CS 377: Database Systems
Final Project: Learning Management System

Due: Tuesday, December 7th, 2021 at 11:59 PM on Canvas

Overview

The Mathematics and Computer Science (MathCS) Department wants to hire you to create a web application to replace Canvas, our learning management system. The learning management system will allow students and faculty to manage grades and discussion questions seamlessly (think of this as simplified Canvas with Piazza support integrated seamlessly).

This project "walks" you through the process of building both the backend (database design, creation, and population) and the frontend, which will use PHP to provide a dynamic website. A few things of note:

- Read the *entire assignment* through before you start. You are likely to change your database design if you take it one question at a time. Once you read what your web application should support, you may decide that certain assumptions may make your life easier!
- Your final project grade will be based **purely on functionality** (whether or not it meets the specified requirements) and not on aesthetics.
- The work must be your own. Given the amount of freedom that exists, each project should look (in both frontend + backend) unique. We will run similarity software (MOSS) on the final project.
- Although you will have your own webserver on the Google Cloud Platform, we will be re-creating your application on our own local Google Cloud server. This means that we *will add/delete tuples* from the data files you submit.
- You want to avoid hard-coding information into your web application (it should read dynamically from the database).

Submission Instructions

All the work should be submitted electronically *in a single zip file* (e.g., FinalProject.zip) via *Canvas*. Do not nest another layer of folders in your `zip` file. The contents of the zip file must contain and adhere to the following guidelines:

- A `README.txt` file where you signed your name, otherwise points will be deducted.

  ```
  /* THIS CODE IS MY OWN WORK.
  IT WAS WRITTEN WITHOUT CONSULTING CODE WRITTEN BY OTHER STUDENTS.
  _Your_Name_Here_ */
  ```

- A single *typeset* PDF that contains your ER design, relational schema, database normalization work, hyperlink to your web application (Questions 1-4), a data description detailing what each relation data file is and what the columns your relation data files encapsulates, a summary of what each PHP file you submit does (related to Questions 7-10). This allows us to easily figure out which file should be used to grade which problem. Make sure to do this even if you only have a single PHP file.
- `createtables.sql` and `dropdatabase.sql` (Question 5)

- Relation data files: `<relation-name>.csv` (Question 6)
- `populatetables.sql` (Question 6)
- `*.php` files for your web application (Questions 7-10).

We will load all your files onto our own Google Cloud server instance. 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 your assignment PDF. Note that if we are forced to grade your project from *your* google server, the **maximum points you will get is 180**.

Question 1: Database Design (15 points)

Design an ER diagram for the website based on the following requirements:

- A class is assigned a course name, course number, semester and year. For example, the database course you are currently taking is course name Database Systems, course number CS 377, Fall semester, and 2021 respectively. For the purpose of this project, we will assume there is only one offering of the course is available in any given semester.
- A student can take any number of classes.
- Each course must be taught by an instructor and may have teaching assistants who are students. Note that an instructor and teaching assistant can be a student in another class.
- A student (instructor and teaching assistant) is assigned a unique student identifier, a login ID, and their first name and last name.
- Each course can have several assignments. To simplify your application, we will consider all quizzes, surveys, and exams to be assignments. For example, the course you are taking has 8 assignments (5 assignments, 1 final project, 2 exams). Each assignment has a due date (just the date and not a timestamp), assignment-related text, and the total number of points for the assignment. For this project, you do not need to support assignment submission (e.g., submission is done on Gradescope).
- A student will receive a numeric grade (i.e., 0-100) for each assignment that can only be entered by the instructor or a teaching assistant.
- A student will receive a final *letter* grade for the course[1].
- A class can have multiple question and answer posts, similar to how Piazza works. A student or instructor can post a question with a title, one or more tags (e.g., assignment1, exam, etc.), the text, and the post date. You should assume that some classes may also not have any posts associated with it.
- For each Q&A post, there is a "thread" or a sequence of replies from students and instructors. Each reply should contain the following information: who posted it, the time, and the text. For the purpose of the project, you do not need to allow post anonymity (i.e., all names should be visible) nor do you need to differentiate whether the poster is a student or a a member of the teaching staff.

Justify any additional assumptions you make about your project.

[1]While you can in theory calculate it from the assignment grades, this may complicate your web application design

Question 2: Relational Model Creation (15 points)

Design the relational model from your ER diagram. Make sure the key attributes are underlined and the foreign keys arrows are drawn to the correct place.

Question 3: Database Normalization (15 points)

What are functional dependencies that you think should hold in your application? Note that you should have at least 3 functional dependencies. Normalize your relational schema to meet 3NF based on your functional dependencies.

Question 4: GCP WebServer (10 points)

Hopefully you already setup a virtual machine on Google Cloud Platform from Weekly Prep 12. If not, you will have to do it here. As a reminder, assuming you've set up the VM based on the specs, you **will not have to pay anything for this project**. Note that the instance should cost around 30 dollars a month, so it can run for at least 5 weeks. To be safe, you may want to stop the instance whenever you're not working on it until you are completely finished.

