. All FAQ’s and contents are there in Amazon S3 and Access to contents in Amazon S3is given.

2. S3 Structure-

For Journals
- ICAI e-Journal (August, 2021 - PDF Download)
- ICAI Journal (Archives)
- Journal Highlights (Archives)

For Students
- FAQs in respect of Revised Scheme of Education and Training
- FAQs for Provisional Admission to Foundation Course
- FAQs - BoS - (22-04-2019)

For Members:
- ICAI - The Institute of Chartered Accountants of India
- Accounting Standards Board
- Auditing & Assurance Standards Board
- Expert Advisory Committee
- Committee on Insolvency & Bankruptcy Code
- Committee on International Taxation
- Corporate Laws & Corporate Governance Committee
- Ethical Standards Board



	A	B	C
1	_question	_answer	_acl_user_allow
2	I am a Class XI student who passed class X recently. Am I eligible to register for the Foundation Course?	Yes. Students after passing Class 10th examination conducted by an examining body constituted by law in India or an examination recognised by the Central Government or the State Government as equivalent thereto, may provisionally register in Foundation Course of ICAI.	testing
3	I am a Class XII student who wish to become CA. When can I register for the Foundation course?	You can register with the Board of Studies of the Institute on or before 1st day of January or 1st day of July for the examination to be held in the months of May/June or November/December respectively. In other words, 4 months study period is required before appearing in Foundation Examination	testing
6	Where can I register myself for the Foundation course ?	For Registration, students may visit https://eservices.icai.org/	testing
7	How will I be able to get the Study Material?	After confirmation of registration in Foundation Course, student can place order at https://icai-cds.org/	testing
8	are conducted by ICAI for students appearing in Foundation	Free classes are conducted by ICAI for Foundation level. For details, students may regularly visit Institute's BOS Knowledge Portal for announcements in this regard.	testing
9			

Updated FAQs for Provisional Ad

Fig 2. FAQ file- customized by adding _acl_user_allow

3. Every Kendra index comprises of configuring the data sources that connect to the S3 buckets having document store. After you create your index, for particular case of Student FAQ you can add your FAQ data.

- On the Amazon Kendra console, choose the new index.
- Choose FAQs.
- Choose Add FAQ.
- For FAQ name, enter a name, such as student-FAQ.
- For FAQ file format, choose JSON file.
- For S3, browse Amazon S3 to find the Student FAQ folder
- Choose the custom CSV file.
- For IAM role, choose Create a new role to allow Amazon Kendra to access your S3 bucket.
- For Role name, enter a name and choose Add.

```
[
    {
        "keyPrefix": "s3://caartha/pdfOCR/",
        "aclEntries": [
            {
                "Name": "pulkit",
                "Type": "USER",
                "Access": "ALLOW"

            }
        ]
    },
    {

        "keyPrefix": "s3://caartha/FAQStudents/",
        "aclEntries": [
            {
                "Name": "testing",
                "Type": "USER",
                "Access": "ALLOW"

            }
        ]
    }

]
```

Fig 3. ACL config.json file

4. We have controlled access to documents in an S3 data source using a configuration file. We specify the file in the console, the configuration file contains a JSON structure that identifies an S3 prefix and lists the access settings for the prefix. The prefix can be a path, or it can be an individual file and the access settings apply to all of the files in that path. We specify both users and groups in the access settings. When query is fired from frontend to the index, we specify user and group information.

5. AWS Lambda to run integrated code serverless for backend to call Kendra API's. The AWS Lambda function serves the API Gateway parses the path parameters and headers and issues an Amazon Kendra query call with AttributeFilters set to the username from the path parameter, the user access level, and category from the headers. Amazon Kendra returns the FAQs and documents for that particular category and filters them by the user access level and category. The Lambda function constructs a response with these search results and sends the FAQ and document search results back to the client application.

6. For selecting multiple categories in checkbox, the search results will display results in ranking order inclusive of all json responses as per the below page ranking algorithm which ensures that results are shown from all selections.

