Monen: A new way to learn

Use-Case Specification

Version 1.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Table of Contents

[**1. Use-case Model 4**](#_heading=h.gjdgxs)

[**2. Use-case Specifications 4**](#_heading=h.30j0zll)

[2.1 User Login 4](#_heading=h.ikrhidrce61m)

[2.2 User Register 5](#_heading=h.k3vmfcwmhiwj)

[2.3 User Forgot Password 5](#_heading=h.whdw11ng4nsx)

[2.4. Logout 6](#_heading=h.ywghnyj1myjh)

[2.5. Create Deck 6](#_heading=h.q75e0pkmj7ds)

[2.6. Edit Deck 7](#_heading=h.xt7b2fpltgyp)

[2.7. Create Flashcard 7](#_heading=h.byu3073wq0gd)

[2.8. Edit Flashcard 7](#_heading=h.kt26q0qtr8p2)

[2.9. Edit Profile 8](#_heading=h.hy54m3f85lsw)

[2.10. Change app theme 8](#_heading=h.y4lx7a9pimhu)

[2.11. Search for a deck 9](#_heading=h.ptk3q9kxgu1t)

[2.12. Upload a deck to the library 9](#_heading=h.vk9rbh20okhc)

[2.13. Create a course 9](#_heading=h.qm1q5lrk3tc6)

[2.14. Add a student to a course 10](#_heading=h.2qg39wppjwrc)

[2.15. Remove a student from a course 10](#_heading=h.f3gppiaubjax)

[2.16. Play game 11](#_heading=h.tgwzo1d8xe60)

[2.17. Asking questions on the course forum 11](#_heading=h.7f3wpdstlp75)

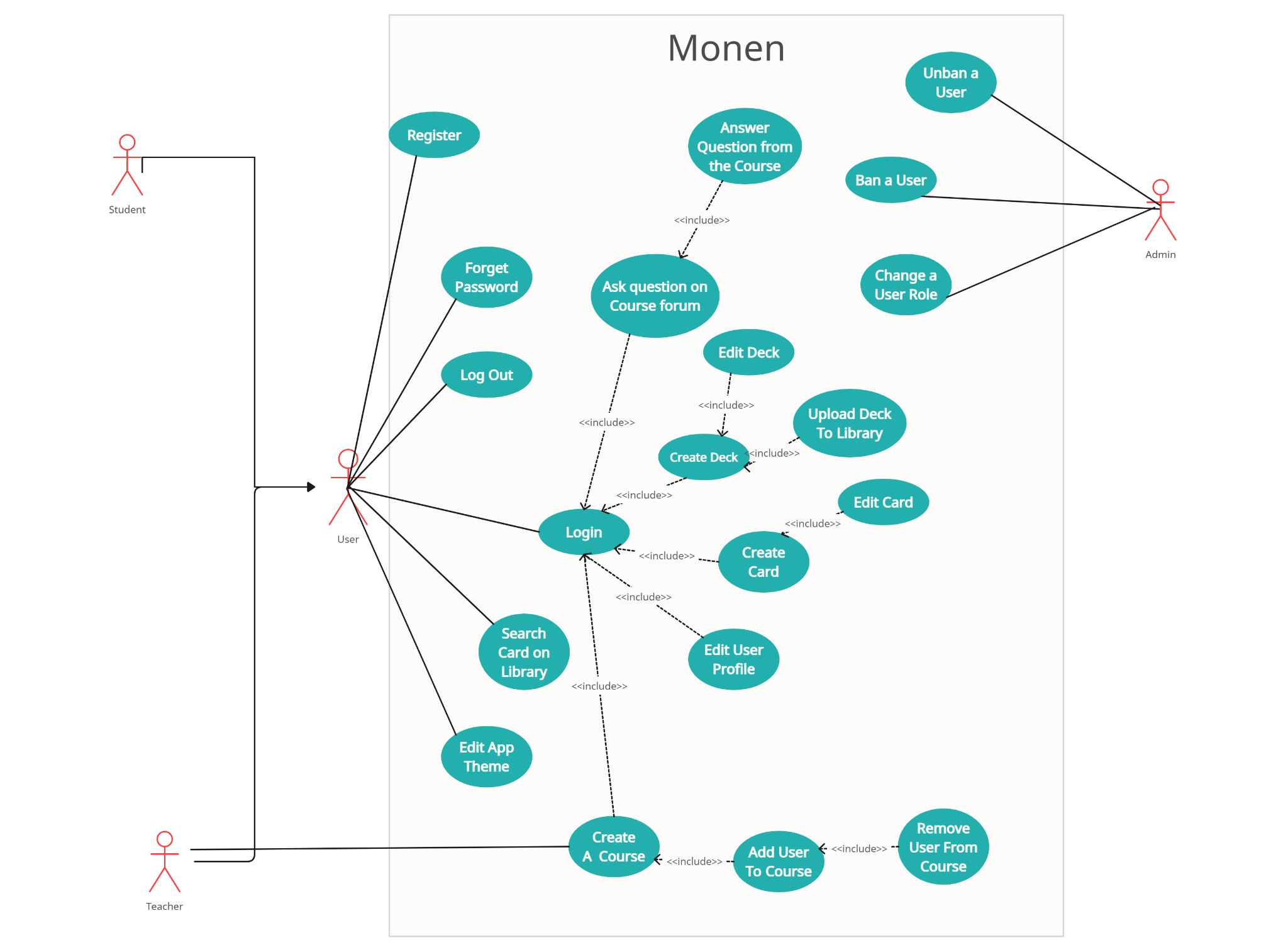
[2.18. Answering questions on the course forum 12](#_heading=h.69bcchz76y8i)

