SWENG 500

Learning Management System

Software Requirements Specification

Authors: William DiStefano, Kevin Scheib, David Singer, Dawn Viscuso

***Overview***

A learning management system can be defined as a computer-based application whose function is to manage training courses, track course progress, maintain a catalog of courses, and assess learning. The learning management system typically allows for creation of class rosters, provides a mechanism for student and course registration, and allows uploading of course content and course assessments. It allows for course content to be delivered over web-based interfaces and in some cases provides capability for real-time interaction among students and instructors.

***System Objectives***

The goal of this project is to provide an online learning platform from which both students and educators can interact. The online learning management system must be flexible and dynamic enough to support many different platforms and content types. The design must be flexible in a way such that it can support a user of the system that would be using a smart-phone, a desktop computer, a laptop computer, or a tablet computer. The system architecture must be able to support and manage many different types of content, to include video, audio, text, and animations.

Specifically, the system will provide for students to enroll in and take training modules, track course progress, set bookmarks, view course catalog, view course history and maintain their student roster. The system will also allow an instructor to perform a variety of functions to manage and assess student learning such as creating and uploading course content, enrolling students in courses and managing the course catalog.

Overall, the system design and architecture will be targeted to be as standardized and open as possible to accommodate a wide range of users. The final system will consist of a SQL database back-end and a dynamic web application front-end.

***System Functions***

The design of the system has been divided into five segments of focus: the course administrator interface, the instructor interface, the student interface, the database, and the backend business logic.

The course administrator interface will allow the course administrator to create a new course, edit or delete existing courses, and change the status of existing courses, i.e. Archive (past), Current (present), or Under Development (future). When new courses are created, the course administrator will specify the Course Number, Course Name, Prerequisites, Completion Requirements, and Assigned Instructor. By default all newly created courses are Under Development in order for the assigned instructor to add content.

The instructor interface will allow the instructor to edit and create content for assigned courses for the e-learning system

Every component of a course can be edited except for course number which is used as a unique identifier. In the event that there is an issue with the course number, the course can be archived or deleted and a new course can be created.

The student interface will allow for students to self-register for approved courses, manage a course plan, take courses and course assessments, track course progress and view their learning history. Students will be able to login from a supported browser or mobile device to access the system. From there, the student will be able to perform a variety of functions to include searching a catalog of courses, adding courses to a student roster, removing courses from their roster, viewing course history, and launching courses. Students may also print course completion certificates.

The database will provide a primary data store for all data related to the system, including data related to lessons, courses, students, educators, etc. All sensitive data, such as passwords and student demographics, will be encrypted and secured before being stored in the database. The system itself will be very content driven and therefore the storage of binary data will be needed, for example: word documents, spreadsheets, videos, etc. These types of files will be stored on the application server, outside of the database. The database will instead store links to these files in the file system so they can be accessed via the web application front-end

The backend of the E-Learning system will accept requests from the web-based front end and perform the necessary business logic for each request. The main purpose of this portion of the application is to validate the inputs from the application, run the business logic for the given request, and interact with the database if needed.

***Functional Requirements***

**Course Administrator Interface**

The system will provide login and authentication for each user.

The system will ensure that only authorized users have access to the Course Administrator Interface.

The Course Administrator Interface will provide an option for creating a new course.

The Course Administrator interface will provide an option for changing the status of the course.

The Course Administrator Interface will provide an option for editing an existing course.

The Course Administrator Interface will provide an option for deleting an existing course.

**Instructor Interface**

The system will provide login and authentication for each user.

The system will ensure that only authorized users have access to the Instructor Interface.

The system will ensure that only an assigned instructor can edit the specific course.

The Instructor Interface will provide an option for editing an existing course.

The Instructor Interface will allow content files of various formats to be uploaded to new or existing courses.

The Instructor Interface will allow content to be created in a text editor.

The Instructor Interface will allow content files to be deleted from an existing course.

**Student Interface**

General:

System shall provide home, help, and exit buttons that are available on all screens.

System shall provide a course navigation bar visible on all screens.

System shall identify loading progress by displaying progress bars or hourglass images.

