Autocomplete

This assignment is heavily inspired by this [one](https://www.cs.princeton.edu/courses/archive/fall16/cos226/assignments/autocomplete.html) (yes, I think you can handle a Princeton assignment!). From the original source:

Autocomplete is pervasive in modern applications. As the user types, the program predicts the complete query (typically a word or phrase) that the user intends to type. Autocomplete is most effective when there are a limited number of likely queries. For example, the [Internet Movie Database](http://www.imdb.com) uses it to display the names of movies as the user types; search engines use it to display suggestions as the user enters web search queries; cell phones use it to speed up text input.

In these examples, the application predicts how likely it is that the user is typing each query and presents to the user a list of the top-matching queries, in descending order of weight. These weights are determined by historical data, such as box office revenue for movies, frequencies of search queries from other Google users, or the typing history of a cell phone user. For the purposes of this assignment, you will have access to a set of all possible queries and associated weights (and these queries and weights will not change).

In this assignment, you will implement autocomplete by sending an Ajax query with every keyup, querying a database for all strings that start with a given prefix, ranking and returning the matching terms by weight, and displaying the choices in an option list. Previous queries made by the user are always given higher priority, in reverse chronological order.

**Preparation**

“The file cities.txt contains over 90,000 cities, with weights equal to their populations.” (ref: [cos226](https://www.cs.princeton.edu/courses/archive/fall16/cos226/assignments/autocomplete.html)) The weights will be used to order the search terms, with the assumption that larger cities are searched for more often that smaller cities. The second part of each line is the city name, followed by the country. In a console application, read the file, break it up into required segments, and populate your database.

**Functionality**

When arriving at the index page, the user must be able register, login and logout using best practices.

Once authenticated, the user will start typing in the search textbox. In this assignment, when the user selects a term that was presented as a match, your application will add the term to the user’s search history. Likewise, if the user manually presses the submit button to submit their term, your application your application will add the term to the user’s search history. Our application is boring – we will then simply confirm the user’s submission and prepare for the next entry.

With every keyup, provide a selection list under with upto 5 top ranked matches.

Whenever the user selects/submits a term, the term is added to the user’s search history. A maximum of 5 search terms are saved in each user’s history; the oldest terms are overwritten with new terms. If the term is already found in the user’s search history, the date associated with the existing entry is updated instead of creating/overwriting a second entry.

Search results are ranked as follows:

* Any query which matches the first letters in a search history term, the search history term has first ranking
  + In the case of multiple matching search history terms, order them by date
* If less than 5 matching terms were found in the search history, order the matching database query results according to their weights.

This project makes heavy use of JavaScript/JQuery and Ajax.

One or more PHP scripts will accept the Ajax queries as well as the submit requests on the server side and process them accordingly. You can decide on the query protocol (i.e., Ajax uses GET request name-value pairs) as well as the encoding and content of the response generated by PHP (i.e., json or xml).

**Submittal:**

This is an individual project! You must submit hardcopies of:

* the SQL files to create your table(s)
* your php scripts
* your js any other files

Submit a softcopy of the entire project.

You will also upload to Korra!