Initial: Page 01 List - User query + Drop down numbering capture
Page 02 List – User query + Drop down numbering capture

- Identify user based on above selections using dictionary
- Send query request as that particular user to corresponding API Gateway
- Respective Lambdas are called parallelly
- Append output from Kendra responses using below logic:
    - Store length of response 1/2/3/….../n in variable x1/x2/x3/…/xn
    - For every n
        Rank()= Response_1/variable x1
        Increasing_Sort(Rank)
        Append(Rank_List)
- Display on Page 2(Rank_List)

```
#logic of shuffling starts
total_results  = len(global_grouping_content)
all_results_length = [len(global_grouping_content[i]) for i in range(total_results)]
min_results  = min(all_results_length)

# extracting the relevant contents

shuff_content=[]
shuff_title=[]
shuff_links=[]
shuff_pages=[]
if    min_results in list(range(3)):
    for i in range(total_results):
        shuff_content.extend(global_grouping_content[i][:3])
        shuff_links.extend(global_grouping_links[i][:3])
        shuff_pages.extend(global_grouping_pages[i][:3])
        shuff_title.extend(global_grouping_titles[i][:3])
    for i in range(total_results):
        shuff_content.extend(global_grouping_content[i][3:])
        shuff_links.extend(global_grouping_links[i][3:])
```

```
else:
    ranges  = min_results//2
    for i in range(total_results):
        shuff_content.extend(global_grouping_content[i][:ranges])
        shuff_links.extend(global_grouping_links[i][:ranges])
        shuff_pages.extend(global_grouping_pages[i][:ranges])
        shuff_title.extend(global_grouping_titles[i][:ranges])
    for i in range(total_results):
        shuff_content.extend(global_grouping_content[i][ranges:])
        shuff_links.extend(global_grouping_links[i][ranges:])
        shuff_pages.extend(global_grouping_pages[i][ranges:])
        shuff_title.extend(global_grouping_titles[i][ranges:])




title = shuff_title
content = shuff_content
filtered_page_numbers = shuff_pages
filtered_links = shuff_links
```

Fig 4. Search results ranking on multiple selections at a time

7. API Gateway is setup for frontend backend interaction/ Rest API which calls a Lambda function. Basically, an API Gateway endpoint is required that is called by the client application.

8. For this, UI webpage is having two things- a search box and a category selection option from drop down. These two parameters- search box query string and drop-down category selected are sent to API Gateway and via Lambda, depending on above incoming parameters, the search query is directed to Kendra index.

9. Deployment of Flask application to Elastic Beanstalk is done. Flask is basically an open-source web application framework for Python which is deployed to an AWS Elastic Beanstalk environment.

