Charles Hardes CS 496 Fall 2016

Assignment 2

Deliverables:

1.) URL to live site: https://cs496-assignment2-146004.appspot.com

2.) Description of the site and what it can do:

The site is a bit of a tonuge-in-cheek online voting app. Originally, the idea was to visualize and create a very basic implementation of a way for people to vote online (as I'm sure is going to happen eventually). But as this election cycle got more depressing, the app got more cynical. The form at ballot.html takes five different types of input:

- A voting ID number (theoretical) as a text input box that only accepts numbers,
- a radio input to select a presidential candidate
- A checkbox input to select any number of up to 12 issues the voter deems important
- An email input with which to send a theoretical confirmation email
- A phone number input for theoretical follow-up/ confirmation with voters

The "vote" or "ballot" is submitted by pressing the "Cast Vote" button, which immediately enters all form data into a ndb database as an entity and promptly queries it back out and displays what vote information was registered into the database.

The "View All Votes" button brings up a table of every vote entity entered and its attributes. At the end of each row in the table is an edit (or "Rig This Vote") button and a delete button which allows the user to either modify any existing record in the database or delete it altogether. Pressing the delete button on a record simply deletes the record and refreshes the page immediately to show the updated table of records minus the one just deleted. Pressing the edit button on any record brings the user back to the original ballot.html screen, only the form will already filled with the original voting information. Modifying the information on the form and pressing "Cast Vote" updates the existing record, without generating a new vote. The "Enter New Vote" button at the bottom of the records viewing screen simply brings you back to the original ballot.html page with a blank form, with which to create a new record.

3.) Test Plan:

Test Description	**Result**
1.) Fill out form with arbitrary info, check GAE datastore to ensure a new record has been entered	Success
2.) Compare the entity attributes in each to enure that the record is entered in the database exactly as it was submitted via the form	Success
3.) Click "View All Votes", ensure record is displayed with all attributes and edit, delete buttons at the end of the table row	Success
4.) Ensure the record displayed on the display page exactly matches both the form data and the database entity	Success

5.) Check that a message was displayed at the end of ballot.html displaying all vote data entered by user and retrieved from database	Success
6.) Ensure the message data is accurate, compare to form and datastore	Success
7.) Use the “Enter New Vote” button to make another vote entry. Ensure that the button navigates back to the ballot.html page and the form content is clear with no message at the bottom	Success
8.) Make a second entry by repeating test 1 with the previous entry still in the database. Repeat tests 2-6, ensuring that the new record is correctly entered and displayed everywhere. Ensure data from both entities isn’t mixed or duplicated in any way.	Success
9.) Make several more entries by repeating steps 1-7, ensuring that each time, data is correctly displayed in the proper locations, experimenting with different form values to ensure that they are entered correctly.	Failure: Data is displayed correctly everywhere except the message at the bottom of ballot.html when more than one records are in the database. The data displayed is not necessarily that which was just entered, but almost always a different record
10.) Try several different combinations of the checkbox input element, including checking all elements. Check that they are entered properly	Success
11.) On the “Registered Votes” page, delete an entry with the “Delete This Vote” button. Ensure that the page is refreshed with the same identical table except for the entry you’ve just deleted.	Success
12.) Check the datastore to ensure the entry has likewise been deleted from the database	Success
13.) Check every other record to ensure no other attributes or records have been deleted/ modified	Success
14.) On the “Registered Votes” page, edit a record by pressing the “Rig This Vote” button at the end of the row of the record. Ensure this action takes you back to the Ballot.html page	Success
15.) After pressing the”Rig This Vote” button, compare the database entry of the record you have just selected to edit to what form fields have been filled/ selected. Ensure they are the same	Success
16.) Modify one or more fields of the form on ballot.html from the auto-populated form after pressing the “Rig This Vote” button. Press the “Cast Vote” button. Repeat steps 1-6 to ensure the newly modified vote is correctly entered.	Success
17.) More specifically, ensure that when performing test #16, that the number of records existing is noted. Ensure that the modified record is only that, a modification to an existing record, not a new record entry	Success
18.) Press the “Enter New Vote” from the Registered Votes page. Ensure the funcionality is the same, as when it was attempted in test 7, check that no complications have arisen.	Success

19.) Attempt to enter a new vote with a blank form.	Failure: This being a pretty complex and time consuming project for a one week introductory assignment, I only had time to implement the basic required functionality without decoratoin or error handling. As it stands, the app does not handle erroneous data gracefully, nor does it have many messages to users in error cases. If this were an app that users were actually going to use and/or time permitted, I would handle each error case, predominantly on the front end, with JavaScript as much as possible.
20.) Attempt normal entries but with one form element being blank	
21.) Make a vote with invalid data entry i.e. letters in the voter ID, etc	
22.) Use the “back”, “forward”, and “refresh” buttons to ensure they don’t break the app	
23.) Enter different urls to see what happens	
24.) Mess with the get data in the urls	
25.) Delete all records, press the View All Votes to see if there is a message indicating no records have yet been entered	

4.) The use of Templating:

I used templating via Jinja2 as heavily as possible, and probably much more than necessary. I relied heavily on the examples from the lecture as well as the Jinja2 documentation directly. The first way in which I used templating was to display the record data back to the user immediately after being entered. Pressing the submit button on the form instantaneously both entered the data into the database in the form of an entity with attributes, and also queried it to be displayed for the user’s confirmation of what was entered. A medium sized section was implemented with a Jinja2 if statement. When the form was submitted, a variable indicating as such was passed via template variables into the template to generate this section with the data that had just been entered into and queried from the database.

Using templates got a lot more complicated in the display page. I had to completely change the structure of the template variables in order to get them into a dictionary form instead of a simple list that would have been much easier. This was done because the template variables were set up as a dictionary, and trying to make them to be set up as a list was not something Jinja2 seemed to like. So, after finally successfully converting the template variables into a dictionary, I was able to pass those variables on to the template, and have the checkbox items iterated over in a for loop to display every single attribute of each entity.

Finally, I used templating once again to repopulate the fields of the original form when editing of records was performed. The most difficult thing about this, and with templating with Jinja2 in general, was trying to pass variables and data back and forth from html page to server page to template to html page, etc. A maze of get/ post requests and definitions, template variables, and queries were needed to perfrom a lot of these operations that might have been simpler to facilitate by simply generating html from the server pages. But this was a good lesson in templating.

5.)Changes

If I had to do it all over again, I might not use templating so extensively. Either that, or I would more exhaustively study the Jinja2 documentation because there seems to be almost as much functionality there as to be its own language. I'm certain that there were easier, quicker, shortcut ways to achieve what I put together here, but being new to templating tools and having a lot of work to do on this project in a short amount of time, I tried not to get bogged down in overcomplicated ways of doing things that would take an even more extensive amount of time to learn.

Certainly, if I had more time to work on this, and I may still…I would implement more user-friendly controls like error handling and style it up a bit. It would be even more interesting to provide not just a display of the records themselves, but perform some complex analyses of the data, like the percentage of each vote, statistics, and visual graphs and such.