Once you have a fully functional webserver and are ready to submit the final project, do not stop and restart your server as it will assign a new IP address each time. *Place a hyperlink to your web application in your PDF document.* This will be used to verify that you in fact have a running web application to get the 10 points for this question, and serve as a backup if we are unable to replicate your web application on our server. We will let you know once grading is complete, to stop your instance.

(a) Create a cs377 user for your MySQL database. You should follow the first set of instructions in the following webpage. `https://www.digitalocean.com/community/tutorials/how-to-create-a-new-user-and-grant-permissions-in-mysql`.

- user: 'cs377'
- password 'ma9BcF@Y' (music apple 9 BESTBUY coffee FRUIT @ YELP)

Your PHP scripts **MUST USE** the cs377 user so that we can replicate your web application on our Google Cloud Platform server. Note that all the PHP examples support this user/password pair so you can test your setup by loading the company database and making sure you can at least run the PHP examples from your GCP instance.

(b) Next, enable the MySQL client and server to load from a local file. To do this, add the following information to the mysql configuration file located at `/etc/mysql/my.cnf`

```
[mysqld]
local-infile

[mysql]
local-infile
```

Once you are done, restart the sql server using the following command:

```
sudo service mysql restart
```

Question 5: SQL Database Creation (5 points)

Create two MySQL files:

(a) (3 points) `createdatabase.sql`: Create the tables (implement the normalized relational schema from the previous question)

(b) (2 points) `dropdatabase.sql`: Clean out your database. This script should remove any trace of itself from the MySQL server.

We (and you) will use this file to build / re-build your database. It is important to assume that you have a "clean" MySQL instance when we run `createtables.sql`. In other words, you must create the schema and delete the schema appropriately.

Run the `createdatabase.sql` script from the command line:

```
mysql -u cs377 -p < createdatabase.sql
```

Question 6: Database Population (20 points)

You will use the bulk load option to populate your MySQL database. Your first task is to take the data that we've provided and convert it into a format that is consistent with your relational schema. We have provided two data files `canvas.csv` and `qa.csv` where each line contains starting information that your application should have[2]. Inside the data file, each line will represent a tuple with commas separating the different attribute values. The `canvas.csv` file is formatted such that:

- User information is captured in the first 4 columns with user ID (unique identifier), login ID, first name, and last name.
- The course-related information is the next 4 entries with course number, course name, semester, and year.
- Teaching staff consists of the next 24 columns and contains their user ID, login ID, first name, and last name. Note that we are assuming there are a max of 5 TAs the way the spreadsheet is formatted so there are a lot of null values for the TA entries (usually a given course has less than 5).
- Assignment details are the next 50 columns. Each assignment has the name, due date, text, total points, and assignment grade.
- The last column is the final letter grade. Courses that are incomplete will have a null value for this.

The `qa.csv` file is formatted such that:

- The course-related information is the first 4 entries with course number, course name, semester, and year (the content is consistent with a course in the `canvas.csv` file).
- The contents related to the post is captured in the next 7 columns with the post title, post date, post text, post tags, and post author information (user ID, first name, last name). If a post has multiple tags, it will be tags will be separated with a ;. As an example, the second tuple has two tags, logistics and assignment1.
- The responses consist of the next 25 columns and contains the date, text, userID, first name, and last name of the person responding. The spreadsheet assumes there are only a maximum of 5 responses but this need not be the case in your schema.
- If there are supported tags that are not yet used, they will be listed associated with the course while all other fields are empty (see row 10 in the file).

(a) (10 points) Take the data in the `canvas.csv` and `qa.csv` file and make new `*.csv` files such that each relation has its own data file (`<relation-name>.csv`). As an example for the company database, the `department.csv` file will have 3 rows:

[2]It is important to note that all the assignment scores have been randomly generated as well as most of the names save for some of the instructors in Computer Science and the TAs for this course. In addition, there are null values everywhere in these files and your schema should avoid null values whenever possible.

```
'Research', 5, '333445555', '1988-05-22'
'Administration', 4, '987654321', '1995-01-01'
'Headquarters', 1, '888665555', '1981-06-19'
```

You are welcome to add more students, instructors, courses, posts, and assignments, but it must contain all the information provided as a minimum.

(b) (3 points) Create another SQL file called `populatetables.sql` that will populate all your relational tables using the bulk import capability. Here is an example of the bulk load command in MySQL using the `department.csv` file above.

```
LOAD DATA LOCAL INFILE 'department.csv' INTO TABLE department
FIELDS TERMINATED BY ',';
```

If you keep the header in the .csv files (i.e., the name of the attributes as the first row), you should add the IGNORE 1 LINES to the bulk load command:

```
LOAD DATA LOCAL INFILE 'department.csv' INTO TABLE department
FIELDS TERMINATED BY ',' IGNORE 1 LINES;
```

Load your database with your new SQL tables:

```
mysql -u cs377 -p < populatetables.sql
```

Double-check that all your data was properly loaded into your schema.