System shall provide access to a Help library with instructions on how to use the application.

System shall track pass/fail status and course grades for display in course roster history.

System shall track a student’s last completed lesson upon exiting course.

System shall be able to check that course prerequisites have been met before allowing course launch.

System shall provide course home/summary page that contains lesson progress information, lesson module name/title, and course completion percentage.

User Functions:

System shall provide ability for user to add a self-paced course to a learning plan.

System shall allow user to delete courses that were self-registered by the user.

System shall provide ability to view course history.

System shall allow user to take/launch a course.

System shall allow user to exit a course.

System shall allow user to print a completion certificate.

System shall allow user to view course details (course length, objectives, prerequisites).

System shall allow user to search for a course from the course catalog.

System shall allow for user to jump to different lessons in a course at any time from the navigation links on course home.

Security:

System will authenticate users before allowing access to system.

System should lock account after 3 failed login attempts.

Course Player:

System shall allow for use of audio and closed captioning.

System shall provide navigation buttons (prev, next) to advance screens.

System shall allow user to pause course.

System shall provide capability to view course player in full screen.

System shall allow user to replay lesson for the current screen.

Testing:

System shall allow for pre-assessments to test out of course.

System shall provide capability for mid-course quizzes and test exercises.

System shall allow for end-of-course quizzes and tests.

System shall allow for end-of-course tests with multiple choice and open-ended questions.

System shall allow for unlimited and immediate test retakes after imperfect score.

System shall provide feedback after test completion that includes questions answered incorrectly and individual section/module scores.

System shall provide capability for timed testing.

**Database**

The system database shall provide the ability to store student demographics information.

The system database shall provide the ability to store secured user access credentials.

The system database shall provide the ability to store course-related information.

The system database shall provide the ability to store lesson-related information.

The system database shall provide the ability to store private messaging data.

The system database shall provide the ability to store file and file location information for files stored on the application server.

The system database shall provide the ability to store student grades.

**Backend**

The system shall authenticate user logins.

The system shall provide error messages for invalid requests.

The system shall save users to the database.

The system shall retrieve users from the database.

The system shall save course information to the database.

The system shall retrieve course information from the database.

The system shall save course content to the database.

The system shall retrieve course content from the database.

The system shall associate course prerequisites.

The system shall enroll students into courses.

The system shall retrieve course rosters from the database.

The system shall enforce prerequisites when register students to a course.

The system shall delete courses from the database.

***Non Functional Requirements***

**Student Interface**

System shall be capable of running on Internet Explorer and Mozilla browsers.

System design shall be consistent in use of font, icons, buttons, icon and button placement.

System shall be intuitive and simple to use without formal training on system. Help topics should be sufficient for any instruction. Help topics should be brief 1 page topic instructions.

System shall be simple to use. Knowledge of computer science, programming, or flash technologies should not be required to use system.

System shall be accessible from Mobile devices.

System must be capable of running on Linux operating system.

System shall support course content of audio, images, and video.

System shall progress from screen to screen without noticeable delay (less than 1 second).

System audio (or closed captioning) shall start immediately (within 3 seconds) after course page loads.

System shall follow AICC and IMS best practices for LMS design and delivery.

**Database**

Logical Requirement:

The database will be provided as an instance of the MySQL DBMS, which will run on the same server as the application. The database will store all data related to this system, except for binary data which will be stored on the application server file system.

Performance Requirements:

The database must be able to support all requests received from the front-end application.

Security Requirements:

The database must not be accessible to the outside network, and only accessible to the internal web application. In addition, all sensitive data must be encrypted before being stored (e.g. passwords, student demographics, etc.).