[2.19. Ban a user 12](#_heading=h.5wvfsea3qux6)

[2.20. Unban a user 13](#_heading=h.nwvihrxnejqx)

[2.19. Changing a user role 13](#_heading=h.ogjsw3h03uce)

# Use-case Model



# Use-case Specifications

## 2.1 User Login

| Use case Name | User Login |
| --- | --- |
| Brief description | This use case describes how a user can log in to their account. |
| Actors | Customer |
| Basic Flow | 1. User navigates to the login page. 2. User enters their registered email/username and password. 3. User clicks on the 'Login' button. 4. System verifies the credentials. 5. If the credentials are correct, the system logs the user into their account. 6. System redirects the user to their account/dashboard page. |
| Alternative Flows | **Alternative flow 1: Incorrect Credentials**   1. If credentials are incorrect, the system displays an error message. 2. Users can re-enter the credentials.   **Alternative flow 2: correct Credentials**   1. From the #1 of the basic flow, the app does not display any decks |
| Pre-conditions | User account has been previously registered. |
| Post-conditions | The user successfully logs in or receives an error message for incorrect  credentials. |

## 

## 2.2 User Register

| Use case Name | User Registration |
| --- | --- |
| Brief description | This use case describes how a user can create a new account. |
| Actors | User, System |
| Basic Flow | 1. User navigates to the registration/sign-up page. 2. User enters required details: username, email, password, etc. 3. User submits the registration form by clicking on the 'Sign Up' or 'Register' button. 4. System validates the entered information. 5. If the information is valid, the system creates a new user account. 6. System redirects the user to a confirmation page or directly to their new account. |
| Alternative Flows | **Alternative flow 1: Existing Email/Username**   1. If the email or username already exists, system prompts the user to choose a different one. |
| Pre-conditions | User navigates to the registration page. |
| Post-conditions | The user successfully creates a new account or receives an error message for existing credentials. |

