# 291 Mini Project 2

jankee
laidlaw
raphaelm

## <u>Phase I</u>

(Khevish's work)

- Connect to server via validated port number from user input
- Connect/create 291db and drop collections if any
- Fetch and validate json file from user input
- Use multithreading to load data into the database (fine-tuning by James)
- Convert year to string in on loading  (James's work)
- Pre-computation for query use (James's work)
- Create index for later query use

## <u>Phase II</u>

**Components:**

**JamesFunctions (James's work)**

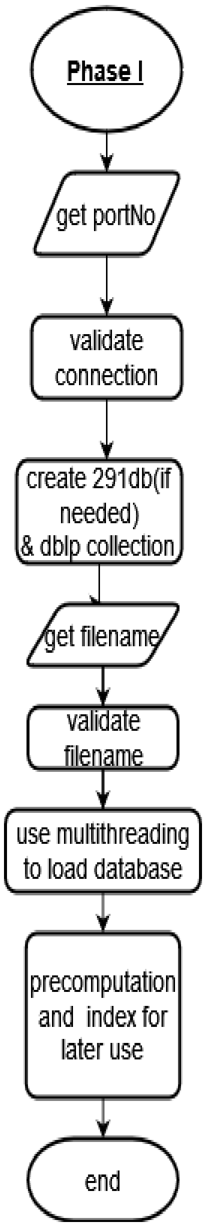
- **searchArticles**: takes a list of strings, returns list of articles containing those strings
- **getRefrencingArticles:** takes an article ID, returns a list of all articles referencing that id
- **articleSearchUI:** a UI wrapper for searchArticle and getRefrencingArticles. Covers all components of portion 1 of section 2 when provided a database collection as an argument
- **authorSearch:** takes a keyword as a string, returns a list of all authors whose name contains that keyword, along with a count of their publications
- **getAuthorPublications:** takes an author's whole name, returns a list of all publications they have written.
- **authorSearchUI:** a UI wrapper for authorSearch and getAuthorPublications. Covers all components of portion 2 of section 2 when provided a database collection as an argument

