Kontroller

Use-Case Specification

Version 4.0

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 15/06/2025 | 1.0 | Detail the first draft of the use case’s description | Nguyen Gia Nghi |
| 07/07/2025 | 2.0 | Add more crucial use case (from 12 to 17) to the report, modify some old use cases (6 and 7), as well as the use-case model | Nguyen Gia Nghi |
| 25/07/2025 | 3.0 | Revise All User-case Specification | All members |
| 29/07/2025 | 4.0 | Revise All User-case Specification | Nguyen Minh Thuan |

Table of Contents

[**1. Use-case Model 4**](#_heading=h.ghvk8ablbjrc)

[**2. Use-case Specifications 5**](#_heading=h.6zfqgtam836d)

[2.1 Use-case: Log in to the platform 5](#_heading=h.syeg5aoc9j0b)

[2.2 Use-case: Create a new account 5](#_heading=h.52oz60k8v2jo)

[2.3 Use-case: Review a game 6](#_heading=h.9otwd69eflhz)

[Review 6](#_heading=h.onkam62clo9w)

[2.4 Use-case: Manage profile 7](#_heading=h.kge5wb6a6g82)

[Profile 7](#_heading=h.i0iilwe4iaka)

[2.5 Use-case: Create a comment for review 8](#_heading=h.lv3kx8kwz0ti)

[Comment 8](#_heading=h.hlpkdgm2qnvj)

[2.6 Use-case: Rate a Game 9](#_heading=h.zavi4ahpzv9j)

[Rate 9](#_heading=h.tgx5bdotciw0)

[2.6.1 Use-case: React a review or comment 10](#_heading=h.fd9zdsh0t5qj)

[React 10](#_heading=h.3ktscowytnwq)

[2.7 Use-case: Block a user 11](#_heading=h.xtnzhk97xoh)

[Block user 11](#_heading=h.yebtoervwa3c)

[2.8 Use-case: Follow a user 13](#_heading=h.bsc7xlmxhh1c)

[Follow user 13](#_heading=h.nh9yydz4egat)

[2.9 Use-case: Search 14](#_heading=h.m1o8o0ttqcqt)

[Search 14](#_heading=h.6tuzrosmdsfp)

