CMPUT 291– MINI PROJECT 2 REPORT

Kunaal Gupta, Aidan Gironella and Shubhkarman Jaura

1. *Overview of our system with a small user guide*

We have created a system that is able to interact with .json files and created a database that we can query and insert into using Python. This system allows users to build a full MongoDB database from a .json file, and then allows the user to perform various different functions on it, such as searching for articles, searching for authors, listing the venues, and adding an article to the database.

The first major file is our Phase 1 load-json.py file, which prompts the user for a full .json file name (including the extension) and a port in which there is an active MongoDB server running. If two valid values are entered here, the program will automatically read the entire .json file and create a database for the user. This step will also automatically set up indexes for fast searching later.

The second major file is the Phase2.py file, which combines all of our individual functions into one common file. When the user runs this program, they will be prompted for a port in which there is an active MongoDB server running. If the user provides a valid port, they will be taken to the main menu, where they can choose from 5 different options:

1. Search for articles
2. Search for authors
3. List the venus
4. Add an article
5. Terminate the Program

The details of these functions are provided below.

2. *A detailed design of our software with a focus on the components required to deliver the major functions of our application*

Primarily our application has three main components:

1. Phase 1
    a. Load Data

2. Phase 2
    a. Search for Articles
    b. Search for Authors
    c. List the Venues
    d. Add an Articles

__Phase 1__: For this part, we wrote a program, named *load-json* with a proper extension (e.g. *load-json.py* if using Python), which will take a json file in the current directory and construct a MongoDB collection. Our program will take as input a json file name and a port number under which the MongoDB server is running, will connect to the server and will create a database named 291db (if it does not exist). Ourprogram then will create a collection named *dblp*. If the collection exists, our program should drop it and create a new collection. Our program for this phase ends after building the collection.

__Phase 2__:

a. **Search for Articles**: This function prompts the user to enter as many keywords as they would like, separated by spaces, and then splits that string into individual keywords and performs a text search on the database. If there are no matches, the program will simply tell this to the user and return to the main menu. If there are matches, the program will print the ID, title, year and venue for each match in a list. From this list, the user can select an article to get more details. This includes the ID, title, year, venus, abstract and author(s), as well as a list of any other articles that reference the chosen article. For the articles that reference the chosen article, their ID, title and year will be displayed.

b. **Search for Authors**: The user is able to provide a keyword and see all authors whose names contain the keyword (the matches should be case-insensitive). For each author, we list the author name and the number of publications. The user is able to select an author and see the title, year and venue of all articles by that author. The results are sorted based on year with more recent articles shown first.

c. **List the Venues:** The user is able to enter a number *n* and see a listing of top *n* venues. For each venue, we list the venue, the number of articles in that venue, and the number of articles that reference a paper in that venue. We sort the result based on the number of papers that reference the venue with the top most cited venues shown first.

d. **Add an Articles**: The user is able to add an article to the collection by providing a unique id, a title, a list of authors, and a year. The fields abstract and venue are set to null, references are set to an empty array and n_citations are set to zero.

3. Testing Strategy

To test the ‘Search for articles’ functionality, we simply entered a keyword into our program and compared the results with the MongoDB Compass tool since it is very fast and highly accurate. This allowed us to check the correctness of the search function on its own. This function is also supposed to be instant, so we had to make sure that this was the case. In order to do this, we tested the program firstly without printing anything, since printing tens of thousands of lines can take time, no matter how fast the actual search is. Using the Python datetime package, we set a start time variable, ran the search, and then set an end time variable and printed end-start. This showed us exactly how many milliseconds the query took, and we made sure that this was instant before proceeding to printing the results.

For 'search authors', the text index was used to retrieve the author and regex was used to ensure that the specifications were met. I first tested my program on a small dataset and subsequently moved onto a large dataset while ensuring that my results are correct and double checking with the results on MongoCompass.

For Add an Article function, we tried to add a number of articles to check if there is any error while it's being added & whether a new article is successfully added in the database or not. For List the Venues function, we tested it by checking if the output of number of articles in that venue matches with what we see in MongoDB Compass. Also, we checked if there is any duplicate venue name in the output to see if our algorithm is working correctly or not.

4. ***Group Work Break-down Strategy***

We are three members in a group, therefore, we decided to break down the project into sub-parts. As mentioned above, Aidan was responsible for implementing the search for articles functionality. To keep the project on track, we actively communicated with each other on the WhatsApp group chat and ensured that expectations and responsibilities were set crystal clear. Moreover, as a team, we had group meetings where we clarified our doubts, sought help when required and offered a helping hand to other group members to ensure the successful completion of the project.

Shubhkarman Jaura: was primarily responsible for implementing the "search for authors" functionality. Apart from that he did the majority of the work. He created a python file for the phase 2 and also contributed immensely in building the phase 1 python file called load.py. He has also prepared the readme file and incharge of section 4 in this report. He has spent around 23 hours on this Mini Project 2.

Aidan Gironella: Tasked with completing the "Search for Artists" functionality. Set up the Git repo for the group to use, created the text indexes for searching. Spent around 10 hours on the project.

Kunaal Gupta: Implemented two functions, namely - Listing Venue & Adding a new article, in the project. Worked to improve the content in the Report MP2 file. Spent about 12-15 hours on the project