CS5500 TEAM-6 : PLAGIARISM DETECTOR - USE CASES

Below are the actor descriptions for all the use cases to follow:

ACTOR	DESCRIPTION
Admin	The administrator of a system. Approves/rejects a course request by a professor.
Professor	The professor who teaches the course. A course should have only one professor.
Grader	Usually assistant professors and teaching assistants. A course can have zero or more graders.

Use Case 1:

Use Case:	Professors and graders registering on the portal.
Primary Actor:	Professor/Grader
Goal in Context:	To register/enroll on the plagiarism detection portal
Preconditions:	1. Must have a valid and unique email ID (should not be already present in the system)
Trigger:	The user decides to enroll himself for the plagiarism detection portal so that he can use it.
Scenario:	1. The user selects “Register as a new user” from the portal/website. 2. The user fills out the required details in the registration portal 3. The user enters a valid password (at least 8 characters in length) and verifies it. 4. User selects the respective “Role” (i.e. Professor or Grader) 5. The user then submits the form to complete the registration. 6. If successful, the user is taken back to the index (login) page. 7. If unsuccessful, the user is shown a popup message stating that he has entered some invalid information.
Exceptions:	1. Invalid email id format 2. Password entered does not comply with the password requirements 3. User with the same email address already exists in the system. 4. User does not select a role (“Professor” or “Grader”). 5. Form validation for all the fields(User submits an empty or incomplete form).
Priority:	Essential and must be implemented
When available:	First increment
Channel to actor:	Via the registration form

Use Case 2:

Use Case:	Professor requesting to add a new class/course under him
Primary Actor:	Professor
Goal in Context:	To request for a new course to be managed by him in this portal.
Preconditions:	1. Professor must have a valid account with the “Professor” role. 2. Professor should know the administrator for the course.
Trigger:	The Professor decides to add a course for plagiarism detection.
Scenario:	1. Professor logs onto the system using valid credentials. 2. The professor views the list of courses and selects “Request a new course”. 3. Professor fills out the course details and selects the correct administrator for approval. 4. The professor submits the request and waits for approval. 5. If unsuccessful, the professor is shown a pop up message describing the error occurred. 6. If successful, the professor is taken back to his home page. 7. Once approved by the admin, the professor can see the course listed under the “Class Name” section. (Refer to Use Case: Admin approving or rejecting a professor’s course request.)
Exceptions:	1. Professor submits invalid/incomplete course/form details. 2. Professor requests for a course which already exists.
Priority:	Essential and must be implemented
When available:	First increment
Channel to actor:	Via a course request form
Secondary Actor	Admin
Channel to secondary:	Admin pending requests view
Open Issues:	1. How to handle duplicate requests? 2. How to notify professors on approval or reject of their requests?

Use Case 3:

Use Case:	Approving or rejecting a professor(s) course request.
Primary Actor:	Admin
Goal in Context:	Approve or reject professor(s) course creation request.
Preconditions:	1. Admin must have a valid account with the “Admin” role. 2. There should be a pending request by the professor(s).
Trigger:	Admin wants to approve or reject pending course creation request so that the professor can check the submitted solutions for that course.
Scenario:	1. Admin logs onto the system using valid credentials. 2. The admin views the pending requests in the pending request tab. 3. The admin approves or rejects the request. 4. The Professor receives a notification informing him about the request status.
Exceptions:	1. Admin is unable to view the request submitted by the professor. 2. A notification is not sent to professor when admin approves or rejects a request. 3. A pending request once approve or rejected, isn’t added to the history of requests.
Secondary Actor	Professor
Priority:	Essential and must be implemented
When available:	Second increment
Channel to actor:	Via admin home page

Use Case 4:

Use Case:	Professor adding/removing graders for a course
Primary Actor:	Professor
Goal in Context:	Add or remove the graders for a particular course.
Preconditions:	1. The professor must have a valid account to log into the system. 2. The course should have been created for which the graders are being managed. 3. The grader must be registered in the system.
Trigger:	Professors decides to manage the graders for his course.
Scenario:	1. Professor logs onto the system using valid credentials. 2. Selects the “Class Information” section for the particular course. 3. Selects “Add Grader” in the grader’s section to add a grader. 4. Professor enters the grader’s details to add him. 5. If the grader is not registered in the system, a “Grader not found” popup message is shown. 6. If the grader is registered in the system, then the professor is taken back to the class information section of his home page with a notification confirming that the grader has been added. 7. For deleting a grader, the professor selects “Delete grader” next to the grader in the graders list. 8. A confirmation popup is shown to confirm deleting the grader. 9. Professor confirms and the grader is removed for the course.
Exceptions:	1. The professor enters incorrect information about the grader. 2. The professor tries to add a grader already added. 3. The professor tries to delete a grader already deleted.
Secondary Actor	Grader
Priority:	Essential and must be implemented
When available:	Second increment
Channel to actor:	Via course page for a particular course

Use Case 5:

