Use Cases Completed

User Management Use Cases:

Use Case 1:

Use Case:	**Create a new account**
Primary Actor:	User (Course Staff, Student)
Goal in Context:	To facilitate any user to sign up for a new account by providing username, first name, last name, email, new password and confirming the new password.
Preconditions:	System has been programmed for any user to create a new account providing credentials which can be used later for logging in again.
Trigger:	A user decides to create a new account.
Scenario:	1. User clicks on "Register" button 2. User then fills the following fields: ● Username ● First Name ● Last Name ● Email address (a valid one) ● Password ● Confirm Password (Same as above) 3. User then presses "Register" button.
Exceptions:	If user tries to submit the form with any of the below issues, new user won't be created: 1. Any of the fields are left unfilled. 2. Incorrect email address is entered. 3. The password typed twice do not match. Remedy: Avoiding the above issues would allow a user to create a new account successfully.

Use Case 2:

Use Case:	**Log into system**
Primary Actor:	User (Course Staff, Student)
Goal in Context:	To facilitate a registered user to log in to his account by providing valid username and password.
Preconditions:	System has been programmed for any registered user to login to his account by providing valid credentials.
Trigger:	A user decides to log into his account.
Scenario:	1. User clicks on "Login" button. 2. User then fills the following fields: ● Username (Registered one) ● Password (Valid one) 3. User then presses "Login" button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to login: 1. Any of the two fields are left unfilled 2. Unregistered username is entered 3. Invalid password is entered Remedy: Avoiding the above issues would allow a user to login to his account successfully.

Use Case 3:

Use Case:	**Update profile**
Primary Actor:	User (Course Staff, Student)
Goal in Context:	To facilitate a registered user to update his profile.
Preconditions:	System has been programmed for any registered user to update his profile after logging in.
Trigger:	A user decides to update his profile.
Scenario:	1. User clicks on “Profile” tab. 2. User then updates any of the following fields which he wishes to update: ● First name ● Last name ● Email 3. User then presses “Update” button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to update: 1. Any of the three fields are left unfilled 2. Invalid email is entered Remedy: Avoiding the above issues would allow a user to update his profile successfully.

Use Case 4:

Use Case:	**Change password**
Primary Actor:	User (Course Staff, Student)
Goal in Context:	To facilitate a registered user to update his password.
Preconditions:	System has been programmed for any registered user to update his password after logging in.
Trigger:	A user decides to update his password.
Scenario:	1. User clicks on “Profile” tab. 2. User then fills the following two fields: ● Password ● Valid Password (same as Password) 3. User then presses “Update” button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to update: 1. Any of the two fields are left unfilled. 2. Two passwords do not match. Remedy: Avoiding the above issues would allow a user to update his password successfully.

Homework Upload Use Cases

Use Case 5:

Use Case:	**Upload single python file by Choose file feature**
Primary Actor:	Student
Goal in Context:	To facilitate a student to submit his python file submission by choosing it from his local system.
Preconditions:	Student has the python file is his local system.
Trigger:	A user decides to submit the assignment.
Scenario:	1. Student clicks on "HomeWorks" tab. 2. Student then clicks "Choose File" button in the section "Choose and upload file". 3. Student then goes to the path where the file is present in local system and selects the python file. 4. Student clicks "Submit" button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to update: 1. Not selecting any file 2. File is not of python type Remedy: Avoiding the above issues would allow a student to upload and submit assignment successfully.

Use Case 6:

Use Case:	**Upload single python file by Github URL feature**
Primary Actor:	Student
Goal in Context:	To facilitate a student to submit his python file submission by entering the Github URL of the file to be uploaded and submitted.
Preconditions:	Student has Github URL of the python file to be submitted.
Trigger:	A user decides to submit the assignment.
Scenario:	1. Student clicks on "HomeWorks" tab. 2. Student then enters Github URL in the section "Enter Github URL of submission file". 3. Student clicks "Submit" button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to update: 1. Invalid Github URL 2. File is not of python type Remedy: Avoiding the above issues would allow a student to upload and submit assignment successfully.

Use Case 7:

Use Case:	**Upload zip file containing multiple python files by Choose file feature**
Primary Actor:	Student
Goal in Context:	To facilitate a student to submit a zip file containing multiple python files by choosing it from his local system.
Preconditions:	Student has the zip file is his local system.
Trigger:	A user decides to submit the assignment.
Scenario:	1. Student clicks on “HomeWorks” tab. 2. Student then clicks “Choose File” button in the section “Choose and upload file”. 3. Student then goes to the path where the file is present in local system and selects the zip file. 4. 4. Student clicks “Submit” button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to update: 1. Not selecting any file. 2. File is not of zip type. Remedy: Avoiding the above issues would allow a student to upload and submit assignment successfully.

