CSCI 4970/6970 Back-End Web Development CLASS PROJECT FALL 2020 (Total Points: 100)

Course Enrolment and Management System

Auburn University at Montgomery - Dept. of Comp. Sci., Montogomery, Alabama

Kelvin Gao
zgao1@aum.edu

Sep 20, 2020

Abstract

Department of Computer Science and Computer Information System (Dept. of Comp. Sci.) at Auburn University at Montgomery (AUM) is going to create their own web system for course enrolment and management. As a well-trained back-end web designer and developer, you want to take this challenge definitely.

Of course, you are not going to work from a sketch. You can reuse all the content from the existing Dept. of Comp. Sci. website, e.g., the courses list. Some extra materials will be provided in this document as well. Please follow the guideline, and we can complete this task together!

1 Objectives

In this project, students are supposed to learn:

1. How to design a dynamic website
2. How to use PHP + MySQL to fulfill the requirement of your design
3. How to use MVC pattern

2 Guideline and Steps

2.1 System Requirement

Figure 1 shows the overview of the system requirement.

This system is divided into two parts. You can choose to implement either part. There are three options:

1. An individual student completes the part for "Student."
2. An individual student completes the part for "Instructor."
3. A group of two students works together to complete the whole system. Due to the COVID-19, please consider using online communication tools like zoom, GitHub to work together.

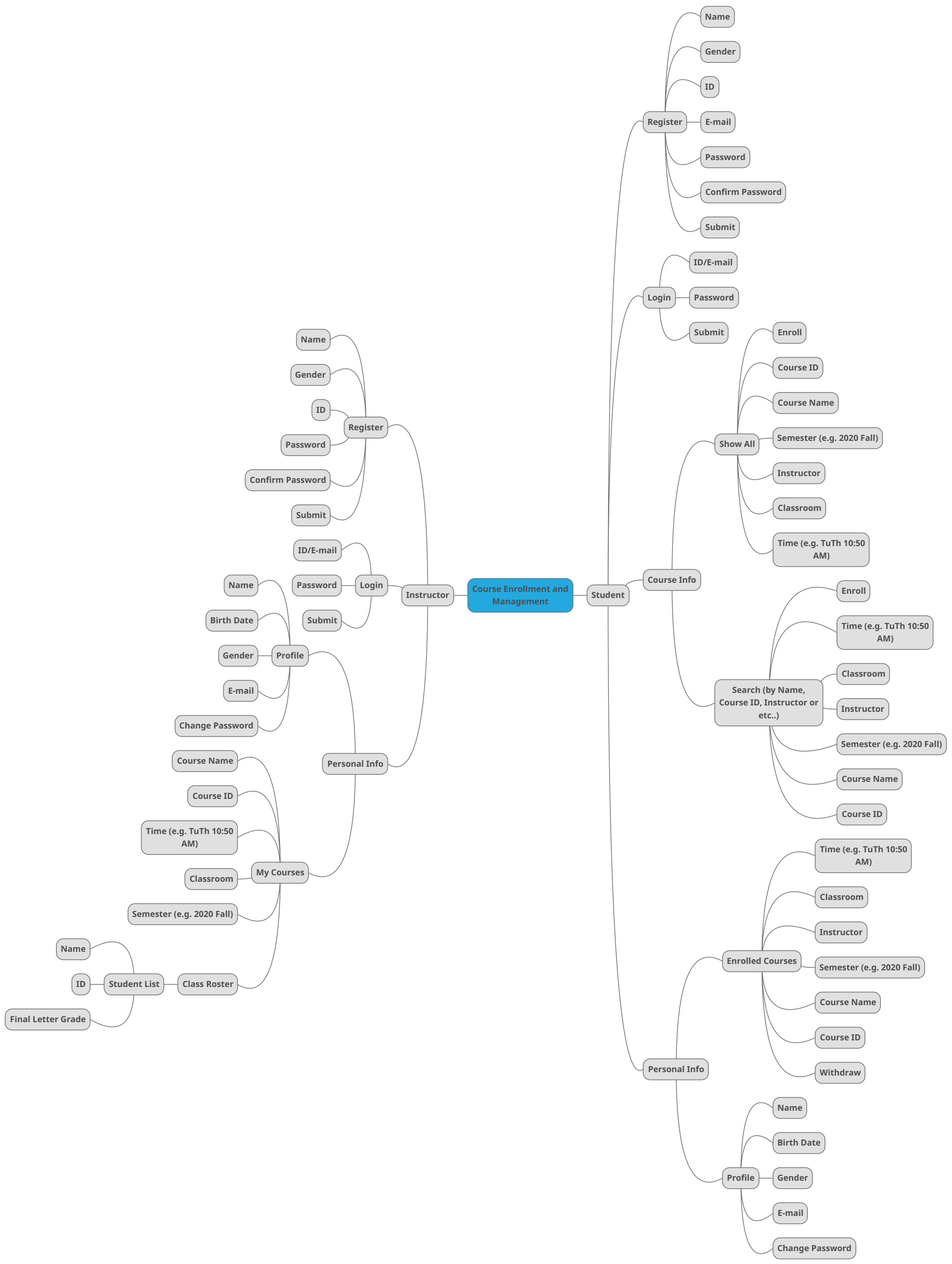

Figure 1: Overview of the System Requirement

Here is the detail of the system requirement.

