General Overview
First, our program loads the database for the program with the execution of load_json.py. In this phase, the user is prompted to enter a json file to load into the database and a port number to connect to the MongoDB server. Then the program tries to connect to the server and open the json file. If successful, the program then loads the necessary data into the database and does some precomputation for Phase 2 of the assignment, updating the user on its progress and the time taken.
After loading the database, the user can then run the document store application. The user will first be directed to a main menu where the user can select an option to perform. There are 5 options: they can search for articles, search for authors, see the top venues, add an article to the store, or quit.
If the user chooses to search for articles, they will be prompted to enter keywords, which will be used to search for articles. The program will search for articles that contain all those keywords in any of the article's title, authors, abstract, venue, or year information. Then it will display the id, title, year, and venue information of all the articles it has found. The user can then select one of the articles displayed to see more information and also to see the information of articles that reference the selected article.
If the user chooses to search for authors, they will be prompted to enter a keyword, which will be used to search for authors. The program will then search for authors whose names contain the keyword and display the name and number of publications of authors found. Only 10 authors will be displayed on each page, and the user can navigate through the results by entering 'next' to go to the next page or 'prev' to go to the previous page. The user can then select an author to see more information on the articles by that author.
If the user chooses to list a venue, then the user will be prompted to enter a number *n*. Then the program will display the top *n* venues according to the number of articles that reference the venue. For each venue, the number of articles in that venue and the number of articles that reference the venue.
If the user chooses to add an article to the store, the user is prompted for a unique id to use when adding the article, the article' title, a list of its authors, and its year of publication. Using this information, the article will be added to the collection.
Finally, if the user chooses to quit, the program will exit.
After each action, the user will automatically be redirected to the main menu for further actions.

User guide:
First, the database has to be loaded before running the program. The program will print instructions to the terminal and prompt users for input whenever users can select an option or enter input. To exit the program, the user can select the option to quit by following the instructions printed to the terminal. To return to the previous screen or stop the current action, the user should enter a blank line.

Terminal command to load the database: **python3 load_json.py**
Terminal command to run the document store program: **python3 main.py**

Detailed Design
Our program implements 3 main classes: Store, ArticleSearchResults, AuthorSearchResults.
Store is an object representing the document store and implements the operation of the document store as per the specification.
ArticleSearchResults represents the results of an article search for a given set of keywords and implements searching for articles that match the given keywords as per the specification.
AuthorSearchResults represents the results of an author search for a given keyword and implements searching for authors that match the given keyword as per the specification.

Responsibility and interface of each function/class:

load_json.py

Functions	**Purpose**
load_json	Implements Phase 1 of the assignment. Includes creating the collection as per the specification, creating indexes, and precomputation steps

Store(class) in store.py

Class Functions	**Purpose**
__init__	Takes a port number and connects the Store to the database and view in MongoDB that was created in Phase 1
show_main_menu()	Shows the main menu of the Store, from which the user can select an option to perform
show_article_search()	Implements the search for articles as per the assignment specification using the class ArticleSearchResults
show_author_search()	Implements the search for authors as per the assignment specification using AuthorSearchResults
show_list_venues()	Implements the listing of top venues as per the assignment specification
show_add_article()	Implements adding an article as per the specification

ArticleSearchResult(class) in article_searcher.py

Function/Class	**Purpose**
__init__	Initializes a MongoDB collection object and a list given keywords as class attributes and executes a search
retrieve	Retrieves all articles that match the given keywords
display_articles	Displays articles that search has found
select_article	Selects an article and displays its referencing articles

AuthorSearchResult(class) in author_searcher.py

Function/Class	**Purpose**
__init__	Initializes a MongoDB collection object and a keyword as class attributes and executes a search
display_results	Displays the authors that the search has found

<u>Testing Strategy</u>

We tested our program using the sample files given as test cases. In addition, we created our own sample datasets. We manually determined what data should be returned using smaller files so that we could verify our program’s correctness. Each feature was tested, with each individual testing their code

portion's functionality before testing out the combined program as a whole. To ensure program robustness and functionality, we tested our program with every input combination available.
Problems we encountered included syntax errors in queries and issues with optimization. Due to the time constraints, we had to rework our queries and program implementation. MongoDB syntax and having to work with unfamiliar operators made the project even more difficult. In this regard, reading documentation and working together with group members to figure out a solution was very important and beneficial.

Group Work Strategy

Breakdown of group work:

Phase 1:
load_json.py - npham
extra precomputation(for listing venues) - wqi3

Phase 2:
Main menu - wqi3, npham
Search for articles - isun
Search for authors - wqi3
Listing venues - wqi3, npham, isun
Add an article - npham

Testing code and creating this report - wqi3, npham, isun

Time spent
isun - 30hrs
npham - 30hrs
wqi3 - 30hrs

Method of coordination
We used discord to discuss and work on our project. We also met up to discuss and work on our code together, and to check in on each individual's progress. Our individual code components were placed inside a Github repository so that other group members could access them.