Homework 6: Search Application Functionality

Due Wednesday 3/4 @ 4pm on Courseworks.

What to submit:

- 2 uploads titled:
    - Hw6.zip: A Flask project containing:
        - server.py
        - templates/ (and the HTML templates you need)
        - static/ (and any static files you need)
    - hw6_writeup.pdf

Create a webpage that lets users interact with data. In this assignment, we focus on functionality, not usability or graphic design. The basic functionality your site must allow is to:

1. Create new data
2. View data
3. Delete data
4. Update data
5. Search the data.

You must use Flask, HTML, CSS, JQuery, and JavaScript. You may use Bootstrap if you want, but it's not required. The changes to the data must be saved to the Flask server.

You must pick a dataset that someone might want to search. Your dataset needs to have at least 30 items in it at the beginning. The user can add more items, but it should already be populated with 30 items. We suggest you create them by hand, like the list of local business were in the log_sales application. Typically, 30 items are not enough to need search functionality for, but we are going to implement it anyway.

Each data item needs to have multiple fields. At the minimum, this must include:

- An id (an integer assigned on the server side)
- A short title or name for the item
- An image, video, or gif. You must have an external link to this. You cannot download them and server the media locally (because when we run your site, we don't want to have to download all your media)
- A text paragraph of explanation (At least 4 sentences)
- Some sort of numerical data (a year, a price, a rating, etc)
- A list of some kind of data (such as a list of reviews for the movie)

Functionality requirements:

1. **Search data.**
    a. At the "/" route, users should see a text box and a button that allows them to search through all the data items to match a string in the text box.

b. When they press the button, make an ajax call to the server and ask for items that match the query text. What and how you match are up to you. We suggest you match the exact string in the input box to the name or title of the item.
c. When the results are returned it should produce a list of items that match the query text.
d. Each item in the list should be encased in a div.
e. If no results are returned print “No results found.”
f. When the user does a new search, it should clear the previous results and display the new matching results.

2. **View data.**
   a. Each of the divs in the search results should have a click function that take you to a new page.
   b. That page has the route ‘view/<id>’, where <id> is the id of the item the user wants to view.
   c. On the view page, it should use a template to display the all fields of the item except the id field. It doesn’t have to be attractive or well-designed in terms of information hierarchy, it just has to function and be readable.
3. **Create new data**.
   a. At the ‘/create’ route, have a series of text boxes and a submit button that allow the user to enter all the information needed to create a new data element and add it to the list.
   b. If the new element is successfully created, the site should return a link to view the new item.
   c. If it is unsuccessful it should report an error to the user on the page. (Do not use an alert box, or any other modal dialog box).
4. **Delete data**
   a. On the search page, there should be a red X next to each item.
   b. If the user presses it, the item is deleted on the back end and the result disappears from the list of items.
   c. To make it disappear from the front end, you can either re-render the query OR delete the item from the screen after the ajax call returns successfully. In the previous homework, we had you re-render the query any time the data changed. In this instance it’s okay to delete from the front end – but only in the success function of the ajax call.
5. **Update data**.
   a. Pick one data field that you think the user should be able to update. (Maybe they want to add their own review to a movie.)
   b. When the user is in the ‘view/<id>’ route, there should be a link that says “edit”. When you press it, it goes to the route ‘edit/<id>’ which shows you the editable text and allows you to change it.
   c. You must have two buttons – “submit” and “discard changes”.
   d. When you press “submit” the update is make on the server and the page reroutes to ‘view/<id>’ so they can verify that the changes were made.

e. If you press “discard changes” no updates are made to the server and the page reroutes to ‘view/<id>’ so they can verify that no changes were made.

Write up.

Please answer the following questions and submit them in a .pdf file

1. What data will your website allow users to search?

Example: *“1990’s comedy films.”*

2. What is an example of a person who would want to search this and why?

Example: *A college student is watching the Big Lebowski and want to know who directed that film, and what other films they directed to figure out what to watch next.*

3. Show up an example of one item in the database

Example:
{
Id: 1,
Title: “The Big Lebowski”,
Poster: “https://m.media-amazon.com/images/lebowski.jpg”,
Year: 1998,
Summary: “When ‘The Dude’ Lebowski is mistaken for a millionaire Lebowski, two thugs urinate on his rug to coerce him into paying a debt he knows nothing about. While attempting to gain recompense for the ruined rug from his wealthy counterpart, he accepts a one-time job with high pay-off. He enlists the help of his bowling buddy, Walter, a gun-toting Jewish-convert with anger issues. Deception leads to more trouble, and it soon seems that everyone from porn empire tycoons to nihilists want something from The Dude.”,
Director: [“Joel Coen”, “Ethan Coen”],
Budget: “$15,000,000”,
Stars: [“Jeff Bridges”, “John Goodman”, “Julianne Moore”],
Score: 8.1,
Genres: [“Comedy”, “crime”]
}

4. When deleting the data, how did you get the data item to disappear? There are two approaches, just list the approach you took.
5. Provide an example of what data a user would input to create a new item. (We will use this or something very close to it to test your code).
6. Provide an example of what a user would use to update an item.
   a. What should we search for to find the element to update?
   b. What text should we edit and what should we replace it with?