## 

## 2.3 User Forgot Password

| Use case Name | Forgot Password |
| --- | --- |
| Brief description | This use case describes how a user can reset their forgotten password. |
| Actors | User, System |
| Basic Flow | 1. User navigates to the 'Forgot Password' or 'Reset Password' page. 2. User enters their registered email address. 3. User submits the email by clicking on the 'Reset Password' button. 4. System verifies the email address. 5. If the email exists, the system sends a password reset link to the user's email. 6. User receives the reset link in their email inbox. 7. User clicks on the reset link to go to the password reset page. 8. User enters a new password and confirms it. 9. System updates the password for the user. |
| Alternative Flows | **Alternative flow 1: None** |
| Pre-conditions | User remembers the email used during registration. |
| Post-conditions | The user successfully resets their password and can log in using the new credentials. |

## 

## 2.4. Logout

| Use case Name | User Logout |
| --- | --- |
| Brief description | This use case describes how a user can log out of their account. |
| Actors | User, System |
| Basic Flow | 1. User is logged in and navigates to the logout option. 2. User clicks on the 'Logout' or 'Sign Out' button. 3. System clears the user's session and logs the user out. 4. System redirects the user to the homepage or a logged-out confirmation page. |
| Alternative Flows | **Alternative flow 1: None.** |
| Pre-conditions | User is logged into their account. |
| Post-conditions | The user successfully logs out and can no longer access account-specific features until logging back in. |

## 

## 

## 2.5. Create Deck

| Use case Name | Create Deck |
| --- | --- |
| Brief description | This use case describes how the Customer creates a deck |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Deck’ button in the middle. 2. The system displays Card and Deck options. 3. The user clicks on the Deck option to create Deck 4. The system displays some choices to customize the deck 5. After customizing the deck, the user clicks on the ‘save’ button. |
| Alternative Flows | **Alternative flow 1: User cannot create a new deck**   1. **Because of reaching the deck quantity limit**    1. From #4 of the basic flow, the system shows a notification and a link to the Upgrade Premium page    2. If the user upgrade Premium successfully, back to #4 of the basic flow 2. **Because of unexpected error**    1. From the #3 of the basic flow, the system shows notification and back to the homepage.     **Alternative flow 2: Deck already existed**   1. From #5 of the basic flow, the system shows a notification if the user wants to update the deck or create a new one. 2. Back to step #4 |
| Pre-conditions | User goes to the app homepage |
| Post-conditions | The user successfully creates or updates Deck |

## 2.6. Edit Deck

| Use case Name | Edit Deck |
| --- | --- |
| Brief description | This use case describes how the Customer edits a deck |
| Actors | Customer |
| Basic Flow | 1. On the homepage, the user clicks on the ‘Deck’ button in the middle. 2. The system displays Card and Deck options. 3. The user clicks on the Deck option to edit Deck 4. The system displays some choices to customize the deck 5. After customizing the deck, the user clicks on the ‘save’ button. |
| Alternative Flows | **Alternative flow 1: The user cannot create a new deck**  **Because of unexpected error**  From the #1 of the basic flow, the system shows notification and back to the homepage. |
| Pre-conditions | User goes to the app homepage |
| Post-conditions | The user successfully create or update Deck |

## 2.7. Create Flashcard

| Use case Name | Create Card |
| --- | --- |
| Brief description | This use case describes how the Customer creates a card |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Deck’ button in the middle. 2. The system displays Card and Deck options. 3. The user clicks on the Card option to create Card 4. The system displays some choices to customize the card 5. After customizing the card, the user clicks on the ‘save’ button. |
| Alternative Flows | **Alternative flow 1: User cannot create a new card**   1. **Because of reaching the quantity limit** 2. From the #3 of the basic flow, the system shows error notification and back to the homepage. 3. **Because of unexpected error**    1. From the #1 of the basic flow, the system shows notification and back to the homepage. |
| Pre-conditions | User goes to the app homepage |
| Post-conditions | The user successfully create or update Card |