Use Case:	Professor/Grader adding/removing students for their course.
Primary Actor:	Professor/Grader
Goal in Context:	Add or remove the students for a particular course.
Preconditions:	1. The user must have a valid account to log into the system with either the “Professor” or the “Grader” role. 2. The course should have been created for which the students are being managed.
Trigger:	The user decides to manage students for their course, so that they can upload solutions for the students and run plagiarism checks on them.
Scenario:	1. User logs onto the system using valid credentials. 2. Selects the “Class Information” section for the particular course. 3. Selects “Add Student” in the student section to add a student. 4. User enters the student details to add the student. 5. The user is taken back to the class information section of his home page with a notification confirming that the student has been added. 6. For deleting a student, the user selects “Delete Student” next to the student in the students list. 7. A confirmation popup is shown to confirm deleting the student. 8. User confirms and the student is removed for the course.
Exceptions:	1. Student already added for the course. 2. Student already deleted for the course.
Priority:	Essential and must be implemented
When available:	Second increment
Channel to actor:	Via course page for a particular course

Use Case 6:

Use Case:	Professor(s)/Grader(s) adding or removing assignments for a course.
Primary Actor:	Professor/Grader
Goal in Context:	User wants to manage the assignments for a course
Preconditions:	1. The user must have a valid account to log into the system with either the “Professor” or the “Grader” role. 2. The course should have been created for which the assignments are being managed.
Trigger:	The user wants to create an assignment to run the plagiarism check on student submissions.
Scenario:	1. User logs onto the system using valid credentials. 2. Selects the correct course for which the assignment is to be added. 3. Selects “Add Assignment”. 4. Enters the assignment details (like assignment name, due date, etc). 5. Selects “Save” to save the assignment information and create a new assignment for the course. 6. The user is taken back to his home screen and can see the newly added assignment.
Exceptions:	1. Incorrect or invalid assignment information does not allow the user to create the assignment. 2. The created assignment is already present in the system.
Priority:	Essential and must be implemented
When available:	Second increment
Channel to actor:	Via user’s home page for a particular course.

Use Case 7:

Use Case:	Professor(s)/Grader(s) uploading the student solutions directly as files for a particular assignment.
Primary Actor:	Professors/Graders
Goal in Context:	Upload students solutions/files to the system.
Preconditions:	1. The user must be logged into the system with the “Professor” or “Grader” role. 2. The course should have been created for which the solutions are being uploaded. 3. The assignment must be created for the course for which the solutions are being uploaded to. 4. The user must have the solutions to upload.
Trigger:	The user wants to runs plagiarism check by by uploading students’ solutions.
Scenario:	1. User logs into the system. 2. User navigates to particular course. 3. User navigates to particular assignment. 4. For each student in the course the user will upload the solutions by selecting “Add solution for student”. 5. The user is presented with a browser popup to select and upload the appropriate file(s). 6. The user selects “save submissions” to save the changes.
Exceptions:	1. User uploads an empty solution(file or folder). 2. User uploads a solution of a wrong file format (eg not a .py file). 3. User uploads a file greater than the size limit.
Priority:	Essential and must be implemented
When available:	Third increment
Channel to actor:	Upload solutions interface (form interface with uploading buttons)
Open Issues:	1. Should we autorun the plagiarism detection when the solutions are first uploaded? 2. Should there be another way to upload solutions? 3. How to handle multiple solution uploads simultaneously?

Use Case 8:

Use Case:	Professor(s)/Grader(s) running plagiarism check on the assignment.
Primary Actor:	Professor/Grader
Goal in Context:	Running plagiarism check against uploaded submissions.
Preconditions:	1. The user must have a valid account to login into the system. 2. The course should have been created for which the plagiarism check has to be run. 3. The assignment must be created for the course for which the plagiarism check has to be run.
Trigger:	User wants to determine plagiarism in the students’ submission.
Scenario:	1. User logs into the system using valid credentials. 2. Navigates to particular course. 3. Navigates to particular assignment. 4. Selects “Run Plagiarism Check” on the specific assignment. 5. The system then starts a plagiarism check on all of the student submissions for the assignment. 6. Once done, it generates results and displays it to the user.
Exceptions:	1. No students are enrolled in the course. 2. Less than two submissions are present in the assignment.
Priority:	Essential and must be implemented
When available:	Third increment
Channel to actor:	Via assignments page.

Use Case 9:

Use Case:	Professor(s)/Grader(s) view the plagiarism report.
Primary Actor:	Professor/Grader
Goal in Context:	View plagiarism results.
Preconditions:	1. The user must have a valid account to login into the system. 2. Must have a course created. 3. Assignment must be created in the course against which plagiarism should be run 4. Solution for each student in the course should have been uploaded. 5. Plagiarism check should have been run against uploaded solutions.
Trigger:	User wants to check result of the plagiarism.
Scenario:	1. User logs into the system using valid credentials. 2. Navigates to particular course. 3. Navigates to particular assignment. 4. Selects a plagiarism check for the assignment that has been already run. 5. Information about the selected plagiarism run will be displayed.
Exceptions:	1. Plagiarism check was not successful(ended due to an error). 2. Report no longer exists on the server.
Priority:	Essential and must be implemented
When available:	Third increment
Channel to actor:	Via Assignment page for a particular course