***Use Cases***

**Course Administrator Interface**

All of the following course administrator use cases presume that the user has been authenticated and is authorized to perform the task.

|  |  |
| --- | --- |
| **Use Case** | Create a new course |
| **Description** | The Course Administrator Interface will provide an option for creating a new course. |
| **Steps** | **Main Success Scenario:**   1. User selects option for creating a new course. 2. The system prompts and accepts input for Course Number. 3. The system prompts and accepts input for Course Name. 4. The system prompts and (optional) accepts input for prerequisites. 5. The system prompts and accepts input for completion requirements. 6. The system prompts and accepts input for assigned instructor. 7. The system saves all course data. 8. The system publishes course in course catalog with an under development status.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User recreates the affected course. |
| **Assumptions** | None. |

|  |  |
| --- | --- |
| **Use Case** | Change the status of a course |
| **Description** | The Course Administrator Interface will provide an option for changing the status of a course. |
| **Steps** | **Main Success Scenario:**   1. User selects option for changing the status of a course. 2. The system displays all courses in the course catalog listed by status: Archived, Current, and Under Development. 3. The user selects the appropriate Course Number. 4. The system prompts and accepts input for new status. 5. The system saves all course data. 6. The system publishes status change in course catalog.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case.   4a. If the new status is “Current” and there already exists a current course with the same Course Number:   1. System prints error message that there cannot be more than one current course with identical Course Number. 2. User must change Course Number or cancel. |
| **Assumptions** | Course exists. |

|  |  |
| --- | --- |
| **Use Case** | Edit a course |
| **Description** | The Course Administrator Interface will provide an option for editing an existing course. |
| **Steps** | **Main Success Scenario:**   1. User selects option for editing an existing course. 2. The system displays all courses in the course catalog listed by status: Archived, Current, and Under Development. 3. The user selects the appropriate Course Number. 4. The system prompts and (optional) accepts update for Course Name. 5. The system prompts and (optional) accepts update for prerequisites. 6. The system prompts and (optional) accepts update for completion requirements. 7. The system prompts and (optional) accepts update for lesson order. 8. The system prompts and (optional) allows content files to be uploaded. 9. The system prompts and (optional) allows content to be created in text editor. 10. The system prompts and (optional) allows content files to be deleted. 11. The system saves all course data.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case.   8a. User tries to save file with duplicate filename:   1. System prints error message. 2. System prompts to change filename or overwrite existing file.   9a. User tries to save file with duplicate filename:   1. System prints error message. 2. System prompts to change filename or overwrite existing file. |
| **Assumptions** | Course exists. |

|  |  |
| --- | --- |
| **Use Case** | Delete a course |
| **Description** | The Course Administrator Interface will provide an option for deleting an existing course. |
| **Steps** | **Main Success Scenario:**   1. User selects option for deleting an existing course. 2. The system displays all courses in the course catalog listed by status: Archived, Current, and Under Development. 3. The user selects the appropriate Course Number. 4. The system prompts for deletion confirmation. 5. The system deletes all associated course data and files. 6. The system removes entry from course catalog.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case. |
| **Assumptions** | Course exists. |

**Instructor Interface**

All of the following instructor use cases presume that the user has been authenticated and is authorized to perform the task.

|  |  |
| --- | --- |
| **Use Case** | Edit a course |
| **Description** | The Instructor Interface will provide an option for editing an existing course. |
| **Steps** | **Main Success Scenario:**   1. User selects option for editing an existing course. 2. The system displays all courses in the course catalog listed by status: Archived, Current, and Under Development. 3. The user selects the appropriate Course Number. 4. The system prompts and (optional) accepts update for Course Name. 5. The system prompts and (optional) accepts update for prerequisites. 6. The system prompts and (optional) accepts update for completion requirements. 7. The system prompts and (optional) accepts update for lesson order. 8. The system prompts and (optional) allows content files to be uploaded. 9. The system prompts and (optional) allows content to be created in text editor. 10. The system prompts and (optional) allows content files to be deleted. 11. The system saves all course data.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case.   8a. User tries to save file with duplicate filename:   1. System prints error message. 2. System prompts to change filename or overwrite existing file.   9a. User tries to save file with duplicate filename:   1. System prints error message. 2. System prompts to change filename or overwrite existing file. |
| **Assumptions** | Course exists. |

|  |  |
| --- | --- |
| **Use Case** | Upload content file(s) |
| **Description** | The Instructor Interface will allow content files of various formats to be uploaded to new or existing courses. |
| **Steps** | **Main Success Scenario:**   1. User browses and selects file(s) to be uploaded. 2. The system saves file(s) in the storage area.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case.   2a. User tries to save file with duplicate filename:   1. System prints error message. 2. System prompts to change filename or overwrite existing file. |
| **Assumptions** | Course exists. |

