Lab 6: Hangman
CMSC 362
Due: April 21, 2017

In the last lab, I had you create a database from an ER diagram. In this lab, you will take a simple online hangman game I've created and give it a database backend using the database you created in the previous lab.

Step 0. Setting Up

Create a folder named Lab6. In that folder, download "Hangman.tar.gz" from the course web site. The easiest way to do this is with wget:

wget http://narnia.homeunix.com/~robert/Spring2017/cs362/Hangman.tar.gz

Now extract the tarball:

tar xvf Hangman.tar.gz

This will extract the files hangman.html, hangman.js, Tile.png and hangman.css into your folder.

You can test out the game by typing:

firefox hangman.html

This will pull up the web page in your browser. The game is implemented in Javascript and can run without any kind of server code at all, but it doesn't keep score and if the user logs out, their game state is lost (which means they have to start the whole game over).

Step 1. Creating a Web Framework

Using Lab 4 as a model, create a flask application to run the hangman game:

A. Create a templates folder and move hangman.html to it.

B. Create a static folder and move hangman.css, hangman.js, and Tile.png to it.

C. Update the paths in hangman.html so that it loads hangman.js, hangman.css, and Tile.png from the static folder instead of the current folder.

D. Create a file named "game.py" that will act as a Flask server. Map the route "/" to hangman.html.

You may want to add other routes later (for example, to fetch a word from the database), but for now, you just need to get the game running on port 5000 so that if I browse to localhost:5000 in firefox, I can play hangman.

Step 2. Extending the Game

Read the W3Schools AJAX introduction (https://www.w3schools.com/jquery/jquery_ajax_intro.asp).

Then use JQuery's AJAX functions to modify the game in the following ways:

A. Instead of loading the words from a hard-coded array that shows up in the Javascript source code, store the words in your database and have the Javascript use AJAX to fetch them from the server. For now, you can just hard code a single word.

B. Modify your code so that when the user finishes a game, the game restarts with a new word. To do this, you will need to add code to the “reset_game” function in hangman.js. You will also need to add additional Python code that records (in the database) that a new game has started and what the new words is. You should also record the time the game started in the "time_started" field. Note that you do not need to actually know a username to do this -- you can hard code one for now.

C. Modify the game so that if I reload the web page or log in from a different system, it skips to the word of the most recent game instead of starting over.

D. Every time the user guesses a letter, update the "Guess" table to reflect that the letter they typed has been guessed in that game. If a game is restarted, "replay" all the guesses.

Step 3. Glitter

For glitter, extend the game in a creative and unusual way. Some ideas:

1. Implement logins so that different users can save and load their games independently.

2. Use CSS and an image editor to make the game look nicer.

3. (Even better) your own original idea!

Submitting

As usual, tar up your code and submit through the course web site.