Cody Dosch

Michael Koscho

CS 490

3 December 2020

Student and Course Management Portal

**Abstract:**

My project is to build a system that universities or high schools can use to manage their students, courses, and registration processes, as well as provide an interface that students can use to view their coursework. My vision combines aspects of applications such as MarywoodYou and Moodle, and integrates them into one shared environment. The design process for the application will include making a user-friendly front-end and building a robust and secure back-end, which will allow me to demonstrate the skills I’ve acquired across the courses in both of my majors.

**Details:**

The project will be a CRUD database application utilizing the ASP.NET MVC architecture. In this way, it will consist of a database, front-end CSHTML pages that will act as views, and back-end controllers and models written in C#. MVC and CRUD applications are something I am familiar with through my internship experience, as I have worked for several months as one of two developers on an internal business application for a client using the technology. During this time, I have come to learn some of the best practices in the architecture, and in software development in general. I would like to take that experience and apply it to my own project, building something new and useful from the ground up.

My vision for the project begins with authentication. A system designed for this use will need a secure login system, with distinctions between student and teacher accounts, as well as overarching admin accounts for any miscellaneous application maintenance. Learning about secure authentication in both computer science and information security courses I’ve taken here, I would like to implement salting and hashing to the login system, so that only hashes of salted passwords are stored in the database. I think this provides a reasonable amount of security for a CRUD application, and am eager to look into its implementation.

Moving forward in the application, there are several main features that are crucial to such an application. First, the ability to add data to the application, the Create portion of CRUD. Pages must exist to add new students and teachers to the database, as well as new courses, and further along, course assignments. Following logically from that, we want to display that data for use by students and teachers alike, the Read portion of CRUD. These pages will range from a course catalog page, student-specific schedule pages, and course-specific information/assignment pages. In addition to simply adding data, users must have the ability to edit the data that is already available, such as updating student information, course meeting times, or course assignments. Pages will exist for this where applicable, and functionality will be implemented into relevant pages for ease of access, and this will make up the Update portion of CRUD. Finally, a system like this must be able to handle the removal of data, such as dropping courses, removal of accounts in cases such as termination of faculty, and removal of course sections. This functionality will also be implemented in various areas across the application, and makes up the final portion of CRUD, Delete.

Another main area I would like to focus on in the application is general security and error handling. With the potential access to sensitive data in a student management system, it is important that there are no vulnerabilities left unaddressed. Aspects of the project that will be assessed will include input validation throughout the application to ensure protection against threats such as SQL injection, as well as making sure that in the case of an error, the application is able to handle it gracefully instead of simply breaking. Special consideration will also be made within the project itself, utilizing best practices such as the use of resource files where applicable for repeatedly used values, and using stored procedures to interact with the database. To ensure that the application is as secure as possible, I will be continuously testing it throughout the development cycle.

Lastly, it is important to note that this application is being designed for end-users. With this in mind, the user experience is something that must also be considered throughout the design process. Not only must the pages in the application be presented in a way that is aesthetically pleasing, but they must also present the data in a logical way, that makes sense to all users. This means that navigation options should be apparent throughout the application, and the overall flow of the processes available should be predictable. Since user experience is a subjective aspect of design, I will bring in others to test my application in order to more closely mimic end-user testing.

I chose this project because I have always enjoyed thinking about the underlying architecture in the software I use. Having learned about MVC and CRUD applications at my internship, I began thinking about the variety of applications that can be built on the architecture, and thought that a system such as this that I have used consistently for the past three years would be a great project. It is an application that I personally know the value of, and knowing the various use cases of student management software will allow me to design the application in a way that I feel best suits the users. Furthermore, I think that the MVC pattern is an excellent fit for the project, as it is very flexible, and knowing it in depth enough to be able to create whole projects will be a valuable skill to have moving forward.

One of my main goals in completing the project is to have a firmer understanding of the entire software development cycle. Though I have experience working in an Agile development environment, all of the projects I have worked on so far are ones that I have joined rather than started. This project will allow me to get hands-on experience with the early stages of the development cycle that I have missed, from conception to design. Another goal of mine for the project is to explore some areas of development that I have learned about conceptually, but have never had the opportunity to implement. One such example of this is secure authentication, using the pattern of hashing and salting discussed above. Being both a computer science major and an information security major, I enjoy topics such as this in which the two overlap, allowing me to use knowledge from both aspects of my major. Exploring these areas is an exciting part of the project for me, as it will allow me to consider where security lies in the timeline of the development process, and in the different layers of the application itself.