## 2.8. Edit Flashcard

| Use case Name | Edit Card |
| --- | --- |
| Brief description | This use case describes how the Customer edit a card |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Deck’ button in the middle. 2. The system displays Card and Deck options. 3. The user clicks on the Card option to edit Card 4. The system displays some choices to customize the card 5. After customizing the card, the user clicks on the ‘save’ button. |
| Alternative Flows | **Alternative flow 1: User cannot edit card**  **Because of unexpected error**  From the #1 of the basic flow, the system shows notification and back to the homepage. |
| Pre-conditions | User goes to the app homepage |
| Post-conditions | The user successfully edit Card |

## 2.9. Edit Profile

| Use case Name | Edit Profile |
| --- | --- |
| Brief description | This use case describes how the Customer edit the user Profile |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Setting’ button on the left. 2. The system displays Setting options. 3. The user clicks on the Profile button. 4. The system displays the list of user profile data. 5. The user edits the data to their preferences. 6. The user clicks on the Save button to save their changes. |
| Alternative Flows | **Alternative flow 1: User data failed to be edited because of invalid inputs**   1. From the #5 of the basic flow, the system shows notification and back to the Setting page. |
| Pre-conditions | User goes to the app homepage |
| Post-conditions | The user successfully edits their profile data. |

## 2.10. Change app theme

| Use case Name | Change app theme |
| --- | --- |
| Brief description | This use case describes how the Customer can change the theme of the app. |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Setting’ button on the left. 2. The system displays Setting options. 3. The user clicks on the Theme button. 4. The system displays the list of themes. 5. The user chooses the theme they like. 6. The user clicks on the Save button to save their changes. |
| Alternative Flows | **Alternative flow 1: Change unsuccessfully**   1. From the #5 of the basic flow, the system shows notification and back to the Setting page. |
| Pre-conditions | User goes to the app homepage |
| Post-conditions | The user successfully changes the app theme |

## 2.11. Search for a deck

| Use case Name | Search for a deck |
| --- | --- |
| Brief description | This use case describes how the customer searchs for a deck |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user enters a search term into the search bar. 2. The app displays a list of decks that match the search term. 3. The user selects the deck they want to view. 4. The app opens the selected deck. |
| Alternative Flows | **Alternative flow 1: No decks match the search term.**   1. From the #2 of the basic flow, the app displays a message indicating that no decks match the search term.   **Alternative flow 2: User enters an empty search term.**   1. From the #1 of the basic flow, the app does not display any decks |
| Pre-conditions | The user goes to the app homepage |
| Post-conditions | The user successfully edits their profile data. |

## 

## 2.12. Upload a deck to the library

| Use case Name | Update a deck to the library |
| --- | --- |
| Brief description | This use case describes how the Customer uploads a deck to the library |
| Actors | Customer |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Deck’ button 2. The user selects a deck. 3. The user clicks on the ‘Upload’ button. 4. The app processes the uploaded deck to the library 5. The app displays a message indicating that the deck has been uploaded successfully. |
| Alternative Flows | **Alternative flow 1: Upload fails due to network error.**   1. From #4 of the basic flow, if there is a network error that prevents the upload from completing, the app will display an error message indicating that the upload failed. The user can then try to upload the file again later. |
| Pre-conditions | The user goes to the app homepage |
| Post-conditions | The user has successfully uploaded the deck to the app. |

## 2.13. Create a course

| Use case Name | Create a course |
| --- | --- |
| Brief description | This use case describes how the Teacher creates a course |
| Actors | Teacher |
| Basic Flow | 1. At the homepage, the user clicks on the ‘Create course’ button 2. The user enters the course title, the description, and other relevant information. 3. The instructor adds course materials, such as decks, videos, documents, and quizzes. 4. The instructor sets the course enrollment options and grading policies. 5. The instructor publishes the course. |
| Alternative Flows | **Alternative Flow 1: The instructor does not have permission to create courses.**   1. From step 2 of the basic flow, if the instructor does not have the necessary permissions, the LMS will display an error message. The instructor can then contact an administrator to request permission to create courses.   **Alternative Flow 2: Course materials are not valid.**   1. From step 4 of the basic flow, if the uploaded course materials are not in a valid format or do not meet the LMS requirements, the LMS will display an error message. The instructor can then fix the errors and try again. |
| Pre-conditions | 1. The instructor is logged in. 2. The instructor has the necessary permissions to create courses. |
| Post-conditions | The instructor has successfully created a new course |