|  |  |
| --- | --- |
| **Use Case** | Create content file(s) |
| **Description** | The Instructor Interface will allow content to be created in a text editor |
| **Steps** | **Main Success Scenario:**   1. User selects text editor function. 2. User creates content within the text editor. 3. The system saves file(s) in the storage area.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case.   3a. User tries to save file with duplicate filename:   1. System prints error message. 2. System prompts to change filename or overwrite existing file. |
| **Assumptions** | Course exists. |

|  |  |
| --- | --- |
| **Use Case** | Delete content file(s) |
| **Description** | The Instructor Interface will allow content files to be deleted from an existing course. |
| **Steps** | **Main Success Scenario:**   1. User browses and selects file(s) to be deleted. 2. The system prompts for deletion confirmation. 3. The system deletes the file(s) from the storage area.   **Extensions:**  \*a. At any time, System fails:   1. System prints error message. 2. User restarts use case. |
| **Assumptions** | Course exists. |

**Student Interface**

|  |  |
| --- | --- |
| **Use Case** | Login |
| **Description** | Student logs into the Learning Management System |
| **Steps** | 1. Student types username into Username field and presses enter 2. Student types password into Password field and presses enter 3. Student presses Login button 4. System authenticates user against database 5. System logs user into LMS   Alternate Flow:  4A1 User enters invalid password. System is unable to authenticate user   1. System notifies user of invalid username / password 2. Student re-enters username password 3. Continue from step 4   Exception Flow:  4A2 User enters invalid password. System unable to authenticate user after 3 attempts   1. System displays to user “account is locked. “ 2. System locks user account. Use case fails. |
| **Assumptions** | User has a valid username and password |
| **Post Conditions** | Successful authentication. User logged into LMS system. |

|  |  |
| --- | --- |
| **Use Case** | Take Course |
| **Description** | Student launches a course for the first time |
| **Steps** | 1. Student searches for course to launch (Include: Search Catalog) 2. Student clicks on link for desired course 3. System displays course details 4. Student clicks Launch button 5. System adds course to student roster 6. System launches course to Course Home page   Alternate Flow:  2A Student selects course from his/her student roster   1. Student clicks on Student Roster link from home page 2. Return to step 4   6A Student has previously accessed this course  1. System launches course to last completed page  Exception Flow:  6B Student has not passed required pre-requisite   1. System display error. “Pre-req course xyz is required” 2. Use case fails |
| **Assumptions** | User has successfully logged into LMS |
| **Post Condition** | Course opens successfully. Course content is loaded. |

|  |  |
| --- | --- |
| **Use Case** | Print Certificate |
| **Description** | Student prints a completion certificate after successfully passing a course |
| **Steps** | 1. From course home page student selects Print Certificate 2. System displays certificate in printable format 3. Student selects print button   Alternative Flow:  3A Course status is incomplete   1. System displays error “Course not completed!” |
| **Assumptions** | User has successfully logged into LMS. Student has passed course |
| **Post Condition** | Certificate displays for print. |

|  |  |
| --- | --- |
| **Use Case** | Create Bookmark |
| **Description** | Student marks current page to return to |
| **Steps** | 1. From current lesson page, student clicks Add Bookmark button 2. System prompts student for a bookmark name 3. Student enters name for the bookmark 4. System displays “bookmark saved” message   Exception Flow:  3A Bookmark name contain special characters   1. System displays error “Invalid format. Bookmark not saved.” 2. Use case fails |
| **Assumptions** | User has successfully launched a course |
| **Post Condition** | Bookmark is saved and a link to it is provided on Course Home page |

|  |  |
| --- | --- |
| **Use Case** | View Help Contents |
| **Description** | User clicks Help button and views help topics |
| **Steps** | 1. From any page, user clicks Help button 2. System opens Help document and displays contents 3. User selects Help topic link 4. Help information for the selected topic is displayed |
| **Assumptions** | User has successfully logged into LMS. |
| **Post Condition** | Help document opens and links take user to appropriate topics within document. |

