

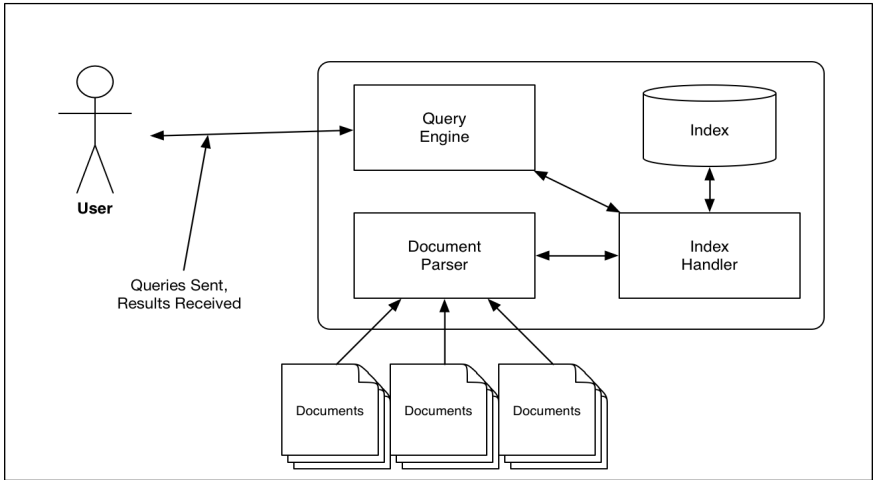
# Final Design for Search Engine

## Architecture

The four major components' of a typical search engine are the :

- 1. Document parser/processor,
- 2. Query processor,
- 3. Search processor, and
- 4. Ranking processor.

The image below provides a general overview of the search engine's architecture.



## Core Data Structure

### DSAvlTree for keyword index

DSAvlTree<T>
- root: DSAvlNode<T>*
+ insert(const T &x): void
+ search(T data): DSAvlNode<T>*
+ count(): int
+ iterator(): Iterator
+ serialize(ostream &os): void
+ deserialize(istream &is): void
- rotateWithLeftChild(DSAvlNode<T>* &k2) void
- rotateWithRightChild(DSAvlNode<T>* &k1) void
- doubleWithRightChild(DSAvlNode<T>* &k3) void
- doubleWithLeftChild(DSAvlNode<T>* &k3) void

DSAvlNode<T>
+ left: DSAvlNode<T>*
+ right: DSAvlNode<T>*
+ height: int
+ element: T

DSAvlTree<T>::Iterator
- tree: DSAvlTree<T>*
- stack: stack<DSAvlNode<T>*>
+ hasNext(): bool
+ next(): DSAvlNode<T>*

---

## DSHashTable for author index

---

DSHashTable<K, V>	DSHashTable<K, V>::iterator
- table: vector<list<pair<K, V>>>	- hashTable: DSHashTable<K, V>*
- tableSize: unsigned int	- currentIndex: int
- resize_table(uint32_t newSize): void	- currentBucketIter: list<pair<K, V>>::iterator
- hash(const K &s): unsigned int	+ &operator*(): pair<K, V>
+ insert(const pair<K, V> &data): pair<iterator, bool>	+ &operator++(): iterator
+ find(const K &key): iterator	+ &operator--(): iterator
+ erase(const K &key): bool	
+ begin(): iterator	
+ end(): iterator	
+ count(): int	

---

## ArticleData to store the parsed article data

ArticleData	ArticleMetaData
+ documentID: string	+ title: string
+ authorLastNames: unordered_set<string>	+ author: string
+ keyWordList: unordered_map<string, unsigned int>	+ abstract: string
	+ datePublished: string
	+ publication: string

---

## IndexNodeData to store the data in a node in the AVL tree

IndexNodeData
+ keyWord: string
+ invertedWordFreq: unordered_map<string, unsigned int>
+ idf: double
+ operator==(IndexNodeData& rhs): bool
+ operator<(IndexNodeData& rhs): bool
+ operator>(IndexNodeData& rhs): bool
+ calculateIdf(unsigned int totalArticlesIndexed): void
+ toJsonString(): string
+ operator<<(ostream& os, const IndexNodeData& nodeData): ostream&
+ operator<<(istream& is, IndexNodeData& data): istream&

---

**QueryResultData** to store the search result from the keyword index, to implement the ranking algorithm, and store data points to display on the UI