## 2.14. Add a student to a course

| Use case Name | Add a student to a course |
| --- | --- |
| Brief description | This use case describes how the Teacher adds a student to a course |
| Actors | Teacher or Administrator |
| Basic Flow | 1. At the homepage, the user clicks on the ‘course’ button 2. The user chooses a course 3. The user clicks on the ‘Add student’ button 4. The system shows all students that want to enroll 5. The user chooses students to access the course |
| Alternative Flows | **Alternative Flow 1: The student is already enrolled in the course.**   1. From the #5 of the basic flow, if the student is already enrolled in the course, the system will display an error message indicating that the student is already enrolled   **Alternative Flow 2: The student’s information is invalid**   1. From the #4, if the student’s information is not valid, the system will display an error message. |
| Pre-conditions | The user is logged in.  The user has the necessary permissions to add students to courses. |
| Post-conditions | The instructor has successfully adds students to courses. |

## 

## 2.15. Remove a student from a course

| Use case Name | Remove a student to a course |
| --- | --- |
| Brief description | This use case describes how the Teacher removes a student from course |
| Actors | Teacher or Administrator |
| Basic Flow | 1. At the homepage, the user clicks on the ‘course’ button 2. The user chooses a course 3. The user selects a Student and press Remove 4. The system shows a confirmation modal 5. The user confirms 6. The system removes the chosen student from the course |
| Alternative Flows | **Alternative Flow 1: The student is already removed from the course.**   1. From the #5 of the basic flow, if the student is already removed from the course, the system will display an error message indicating that the student is already removed |
| Pre-conditions | The user is logged in.  The user has the necessary permissions to remove students from courses. |
| Post-conditions | The instructor has successfully removed students from the course. |

## 

## 

## 2.16. Play game

| Use case Name | Play game to learn |
| --- | --- |
| Brief description | This use case describes how the Customer play games to learn |
| Actors | Customer |
| Basic Flow | 1. On the homepage, the user clicks on the ‘Deck’ button in the middle. 2. The system displays Card and Deck options. 3. The user clicks on the Deck option and chooses the deck to learn 4. The system shows a list of games. 5. The user chooses the game to play. 6. The system displays the game screen. |
| Alternative Flows | **Alternative Flow 1: There is no card in the deck**   1. From the #5 of the basic flow, if the user chosen deck has no card, show an error message |
| Pre-conditions | The user must have at least one deck to start learning |
| Post-conditions | The user successfully play the game to learn |

## 

## 2.17. Asking questions on the course forum

| Use case Name | Asking questions on the course forum |
| --- | --- |
| Brief description | This use case describes how the Customer can ask questions on the course forum. |
| Actors | Customer |
| Basic Flow | 1. The user navigates to the course. 2. The user clicks on the "New Topic" or "Ask a Question" button. 3. The system provides a form for the student to enter the question details, including a title and the question content. 4. Student fills out the form and clicks the "Submit" button. 5. The system posts the question to the selected course or topic category. 6. The system notifies the student that the question has been successfully posted. |
| Alternative Flows | **Alternative Flow 1: The question is invalid.**   1. From the #4 of the basic flow, if the user enters an inappropriate question detected by the language processor, displays an error message. |
| Pre-conditions | The user has the necessary permissions to remove students from courses. |
| Post-conditions | The user successfully posted a question on the course forum. |

