# Report

Design Document Report

By Ahmed Zihan Hossain, Alfred Lee, Connor Macleod

## User Guide

Phase 1:

The program load-json.py takes a json file in the current directory and constructs a MongoDB collection called dblp. The program takes as input a json file name and a port number under which the MongoDB server is running, connects to the server and creates a database named 291db (if it does not exist). The program then creates the collection named *dblp*.

Phase 2:

1.) User enters port number
2.) System Functionalities For User
    a.) Search for Articles
        i.) Provide 1 or more keywords to search with
            (1) User selects an article (by number) matching the keywords to view more info
    b.) Search for Authors
        i.) Provide 1 keyword to search with
            (1) User selects an author (by full name, case sensitive) to view for info about articles they've written
    c.) List the venues
        i.) Provide a number (n) to view top n venues by articles referencing other articles by the venue
    d.) Add an Article
        i.) User inputs a new, unique ID
        ii.) User inputs a title
        iii.) The user adds authors until they enter "done"
        iv.) User inputs a year to the article
    e.) Exit the program

## Detailed Software Design Components

**Phase1 main()**

- Connect to the database, import json file, create indexes

**SearchArticles(collection):**

- Find articles when given a keyword in our database. Then print the articles that match while allowing to see further detail of one article and what it is referenced by.

**searchauthors()**
- Find all authors and how many articles they've published, and print them for the user. When user has input the author, find all articles that the author is credited with and print them for the user.

**listVenues()**
- List the Top N Venues, N is selected from the user. It sorts if based on number of references for that venue, prints the number of articles in that venue, and number of references for that venue

**addArticle()**
- loops until the user has selected a unique ID, lets them enter a title, lets user keep adding authors until they want to stop, lets users enter a year. That info is entered into the database.

**seloptions()**
- Main loop that allows the user to select 5 options on a loop.

## Testing Strategy

We will be testing on the 10 json file and adding fields to see if the data changes as expected

| TestID | Description | Expected Result | Actual Result |
|---|---|---|---|
| 1 | Search for article with known abstract | Only that article found | Only that article found |
| 2 | Search same as TesdID 1 but add a crazy value | No article found | No Article found |
| 3 | Test title, authors, abstract, venue, and year for values only in that field | All found | All found |
| 4 | All 5 fields at once | Article Found | Article Found |
| 5 | Add reference and see if it shows up | Reference shows up | Reference shows up |
| 6 | Known Author | Author Returned | Author Returned |
| 7 | Unknown Author | Not Returned | Not returned |
| 8 | Partial Author | Not returned | Not Returned |
| 9 | New collection, no references | No references | No references |
| 10 | Add a reference | Reference value increased | Reference value increased |
| 11 | Add Article | Article Added successfully | Article Added successfully |
| 12 | Exit works | Success | Success |
| 13 | Load timing | <5 min | 4min 13 secs |

| | | | |
|---|---|---|---|
| 14 | SearchArticle timing | <1s | ~0.5s |
| 15 | SearchAuthors timing | <4s | ~0.5s |
| 16 | Top Venue | <30s | ~10s |
| 17 | Add Article | <1 | <1 |
| 17 | Check double counting | Does not double count | Does not double count |

## Group Work Break-Down Strategy

We broke down the project by the tasks to be done. Each task was roughly weighted by everyone according to difficulty and each person chose 3 points worth of work. Care was taken during breakdown so that everyone got equal amounts of work. The breakdown is given below:

Ahmed Zihan Hossain
(Time spent ~ 15 hours)
- load-json.py (3)

Alfred's Functions:
(Time spent ~20)
- Search for articles (3)

Connor Macleod
(Time spent ~20)
- Search for authors (1.5)
- List the venues (1)
- Add an article (0.5)
- End the program (0.1)

*Numbers beside tasks represent our original weight given to the function as an expectation of difficulty.

**For Search Venue we worked together as a team to solve the problem since it was proving to be a lot more difficult than expected

After our individual tasks were done we grouped everything together and tested interactions between each other's functions.

For communication, we scheduled meetings and shared code through Discord, and planned on Google Docs.