The depth of this project will give me an opportunity to display the skills I have learned across the courses I have taken, and further expand on them as I step through the software development cycle.

**Changes from Proposal:**

Most of the goals I set in the original proposal above have been fulfilled, however in the interest of time and the having the most polished application, some features were not included in the final project.

First, the course assignments functionality was not implemented. This feature relied on the existence of all of the current functionality in the application, so it was last in order of priority. Though I did not have time to implement the actual functionality for it, I did include an assignments table on the Course Home pages for students and teachers, as well as an “Add Assigment” button on the teacher side. This will give an idea of where the functionality would have existed, and will allow me to add it in the future with ease.

The second part of the proposal that did not make it into the application is the removal of data. Though it would be necessary to delete users and courses in actual use of such an application, this fell lower on the priority list, as adding and editing data comes first. The functionality would exist for admin users only, and can be implemented at a later date for practice.

Other than these features, everything else in the proposal has been included in the final project. I also spent more time and put special considerations into the security of the application, adding more measures than described in the original proposal. In addition to the protections described in the proposal, I worked on limiting access to pages to only authenticated users of specified roles. This ensures that users cannot navigate to areas of the application they should not have access to.

All of the application’s functionality is outlined in the use cases below.

**Use Cases:**

1.1 Add User

|  |  |
| --- | --- |
| **Use case name** | Add User |
| **Actors** | An existing admin user |
| **Use case overview** | An admin user has the ability to add users of any role to the database through a form in the application. This creates an account for the new user and allows them to login. |
| **Preconditions** | The user must be logged in to an admin account to initiate this use case. |
| **Typical Flow of Events** | The user clicks on the “Add User” link and is brought to a form containing fields for new user entry.   1. The user enters all required fields. 2. The user clicks the “Add” button. 3. The input fields are sent to the Users Controller for validation. 4. A record for the new user is added to the Users table in the database. 5. The user is brought to a success page. |
| **Termination outcome** | The use case terminates upon successful addition of a user. |
| **Alternative Flow 1** | 1. The user does not enter all required information, or enters invalid data for any field. 2. The user clicks the “Add” button. 3. Data validation in the Users Controller fails. 4. The user remains on the page, and an error is displayed noting what must be fixed. |
| **Alternative Flow 2** | If the user does not have the role “admin” and tries to get to the page manually, they will be brought to an Access Denied page. |
| **Use case associations** | 1.2 Login |
| **Input summary** | Username  Password  Role  Name (Optional)  Address (Optional)  City (Optional)  State (Optional)  Zip (Optional) |
| **Output summary** | Success page  Error message  Access Denied page |
| **Use case notes** | Passwords are to be salted and hashed before storing. The salt must also be stored in the database.  Input validation can be done on the front-end through data annotations, but further checks must be implemented on the back-end.  Limit special characters to only those that may be necessary for all fields. |

1.2 Login

|  |  |
| --- | --- |
| **Use case name** | Login |
| **Actors** | A user with an existing account of any role |
| **Use case overview** | A user of any role enters their account credentials to login to the application. |
| **Preconditions** | The user’s account must have been added to the database by an admin user. |
| **Typical Flow of Events** | The user enters the application and is sent to the Login page.   1. The user enters a username and password. 2. The user clicks the “Login” button. 3. The input credentials are sent to the Login Controller for validation. 4. The user is logged into the application and is sent to the Home page. |
| **Termination outcome** | The use case terminates upon successful login of a user. |
| **Alternative Flow 1** | 1. The user does not enter a credential, or enters invalid credentials. 2. The user clicks the “Login” button. 3. Data validation in the Login Controller fails. 4. The user remains on the page, and an error is displayed noting what must be fixed. |
| **Use case associations** | 1.1 Add User  1.12 View Home Page |
| **Input summary** | Username  Password |
| **Output summary** | User is brought to Home page  Error message |
| **Use case notes** | The error message presented to the user should be the same regardless of what was entered incorrectly to better protect user credentials. |

