Project 1C Specification

Project 1C

Due Monday, April 30, 2018 by 11:59pm

Late Submission Policy

To accommodate the emergencies that students may encounter, each team has 4-day grace period for late submission. The grace period can be used for any part of the project in the unit of one day. For example, a student may use 1-day grace period for Project 1B and 1-day grace period for Project 2B. Any single project (1 or 2) may not be more than 2 days late. Note that even if a team submits a project 12 hours late, they would need to use a full day grace period to avoid late penalty. If your project is submitted late, we will automatically use the available days in your grace period unless you specifically mention otherwise in the README file.

Disclaimer: To reduce the overhead to students in testing and validating your project, we will require you to demo the functionality of your project with a recorded video rather than by writing a Selenium driver, as in the past. Selenium is a framework that browses web sites in an automated fashion to mimic a human web surfer. Thus, these directions for part 1C come from a different source than the directions from 1A and 1B. *If you notice any discrepancies in these directions, please let us know on Piazza so that we can fix them.*

Project 1C is the final part of the Movie Database project and is very open-ended. You should start on it as soon as you finish Project 1B.

In Project 1C, you will create a fully functioning Movie Database system accessed by users exclusively through a Web interface. While there is much flexibility in the functionality required from your Movie Database system, all students are expected to implement the baseline capabilities described next.

Four input pages:

- Page I1: A page that lets users to add actor and/or director information. Here are some name examples: X.M.L Smith, J'son Lee, etc.
- Page I2: A page that lets users to add movie information.
  Some name examples: Beware the BLOB -- A Sequel, Willy Wonka and the Chocolate Factory
- Page I3: A page that lets users to add comments to movies. i.e. This movie was terrible.
- Page I4: A page that lets users to add "actor to movie" relation(s). i.e. Ryan Rosario *stars* in Alice in Wonderland (as the Madhatter, of course!)
- Page I5: A page that lets users to add "director to movie" relation(s). i.e. Johnny Depp *directs* Alice in Wonderland

Two browsing pages:

- Page B1: A page that shows actor information.
  - Show links to the movies that the actor was in.
- Page B2: A page that shows movie information.
  - Show Title, Producer, MPAA Rating, Director, Genre of this movie.
  - Show links to the actors/actresses that were in this movie.
  - Show the average score of the movie based on user feedbacks.
  - Show all user comments.
  - Contain "Add Comment" button which links to Page I3 where users can add comments.

One search page:

- Page S1: A page that lets users search for an actor/actress/movie through a keyword search interface. (For actor/actress, you should examine first/last name, and for movie, you should examine title.)
  Your search page **should** support **multi-word** search, such as "Tom Hanks". For multi-word search, interpret space as "AND" relation. That is, return all items that contain both words, "Tom" and "Hanks". Since the search page is for actor/actress/movie, so if there was a movie named "I love Tom Hanks!", it should be returned. As for the output, you should sort them in a way that users could find an item easily.

A demo site is available here. This page is available strictly to give you an idea of the basic requirements, and is *not* meant to guide your choice of style or user interface in any way. Please be creative, and do not simply mimic the UI of the demo site.

Important Notes:

- "Pages" mentioned above are units of conceptual functions that you need to provide, not distinct "files". It is OK to implement multiple "pages" as a single php page. For example, it is OK to combine S1 and B2 by adding a search box to a page that shows movie information. You can also use multiple php pages to implement a single "page" in our spec.
- While the functionality of your Movie Database system is quite open-ended, the interface itself is *extremely* open-ended. This class is not a user interface class and *you can certainly get most credit for a solid system* with simple input boxes, menus, and/or radio buttons, and simple HTML output tables as long as your web site is reasonably intuitive to use. That is, the user should be able to navigate through your web

site without too much difficulty. Having said this, we welcome much snazzier interfaces, like something resembling IMDb.com itself, or perhaps even something better!
- It is okay to embed some Javascript or CSS to make your website look good, but please make sure your work is **completely self-contained**. Namely, all javascript/CSS files related to your website should be incorporated in your submission.
- You may assume that the users are not malicious. They do not intentionally perform anything bad, such as an SQL injection. However, any PHP/database errors due to simple data entry errors or bad input values should be managed gracefully, i.e., it should be possible for users to continue interacting with the system, and the database should not get corruptted.
- You need to use the same set of tables (and only them) that you created in Part A of this project. You should use the database CS143 by connecting with username "cs143" and empty password.
- Please ensure that your work does not use more than 20MB space.