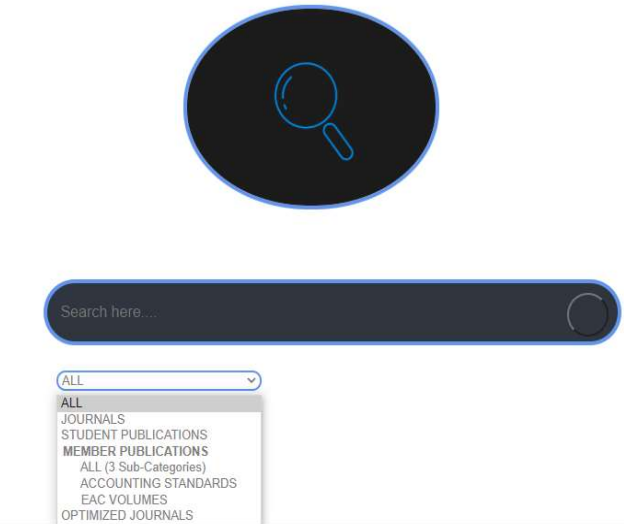


Fig 5. Dropdown with category selection

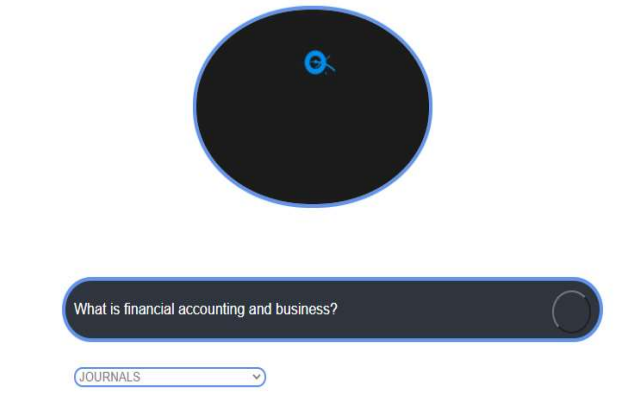


Fig 6. Journals query

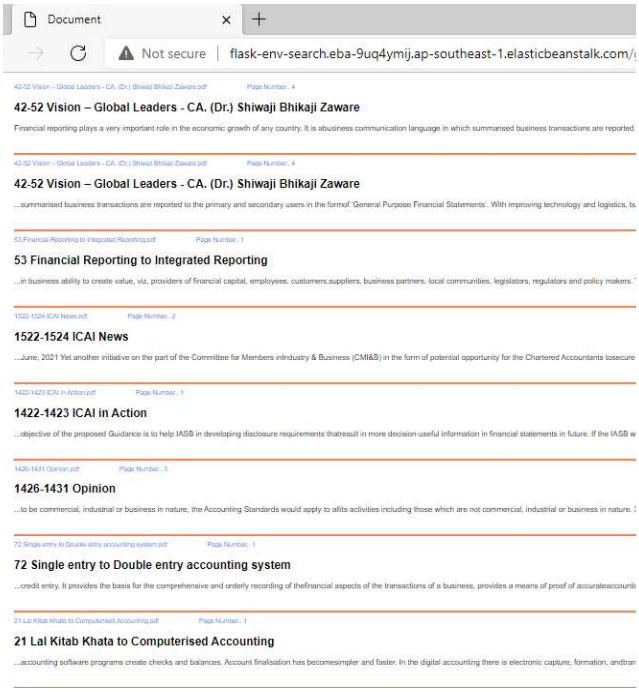


Fig 7. Journals query response- Lal Kitab last result

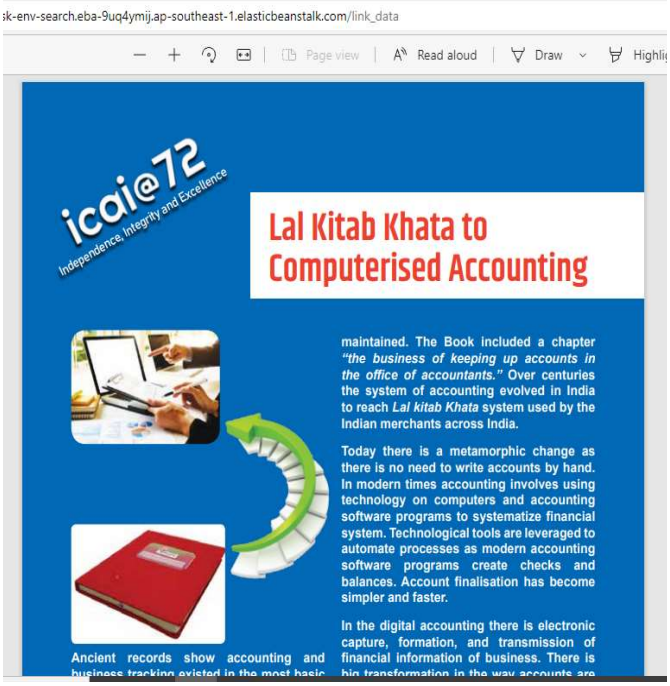


Fig 8. PDF viewer

❖ Optimization of Solution

By default, query responses are sorted by the relevance score that Amazon Kendra determines for each result in the response. We can modify the effect of a field or attribute on the search relevance through *relevance tuning,* then we see the different search results that you get from different configurations. We have tuned results for built-in (_category) and custom attribute(_tags) to improve relevance of document ranking.