1.3 Edit User

|  |  |
| --- | --- |
| **Use case name** | Edit User |
| **Actors** | A user with an existing account of any role |
| **Use case overview** | A user of any role updates their account information through a form in the application. |
| **Preconditions** | The user must be logged into the application with a valid account. |
| **Typical Flow of Events** | The user clicks the “Your Account” link and is brought to the “Account Details” page. The user clicks the “Edit” button.   1. The user enters data for the fields on the form. 2. The user clicks the “Save” button. 3. The data is sent to the Users Controller for validation. 4. The user’s record is updated in the database with the input data. 5. The user is brought to a success page. |
| **Termination outcome** | The use case terminates upon successful update of a user record. |
| **Alternative Flow 1** | 1. The user enters invalid data for any field on the form. 2. The user clicks the “Save” button. 3. Data validation in the Users Controller fails. 4. The user remains on the page, and an error is displayed noting what must be fixed. |
| **Alternative Flow 2** | The user can also reach the Edit Page by clicking on the Edit button in the Your Info section. |
| **Use case associations** | 1.1 Add User  1.2 Login  1.13 View Profile |
| **Input summary** | Name  Address  City  State  Zip |
| **Output summary** | Success Page  Error message |
| **Use case notes** | No fields are required; however, updates to null values should be allowed and handled properly.  Limit special characters to only those that may be necessary for all fields. |

1.4 Add Course

|  |  |
| --- | --- |
| **Use case name** | Add Course |
| **Actors** | An existing admin user |
| **Use case overview** | An admin user has the ability to add courses to the database through a form in the application. |
| **Preconditions** | The user must be logged in to an admin account to initiate this use case. |
| **Typical Flow of Events** | The user clicks on the “Add Course” link and is brought to a form containing fields for new course entry.   1. The user enters all required fields. 2. The user clicks the “Add” button. 3. The input fields are sent to the Courses Controller for validation. 4. A record for the new course is added to the Courses table in the database. 5. The user is brought to a success page. |
| **Termination outcome** | The use case terminates upon successful addition of a course. |
| **Alternative Flow 1** | 1. The user does not enter all required information, or enters invalid data for any field. 2. The user clicks the “Add” button. 3. Data validation in the Courses Controller fails. 4. The user remains on the page, and an error is displayed noting what must be fixed. |
| **Alternative Flow 2** | 1. The user attempts to add a course with a Department and Course Number that already exists. 2. The user remains on the page, and an error is displayed noting the conflict. |
| **Alternative Flow 3** | If the user does not have the role “admin” and tries to get to the page manually, they will be brought to an Access Denied page. |
| **Use case associations** | 1.2 Login  1.5 Edit Course |
| **Input summary** | Department  Course Number  Course Name  Semester  Seats Available  Instructor (Optional)  Course Description (Optional)  Start Time (Optional)  End Time (Optional)  Day of Week (Optional) |
| **Output summary** | Success page  Error message  Access Denied Page |
| **Use case notes** | Instructor, Semester, and Department should be dropdown lists rather than textboxes, as they will be foreign keys.  Input validation can be done on the front-end through data annotations, but further checks must be implemented on the back-end.  Limit special characters to only those that may be necessary for all fields. |

1.5 Edit Course

|  |  |
| --- | --- |
| **Use case name** | Edit Course |
| **Actors** | An existing admin user |
| **Use case overview** | An admin user has the ability to edit courses in the database through a form in the application. |
| **Preconditions** | The user must be logged in to an admin account to initiate this use case. There also must be existing courses in the database. |
| **Typical Flow of Events** | The user navigates to the Course Catalogue, clicks the “Edit” button on the row of a course and is brought to a form containing fields for updating a course.   1. The user enters updated data into fields. 2. The user clicks the “Edit” button. 3. The input fields are sent to the Courses Controller for validation. 4. The record for the course is updated in the Courses table in the database. 5. The user is brought to a success page. |
| **Termination outcome** | The use case terminates upon successful update of a course. |
| **Alternative Flow 1** | 1. The user does not enter all required information, or enters invalid data for any field. 2. The user clicks the “Edit” button. 3. Data validation in the Courses Controller fails. 4. The user remains on the page, and an error is displayed noting what must be fixed. |
| **Alternative Flow 2** | The user can also access the Edit page by navigating to the Course Details page and clicking the Edit button. |
| **Alternative Flow 3** | If the user does not have the role “admin” and tries to get to the page manually, they will be brought to an Access Denied page. |
| **Use case associations** | 1.2 Login  1.4 Add Course  1.6 View Course Catalogue |
| **Input summary** | Course Name  Semester  Seats Available  Instructor (Optional)  Course Description (Optional)  Start Time (Optional)  End Time (Optional)  Day of Week (Optional) |
| **Output summary** | Success page  Error message  Access Denied page |
| **Use case notes** | Instructor, Semester, and Department should be dropdown lists rather than textboxes, as they will be foreign keys.  Input validation can be done on the front-end through data annotations, but further checks must be implemented on the back-end.  Limit special characters to only those that may be necessary for all fields. |