QueryResultData
+ keyword: string
+ documentID: string
+ wc: unsigned int
+ idf: double
+ weight: double
+ title: string
+ publication: string
+ datePublished: string
+ authorString: string
+ abstract: string
+ operator== (QueryResultData& rhs): bool
+ operator> (QueryResultData& rhs): bool
+ operator< (QueryResultData& rhs): bool

---

**DocumentParser** to parse the articles in the corpus and load the inverted file index to the AVL tree and author to the HashTable

DocumentParser
+ DocumentParser(string &corpusPath, string &stopwordPath, unordered_map<string, ArticleMetaData> &metaDataMap)
+ corpusPath: string
+ stopwordsPath: string
+ metaDataMap: unordered_map<string, ArticleMetaData>
+ parse(DSAviTree<IndexNodeData>: *keywordIndex, DSHashTable<string, unordered_set<string>> *authorIndex): void
+ loadStopWords(string &filePath): unordered_set<string>
+ addArticleToKeywordIndex(DSAviTree<IndexNodeData> *aviTree, ArticleData &articleData): void
+ addAuthorsToHashTable(DSHashTable<string, unordered_set<string>> *authorIndex, ArticleData &input): void

**IndexHandler** to store the keyword index, author index, and the metadata and handles all the index-related operations

IndexHandler
<div>+ IndexHandler (const string &amp;corpusPath)</div> <div>- authorIndex: DSHashTable&lt;string, unordered_set&lt;string&gt;&gt;*</div> <div>- keyWordIndex: DSAvITree&lt;IndexNodeData&gt;*</div> <div>+ metaDataMap: unordered_map&lt;string, ArticleMetaData&gt;</div>
<div>+ loadMetaData(string &amp;corpusPath): unordered_map&lt;string, ArticleMetaData&gt;</div> <div>+ createIndex(): bool</div> <div>+ persistIndices(): bool</div> <div>+ restoreIndices(string &amp;dir = ""): bool</div> <div>+ searchByKeyword(string &amp;keyWord): IndexNodeData*</div> <div>+ searchByAuthor(string &amp;author): unordered_set&lt;string&gt;</div> <div>+ clearIndex(): void</div> <div>+ isEmpty(): bool</div> <div>+ persistKeywordIndex(): bool</div> <div>+ restoreKeywordIndex(): bool</div> <div>+ persistAuthorIndex(): bool</div> <div>+ restoreAuthorIndex(): bool</div> <div>+ persistStats(): bool</div> <div>+ restoreStats(): bool</div> <div>+ totalArticlesIndexed: unsigned int</div> <div>+ avgWordsIndexedPerArticle: unsigned int</div> <div>+ totalWordsIndexed: unsigned int</div> <div>+ totalUniqueAuthors: int</div> <div>- maxFilesLoaded: int</div> <div>- persistentDir: string</div> <div>- stopWordFile: string</div> <div>- keyWordIndexFile: string</div> <div>- authorIndexFile: string</div> <div>- statsFile: string</div>

## QueryProcessor to parse the user input and invoke IndexHandler

QueryProcessor
+ QueryProessor(IndexHandler *indexHandler) - indexHandler: IndexHandler*
+ clearIndex(): void + isIndexEmpty(): bool + createIndex(): bool + loadIndices(string &dir): bool + parseQueryString(string &queryString): vector<string>* + preprocess(string &input, bool shouldStem): void + search(string logicOp, vector<string> searchWords, vector<string> excludedWords, vector<string> authors): set<QueryResultData> + searchKeywordindex(vector<string> &searchWords, vector<IndexNodeData*> &searchResults, vector<set<string>> &documentIDSets): void + searchAuthorindex(vector<string> &authors, set<string> &documentIDSet): void + getTop50OriginalWords(): vector<string> + getTotalUniqueAuthors(): int + getTotalWordsIndexed(): int + getAvgWordsIndexedPerArticle(): int + getTotalArticlesIndexed(): int

---

**SearchEngineUI** to display user interface and handle the user interactions

<b>SearchEngineUI</b>
+ SearchEngineUI(QueryProcessor *input) - queryProcessor: QueryProcessor*
+ run(): void + displayMainMenu(int &mainMenuChoice): void + clearScreen(): void + clearIndex(): void + parseCorpus(): void + openPersistenceFile(): void + search(): void + displayAbstract(QueryResultData &resultData): void + printStatistics(): void