Use Case 8:

Use Case:	**Upload zip file containing multiple python files by Github URL feature**
Primary Actor:	Student
Goal in Context:	To facilitate a student to submit a zip file containing multiple python files by entering the Github URL of the file to be uploaded and submitted.
Preconditions:	Student has Github URL of the zip file to be submitted.
Trigger:	A user decides to submit the assignment.
Scenario:	1. Student clicks on "HomeWorks" tab. 2. Student then enters Github URL in the section "Enter Github URL of submission file". 3. Student clicks "Submit" button.
Exceptions:	If user tries to submit the form with any of the below issues, he would not be able to update: 1. Invalid Github URL 2. File is not of zip type Remedy: Avoiding the above issues would allow a student to upload and submit assignment successfully.

Statistics Related Use Cases :

Use Case 9:

Use Case:	**Check System Status**
Primary Actor:	Course Staff
Goal in Context:	To allow a course staff to check various parameters of system like memory, processors, instances, heaps, threads, classes, counter status, etc.
Preconditions:	System status should be recorded.
Trigger:	A course staff decides to check system status.
Scenario:	1. Course staff clicks on “System Status” tab.

Use Case 10:

Use Case:	**Get number of plagiarism detection runs**
Primary Actor:	Course Staff
Goal in Context:	To allow a course staff to get number of plagiarism detection runs by all course staffs collectively.
Preconditions:	Plagiarism detection run count is incremented and maintained each time a course staff checks for plagiarism.
Trigger:	A course staff decides to get number of plagiarism detection runs.
Scenario:	1. Course staff clicks on “SystemStatus” tab. 2. Course staff clicks on “Get Count” button.

File Comparison Use Cases:

Use Case 11:

Use Case:	**Compare all submissions of students who submitted their assignments**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of the students for plagiarism detection and get a summary of plagiarism found via email.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Each file should be a python file or a zip file.
Trigger:	A course staff decides to detect plagiarism for the entire class.
Scenario:	1. Course staff clicks on "Compare" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff clicks "Find For Entire Class" button. 5. After some time, course staff gets an email containing pairs of students' usernames with plagiarism similarity percentage for each pair if plagiarism is found. If not found, email will contain message "No Plagiarism Found".

Use Case 12:

Use Case:	**Compare submissions of two students by all strategies**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and get similarity percentage corresponding to each strategy.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by all strategies for submissions of two students.
Scenario:	1. Course staff clicks on “Compare” tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Course staff clicks “Get Similarity By All” button to get similarity score corresponding to each strategy.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 13:

Use Case:	**Compare submissions of two students by Levenshtein Distance strategy.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and check if plagiarism is found or not by Levenshtein Distance strategy.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by Levenshtein Distance strategy for submissions of two students.
Scenario:	1. Course staff clicks on "Compare" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Then, course staff selects "LEVENSHTEIN_DISTANCE" from the Strategy dropdown. 6. Course staff clicks "Submit" button to get a message whether Plagiarism was found or not. Also, if found, course staff can check files which matched, similarity percentage and an option to view matching lines.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 14:

Use Case:	**Compare submissions of two students by LCS strategy.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and check if plagiarism is found or not by LCS strategy.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by LCS strategy for submissions of two students.
Scenario:	1. Course staff clicks on “Compare” tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Then, course staff selects “LCS” from the Strategy dropdown. 6. Course staff clicks “Submit” button to get a message whether Plagiarism was found or not. Also, if found, course staff can check files which matched, similarity percentage and an option to view matching lines.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 15:

Use Case:	**Compare submissions of two students by AST LCS strategy.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and check if plagiarism is found or not by AST LCS strategy.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by AST LCS strategy for submissions of two students.
Scenario:	1. Course staff clicks on "Compare" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Then, course staff selects "AST_LCS" from the Strategy dropdown. 6. Course staff clicks "Submit" button to get a message whether Plagiarism was found or not. Also, if found, course staff can check files which matched, similarity percentage and an option to view matching lines.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 16:

