Columbia Tutoring Services Platform

README FILE

Project 1 – Part 3

Course: Introduction to Databases | COMS W 4111
Section: sec 001

Team:

Deema Alnuhait	daa2182
Haojun Li	hl3509

PostgreSQL account name: daa2182
URL: http://35.231.126.135:8111/

Description of what we implement (Web Front-End Option):

In this project we developed a platform for Columbia tutoring services. The data is collected from Columbia Vergil (*Vergil, Your Personal Course Planning Guide | Columbia University*, 2021) courses' and departments' information along with some test data for the users information. We have successfully implemented all the functions discussed in the first part of this project and we have also successfully caught all constraints discussed in the first part of this project. This includes all the 12 functions listed below. These functions successfully showed the full functionality of our database through an assortment of user's inputs.

1. **Registration, login, and logout**
   We designed a registration page to ask users to input their personal information and register into our system. We also designed a login page for users and, once they input a correct username and password, they can have access to the dashboard. Finally, we included a "Logout" button within our menu for users to logout.

2. **Advertisements management**
   Users now can click the "Advertisement Center" button within our menu to be redirected to the advertisement center. Within that web page, users can browse advertisements, place orders, click tutors' usernames to view tutors' information, click buttons to access search advertisement functionalities, and click the "My advertisement" button to manage their advertisements (browse, post, delete, and update their own advertisements).

3. **Placing orders**
   Our implementation enables students to browse, place, and cancel their orders. They can cancel only the active orders in which their appointment time has not passed.

4. **Updating profiles**

Our website provides a page to allow the user to view and update his/her profile information. They can input updated information and submit the request. Older information will be replaced.

5. **Becoming a VIP**

   Our implementation provides a page for users to become one of our VIPs or to extend their VIP memberships. VIPs will enjoy a discount (20%) on their future orders (All VIPs have expiration dates).

6. **Rating tutors' services**

   Our implementation enables the student to rate the tutor after having a session with him/her on a scale from 1 to 5 where 5 is the highest. We enforce the constraint that students are only allowed to rate tutors who taught them before.

7. **Browsing tutoring sessions under specific departments**

   Users can access this functionality by clicking the "Search By Department" button in the "Advertisement Center." Our implementation allows users to select a department and display all advertisements under that department. Users can also browse advertisements, place orders, and view tutors' information (just click tutors' usernames).

8. **System notification boxes**

   The website notified the related users of any changes related to orders and advertisements. The user can access his/her notifications through the notification Box option displayed in the main menu of each page.

9. **Searching tutoring sessions based on course names or tutors' names**

   Our implementation allows users to select a specific course from a drop down list and display advertisements related to that course. Our implementation also allows users to select a specific tutor's username from a drop down list and display advertisements related to that tutor. Users can

also browse advertisements, place orders, and view tutors' information (just click tutors' usernames).

10. Recommending the top five best tutors

Once the student is done with rating a tutor, the system shows him/her a list of top five tutors based on tutors' average scores so the user could look for them next time he wants to book a session in the course he chooses. The user can also click the username of the tutor to see his/her profile.

11. Browsing other user's profile

The website allowed the user to view other users' profiles either tutors or not. They could search for the username or click his/her name throughout the session booking process. All the profile info and other people's ratings about him/ her are displayed except the phone number and bank information as displaying them will raise several privacy concerns.

12. Browsing orders

The system allows the student to list his/her orders along with the orders' details. There is an option in the menu to view all orders along with their details. However the user is only allowed to cancel placed orders as long as their appointment time has not passed (active). The user can view any tutors profile by clicking on their usernames.

Interesting Database Operations:

In the following section we list two of the web pages that require the most interesting database operations.

1. **Web page 1 with the most interesting database operations (Searching and browsing advertisements by course name):**

   In this page the user can search for tutoring sessions belonging to a specific course. So he selects a certain course and the system retrieves all the advertised sessions under that course. This is useful for students looking for help in certain courses.

