## Methodology

The way the database will first be created is to scrape the indeed website. This happens in two phases.

The information shall be stored on a back end cloud database. At this time it’s AWS RDS database.

# Searching Indeed

## Generating a list of results

Is done by the function indeed\_search\_function.py. It takes in two parameters (keywords, zip\_codes) and returns a DataFrame with the following

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Comments** |
| guid | Object | The guid is a unique id assigned to a job by indeed |
| job\_title\_row | Object | This group of data resides in one tag under the title header. Consists usually of Job Title – Company – Location. |
| real\_job\_title | Object | This is the result of splitting before the first hyphen. |
| company | Object | The second split on the hyphen |
| in\_location\_zip | Object | The third split on the hyphen. |
| Listing\_source | Object | Not captured at this time |
| publish\_date | Object | An unparsed date time string (Tue, 06 Aug 2019 04:21:46 GMT) |
| short\_description | Object | The one line description from the search results. |
| lat | Float64 | Latitude from the listing |
| L ongitude | Float64 | Longitude from the listing |
| extracted\_url | Object | The URL with only the essential information to retrieve the job found. No external tracking |
| scraped | Bool | A flag created as false and to be used in the future to be see to positive when the job has been scraped. |

## Logging in a user

The module aws\_login\_credentials has a function awlc which works to maintain and read a .json file in the root directory continaing the user, password, and target DB information.

* Looks for a password json in the root file
* If it exists, check that the user wants to proceed with the information
* If not, then get new user login info.

For now, the user information is stored in a json in the root director. Future improvements should get the password using getpass encrypt the json using objcrypt. See appendix A.

## Maintaining a list of searches

The list of searches is stored on the database under the table indeed\_search\_set. (Future module should create this search set from the user. For now, we’ll create a CSV by hand )

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Comments** |
| Iss\_pk | Serial | Unique key for the table |
| search\_keyword\_list | Varchar(63) | List of keywords to search. These will be concatenated. |
| search\_zip\_code | Varchar(31) | At now, this will only support a single zip\_code. Future features should be able to take a list of zip codes or geographic location. The only problem with geographic location is the comma in city state which must be converted to a %2C character for inclusion in to the python string. |
| creation\_date | date | The second split on the hyphen |
| search\_completed | Boolean | A Boolean flag flipped to true when a search is completed |
| Search\_run\_date | Date | The acutal date the search is run. |

# Appendix A

Required modules in Alphabetical order

|  |  |  |
| --- | --- | --- |
| Module | Location | Version used |
| Pandas | <https://pandas.pydata.org/>,<https://pypi.org/project/pandas/> | 0.25.1 |
| pathlib | Standard Module |  |
| getpass | Standard Module |  |
| objcrypt | <https://pypi.org/project/objcrypt/> | 0.6 |
| SQLAlchemy | <https://pypi.org/project/SQLAlchemy/> | 1.3.8 |