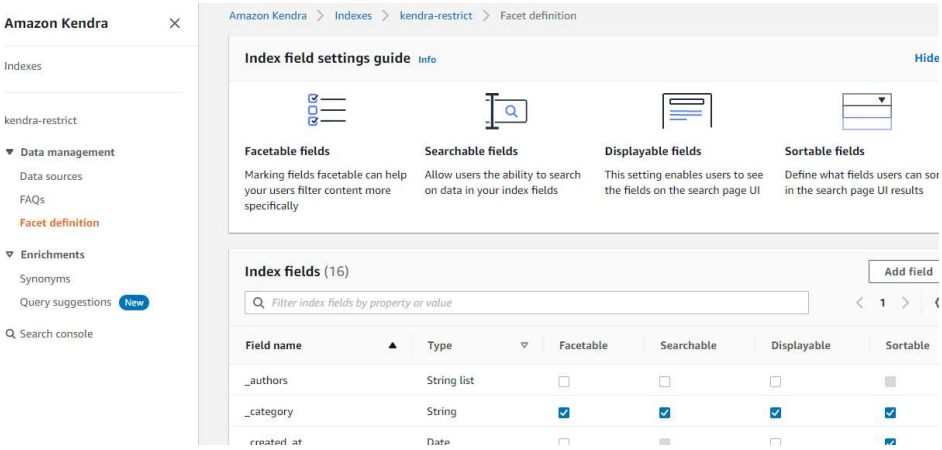


Fig 9. Category optimization

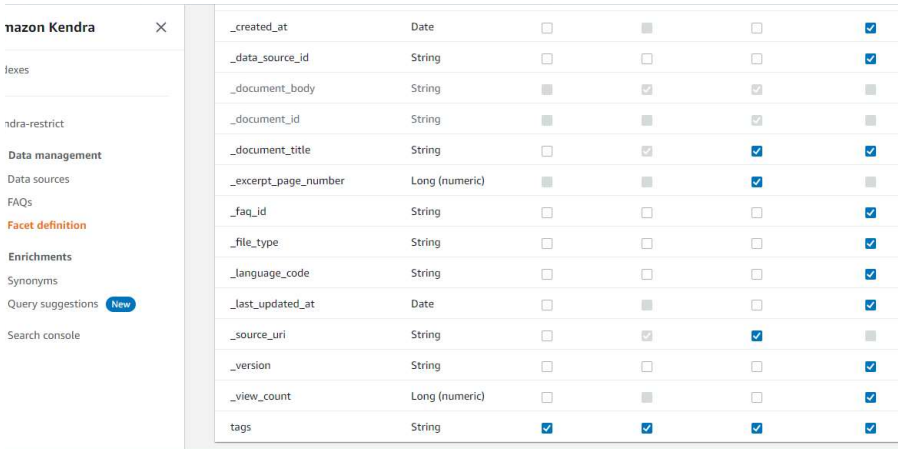


Fig 10. Tags and excerpt page number

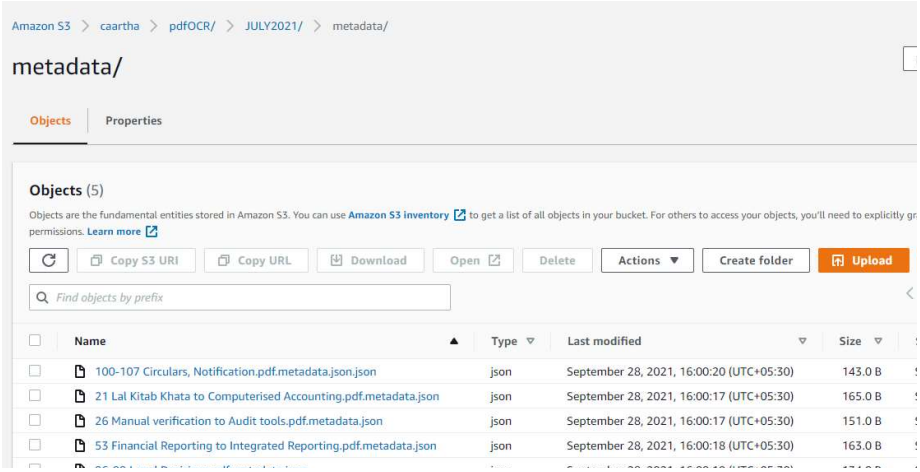


Fig 11. Metadata files in S3

To do this, we add metadata to documents and FAQs using the built-in attributes in Amazon Kendra, custom attributes, and user context filters. We filter the content using a combination of these. For this, below steps are there:

- Built-in attribute _category to represent the document category.
- User context filter attribute for the employee access level.
- Custom attribute _tags representing the document tagged.
- On the Amazon Kendra console, choose the index created in the previous step.
- Choose Facet definition.
- Choose the Add
- For Field name, enter _tags.
- For Data type, choose String.
- For Usage types, select Facetable, Searchable, Displayable, and Sortable.
- Choose Add.

```
{
    "Title":"21 Lal Kitab Khata to Computerised Accounting",
    "Attributes":{
        "_category": "ICAI",
        "tags": ["Accounting", "Financial", "Business"]
    }
}
```


Fig. Metadata file for category 1


```
{
    "Title":"100-107 Circulars, Notification",
    "Attributes":{
        "_category": "Legal",
        "tags": ["Investment", "Proceedings", "Tax"]
    }
}
```


Fig. Metadata file for category 2

Category 1- Legal
Tags for category 1- Investment, Proceedings, Tax
Document set- 96-99 Legal Decisions.pdf, 100-107 Circulars, Notifications.pdf

Category 2- ICAI
Tags for category 2- Accounting, Financial, Business
Document set- 26 Manual verification to Audit tools.pdf, 53 Financial Reporting to Integrated Reporting.pdf, 21 Lal Kitab Khata to Computerised Accounting.pdf

