# 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

## Overall Process

In the overall process the flow is as follows

1. The first step is to check the database to see if there are any searches pending.
2. The *indeed\_search\_executer.py* starts.
   1. *indeed\_search\_executer.py*  calls *aws\_login\_credentials.awlc* to log in a user.
   2. *indeed\_search\_executer.py*  creates a search query for any rows in **indeed\_search\_set** where the field *search\_completed* is False
   3. for each record returned, it calls the *indeed\_search\_function.py* on that element.
   4. *indeed\_search\_function.py*  returns the row information modified to show if the search was successfully completed or not.
   5. The data is committed and closed.

## Logging in a user

The module *aws\_login\_credentials* has a function *awlc* which read a .jsons 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.

## Traversing and executing a list of searches to be done.

The list of searches is stored on the database under the table **indeed\_search\_set**

This leverages SQLAlchemy’s ability to mirror table contents as an object.

The table **indeed\_search\_set** is defined in the remulak database as

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Comments** |
| creation\_date | date | The second split on the hyphen |
| Iss\_pk | Serial | Primary Key for the table |
| search\_completed | Boolean | A Boolean flag flipped to true when a search is completed |
| search\_keyword\_list | Varchar(63) | List of keywords to search. These will be concatenated. |
| Search\_run\_date | Date | The acutal date the search is run. |
| 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. |

Likewise, the Search Class is defined in SQLAlchemy terms as

class Serch(Base):  
 \_\_tablename\_\_ = 'indeed\_search\_set'  
 creation\_date = Column(Date)  
 iss\_pk = Column(Integer, primary\_key=True)  
 search\_completed = Column(Boolean)  
 search\_keyword\_list = Column(String)  
 search\_run\_date = Column(DateTime)  
 search\_zip\_code = Column(String)

I run a query to find everything where search completed = False, and iterate that through each result and call the search function.

At the end I commit and close the session. Because it’s a horrible thing to leave a session open.

## Building a table of search results.

Building a table of search results done by the function indeed\_search\_function.py.

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Data Type** | **Comments** |
| company | VARCHAR(64) | The extracted company. |
| extracted\_url | Object | The URL with only the essential information to retrieve the job found. No external tracking |
| guid | VARCHAR(32) | The guid is a unique id assigned to a job by indeed |
| Irs\_pk | Serial(int) | The Primary Key for the results |
| Iss\_pk | Integer | This is the foreign key to table indeed search set so we know which search created these results. |
| Job\_title\_row | Text | All of the information in the title line of indeed |
| lat | Float64 | Latitude from the listing |
| longitude | Float64 | Longitude from the listing |
| publish\_date | Timestamp | The derived timestamp from which it was published. |
| scraped | Bool | The flag which is flipped to True when a job is scraped |

The class Rezult creates a sql\_Alchemy structure Mirroring this table

class Rezult(Base):  
 \_\_tablename\_\_ = 'indeed\_search\_results'  
 company = Column(String)  
 extracted\_url = Column(String)  
 guid = Column(String)  
 isr\_pk = Column(Integer, primary\_key=True)  
 iss\_pk = Column(Integer)  
 job\_title\_row = Column(String)  
 latitude = Column(Float)  
 longitude = Column(Float)  
 publish\_date = Column(DateTime)  
 scraped = Column(Boolean)

The search for keywords is executed in a loop until the number of results in a block is less than 20, which means indeed has reached its end.

# 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 |