(c) (7 points) Provide a database description (or documention) for your schema in your typeset PDF. As an example for the DEPARTMENT table in the company database:

```
The DEPARTMENT table defines all the departments present in the company.

ATTRIBUTE NAME | ATTRIBUTE TYPE | DESCRIPTION
-----------------------------------------------------------------
dnumber        | numeric        | Unique identifier for department
dname          | varchar(15)    | Unique name of department
mgrssn         | char(9)        | SSN of department manager
mgrstartdate   | date           | start date for department manager
```

Question 7: Student Course Page (20 points)

Design a webpage(s) that allows a student to view a particular course they have taken or are taking. You have freedom to design the webpage(s) however you think is the easiest for you so long as it meets the following requirements:

(a) (5 points) At least one webpage that allows the student to view a particular course content (e.g., CS 377 Fall 2021).

(b) (5 points) Allow the student to view their current assignment grades.

(c) (5 points) Allow the student to view the details of any assignment (i.e., due date, text, and total number of points).

(d) (5 points) Allow the student to view their final letter grade for the course.

Question 8: Teaching Staff Page(s) (30 points)

Design a webpage(s) that allows the instructor/teaching assistant to see all the students who are taking a specific course they are teaching. You have freedom to design the webpage(s) however you think is the easiest for you so long as it allows the teacher to perform the following actions:

(a) (5 points) At least one webpage that allows the instructor/teaching assistant to view a particular course content (e.g., CS 377 Fall 2021).

(b) (8 points) Create a new assignment for the course with text (e.g., Final Project worth 200 points due 12/7 at 11:59 PM).

(c) (7 points) Enter grades for each student for a specific assignment.

(d) (5 points) Show the current grades of all students in the course for all the assignments.

(e) (5 points) Enter the letter grade for the student at the end of the semester.

Question 9: Q&A Page(s) (35 points)

Design a webpage(s) that allows users (i.e., students, instructors, teaching assistants) to view, post, and answer questions associated with their courses. For the purpose of this project, you can assume the user can only choose tags that have already been defined in the database for the course (i.e., those already existing in the qa.csv file for that particular course/semester). You have the freedom to incorporate this functionality into the above pages from the previous two questions or as new standalone pages.

(a) (15 points) Allow the user to view existing posts and their associated threads. Posts should be displayed in the order of their timestamp. You have the freedom to choose ascending or descending.

(b) (5 points) Allow the user to filter the questions and threads based on the tag of the posts.

(c) (8 points) Allow the user to create a new post by entering the title, one or more tags that are pre-defined in the database, and the text of their post. Note that the date should automatically be populated once they submit.

(d) (7 points) Allow the user to respond to a post by entering their reply text.

Question 10: Home/Login Page (30 points)

Design a webpage(s) that allows the user to login to your application, which will then lead them to a "home" page that shows all the courses applicable to that user. This home page should be used to navigate to the specific student course page or teaching course page built in the previous two questions.

(a) (10 points) A webpage to "login" to the application. Users will enter their student ID and their login ID. If the student ID / login ID pair is not valid, your application should inform the user of this. Note that authentication has been significantly simplified to make it easier for the project. You do not need to support HTTP authentication, this is simply a check to make sure that there is such a user in the system.

(b) (10 points) Once the user has logged in, they should be shown all the courses that they can access either as a student or a faculty.

(c) (10 points) The user should then be able to click on a specific course that takes them to the webpages built in Questions 7-9 depending on whether they are a student or member of the teaching staff for the course in question.

Note that you do not need to have the capability to add a course from the web application perspective. Instead, you can assume courses will be added to the database and your home page should reflect any new courses that were added. For the purpose of the project, you can also keep around all the courses even if they have completed. Students may want to come back and refer to old content as well.

Question 11: Optional Student Transcript (Bonus) (20 points)

This is a **bonus question** and you should do this after you finish Questions 7-10. You *will not receive points for this part unless the other functionality is complete.* Design a page for the student to automatically get the transcript from all their classes that they have taken. This will look similar to your academic transcript. If the user is both an instructor and a student, the transcript should only include the courses that s/he has taken as a student.

(a) (15 points) The page should have the student's name and contain all the letter grades from all the classes the student is taking / has taken. For each course, it should list the class number, the name of the class and the grade. For example, CS377 Database Systems A. It should be obvious from the webpage what is the class number, the class name, and the grade. For classes that don't yet have a letter grade, it should have a blank entry.

(b) (5 points) Classes should be grouped and sorted based on the academic semester and year (e.g., Spring 2021 courses followed by Fall 2021 courses which are then followed by Spring 2022 block).