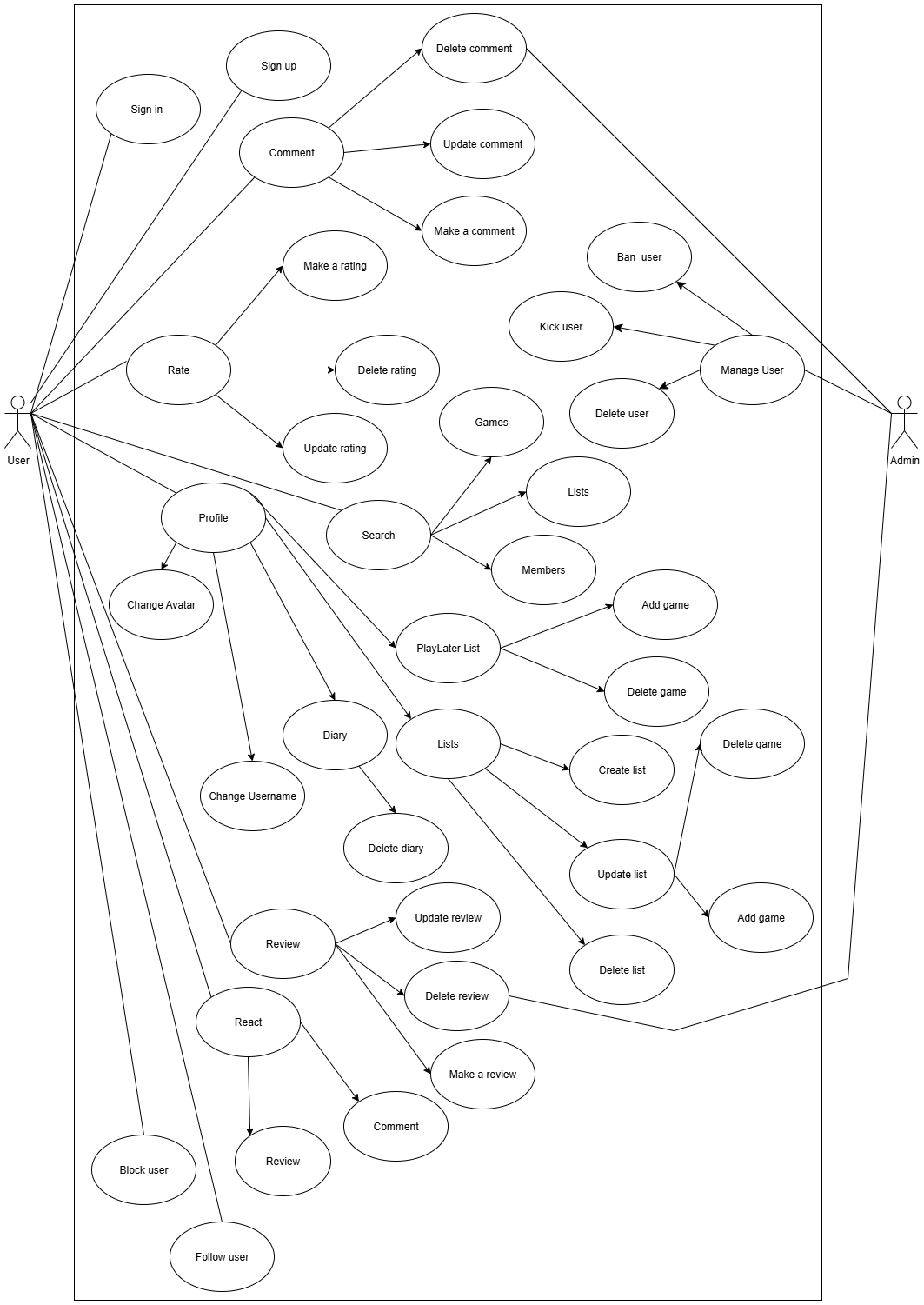
[2.10 Use-case: Manage user 15](#_heading=h.9lgxczhs90c)

[Manage User 15](#_heading=h.50ag0glmu6tq)

[2.11 Use-case: Delete review or comment of user 15](#_heading=h.fobm2i72ywxx)

[Delete Review, Delete Comment 15](#_heading=h.n36f1k4e4oai)

# Use-case Model



# Use-case Specifications

## Use-case: Log in to the platform

| Use case Name | Sign in |
| --- | --- |
| Brief description | This use case describes how a user logs into their account in order to access the website's features. |
| Actors | User |
| Basic Flow | 1. The user accesses the index page. 2. The user clicks on the “Sign in” button then going to auth page 3. The system displays the login form with fields for username and password. 4. The user enters their credentials. 5. The user clicks the “Sign in” button. 6. The system verifies the credentials. 7. If the credentials are valid, the system logs the user in. 8. The user is redirected to the profile page |
| Alternative Flows | **Alternative flow 1: Invalid credentials**   1. Step 6: If the credentials are invalid, the system displays an error message “Invalid username or password.” 2. The user remains on the auth page and can try again   **Alternative flow 2: Forgot password**   1. Step 3: If the user clicks on “Forgot password”, the system asks the user to prompt email to reset password. 2. The user can enter their email to receive reset instructions.   **Alternative flow 3: Cancel sign-in**   1. Step 2 - 4: The user clicks “Back” button 2. The system redirects the user back to the index page.   **Alternative flow 4: Sign up**   1. After Step 2, the user clicks “Sign up” to go to the account registration page. |
| Pre-conditions | The user has already registered.  User is not currently logged in |
| Post-conditions | User is successfully authenticated and redirected to the profile page  The system creates a user session. |

