Project Description:

Class Registration Application

Description
The Registration application will allow students to register for classes. Students and Classes will be added to the database by a registrar. Students would then be able to select classes and add them to their schedule.

This sample application is not expected to enforce real world constraints such as:
1. limiting the number of students in a class
2. limiting the number of credits per student.
3. and many others!

Operating Environment
1. The application shall work with Firefox 2.0.
2. The application shall operate on a linux server with rails 2.0.

Design Constraints
1. The application shall output valid XHTML 1.0 transitional.
2. The application shall store data in a MySQL 5.0 database.

Security Requirements
1. Users must login to the application before performing any operations, except viewing a list of classes.
2. The system shall permit only users logged in as a “registrar” to perform registrar tasks.
3. A student may not perform any tasks for another student, including viewing another student’s schedule.

Use Cases

Registrar

Add Student
Normal Flow: Registrar adds a student
1. The registrar navigates to the “Add Student” page.
2. The system displays a student form.
3. The registrar fills out the form and clicks a submit button.
4. The system confirms the addition.

Alternative: Registrar cancels after the for is displayed
1. Registrar clicks a Cancel button.
2. System ignores the form and returns to previous screen.

Exception: attempt to add another student with same id
1. System provides error message and rejects the addition.
2. The add student form is redisplayed.

**Search for Student**
The registrar shall be able to search for a student by id or by name.

**Edit Student**
Identical to Add Student, except that the form is initially filled with current data.

**Delete Student**
Normal Flow:
1. Registrar clicks a button (or a link) to delete a student.
2. System asks for confirmation.
3. Registrar confirms.
4. System deletes student from database, as well as all registrations for the student.
5. System returns to previous screen.

Alternate Flow: Registrar does not confirm
1. System does not delete, and returns to previous screen.

**Add Class**
*Similar to add student*

**Search for Class**
The registrar shall be able to search for a class by course number (e.g. CSC4800).

**Edit Class**
Identical to Add Class, except that the form is initially filled with current data.

**Delete Class**
*Here is an example of a design decision that is governed by business rules: should the registrar be able to delete classes for which students have registered? If so, how should that be handled? In real life, there is no way for the programmer to correctly design the system without interviewing “stakeholders.”*

Normal Flow:
1. Registrar clicks a button (or a link) to delete a class.
2. System asks for confirmation.
3. Registrar confirms.
4. System sends email to all students registered for the class.
5. System deletes class from database, as well as all registrations for the class.
6. System returns to previous screen.
Alternate Flow: Registrar does not confirm  
2. System does not delete, and returns to previous screen.

**View Registrations for a student**  
1. Registrar searches for student by id number or name.  
2. Registrar clicks button (or link) to view registration for the selected student.  
3. System displays the registration.

**View Registrations for a class**  
1. Registrar searches for class by course number.  
2. Registrar clicks button (or link) to view registration for the selected class.  
3. System displays the registration.

**Add/Drop registration for a student**

**Student**

**View Class Descriptions**

**Register for Class**  
Normal Flow:  
1. Student navigates to registration page.  
2. Student indicates which class to add.  
3. Student clicks “Register” button.

Alternate Flow: Student cancels registration process while in registration page  
1. Student clicks “Cancel” button.  
2. System does not register student for any classes.  
3. System returns to previous page.

Exception: Student attempts to register for a class twice  
1. System does not accept the second attempt, but instead prints error message and returns to previous screen.

Exception: Student attempts to register for a class that is full.  
*Not implemented*

Exception: Student attempts to register for a class without the prerequisites.  
*Not implemented*

**Drop Class**  
Normal Flow:  
1. Student clicks a button (or a link) to drop a class.  
2. System asks for confirmation.
3. Student confirms.
4. System deletes registration from database.
5. System returns to previous screen.

Alternate Flow: Student does not confirm
1. System does not drop anything, and returns to previous screen.

Design

**Database ER Diagram**

<table>
<thead>
<tr>
<th>Student</th>
<th>Registration</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>PK</td>
<td>PK, FK1, FK2</td>
<td>PK</td>
</tr>
<tr>
<td>id</td>
<td>student_id</td>
<td>id</td>
</tr>
<tr>
<td>lastname</td>
<td>course_id</td>
<td></td>
</tr>
<tr>
<td>firstname</td>
<td></td>
<td>title</td>
</tr>
<tr>
<td>email</td>
<td></td>
<td>description</td>
</tr>
</tbody>
</table>

**Screens**

- Main Register Page
- Registrar: Student List Page
  - search results
  - search form
  - new
- Registrar: Add/Edit Student
  - Add Student
    - student form
    - ok
    - cancel
- Registrar: Individual Student Page
  - ID Lastname, FirstName
    - student information
    - course list for student
    - Add Course: course id
    - ok