Documentation

(a) General Overview

We integrated MongoDB inside Python to complete our project. The goal of this software is to build a document store. The user can search for articles in the software and the user can select the article to see the fields of the article. The user can also search for authors by providing keywords and the software returns the authors with matching keywords in it. The user can then select the author to see more information regarding them. The user can also enter a number n and see a listing of top n venues and for each venue the venue, the number of articles in that venue, and the number of articles that reference a paper in that venue are displayed with top most cited venues shown first. Lastly, the user can add an article to the document store. We have a load-json file also which converts the json file into a indexed database to use by the document store

User Guide

When the program just runs , It asks the user to first enter a port to connect to. After the user enters a port to connect to , they are prompted with an option to
Press 1 to search for Articles
    Press 2 to Search for Authors
    Press 3 to List the Venues
    Press 4 to Add an Article
    Press 5 to exit the Program

When the user enters (1), They are prompted to enter a keyword to search for articles. After then, they can select an article to get more information about it
If the user enters (2), they are prompted to enter a keyword to search for Authors. After the result is displayed, they can select an Author to get more information about them.
If the user enters (3), They are prompted to enter a number (Which is the number of top venues they want to see), and then the top n venues are shown.
If the user enters (4), They are prompted with details to input so that they can add a new Article
And If they Press (5), It exits the program.

(b) Detailed Design

We broke down our code into multiple user-defined functions, each having its own individual function.
load-json.py

openJsonFile -> It reads the Json file and converts it into a mongodb database which is indexed.

script.py

startProgram -> Asks the user to input a port number and connect to the mongodb server using that port. It also makes the database with the data in it displayArticleSearchResult -> This function is used in searchArticle to make sure the displayed search results are formatted in a nice and readable manner.

searchArticles -> This function prompts the user to input keywords. Using those keywords, it displays the articles with the keywords in them. It doesn't only check the articles but it checks the title, authors, abstract, venue, and year fields to check if the keyword exists in any of them. After the search result is displayed, the user is prompted with the option to select an article from the given search results, and when the user selects the article, the user can see all fields including the abstract and the authors in addition to the fields shown before. If the article is referenced by other articles, the id, the title, and the year of those references will also be listed

displayAuthorSearchResult -> This function is used in searchAuthor to make sure the displayed search results are formatted in a nice and readable manner.

searchAuthors -> This function prompts the user to input keywords. Using those keywords, it displays the authors with the keywords in them. For each author, the author's name and the number of publications they have are also listed. After the search results are published, The user can select an author and see the title, year, and venue of all articles by that author.

addArticle -> This prompts the user to input an id, title, authors, and year. Using that information, it inputs the imputed value along with The fields abstract and venue be set to null, references will be set to an empty array and n_citations will be set to zero. If the user inputs a non unique id , an error message will show as it's not unique, and the new article will not be uploaded.

listVenues-> Prompts user to enter a number n which will be the number of venues that will be displayed on screen. Sorts the data by descending order of references to the venue . The query uses $lookup to get an array of all articles referencing an article (called joinedV). The following results are then used to group based on the venues and the corresponding references were counted by summing the size of joinedV for each article in the venue.

Main -> The main function has the call to all other functionality. First, it starts the program using the startProgram. Then it asks the user to input a number between 1 and 5 where each number has a specific functionality which it does. 1 to search for articles, 2 to Search for Authors, 3 to List the Venues, 4 to Add an Article, and 5 to exit the Program. When the user enters 1, it calls the searchArticles , when the user enters 2, it calls searchAuthors , when the user enters 3, it calls listVenues , when the user enters 4 it calls addArticle and when the user enters 5, it exits the program. There is also a check which checks if the user has a valid input or not

(c)Testing Strategy:

We used the Json test data files to check our functionality. We downloaded the files and converted those files into the database to run our MongoDB commands on them to see if our code worked or not. Furthermore, we used this procedure to get a base understanding if our code works or not. We did this for all the codes to get the base functionality working.

We later brainstorm any edge cases possible and try to test those out. For example, for adding an article we tested that function by first providing a normal unique id and checking if it was inserted properly. Then we checked by adding a non-unique ID to check if the code gives us an error as the software should not allow having a non-unique ID. So that is how we checked the adding article function.

For the search article, We first wrote the code that we thought worked and made sure the general gist of the code worked. Then we provided a different variety of keywords to check if it worked. We provided multiple different keywords to see if it worked. Then we checked by providing a random keyword to check if it gives no output as no matches should be there. Then we checked by not giving any keywords as it should give all the results. The last test we did was to use the add article to add a new article and we then searched for the newly added article. Then we added one more article with similar data to see if the references increased by 1. This was used to test both search and adding articles. We made sure we could select the article to show the information about the article.

For the search author, it was very similar to the search article as we made sure the general gist of the code worked. Then we provided a different variety of keywords to check if it worked. We provided multiple different keywords to see if it worked. Then we checked by providing a random keyword to check if it gives no output as no matches should be there. Then we checked by not giving any keywords (empty input), as it should give all the results.

(d)Group Work: Distribution Strategy

We used to schedule a meeting time and spend 2-3 hours per meeting working on the project. We used to call over discord and work on the project via VScode live share, allowing us to work on the code together. If one used to get stuck on a problem, we all brainstormed ideas to help overcome that challenge. We made sure we completed our tasks which were assigned before our next meeting. We also had a discord group where we kept in touch every day stating 1) What work we have done, 2) what work we have pending and 3) What work we are stuck at if any. And if any member was available, we used to work together to solve the problems. We also had separate files in which we got the basic work done and once it worked we transferred the code to the main file. When we used to call and work together, we used to work on the main file as everyone could see the code but when we worked individually on our own , we worked on our separate files as to avoid breaking the main working file. We used to do the testing and functionality building of each function in our own individual files. A breakdown of work followed

Dave - Worked on the search Article, Selecting Articles, list the venues, Searching Authors
Hours Worked Dave - 15 Hours

Pratham - Worked on Adding Articles, Searching Articles, Documentation, main
Hours Worked Pratham - 15 Hours

Vedd - Worked on List the venue, Search Authors,
Hours Worked Vedd - 15 Hours

We also debugged each other's code in functionality so that we can work faster and be more efficient