## Use-case: Create a new account

## 

| Use case Name | Sign up |
| --- | --- |
| Brief description | This use case describes how a new user creates an account to access the system's features. |
| Actors | User |
| Basic Flow | 1. The user accesses the index page. 2. The user clicks the “Sign up” button then going to register page 3. The system displays the registration form with fields: username, password, email, and confirm password. 4. The user enters the required information. 5. The user clicks the “Sign up” button. 6. The system validates the input (e.g., unique username, email format, matching passwords). 7. If all information is valid, the system creates a new user account. 8. The user is redirected to the sign-in page or logged in automatically. |
| Alternative Flows | **Alternative flow 1: Invalid credentials**   1. Step 6: If the username is already taken or email format is invalid or passwords do not match, the system shows appropriate error messages. 2. The user remains on the register page and can edit the form to try again.   **Alternative flow 2: Cancel sign-up**   1. Step 2–4: The user clicks “Back” button 2. The system redirects the user back to the index page.   **Alternative flow 3: Password too weak**   1. Step 6: If the password does not meet the security policy (e.g., too short, missing numbers/symbols), the system shows an error message: "Password must be at least 8 characters and include a symbol |
| Pre-conditions | The user is not currently logged in.  The user has not registered before using the same email or username. |
| Post-conditions | A new user account is created and stored in the system.  The user is either redirected to the login page or automatically logged in with a new session. |

## Use-case: Review a game

| Use case Name | Review |
| --- | --- |
| Brief description | This use case describes how a user views reviews of a specific game, including browsing through content, checking comments, seeing who reacted to reviews, and visiting author profiles. Users can also navigate to other sections using the top menu. |
| Actors | User |
| Basic Flow | 1. The user opens the Review Page of a game (via Game Page or navigation menu). 2. The system displays a list of reviews for that game, each containing: 3. Author name and avatar (clickable). 4. Review content (collapsed/expanded). 5. Reaction count (likes, dislikes, etc.). 6. Preview of top comments (e.g., 2–3 most recent). 7. The user scrolls through the list of reviews. 8. The user can perform the following interactions: 9. View Full Review → Click collapsed review → system expands it. 10. View Comments → Click “View all comments” → comment section expands. 11. View Reactions → Click number of reactions (e.g., “15 Likes”) → overlay/popup shows users who reacted. 12. Visit Author Profile → Click author name/avatar → navigate to Profile page. 13. Use Top Navigation Menu → Click “Home”, “Profile”, “Search”, etc. → system navigates accordingly. |
| Alternative Flows | **Alternative Flow 1: Not Logged In**   1. When clicking to react or comment 2. System prompts login.   **Alternative Flow 2: No Comments**   1. If a review has no comments 2. Display: **“No comments yet.”**   **Alternative Flow 3: No Reactions**   1. If no reactions 2. Display: **“No users have reacted yet.”**   **Alternative Flow 4: Review Load Failure**   1. At step 2: If the review list cannot be loaded due to an error 2. System displays: “Failed to load reviews. Please try again later.” |
| Pre-conditions | The user is on the Review Page of a specific game.  The system has loaded the available reviews and related metadata. |
| Post-conditions | The user browses review content and interacts with comments/reactions.  If applicable, the system records analytics or scroll position for personalized content later. |

