CS 377: Database Systems

Project #4: Course Suggestion Website Part II

Due: April 24th 2017 at 11:59 PM on Canvas

Instructions

This is a continuation of the course suggestion web application from Project #3, where you built much of the backend aspect of the website. This project focuses on using PHP to build the website. Note that you will not be graded on aesthetics but purely on functionality (whether or not it does what the requirements specify). You will submit your PHP scripts which will be evaluated on a separate server using your database SQL scripts to populate the database accordingly. We will then modify the courses, pre-requisites, and degree requirements on the server, so make sure you are not hard–coding too much onto your webpages.

1 Google Cloud Server Setup (20 points)

Setup a server using Google Cloud that you can use as your sandbox.

1. Request a Google Cloud coupon for the course from this URL: `https://goo.gl/gcpedu/bbSOq5`. You will be asked for a name and email address (emory.edu address). A confirmation email will be sent to you with a coupon code. Please make sure you request **ONLY ONE** coupon – there is a maximum number of coupons that were assigned to the course.

2. Setup a VM instance using Ubuntu 16.04. You will want to use the cheapest machine (you don't need more) – 1 core with 3.76 GB of RAM, us-east-1c so that you can get the most out of your Google credit. Make sure you use the us-east regions as this is the cheapest and is geographically closer. If you want to connect the server using ssh, you will want to follow these instructions: `https://cloud.google.com/compute/docs/instances/connecting-to-instance`

3. Install the LAMP stack on your server: `https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-on-ubuntu-16-04#step-1-install-apache-and-allow-in-firewall`

4. Set up your MySQL server to have the user cs377 and password cs377_s17 (same as `cs377db.mathcs.emory.edu`). Make sure you grant access to this user/password to your database. Your PHP scripts **MUST USE** the cs377 user so that we can replicate your web application on our server.

5. Load your database from Project # 3 using the SQL scripts you created.

2 User Input Form (20 points)

Design a webpage that allows the users to enter their major(s) and minor, the list of courses they've taken thus far (and passed), what the current semester / year is, and their desired date of graduation. How you choose to allow user input is up to you (free-form, radio button, check box, dropdown menu, etc.). You'll want to sanitize your user input such as:

- The major and minors should be the ones offered by the MathCS department.
- At least 1 major is entered.
- No more than 2 majors (or 1 major and 1 minor) can be entered by the user.
- The courses should be ones that have been offered by the MathCS department.
- Check that the desired graduation date is after the current semester / year.

If any of the user inputs is invalid, your webpage should be able to specify which one was incorrect so the user can fix it.

3 Course Suggestion Page (45 + 15 = 60 points)

Design a webpage that will consist of two parts, one with the course planner and one with course information.

3.1 Course Planner

For each semester up until graduation, list the courses that the student should take to satisfy their major/minor requirements. Assume that no classes are offered in the summer (i.e., students can only take courses in the fall and spring). Also assume that no more than 4 MathCS courses can be taken in any semester. If a student does not have sufficient time to complete the degree, your webpage should mention that. To design the "optimal" plan you will want to apply the following heuristics in the order specified:

1. Schedule the courses which are prerequisites for other courses as soon as possible (i.e., CS 171 if not already taken).
2. Schedule the courses that are only offered in either fall or spring as early as possible (assuming the prerequisites are met by that semester).
3. For the "elective" courses, favor the courses that can be taken the earliest and if multiple courses can be taken in semester, denote the options by listing them all (i.e., CS 325 / CS 378 for Spring semester).

3.2 Course Information

For the courses that are suggested, provide some basic information about the course. This includes the course number, course description, prerequisites for the course, when it is offered, and the number of credits.

Bonus (20 points)

Students may be interested to see how much more they need to do to finish another major / minor (assuming they are only doing one major at the moment). If the student has only entered one major on the input page, your course planner should also calculate whether or not they can pick up another major or minor within the specified amount of time. The web application should favor a major over a minor whenever possible (i.e., if an Applied Mathematics major has taken almost all the coursework for a Computer Science major, then suggest the CS major before CS minor). The website should suggest the closest major/minor that the student can pick up and the courses they need to take in addition to fulfill the additional major/minor's requirements.The extra courses should be highlighted on the course planner either using a different color font or different font style (e.g., italics or bold).

Submission Instructions

Although project 3 and 4 will be submitted together, what you will end up submitting for this portion of the project is the following:

- PHP files (one for course planner, one for course information)
- `README.txt` file containing the honor code
- `server.txt` text file that contains a working link to your google server (IP address).

Please compress all your files from project 3 and 4 into a `zip` file. Do not nest another layer of folders in your `zip` file. We will load all your files onto our own Google Cloud server instance and modifying the tuples using the relational data files `<relation-name>.csv` you provided. To test your web application, we will add / delete some tuples to ensure that your program dynamically adapts to the database instance while still adhering to the requirements we have specified. If we are unable to successfully recreate your project on our server, we will use the link provided in the `server.txt` file. However, if we are forced to grade your project from your google server, the **maximum points you will get is 80**.