Although the project interface is very open-ended, we have to make the basic features of your system accessible in a typical browser environment. More precisely, you must make sure your system work with the most recent vesion of the Mozilla Firefox browser without any additional plugins or extensions.

Hints on implementation

- Please note that when your PHP scripts inserts a new tuple, your program will likely need to lookup either MaxPersonID or MaxMovieID table, assign a unique ID to the new actor/movie, and increase the MaxPersonID or MaxMovieID table value.
- In adding user reviews to a movie, you may need to obtain the current timestamp. You can do it either in your PHP code or in MySQL itself. The MySQL date and time functions page explains how you can obtain the current timestamp in MySQL.
- Note that your php pages can be accessed directly through a URL like http://localhost:1438/~cs143/mypage.php?param1=value1&param2=value2&param3=value3 where mypage.php is the name of your php page, and the string after the question mark contains parameters passed to the PHP page just like in a "GET" method. So you can *embed a clickable link to the page* generated by your php code using hyperlinks of the above format, with appropriate parameters that instruct your PHP code to execute the right query.

Your Project 1B

- Note that your project 1C is dependent on your project 1B. To make sure you can get 100% score, please make sure your project 1B is working correctly and fix any existing bugs. As part of project 1C submission, you will have to submit the relevant files of your (revised) project 1B again.

Demonstrate your website

By now you should already have a working website, and in this section you are going to show it. In order to do this, you must "record" **a (maximum) three-minute video**, showing the basic functionality and interaction of your site. More specifically, you need to demonstrate the following five cases in your video:

- `Case 1`: Use your search page to search for **Tom Hanks**, and it should return just ONE match (actor). Then you should click the actor "Tom Hanks" in your search result (it should be *a link pointing to a page that displays information about Tom Hanks*).
- `Case 2`: Use your web pages to create:
  - Actor: **Ryan Rosario, Male, born on 1997-11-11, alive now** (note: first name: **Ryan** & last name: **Rosario**).
  - Director: **Alice Liddell, born on 1976-12-09, died on 2016-01-01** (note: first name: **Alice** & last name: **Liddell**).
- `Case 3`: Create a fake movie, title:**Alice-in-SQLand**; company:**UCLA Computer Science** ; year:**2014**; director:**Alice Liddell**; genre: **Fantasy** and **Adventure**; MPAA: **G**; "Ryan Rosario" acted as role **MadHatter** in this movie.
- `Case 4`: Use your review page to create two reviews: (1) content: **very good** and score: **5** (2) content: **excellent** and score: **4.** And then use your search page to search **Alice-in-SQLand**, and your case should stop at the page where we can see the average is 4.5, two comments your wrote, and other movie info.
- `Case 5`: Use your search page to search keyword **santa** (note: all letters in lowercase in this case), and this case should stop at the search result page, which should contain 7 actors (Actor: Toni Santagata(1940-12-09), Actor: Yusuke Santamaria(1971-03-12), Actor: Carlos Santana(1947-07-20), Actor: Henrique Santana(1924-05-07), Actor: Merlin Santana(1976-03-14), Actor: Peter Santana(1966-03-29), Actor: Carolina Santangelo(1986-08-05)) and 3 movies (Movie: Santa Claws(1996), Movie: Santa Fe(1997), Movie: Santa with Muscles(1996)).

There exist a number of excellent free/paid software for recording your computer screen while you demonstrate the functionality of your Web site. For example, OBS Studio is a free open-source software that is widely using for screen capturing and live video streaming. You can go over this short tutorial on OBS Studio if you decide to use OBS Studio and wants to learn how to use it to record your computer screen. Whatever software you use, make sure that your video is playable with the latest version of VLC Media Player (without installing any proprietary codec) to avoid any codec/format incompatibility issue. Recording your video using the H.264/MPEG4/WebM codec in the MP4/MOV/MKV container format will be a safe choice. If you use a Mac, you may be able to use QuickTime to record the screen as well as your voice.