## 

## 2.18. Answering questions on the course forum

| Use case Name | Answering questions on the course forum |
| --- | --- |
| Brief description | This use case describes how the Teacher can ask questions on the course forum. |
| Actors | Teacher |
| Basic Flow | 1. The user navigates to the course. 2. The user clicks on a course. 3. The system provides a list of questions posted by students. 4. The user chooses the question to answer and enters their answer. 5. The user clicks the Confirm button 6. The system notifies the teacher that the answer is submitted. 7. The system notifies the student that the question has been answered. |
| Alternative Flows | **Alternative Flow 1: The question no longer exists.**   1. From the #5 of the basic flow, if the student has already deleted the question, show an error notification to the teacher. |
| Pre-conditions | The user has the necessary permissions to answer a question. |
| Post-conditions | The user successfully answers a question on the course forum. |

## 

## 2.19. Ban a user

| Use case Name | Ban a user from the app |
| --- | --- |
| Brief description | This use case describes how the Administrator can ban a user from the app |
| Actors | Administrator |
| Basic Flow | 1. Admin logs in to the administrative panel of the system. 2. The system presents the admin with a list of users or a search functionality. 3. Admin selects the user profile to be banned. 4. Admin navigates to the user management section. 5. Admin locates the "Ban User" option. 6. Admin confirms the decision to ban the selected user. 7. The system updates the user's status to "Banned" in the user database. 8. The system terminates the user's active sessions, if any. 9. The system notifies the banned user about the account status change. 10. The system logs the ban action in the administrator's activity log. |
| Alternative Flows | **Alternative Flow 1: User is already banned**   1. If the selected user is already banned, the system notifies the admin. 2. The admin can choose to take additional actions or return to the user selection step. |
| Pre-conditions | Admin has valid credentials to access the administrative panel. |
| Post-conditions | The selected user is banned from accessing the system. |

## 

## 2.20. Unban a user

| Use case Name | Unban a user from the app |
| --- | --- |
| Brief description | This use case describes how the Administrator can unban a user from the app |
| Actors | Administrator |
| Basic Flow | 1. Admin logs in to the administrative panel of the system. 2. The system presents the admin with a list of banned users or a search functionality. 3. Admin selects the banned user profile to be unbanned. 4. Admin navigates to the user management section. 5. Admin locates the "Unban User" option. 6. Admin confirms the decision to unban the selected user. 7. The system updates the user's status to "Active" in the user database. 8. The system notifies the unbanned user about the account status change. 9. System logs the unban action in the administrator's activity log. |
| Alternative Flows | **Alternative Flow 1: User is already active**   1. If the selected user is already active, the system notifies the admin. 2. The admin can choose to take additional actions or return to the user selection step. |
| Pre-conditions | Admin has valid credentials to access the administrative panel. |
| Post-conditions | The selected user is unbanned and regains access to the system. |

## 

## 2.19. Changing a user role

| Use case Name | Change a user role |
| --- | --- |
| Brief description | This use case describes how the Administrator can change a user role |
| Actors | Administrator |
| Basic Flow | 1. Admin logs in to the administrative panel of the system. 2. System presents the admin with a list of users or a search functionality. 3. Admin selects the user whose role needs to be changed. 4. Admin navigates to the user management section. 5. Admin locates the "Change User Role" option. 6. Admin selects the new role for the user (e.g., from "User" to "Moderator" or vice versa). 7. Admin confirms the decision to change the user's role. 8. System updates the user's role in the user database. 9. System notifies the user about the role change. 10. System logs the role change action in the administrator's activity log. |
| Alternative Flows | **Alternative Flow 1: Invalid Role Selection**   1. If the admin tries to assign an invalid role, the system notifies the admin and provides a list of valid roles. 2. Admin selects a valid role and proceeds with the role change. |
| Pre-conditions | Admin has valid credentials to access the administrative panel. |
| Post-conditions | The selected user's role is successfully changed within the system. |

## 

## 