2.1.1 Student

1. Register (20 Points)
   - You should create a PHP form page for a student to register an account.
   - The form data include *name, gender, id, e-mail, password (confirm password)*. *Confirm Password* is used to verify if the user enter the password correctly. You don't need to submit it.
   - Password can be sent with/without encryption.
   - You can verify the data using HTML (Front-end) or PHP (Back-end).
   - A submit button to submit the form.
2. Login (20 Points)
   - User can use ID or E-mail to log in.
   - User needs to enter a correct password.
   - Password can be sent with/without encryption.
   - Once account information is verify, create an session to store the information of current login; otherwise, show an error page.
3. Course Info (20 Points)
   - Show all available courses
     - a) Course information should include *course id, name, semester, instructor, classroom, time.*
     - b) A enroll button to submit the course that the student wants to enroll.
   - Search
     - a) Student can search courses by *course id, name, or instructor.*
     - b) A search button submit the query to the database.
     - c) The returned result should be organized like the section "Show all available courses". If no course returned, show the error message.
4. Personal Info (20 Points)
   - Enrolled Courses
     - a) The list of enrolled courses should be organized like the section "Show all available courses".
     - b) A "Withdraw" button to withdraw an enrolled course.
   - Profile
     - a) This page is to show the basic information of a student, including *name, birthday, gender, e-mail.* All information can be edit including password.
     - b) For changing the password, the student is required to enter the current password. And you need to verify it before updating it.

2.1.2 Instructor

1. Register (20 Points)
    - You should create a PHP form page for a student to register an account.
    - The form data include *name, gender, id, e-mail, password (confirm password). Confirm Password* is used to verify if the user enter the password correctly. You don't need to submit it.
    - Password can be sent with/without encryption.
    - You can verify the data using HTML (Front-end) or PHP (Back-end).
    - A submit button to submit the form.
2. Login (20 Points)
    - User can use ID or E-mail to log in.
    - User needs to enter a correct password.
    - Password can be sent with/without encryption.
    - Once account information is verify, create an session to store the information of current login; otherwise, show an error page.
3. Personal Info (40 Points)
    - My Courses (30 Points)
        a) The list of enrolled courses should be organized like the section "Show all available courses".
        b) A "Class Roster" button to access the student list of the selected course.
            - Use an table to show a list of students that are enrolled in the course.
            - The information includes *student's name, id and letter grade.*
            - Letter grade is editable. You can use a dropdown list to provide the option from "A" to "D" and "F". If you use text field, make sure that the user enter the correct letter grade.
    - Profile (10 Points)
        a) This page is to show the basic information of a student, including *name, birthday, gender, e-mail.* All information can be edit including password.
        b) For changing the password, the student is required to enter the current password. And you need to verify it before updating it.

2.2 Database Design (20 Points)

You should design a database for the proposed system.

- Create all required tables with all necessary columns/attributes.
- Create some test data in order to test your system.
- Bonus (10 points) - Create SQL stored functions or procedures for some frequent transactions, e.g. login. And you should use these functions/procedures in your PHP codes.

2.2.1 Submission

Put all tables with their test data in a word document (.docx file) and submit it to the blackboard.

2.3 Website Implementation (80 Points)

In this phase, you will implement the website with all skills that you learn in this course. Here are the requirements. Note that you don't need to fulfill the all requirements in a single page. As long as some pages satisfies the requirements then it's fine.

2.3.1 Required Skills (Missing any of the following will result in a deduction of 5 points.

1. PHP and MySQL
2. HTML form processing
3. Control statements including if-else and loops.
4. String processing, e.g., splitting, concatenating, etc.
5. Arrays.
6. Sessions.
7. Custom functions.
8. Objects.

2.4 Optional Skills

1. HTML 5
2. CSS 3 (including Bootstrap, etc.)
3. Javascript (including jQuery, Vue, Angular, etc.)

2.5 Bonus Skills

1. Use MVC pattern. (20 points)
2. Use cloud services like Beanstalk and MySQL cloud to host your website. EC2 doesn't count. (10 points)

2.5.1 Submission

Depending on the option that you choose from Section 1, you need to submit one of the follows.

- Individual Report
    1. Host your website somewhere (GitHub Page, S3, etc.) and put the URL to your website in the report.
    2. Indicate the php page that utilize the skills in Section 2.3.
- Group Report - You need to submit all listed in a "Individual Report" and a table showing the contribution of every group member.

3 Grading

1. Database Design (20 Points + Bonus 10 Points). See Section 2.2.
2. System Design (80 Points). See Section 2.1.
3. Skills - You may be deducted some points if you don't use some of the required skills. You may be added some points if you use some of the bonus skills.
4. Full Grade without bonus: 100.
5. Full Grade with bonus: 140.

4 Useful Materials

`http://www.aum.edu/computer-science-0`