Use Case:	**Compare submissions of two students by AST Tree Edit Distance strategy.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and check if plagiarism is found or not by AST Tree Edit Distance strategy.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by AST Tree Edit Distance strategy for submissions of two students.
Scenario:	1. Course staff clicks on "Compare" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Then, course staff selects "AST_TREE_EDIT_DISTANCE" from the Strategy dropdown. 6. Course staff clicks "Submit" button to get a message whether Plagiarism was found or not. Also, if found, course staff can check files which matched, similarity percentage and an option to view matching lines.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 17:

Use Case:	**Compare submissions of two students by MOSS strategy.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and check if plagiarism is found or not by MOSS strategy.
Preconditions:	3. The files should be related to the same assignment of the same course. 4. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by MOSS strategy for submissions of two students.
Scenario:	7. Course staff clicks on "Compare" tab. 8. Course staff selects the course from the Course dropdown. 9. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 10. Course staff then selects username of both students from the First Student and Second Student dropdowns. 11. Then, course staff selects "MOSS" from the Strategy dropdown. 12. Course staff clicks "Submit" button to get a message whether Plagiarism was found or not. Also, if found, course staff can check files which matched, similarity percentage and an option to view matching lines.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 18:

Use Case:	**Compare submissions of two students by Weighted Score strategy.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to compare all the files of two students for plagiarism detection and check if plagiarism is found or not by Weighted Score strategy.
Preconditions:	1. The files should be related to the same assignment of the same course. 2. Both files should be of either python or zip type.
Trigger:	A course staff decides to detect plagiarism by Weighted Score strategy for submissions of two students.
Scenario:	1. Course staff clicks on "Compare" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Then, course staff selects "WEIGHTED_SCORE" from the Strategy dropdown. 6. Course staff clicks "Submit" button to get a message whether Plagiarism was found or not. Also, if found, course staff can check files which matched, similarity percentage and an option to view matching lines.
Exceptions	Less than two students have submitted the assignment selected.

Use Case 19:

Use Case:	**View matching similar lines between two plagiarized submissions.**
Primary Actor:	Course Staff
Goal in Context:	To facilitate a course staff to view similar lines by highlighting similar code between two plagiarized submissions.
Preconditions:	1. Two selected students' submissions need to be plagiarized.
Trigger:	A course staff wishes to find similar code between two student submissions using one of the strategies.
Scenario:	1. Course staff clicks on "Compare" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff then selects the assignment for which he wishes to detect plagiarism from the Assignment dropdown. 4. Course staff then selects username of both students from the First Student and Second Student dropdowns. 5. Then, course staff selects a strategy from the Strategy dropdown. 6. Course staff clicks "Submit" button to get a message that Plagiarism was found. 7. Course staff clicks on "Plagiarism found" message to check files which matched, similarity percentage and an option to view matching lines. 8. Course staff clicks on "Show Matching Lines" option which opens a modal of Plagiarism Report consisting of both students' usernames, file names, submitted python code and highlighted plagiarized similar code.
Exceptions	The submissions of the two students selected are not plagiarized.

Use Cases Not Completed

Use Case 20:

Use Case:	**Adding assignments to a course.**
Primary Actor:	Course staff
Goal in Context:	Course staff being able to add more assignments to the course.
Preconditions:	Course staff should be handling the course for which they want to add the assignments.
Trigger:	Course staff wishing to add a new assignment.
Scenario:	1. Course staff clicks on "Assignments" tab. 2. Course staff selects the course from the Course dropdown. 3. Course staff clicks on "Choose file" option and selects assignment file from his local system to upload new assignment. 4. Course staff clicks on "Submit" button.
Exceptions:	1. The course staff is not registered in the system. Remedy: They do not get access to the system. 2. The course staff does not handle the course for which they want to add the assignment. Remedy: Course staff cannot view the course on the system.

Use Case 21:

Use Case:	**Change the roles of users**
Primary Actor:	Administrator
Goal in Context:	Change the role of a registered user from one type of actor to another.
Preconditions:	The user must be registered into the system.
Trigger:	Administrator gets a request to change role of a user.
Scenario:	1. User registered as a student in the system. 2. Student becomes a teaching assistant for the same course in the later semester, and needs access privileges as a member of the course staff. 3. User requests to change the role from that of student to course staff. 4. Admin verifies the request and grants/denies access accordingly.
Exceptions:	1. When the user is not registered into the system. Remedy: the user is not able to access this feature. 2. If the user does not belong to the institution where the course is being offered. Remedy: the user cannot request change of role.