Submission Instruction

Preparing Your Submission

Your submission should consist of 1) **P1C.zip** file that contains all of your website related files, 2) **a (maximum) three-minute video** file that

demonstrate the functionality of your website. You can name the video file as **UID_demo.EXT**, replacing UID with the submitter's UID and EXT with the actual video extension that you use. You need to submit P1C.zip and the video separately on CCLE.

P1C.zip Structure

Create **a folder named as your UID**, and create "`sql`" and "`www`" folders within your UID folder. Put your `create.sql` and `load.sql` files from project 1B into the `sql` folder, all your web site source files into the `www` folder, and your README file into the UID folder. Compress your UID folder into a single zip file called "P1C.zip". In summary, your zip file should have the following structure.

```
P1C.zip
 |
 +- Folder named with Your UID, like "904200000" (without quotes)
     |
     +- readme.txt
     |
     +- team.txt
     |
     +- sql
         |
         +- create.sql
         |
         +- load.sql
         |
     +- www (all your website files are put under this folder, files listed are just examples, you can
         |
         +- index.php
         |
         +- search.php
         |
         +- others.php
         :
         :
```

Please note that the **file names are case sensitive**, so you should use the exact same cases for the file names. (For team work, use the submitter's UID to name the folder)

- `readme.txt`: A README file, with a detailed explanation of which of the project criteria you met, as well as any additional features you have implemented.
  - If you worked as a team with a partner, briefly explain how you split the work and collaborated in your README file. Also briefly explain what aspect you feel you can improve in a team setting for better collaboration.
- **`team.txt`**: A plain-text file (no word or PDF, please) that contains the UID(s) of every member of your team. If you work alone, just write your UID (e.g. 904200000).
  If you work with a partner, write both UIDs separated by a comma (e.g. 904200000, 904200001). **Do not include any other content in this file!**
- `sql/`: a folder which contains your (revised) `create.sql` and `load.sql` files from project 1B.
  - Please make sure that when you run **mysql CS143 < ./create.sql** and **mysql CS143 < ./load.sql** in this directory, there is no error.
- `www/`: a folder which contains all the source code and other supplementary files of your web site
  - After loading the database, the TA should be able to simply copy the files in this directory to ${HOME}/www and run your web site without any issues.
  - Your web site should use the database CS143 by connecting with username "cs143" and empty password.
  - Remember to use relative URLs in your machine/port references. You may be get zero point for code with faulty URL links that don't work during grading.

Testing of Your Submission

Grading is a difficult and time-consuming process, and file naming and packaging convention is very important to test your submission without any error. In order to help you ensure the correct packaging of your submission, you can test your packaging by downloading this test script. In essence, this script unzips your submission to a temporary directory and tests whether or not you have included all your submission files. The script for Project 1C can be executed like:

```
cs143@cs143:~$ ./p1c_test <Your UID>
```

(Put your P1C.zip file in the same directory with this test script, you may need to use "chmod +x p1c_test" if there is a permission error).
You **MUST** test your submission using the script before your final submission to minimize the chance of an unexpected error during grading. Again, significant points may be deducted if the grader encounters an error during grading.When evertything runs properly, you will see an output similar to the following from this script:

```
Check File Successfully. Please upload your P1C.zip file to CCLE.
```

Submitting Your Zip File and Video

Visit the Project 1C submission page and submit **your "P1C.zip" file and the video you recorded** electronically by the deadline. In order to accomodate the last minite snafu during submission, you will have 30-minute window after the deadline to finish your submission process. That is, as long as you start your submission before the deadline and complete within 30 minutes after the deadline, we won't deduct your grade period without any penalty.

Grading:

- (10%) We can navigate your website using Firefox without any errors, and your websites contain all assigned pages.
- (20%) All your cases start at a web server with address: **localhost:1438**, and your cases complete without encountering any error/warning.
- (50%) Each case is 10% if it acts as expected (correct result). Please follow "Demonstrate your website" precisely, especially all the values (case sensitive) you are asked to enter.
- (10%) 1 additional test case designed by our grader/TA.
- (10%) Better designed and decorated websites may get higher points in this part. **But do not devote too much time on it. Make sure all cases can work first!**

Notes: Start your project early and use our discussion forum, asking for clarifications if any part of this spec looks unclear to you.

Last modified: Wednesday, 25 April 2018, 2:42 PM PDT