|  |  |
| --- | --- |
| **Use Case** | Add Course |
| **Description** | Student adds a course to his/her student roster/learning plan |
| **Steps** | 1. Student searches for course (Include: Search Catalog) 2. Student clicks desired course 3. System displays course details 4. Student clicks “Add to Roster” link 5. System adds course to roster   Exception Flow:  5A Course requires pre-approval / registration by instructor   1. System displays error “Approval Required. See Instructor” 2. Use Case fails |
| **Assumptions** | User has successfully logged into LMS. System |
| **Post Condition** | Course added to student roster |

|  |  |
| --- | --- |
| **Use Case** | Delete Course |
| **Description** | Student deletes a course from his/her student roster/learning plan |
| **Steps** | 1. Student select Student Roster link from system home page 2. System display list of courses student is enrolled in 3. Student clicks Delete icon next to course to be deleted 4. System deletes course from roster   Exception Flow:  4A Course was registered by instructor and cannot be deleted by student   1. System displays error “Approval Required. See Instructor” 2. Use case fails |
| **Assumptions** | User has successfully logged into LMS. System |
| **Post Condition** | Course deleted from student roster list |

|  |  |
| --- | --- |
| **Use Case** | Exit Course |
| **Description** | Student exits a course |
| **Steps** | 1. Student clicks the Exit button from any screen 2. Students prompts user for confirmation to exit 3. Student clicks Yes to exit 4. Course exits and student is returned to system home page 5. Course status is updated to incomplete   Alternate Flow:  5A Course is exited after completion   1. Course status is updated to complete   Exception Flow:  4A User selects No to not exit course   1. Course does not exit 2. Use case ends |
| **Assumptions** | User has successfully launched a course |
| **Post Condition** | Course is exited and completion status updated appropriately |

|  |  |
| --- | --- |
| **Use Case** | Complete Lesson |
| **Description** | Student views/completes all pages of a course lesson |
| **Steps** | 1. From Course Home, student clicks on any lesson link 2. System launches first page of the selected lesson 3. Student uses “Next” navigation button to advance page 4. System advances page 5. Steps 3 and 4 repeated until last page reached 6. System updates lesson status to Complete 7. First page of next lesson in lesson order is displayed   Alternative Flow:  1A Student clicks Begin Course from course home   1. First lesson in the course is launched   1B Student accesses lesson from a Bookmark   1. Student clicks on selected Bookmark link from Course Home 2. System launches lesson at bookmarked page 3. Return to step 3   7A No other lessons exist after the current lesson   1. Student returned to Course Home page |
| **Assumptions** | User has successfully launched a course |
| **Post Condition** | Lesson status is marked Complete |

|  |  |
| --- | --- |
| **Use Case** | Take Course Test |
| **Description** | Student takes a course assessment to test out of course at any time |
| **Steps** | 1. Student launces course (Include: Take Course) 2. Student clicks Take Course Test link from course home page 3. System launches Test and displays a start test button 4. Student clicks Start Test button 5. System displays a test question 6. Student selects an answer 7. System returns to step 5 and repeat until last question 8. System ends test after last question answered 9. System displays test score and feedback 10. System offers option to retake course or return to course home 11. Student selects Return to Home 12. System displays course home page 13. System updates course status to complete   Alternate Flow:  8A Time runs out on a timed test   1. System ends test immediately after time expires   10A Student gets a perfect score   1. System returns to course home page. Use case ends.   11A Student selects Retake Test   1. Return to step 3   13A Student fails test   1. System updates course status to incomplete |
| **Assumptions** | User has successfully logged into LMS |
| **Post Condition** | Student receives test results. Course status updated (complete/incomplete) |

|  |  |
| --- | --- |
| **Use Case** | View Course Details |
| **Description** | Student checks course information before launching course |
| **Steps** | 1. Student searches for course of interest in catalog (Include: Search Catalog) 2. Student clicks on link for desired course 3. System displays course information (duration, objective etc.)   Alternate Flow:  1A Student selects course from his/her student roster   1. Student clicks on Student Roster link from home page 2. Return to step 2 |
| **Assumptions** | User has successfully logged into LMS |
| **Post Condition** | Course details display for selected course |