## Use-case: Manage profile

| Use case Name | Profile |
| --- | --- |
| Brief description | This use case describes how a user can view and manage their profile information, including updating avatar and username, accessing followers/following lists, and exploring suggested content. From this page, users can also navigate to other areas using the top navigation menu. |
| Actors | User |
| Basic Flow | 1. The user navigates to the Profile page from the navigation bar. 2. The system displays user information: avatar, username, bio, and follower/following counts. 3. The user may perform one or more of the following actions: 4. Change Avatar:    1. The user clicks on the avatar.    2. The system opens a file dialog to upload a new image.    3. The user selects and uploads the image.    4. The system updates and displays the new avatar. 5. Change Username    1. The user clicks an edit icon near their username.    2. The system shows a text field for new input.    3. The user enters a new username and confirms.    4. The system validates and saves the change. 6. View Followers/Following    1. The user clicks on "Followers" or "Following" count.    2. The system displays a list of users with names and avatars. 7. As the user scrolls down the Profile page, the system shows recommended content (e.g., trending games, reviews, or users to follow). |
| Alternative Flows | **Alternative Flow 1: Invalid Username Format**   1. At Step 3b: 2. If the new username contains invalid characters or is already taken, 3. The system displays an error message: “Invalid or taken username.”   **Alternative Flow 2: Image Upload Failure**   1. At Step 3a 2. If the image file is too large or of an unsupported format 3. The system shows: “Image upload failed. Please try another file.”   **Alternative Flow 3: Not Logged In**   1. At any interaction step, if the session expires 2. The system redirects to the auth page.   **Alternative Flow 4: Access Diary Page**   1. User clicks on “Diary” tab/menu 2. The system navigates to the Diarypage, where the user can view, create, update, or delete diary entries.   **Alternative Flow 5: Access PlayLater Page**   1. User clicks on “PlayLater” tab 2. The system navigates to the PlayLaterpage, where the user can manage saved games.   **Alternative Flow 6: Not Logged In**   1. User clicks on “Lists” tab 2. The system navigates to the Listspage, where the user can create, update, or delete game collections. |
| Pre-conditions | The user is signed in.  The user is on the Profile page. |
| Post-conditions | Any profile changes (avatar/username) are saved.  The user may navigate to other sections for more actions.  Suggested content and user-related info are updated/displayed dynamically. |

## Use-case: Create a comment for review

| Use case Name | Comment |
| --- | --- |
| Brief description | This use case describes how a user can write and post a comment on a specific review in the Review page. |
| Actors | User |
| Basic Flow | 1. The user accesses the Review page. 2. The system displays a list of reviews. 3. The user finds a review they wish to comment on. 4. The user clicks the “Comment” button associated with that review. 5. The system scrolls or expands a comment input field below the selected review. 6. The user types their comment in the input field. 7. The user clicks the “Post” button. 8. The system validates the comment (e.g., checks for empty content or length limits). 9. The system saves the comment to the database and links it to the selected review. 10. The system refreshes the comment section of that review to show the newly posted comment. |
| Alternative Flows | **Alternative Flow 1: Empty Comment**   1. At step 8: If the comment is empty or contains only whitespace 2. The system displays a validation error: “Comment cannot be empty.” 3. The user remains on the same interface to revise their input.   **Alternative Flow 2: Too Long Comment**   1. At step 8: If the comment exceeds the maximum allowed characters (e.g., 500 chars), 2. The system displays a message: “Comment is too long.” 3. The user can edit the comment and retry.   **Alternative Flow 3: Not Logged In**   1. At step 4: If the user is not logged in and clicks “Comment,” 2. The system prompts the user to log in first. 3. After logging in, the user is redirected back to the same review. |
| Pre-conditions | The user is logged in.  The user is on the Review page.  The page displays a list of reviews, each with a visible “Comment” button. |
| Post-conditions | The new comment is saved and displayed under the selected review.  Other users can see the comment immediately. |

## Use-case: Rate a Game

| Use case Name | Rate |
| --- | --- |
| Brief description | This use case describes how a user can rate a specific game on a scale of 1 to 10 from the Game Page. The user can also update or delete their rating. Additionally, the user can choose to write a review for the rated game. |
| Actors | User |
| Basic Flow | 1. The user accesses the Game Page displaying a list of games. 2. The system displays each game with its: 3. Title, cover image, and short description. 4. Average rating score (out of 10). 5. User’s current rating (if previously rated). 6. The user selects a game to rate. 7. The user chooses a score from 1 to 10 (e.g., via slider or clickable stars/numbers). 8. The user submits the rating. 9. The system: 10. Validates the score (must be between 1–10). 11. Saves the user’s rating. 12. Updates the game’s average rating in real-time. 13. Confirms the action to the user (e.g., “Your rating has been saved”). 14. If the user previously rated the game, the new rating will overwrite the previous one. |
| Alternative Flows | **Alternative Flow 1: Update Existing Rating**   1. At step 3, if the user already rated the game 2. The existing score is shown, allowing the user to adjust it. 3. Upon submission, the system updates the old rating with the new value.   **Alternative Flow 2: Delete Rating**   1. Instead of updating, the user clicks a “Delete Rating” button. 2. The system removes the user’s rating and updates the game’s average accordingly. 3. Confirmation is shown: “Rating deleted.”   **Alternative Flow 3: Not Logged In**   1. At step 4, if the user is not logged in and tries to rate 2. The system prompts the user to log in. 3. After logging in, they are returned to the same game page.   **Alternative Flow 4: Write a Review**   1. After rating, the user clicks “Write a Review”. 2. The system opens the Review Creation Page for the selected game. 3. The user may write and submit a review (handled in a separate use-case: *Create Review*). |
| Pre-conditions | The user is logged in.  The user is on the Game Page.  The page displays a list of games with their information and current average rating. |
| Post-conditions | The new comment is saved and displayed under the selected review.  Other users can see the comment immediately. |