1.6 View Course Catalogue

|  |  |
| --- | --- |
| **Use case name** | View Course Catalogue |
| **Actors** | An existing user of any role |
| **Use case overview** | Any user has the ability to view and filter the Course Catalogue. |
| **Preconditions** | The user must be logged in to any account to initiate this use case. |
| **Typical Flow of Events** | The user clicks on the “Course Catalogue” link and is brought to the Course Catalogue.  The user can select options for Department and Semester and click the “Filter” button to filter the results.  The user can click the “Remove Filter” button to view all courses. |
| **Termination outcome** | The use case terminates upon successful loading of the course catalogue. |
| **Use case associations** | 1.2 Login  1.5 Edit Course  1.7 View Course Details |
| **Input summary** | Department (On filter)  Semester (On filter) |
| **Output summary** | Course Catalogue Page |
| **Use case notes** | Department and Semester fields for filtering will be dropdowns, like on the Add and Edit Course pages. |

1.7 View Course Details

|  |  |
| --- | --- |
| **Use case name** | View Course Details |
| **Actors** | An existing user of any role |
| **Use case overview** | Any user has the ability to view the details of a course in the Course Catalogue. |
| **Preconditions** | The user must be logged in to any account to initiate this use case. |
| **Typical Flow of Events** | 1. The user clicks on the “Course Catalogue” link and is brought to the Course Catalogue. 2. The user clicks on the “View Details” button on the row of any course. 3. The user is brought to the details page for that course. |
| **Termination outcome** | The use case terminates upon successful loading of the course details page. |
| **Use case associations** | 1.2 Login  1.5 Edit Course  1.6 View Course Catalogue  1.8 Register  1.11 View Course Home |
| **Input summary** | None |
| **Output summary** | Course Details Page |
| **Use case notes** | Course Details page will be a generic page that takes the Course ID of the row clicked. It will pull the course information from the database by that passed ID. |

1.8 Register for Course

|  |  |
| --- | --- |
| **Use case name** | Register for Course |
| **Actors** | An existing student user |
| **Use case overview** | A student user has the ability to register for courses in the database. |
| **Preconditions** | The user must be logged in to a student account to initiate this use case. There also must be existing courses in the database. |
| **Typical Flow of Events** | The user navigates to the Course Catalogue and clicks the “Register” button on the row of a course. The user is then brought to a success page, which directs them back to the Course Catalogue. |
| **Termination outcome** | The use case terminates upon successful registration for a course. |
| **Alternative Flow 1** | The student also has access to a Register button after clicking the View Course Details button from the Course Catalogue page. |
| **Use case associations** | 1.2 Login  1.6 View Course Catalogue  1.9 View My Courses  1.10 Drop Course  1.11 View Course Home |
| **Input summary** | Course ID and ID of current user will be passed to Register endpoint. |
| **Output summary** | Success page |
| **Use case notes** | The Register button will redirect to a Register endpoint with the ID of the course clicked on. The Register button should not be displayed on courses the student is already registered for. Instead, a Course Home button should be displayed. SeatsOpen will need to be decremented on the course registered for. |

1.9 View My Courses

|  |  |
| --- | --- |
| **Use case name** | View My Courses |
| **Actors** | An existing student or instructor user |
| **Use case overview** | Any student or instructor user has the ability to view the list of courses they are teaching or registered for in the My Courses page. |
| **Preconditions** | The user must be logged in to a student or instructor account to initiate this use case. |
| **Typical Flow of Events** | The user clicks on the “My Courses” link and is brought to the My Courses page. |
| **Termination outcome** | The use case terminates upon successful loading of the My Courses page. |
| **Use case associations** | 1.2 Login  1.8 Register for Course  1.10 Drop Course  1.11 View Course Home |
| **Input summary** | None |
| **Output summary** | My Courses Page |
| **Use case notes** | The My Courses page should display only the courses registered/teaching for the current semester. This value can be set in and read from the web config. |