   The inputs from the web page to this database operation are course ID and department ID. The tables related to the functionality of this page are the 'Adertisements_manage_associate', 'Courses_belongs', and 'Departments.' We create a SQL query that concatenates these three tables and applies course ID and department ID as filters to select all unexpired sessions related to that user-specified course. An example of a query for this function is shown below which uses 'Adertisements_manage_associate', 'Departments', and 'Courses_belongs' to return all the active (unexpired) advertised sessions pertaining to the course the user selected:

   ```
   'SELECT AD.AdID, AD.Location, AD.AppointmentTime, AD.AvailableSeats, AD.Price, AD.Comments, AD.Username, D.DepartmentName, C.CourseName, C.CoureDescription FROM Adertisements_manage_associate AD, Departments D, Courses_belongs C WHERE AD.DepartmentID = D.DepartmentID AND AD.CourseID = C.CourseID AND AD.DepartmentID = C.DepartmentID AND C.CourseID = 1 AND AppointmentTime > now() AT TIME ZONE \'EST\' AND D.DepartmentID = 3'
   ```

   We think this is interesting due to the following reasons. First, from the database perspective, since the 'Courses_belongs' is a weak entity set, a single course can only be identified with a specific department ID and a specific course ID and this is reflected in the above mentioned query. Second,

from the user's perspective, searching for sessions under specific courses is practical for students to find the right session to enroll in. Third, from the user's perspective, they know that, in order to use this functionality, they only need to select a course, however, our code needs to catch both department ID and course ID to locate a specific course.

2. **Web page 2 with the most interesting database operations (Rating)**

   In this page the user can rate a tutor he/she had a session with before. So, from a dropdown menu of all the tutors this specific user attended sessions with, he can proceed with any comments and the score he would like to give to that tutor. The student can also update previous ratings. This is useful to build up the ranking of the tutors, to help other students select the right tutors based on the rating and the comments that the previous students shared.

   The tables related to the functionality of this page are 'Rate_by ', 'Tutors', 'Orders_manage_link, and 'Adertisements_manage_associate'. The inputs of our database operations are score, comment, and tutor's username. If this user never rated this tutor before, our database operations will insert a new record into the database. If this user previously rated this tutor, our database operations will update the older rating to the newer one. Then our database operations will also update that tutor's average rating score and the number of people who rated this tutor correspondingly. An example of queries for this function are shown below which work as follows: considering the case when the user updates a rating, we will update the score and the comments of that rating in the 'Rate_by' table. Then, subsequently, will retrieve all records pertaining to that tutor from the 'Rate_by' table. Finally, we will calculate the average of the cumulative score of the tutor and count how many people rated this tutor and update his/ her score and the number of people who rated this tutor in the 'Tutors' table accordingly.

   ```
   'UPDATE Rate_by SET Score = 4, Comments = 'He focused on details and ask following questions to check our
   ```

```
understanding', Rate_time = now() AT TIME ZONE \'EST\' WHERE
Studentsusername = "daa2182" AND Tutorsusername = "hl2903"'
'SELECT Score FROM Rate_by WHERE Tutorsusername = "hl2903"'
Here we will use these scores and the number of records in
the result to calculate the average of the cumulative scores
(sum of score / number of records) (Since Score attribute is
an ENUM type, so we cannot use aggregation). Then, in the
below query we update this calculated score and counts of
people who rated this tutor.
'UPDATE Tutors SET Score = 4.5, NumberRate = 2 WHERE Username
= "hl2903"'
```

We think this is interesting due to the following reasons. First, from the database perspective, the database operations need to examine whether they should insert a new record into the "Rate_by" table or update the existing record. Second, these database operations also need to recompute and update the average score and the number of people who rated this tutor in the "Tutors" table. Third, from the user's perspective, this function is the essence of our whole system. It builds up the ranking scores of the tutors registered in the system. This helps students leverage the feedback shared by previous students when selecting future sessions.

Resources

Vergil, your personal course planning guide | Columbia University. (2021). Vergil@CU. https://vergil.registrar.columbia.edu/