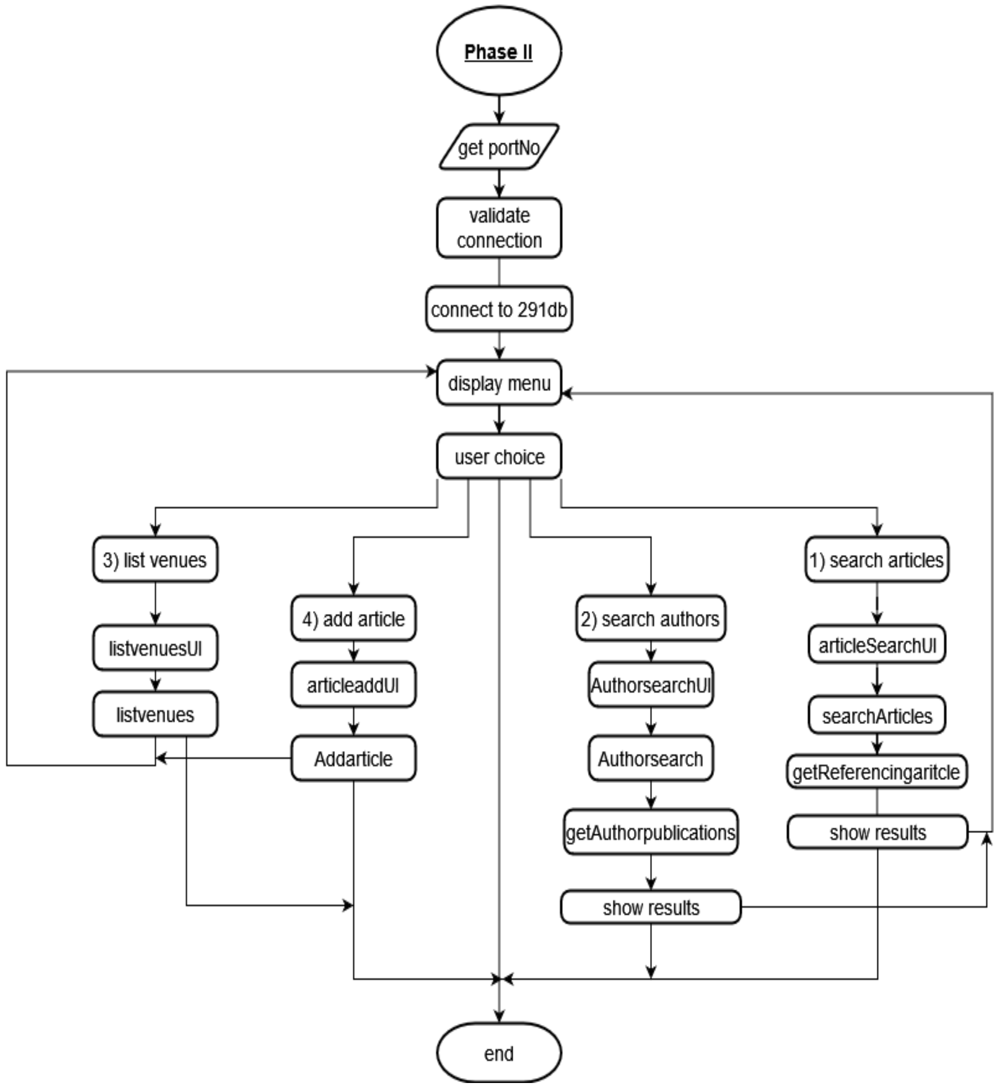
**UI module (Khevish's work)**

- connect to database with validated port number given by user
- navigation menu between i) search articles, ii) search authors, iii) list venues, iv) add an article
- Program exit/end

**raphaelFunctions (Raphael's work)**

- **listVenues**: Takes an integer and returns cursor that contains the top n venues
- **listVenuesUI**: UI wrapper for listVenues, handles input and prints the output from listVenue
- **addArticle**: Takes a string(id, title), list(authors), and integer(year) and adds an article to the collection with the given arguments
- **addArticleUI**: UI wrapper for addArticle, handles all inputs and type matching for addArticle, prints the added article at the end


Phase I
get portNo
validate connection
create 291db(if needed) & dblp collection
get filename
validate filename
use multithreading to load database
precomputation and index for later use
end



## Work Breakdown

Actual:

- James: 19.5 hours
    - Parts 1 & 2 of phase 2 (with UI): ~9 hours
    - ~⅓ of report: ~1 hour
    - Debugging, testing & optimizing phase 1: ~4 hours
    - Helping with part 3: ~1 hour
    - Making phase 1 convert year to string: ~1 hour
    - Making phase 1 Precompute ref count for part 3: 2.5 hours
    - Testing & debugging: ~1 hour

- Khevish: 19 hours
    - Phase 1 : ~9 hours
    - Debugging, testing & fine-tuning phase1 : ~5 hours
    - ~⅓ of report: ~1 hours
    - User interface (UI.py): ~2 hours
    - Testing and debugging assembled code: ~2 hours

- Raphael: 14 hours
    - Parts 3 & 4 of phase 2: ~8 hours
    - User interface for parts 3 & 4: ~3 hours
    - ~⅓ of report: ~1 hour
    - Testing: ~2 hours

## Testing Strategy

### Phase I

Each group member ran the code on their local machine and lab machine in order to measure the different variance in time the loading process of the database would take in order to respect the time limit imposed. Testing was made by using the 3 sample files provided specifically the 1 million entries one in order to optimize code and reduce running time.

### Phase II

Each member of the group tested the function they wrote individually (unit testing).
On merging the code all together, all group members tested the whole code by checking regular cases, edge cases and abnormal cases.

### Implementation strategy:

Break down the assignment into smaller manageable modules
Fixing deadlines to complete specific modules
Maintaining a code style which is easier to integrate with other modules.

Main mode of communication used by the group is Discord.
File sharing was done through Github and Google Docs