1.10 Drop Course

|  |  |
| --- | --- |
| **Use case name** | Drop Course |
| **Actors** | An existing student user |
| **Use case overview** | A student user has the ability to drop courses they are registered for. |
| **Preconditions** | The user must be logged in to a student account and registered for a course to initiate this use case. |
| **Typical Flow of Events** | The user navigates to the My Courses page and clicks the “Drop” button on the row of a course. The user is then brought to a success page, which directs them back to the My Courses page. |
| **Termination outcome** | The use case terminates upon successful drop of a course. |
| **Use case associations** | 1.2 Login  1.8 Register for Course  1.9 View My Courses |
| **Input summary** | Course ID and ID of current user will be passed to Drop endpoint. |
| **Output summary** | Success page |
| **Use case notes** | The Drop button will redirect to a Drop endpoint with the ID of the course clicked on. The Drop button should only be displayed for student users who are registered for the course in context. SeatsOpen will need to be incremented on the course dropped. |

1.11 View Course Home

|  |  |
| --- | --- |
| **Use case name** | View Course Home |
| **Actors** | An existing student or instructor user |
| **Use case overview** | Any student or instructor user has the ability to view the Course Home page for a course they are registered for or are teaching. |
| **Preconditions** | The user must be logged in to any student or instructor account to initiate this use case. They also must be either registered for or teaching a course. |
| **Typical Flow of Events** | 1. The user clicks on the “Course Catalogue” link and is brought to the Course Catalogue. 2. The user clicks on the “Course Home” button on the row of a course and is brought to the Course Home page for that course. |
| **Termination outcome** | The use case terminates upon successful loading of the course home page. |
|  | 1. The user clicks on the “Course Catalogue” link and is brought to the Course Catalogue. 2. The user clicks on the “View Course Details” button on a course they are registered for. 3. The user clicks on the “View Course Home” button and is brought to the Course Home page for that course. |
| **Alternative Flow 2** | 1. The user clicks on the “My Courses” link and is brought to the My Courses page. 2. The user clicks on the “Course Home” link on the row of a course and is brought to the Course Home page for that course. |
| **Alternative Flow 3** | If the user is no registered for or teaching the course and attempts manual access, they are brought to an access denied page. |
| **Use case associations** | 1.2 Login  1.6 View Course Catalogue  1.7 View Course Details  1.8 Register for Course  1.9 View My Courses |
| **Input summary** | Course ID of the course in context is passed to the StudentHome or InstructorHome endpoint depending on the role of the current user. |
| **Output summary** | Course Home Page  Access Denied Page |
| **Use case notes** | Make sure the student/instructor is actually registered for or teaching the course by checking for records in StudentCourseAssoc or InstructorCourseAssoc respectively. If they are not, redirect to the “Access Denied” page. |

1.12 View Home Page

|  |  |
| --- | --- |
| **Use case name** | View Home Page |
| **Actors** | An existing user of any role |
| **Use case overview** | Any user has the ability to view the Home Page. |
| **Preconditions** | The user must be logged in to any account to initiate this use case. |
| **Typical Flow of Events** | The user logs in and is brought to the Home Page.  They are presented with overview data and links to pages that vary by the user’s role. |
| **Termination outcome** | The use case terminates upon successful loading of the home page. |
| **Use case associations** | 1.1 Add User  1.2 Login  1.4 Add Course  1.3 Edit User  1.6 View Course Catalogue  1.9 View My Courses |
| **Input summary** | None |
| **Output summary** | Home Page |
| **Use case notes** | Conditional checks on the current user’s role will be used to change displayed data and links. |

1.13 View Profile

|  |  |
| --- | --- |
| **Use case name** | View Profile |
| **Actors** | An existing user of any role |
| **Use case overview** | Any user has the ability to view their Profile. |
| **Preconditions** | The user must be logged in to any account to initiate this use case. |
| **Typical Flow of Events** | The user clicks on the “Your Account” Link and is brought to the Profile Page. |
| **Termination outcome** | The use case terminates upon successful loading of the Profile page. |
| **Use case associations** | 1.2 Login  1.3 Edit User |
| **Input summary** | None |
| **Output summary** | Profile Page |
| **Use case notes** | None |