|  |  |
| --- | --- |
| **Use Case** | Search Catalog |
| **Description** | Student searches course catalog |
| **Steps** | 1. From Home page, student enter course name in Search box 2. Student clicks the search icon or presses enter 3. System searches catalog and displays link to course   Exception Flow:  3A Course not found   1. Search results displays “No records found” 2. Student returns to step 1 to perform another search |
| **Assumptions** | User has successfully logged into LMS |
| **Post Condition** | Link to access desired course is displayed |

**Backend**

|  |  |
| --- | --- |
| **Use Case** | Create User |
| **Description** | The backend will validate the user information before saving. |
| **Steps** | 1. The request comes into the system. 2. The information is pulled out of the request and validated for correctness. 3. The information is saved in the database. 4. A successful response is returned.   Alternate Flow:  2A. The information is pulled out of the request and is invalid.   1. Return a response that contains the invalid criteria.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Get User |
| **Description** | The backend will create a user object from the information in the database. |
| **Steps** | 1. The request comes into the system. 2. The information is retrieved from the database. 3. A user object is created from that information. 4. A user object is returned in the response.   Alternate Flow:  2A. The user does not exist.   1. A response is returned stating that the user does not exist.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Authenticate user |
| **Description** | The backend will validate the credentials and return a successful response. |
| **Steps** | 1. The request comes into the system. 2. The user information is validated against information in the database. 3. An success response is returned.   Alternate Flow:  2A. The username/password is incorrect.   1. Return a response that states that the user credentials are incorrect.   2B. The username/password is incorrect for a user after 3 tries.   1. Lock the user account, thus disabling them from logging in. 2. Return a response that states that user is locked.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Retrieve data |
| **Description** | This is a generic use case for retrieving any information from the database. The queries and objects that are returned are different for each type of data. |
| **Steps** | 1. The request comes into the system. 2. The information from the request is validated. 3. The data for the object creation is retrieved from the database. 4. The object(s) is created. 5. A success response is returned that contains the object(s).   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Associate course as a prerequisite |
| **Description** | The course that is being edited will have a prerequisite |
| **Steps** | 1. The request comes into the system. 2. The information from the request is validated. 3. The prerequisite course is added to the course that is being edited. 4. The course is saved. 5. A success response is returned.   Alternate Flow:  2A. The request is invalid   1. Return a response that contains the invalid criteria.   2B. The prerequisite course does not exist.   1. Return a response that states the prerequisite course does not exist.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Associate student to a course |
| **Description** | The student should be added to the course roster |
| **Steps** | 1. The request comes into the system. 2. The information from the request is validated. 3. The student is validated against the course prerequisites. 4. The updated course roster is updated in the database. 5. A success response is returned.   Alternate Flow:  2A. The request is invalid.   1. Return a response that contains the invalid criteria.   3A. The student does not meet prerequisites   1. Return a response that lets the user know that the student does not meet prerequisites.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Course is deleted |
| **Description** | The course will be deleted from the database |
| **Steps** | 1. The request comes into the system. 2. The information from the request is validated. 3. The user permissions are validated to ensure that they can delete the course. 4. The course is deleted from the database. 5. An success response is returned.   Alternate Flow:  2A. The request is invalid.   1. Return a response that contains invalid criteria.   3A. The user does not have access to delete a course.   1. Return an error response.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |

|  |  |
| --- | --- |
| **Use Case** | Course content is updated |
| **Description** | The course content will be saved to the database |
| **Steps** | 1. The request comes into the system. 2. The information from the request is validated. 3. The user permissions are validated to ensure that they can add content to the course. 4. The course content is saved to the database. 5. A success response is returned.   Alternate Flow:  2A. The request is invalid.   1. Return a response that contains the invalid criteria.   3A. The user does not have permission to edit a course.   1. Return a response that states the user does not have access.   Exception Flow:   1. Return an error response. |
| **Assumptions** | The request is valid |