IT329 Course Project - Phase 2 Description

Using the pages you created in the previous phase, we would like you to add the back-end functionality of your project using PHP + database. Back-end in this project involves creating and populating the database, adding functional log-in, log-out and sign-up functionalities (using Sessions) and the ability to:

Instructors: add new sections in the system and add their attendance records.
Students: view their attendance records and upload excuses for any absences.

In order to implement these functionalities, you need a database that contains tables for Instructors, students, attendances, courses and sections.

In order to fulfil these requirements, please develop the following:

1. **Create a database** following this schema (underlined is PK, dotted underlined is FK):

   Instructor (id, first_name, last_name, email_address, password)
   Student (KSUID, firstName, lastName)
   StudentAccount (id, KSUID, password)
   Course (id, symbol, name)
   Section (sectionNumber, courseID, type, hours, instructorID)
   SectionStudents (sectionNumber, studentKSUID)
   ClassAttendanceRecord (id, sectionNumber, date)
   StudentAttendanceInRecord (attendanceRecordID, studentKSUID, attendance)
   UploadedExcuses (id, studentAccountID, attendanceRecordID, absenceReason, uploadedExcuseFileName, decision)

   Where `decision` can be either "under consideration", "approved", or "disapproved", and `attendance` is a Boolean. Note that the data in the two tables Student and Course should be pulled from KSU's system, and thus they are not entered by users in this system. In this project, insert manually some sample records for these tables in the database.
   You can add more data if you have any extra needs in your application.

2. Add ten rows in Student table and at least three rows per other tables.
3. **Instructor log-in:** Add log-in functionality to the log-in page. When filling the log-in form, your form should call a PHP page that checks if email_address and password are correct. If yes, the user's id and role (i.e., instructor) are added as **session variables**; then the user is redirected to the instructor's home page. Otherwise, they are redirected back to the instructor's log-in page with an error message.
4. **Student log-in:** Add log-in functionality to the log-in page. When filling the log-in form, your form should call a PHP page that checks if KSU ID and password are correct. If yes, the user's id and role (i.e., student) are added as **session variables**; then the user is redirected to the student's home page. Otherwise, they are redirected back to the student's log-in page with an error message.

Please note: passwords are never stored as plain text into the database, hash passwords before storing them in the database using the PHP function password_hash.

5. Add **security** to the rest of your project pages such that all pages check that the user has logged-in, and if not, redirect the user to the home page. It also checks the role of the user, such that only instructors can view the instructor's homepage and only students can view the student's homepage. (Hint: use session variables.)
6. The **instructor's sign-up page** has a form that, when submitted, calls a PHP page that will add the new instructor to the database then redirect to the instructor page if they signed up correctly. If the instructor's email address is already in the database, redirect the user to the sign-up page with an appropriate message.

7. The **student's sign-up page** has a form that, when submitted, calls a PHP page that will check if the student exists (KSUID exists in "Student" table) then redirect to the student's page if they signed up correctly. If the KSUID is already in "StudentAccount" table, redirect the user to the sign-up page with an appropriate message.
8. **Instructor homepage** has PHP code that:
   a. Checks the instructor's id (from the session variable), retrieves the instructor's information and displays it in the page.
   b. Retrieves all sections that the instructor is teaching and displays them in a table that contains 6 columns: the section number, course, type (either lecture or lab), number of hours, attendance record, and button to delete the section.
      - The attendance is a code-generated link to the **instructor attendance record page** for the corresponding section.
      - The 'Delete' button calls a PHP page that will delete the corresponding section in the database along with its records in SectionStudents, ClassAttendanceRecord, StudentAttendanceInRecord, UploadedExcuses tables, and then redirects to the instructor's homepage.
   c. Displays an 'Add New Section' link that redirects to **Add new section page.**
   d. Retrieves all uploaded absence excuses that are under consideration by students in the instructor's sections and displays them in a table of 7 columns: section, student name, student ID, absence reason, uploaded excuse, date of absence, and 2 buttons for the instructor's final decision whether to approve or disapprove the student's absence excuse.
      - The excuse is a code-generated link to a new page that displays the uploaded excuse PDF file.
      - The 'Approve' and 'Disapprove' buttons in general call the same PHP page that will update the `decision` of the corresponding excuse in the database accordingly based on the clicked button. Also, if the decision is "approved", it will update the student's attendance and then redirects to the instructor's homepage. If the decision is "disapproved", then it redirects to the instructor's homepage without updating the student's attendance.
9. **Instructor attendance record page** has PHP code that:
   a. Checks the section's id that is sent in the query string, retrieves the section's information and displays it in the page. You also need to access course table in order to get the course code and name.
   b. Retrieves the dates of the previously recorded class and put them in a drop-down list. The drop-down list selects the last class by default. The page then displays the attendance for the last class. Also, the instructor can choses a date for a previously recorded class from the drop-down list, clicks the "Display" button to displays the attendance for this class. The page displays the attendances in a table with 3 columns: KSU ID, name, and attendance, for all the students in this section.
   c. Displays a form that allows the instructor to enter the attendance for a new class. In this form, the instructor selects a new date. The list of students in this section are retrieved and displayed in a table of 3 columns: KSU ID, name, and a drop-down list with two options: "attended" and "absent", where "attended" is the default value.
      - The 'Save' button calls a PHP page that will insert a new record in ClassAttendanceRecord for this class, and a record in StudentAttendanceInRecord for each student in the section.
10. The addition of a new section for an instructor is performed in a 2-step process with 2 PHP pages: "AddSection_Step1" and "AddSection_Step2". When the instructor clicks **"Add new section"** on the instructor's home page, the "AddSection_Step1" page will be requested using GET request method.