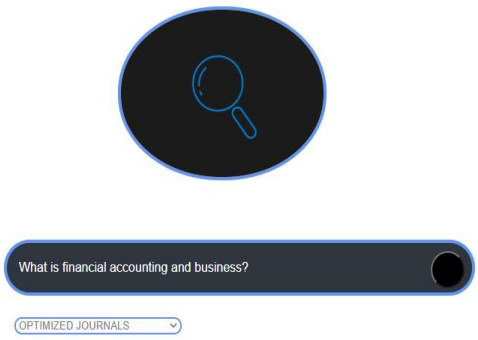


Fig. Journals Optimised query

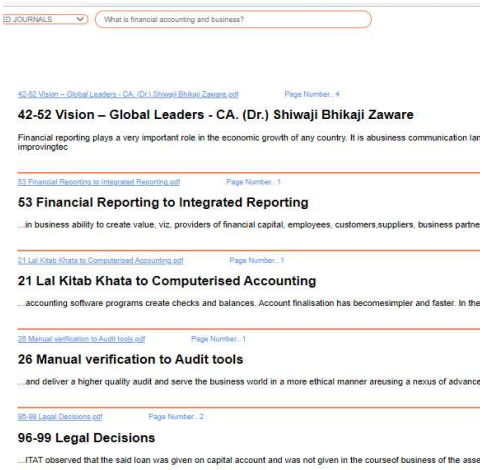


Fig. Optimised response- Lal Kitab third link now

Test cases

S. No.	Description	Test Data	Expected Result	Actual Result	Status (pass/fail)
#1	All category	What is ICAI?	Links from all the s3 buckets	Links from all the s3 buckets.	Pass
#2	Journals	What about financial accounting and business?	Links from journals subfolder.	Links from journals subfolder.	Pass
#3	Journals optimized	What about financial accounting and business?	Links from journals subfolder.	Links from journals subfolder.	Pass
#4	Members -EAC volumes	What are some diverse accounting policies and practices?	Links from EAC subfolder.	Links from EAC subfolder.	Pass
#5	Members -Accounting standards	What are your opinions on Indian accounting standards?	Links from accounting standards subfolder.	Links from accounting standards subfolder.	Pass

#6	Students	What is syllabus of ICAI?	Links from student subfolder.	Links from student subfolder.	Pass

Key Learnings

❖ **How to take name of link that user clicked from frontend?**

Solution was to create html form , so to capture, what link user clicked on, is to use html <input> tag, and first stores link value as <input> tag value and keep it hidden, then use same value in <input> submit value, so actual data will be send by first <input> tag which is kept hidden, and second <input> tag is used for only displaying link value, so value of the link will be sending using post request to flask url endpoint, and we can now further process the logic, once we get link value.

❖ **How to keep/maintain dropdown ordering when user choose from dropdown?**

First checked whether user clicked from member functions, if user select from member function then universal_list = [member_options+non_member_options]
So first removed all member options, then append selected option at start, then add non-member list then store member list in separate list, then send two list i.e. [selected_member_option+other_non_member_options],[remaining_options] so our drop down ordering will change, every time user selects any options.

❖ **How did you solve, particular link will be open at particular click?**

Used link masking solution {'link_name':'full_link_path} i.e. {'icai_doc_name':'s3://caartha/journal/icai_doc_name}, so after getting link name from frontend when user clicks on any link, it will fetch out its full path for the corresponding link name that contains only file name not full path, so to get full path we fetch from dictionary already created , now file can be downloaded from s3, as s3 needs full path to download file, not just document name. After downloading file, we will store in app/static folder as per flask standard, and return the downloaded file when user clicks on any link.

❖ **How does shuffling logic works?**

Shuffling logic is working only for multiple user selection, by taking factor in consideration, that for multiple user selection, we will get unequal length result, so if length of min

returned results is < 3, we will take that minimum number of results from every multiuser results, and if length of minimum query is > 3, will divide it, and take that number to fetch results from returned results and append remaining results, and finally rendering the results on page.

- **How to handle multiple user selection at backend level?**

Used check button, and use request.args.getlist function to grab the multiple user data as a list. The corresponding users are sent to Lambda via API gateway which then forwards it to Kendra index to fetch json responses. These responses are then shuffled in Lambda function and presented in frontend screen 2.

Solution Outcome

- ✓ The user is now able to perform NLP based Searching (Cognitive Searching) based on the contents of Publications, Journals, Students FAQs, Members FAQs stored in Amazon S3.
- ✓ The search display pdf page number along with pdf link allowing the user to perform Restrictive Search within a selected topic / sub-topic.
- ✓ The optimization technique of tagging resulted in 66% improved results.
- ✓ Reduction in search time due to relevance tuning in Indexes.
- ✓ Ability to query multiple categories at a time.