Week 13 Assignment - Web Development with Flask (2/2)

Overview

This week, which is our last week for web development, we reviewed how we can extend our Flask applications to use a relational database (SQLite). In this vein, you will develop a small application in Flask. The purpose of this application is to keep track of students that are enrolled in a class, and the score they have received on quizzes in the class. You will have to import a database model in a SQLite database, and use this database to allow a teacher to:

1. Add students, add quizzes and to add quiz results to the database
2. View the current list of students, quizzes and the student's quiz results

Make sure to create a github repository for this assignment named **IS211_Assignment13**. All development should be done in this repository.

Useful Reminders

1. Read the assignment over a few times. At least twice. It always helps to have a clear picture of the overall assignment when understanding how to build a solution.
2. Think about the problem for a while, and even try writing or drawing a solution using pencil and paper or a whiteboard.
3. Before submitting the assignment, review the "Functional Requirements" section and make sure you hit all the points. This will not guarantee a perfect score, however.

Part I - Database Setup and Initialization

Given the description above, we know there are three entities in the problem:

1. Students
2. Quizzes
3. Student's Results on Quizzes

Lets describe each one:

1. Students represent students taking a class. Students have a first name, a last name, and a unique integer ID.
2. Quizzes are the quizzes that have been given in the class. A quiz has a unique ID, a subject (i.e. "Python Basics"), a certain number of questions, and a date representing the day the quiz was given.
3. Student's Results can be modeled as linking one student to one quiz, with an integer representing their score (from 0 - 100).

Given this, you should design a schema for this problem. and save this schema to a file named 'schema.sql' in your repo. After you create the schema, please create a SQLite database file called 'hw13.db'. To help with debugging the next sections of code, load some data into the database that represents:

- a student named "John Smith"

- one quiz with a subject of “Python Basics”, that has 5 questions and was given on “February, 5th, 2015”
- and that “John Smith” received a 85 on the quiz

Part II - Teacher Login

The ‘/login’ route of this application should render a simple login form, which asks for a username and password. The form on this page should submit to the ‘/login’ route, and upon submission:

- should redirect to ‘/dashboard’ route, which we will develop later, if the username and password credentials are correct
- should redirect back to the ‘/login’ route, with an error message, when the credentials are incorrect

For the application, the username should be ‘admin’, and the password ‘password’ (this is obviously not secure!). This is similar to the Flaskr application from the reading. All subsequent controllers should check if the request is being made by a logged in user. If not, the controller should redirect back to the login page.

Part III - Dashboard: View students and quizzes in the class

The next step is to design a ‘dashboard’ page (located at ‘/dashboard’), which shows a listing of students in the class and a listing of quizzes in the class, in two separate tables. Each row of the student table should list the ID, first and last name of all student enrolled in this class. Each row of the quiz table should list the ID, subject, number of questions and the quiz date.

Part IV - Add students to the class

We need to be able to add students. Update the dashboard page to include a link to the “Add Student Page”, which will be located at ‘/student/add’. Create a controller at this route that should:

- Display an HTML form capable of adding a new student (HINT: there should be no reason to have an ID field in this form, as that should be taken care of by the database)
- Accepts the HTML form and attempts to add a new student to the database with the given form information. Upon success, redirect back to the ‘/dashboard’ route. If there is a failure, return the same HTML form with an error message.

Part V - Add Quizzes to the Class

We also need to be able to add quizzes. Repeat the prior step, but for a quiz instead of a student. This route should be located at ‘/quiz/add’. Think about how best to represent a date using HTML forms, as you have a few options.

Part VI - View Quiz Results

Now that we have a way of listing and adding both students and quizzes, we need to see student’s results on their quizzes. We’ll accomplish this by updating the Student listing in the dashboard output to include a link. This link should point to route that has a format of ‘/student/<id>’, where id is the ID of the student. This route should display all quiz results for the student with the given ID. If there are no results, you should

output “No Results’ to the page; otherwise, an HTML table should be displayed showing the Quiz ID and the Score on the quiz.

Part VII - Add a Student’s Quiz Result

We’re almost done, but we now need to handle recording a student’s grade for a quiz. Update the dashboard to include a link called “Add Quiz Result” that links to the ‘/results/add’ route. This route should display an HTML form capable of adding a quiz result. Since a result links a student and a quiz together, we need to be able to select a student and to select a quiz. Implement both of these as a dropdown menu, which lists the possible students and quizzes to choose from. Don’t forget to add an input field for the grade as well. If successful, this should redirect to the dashboard; if there is a failure, show the HTML form again with an error message.

Optional Part: Expand the Results Output

Using a JOIN SQL statement, expand the output to show not only the Quiz ID, but also the date the quiz was given and the subject of the quiz.

Optional Part: Deletions

Allow the application to delete quizzes, students and their results.

Optional Part: Anonymous View of Quiz Results

Allow non-logged in users to see an anonymous view of the quiz results. Given a quiz ID, the route ‘/quiz/<id>/results/’ should display all the student’s results but only display the student’s ID (so this non-logged in user, like a student, can see the class results but cannot see the name associated with the grades). You can further expand this to allow the admin user to see the student’s name associated with the result.

Functional Requirements

The web application must:

1. Allow a teacher to login via a username and password
2. Allow this teacher to view and add students to the roster
3. Allow this teacher to view and add quizzes in the class
4. Allow this teacher to view and add student’s quiz results

The web application can assume:

1. There is only one class to worry about

The web application can optionally:

1. Display extended information for quix results
2. Allow this teacher to delete students, quizzes or results

3. Allow a non-logged in user to see quiz results but only in a anonymized way (i.e. show a student ID instead of the user name)