A) "AddSection_Step1" page has a PHP code that:

a. Displays a form that takes the section, the section type and number of hours. It retrieves all the courses code and name, and puts them in a drop-down menu in this form.
b. The 'Next' button will submit the form data to this same page using a POST request method.

c. If the request method for this page is POST, then it:
   - Retrieves the form input data. It checks if the section number already exists in the Section table, and in that case, it will stay in the same page with an appropriate message. Otherwise, it creates a new record for the section in the Section table, and redirects the user to the second step page "AddSection_Step2".

B) "AddSection_Step2" page has a PHP code that:

a. Displays a form that allows the instructor to add students in the section by entering the KSUID for students.
b. The 'Add' button calls a PHP page that adds the student in **`SectionStudents`** table, then redirects the user again to this page.
c. The 'Done' button redirects the instructor to the **instructor's homepage**.

11. **Student homepage** has a PHP code that:
   a. Checks the student's id (from the session variable), retrieves the student's information and displays it in the page.
   b. Retrieves all courses that the student is registered in, and displays it in a table of 3 columns: the course, attendance record for this course, and percentage of absence from start of semester to last recorded class (if there are no recorded classes, then it will display an appropriate message in this cell).
      - The attendance is a code-generated link to the **student attendance record page** for a course.
      - The calculation method for an absence percentage is shown in the following image, and it uses KSU's rule where each lecture hour is counted as 1 hour and a 2-hour lab is counted as 1 hour:

```
totalHours = 0
totalAbsentHours = 0

for each registered section in course:
        for each ClassAttendanceRecord for this section:
                if section.type = lecture
                        totalHours += section.hours
                else (section.type = lab)
                        totalHours += section.hours / 2
                get student attendance from StudentAttendanceInRecord
                if absent
                        if section.type = lecture
                                totalAbsentHours += section.hours
                        else (section.type = lab)
                                totalAbsentHours += section.hours / 2

percentage = totalAbsentHours / totalHours *100.
```

12. **Student attendance record page** has a PHP code that:
   a. For both lecture and lab attendance, it retrieves the attendance in this course for the student and displays it in a table of 3 columns: the date, the attendance status, and a link to **Upload excuse page** if the student was absent in this date.
      - The '**Upload excuse'** is a code-generated link to **Upload excuse page** with the id of **`ClassAttendanceRecord`** table (student id from session)
   b. Retrieves all the previous absence excuses that were submitted, and displays them in a table of 5 columns: class type, date of absence, reason of absence, uploaded excuse file, and the instructor's decision.
      - The excuse is a code-generated link to a new page that displays the uploaded excuse PDF file.

13. **Upload excuse** page has a PHP code that:
    a. Retrieves the course code, class type and date for this class, and displays the in the page.
    b. Allows the student to enter the absence reason and upload a PDF file for the document that proves the absence reason.
    c. The 'Send' button will call a PHP page that stores the excuse file in a directory on the server with a unique file name (combine section number, student KSU ID, and absence date in file name), creates a new record for the excuse in the database, and redirects the user to **Student homepage**.
14. A '**sign-out**' link for both instructor and student in all the different pages. The link is a PHP page that wipes out the session and then redirects the users to the home page.

You need to submit your NetBeans project folder after exporting it. It should include all your website pages (HTML, CSS, JS, images, and PHP) in a working hierarchy. In addition, you also need to submit the exported database. One member of each group should submit the phase files via LMS.

Submission deadline: Before 11:59 pm on Wednesday, November 1.