## Use-case: React a review or comment

| Use case Name | React |
| --- | --- |
| Brief description | This use case allows a logged-in user to add or remove a reaction (e.g., like, heart, haha, angry…) on a review or comment. The user may react directly from the review list or open the detailed view of a review to react to it or its associated comments. |
| Actors | User |
| Basic Flow | 1. The user opens the **Review page**. 2. The system displays a list of reviews for a selected game. 3. The user performs one of the following: 4. Clicks a reaction icon (e.g., "like", "heart") on a review directly from the list. 5. Clicks a review to open the **full review view**, then:    1. Click a reaction icon on the detailed review.    2. Scrolls to the comment section, and clicks a reaction icon on any comment. 6. The system registers or updates the reaction accordingly: 7. If the user hasn’t reacted yet → adds the reaction. 8. If the user clicks the same reaction again → removes it. 9. If the user switches to another reaction → updates the reaction. 10. The system updates the reaction count and provides visual feedback (e.g., highlight or animation). |
| Alternative Flows | **Alternative Flow 1: Not Logged In**   1. At step 3: When user clicks any reaction icon 2. System prompts for login before allowing the action.   **Alternative Flow 2: Network or Server Failure**   1. At step 4: If system fails to process reaction 2. Displays: "Failed to react. Please try again."   **Alternative Flow 3: Reaction Already Exists (Same Type)**   1. At step 4: If user clicks on a reaction they already gave 2. The system removes the reaction instead of adding a duplicate.   **Alternative Flow 4: Reaction Updated**   1. If the user switches from one reaction to another (e.g., from “Like” to “Haha”) 2. The system replaces the old one with the new one and updates counts accordingly. |
| Pre-conditions | The user is logged in.  The Review page is accessible, and the review or comment is rendered with reaction options. |
| Post-conditions | The system records or updates the user’s reaction to the selected review or comment.  UI is refreshed to reflect new reaction states and counts. |

## Use-case: Block a user

| Use case Name | Block user |
| --- | --- |
| Brief description | This use case allows a logged-in user to block another user. Blocking prevents further interactions from the blocked user, including viewing the blocker’s profile, commenting, following, reacting, or messaging. The block action can be performed from any page where the target user is visible (e.g., comments, reviews, profiles). |
| Actors | User |
| Basic Flow | 1. The user is browsing a page that displays another user (e.g., in a review, comment, follower list, or profile page). 2. The user performs one of the following: 3. Right-clicks (or taps menu icon) on the target user’s avatar/name → selects “Block User” 4. Navigates to the target user’s profile → clicks a **“**Block**”** button (e.g., top-right corner or in dropdown menu). 5. The system prompts for confirmation: “Are you sure you want to block this user?” 6. The user confirms. 7. The system processes the block: 8. Updates database to mark the target user as blocked by the current user. 9. Hides or restricts interactions between the two users. 10. Optionally notifies the user: *“User successfully blocked”*. |
| Alternative Flows | **Alternative Flow 1: User Cancels at Confirmation Prompt**   1. At step 4: The user cancels the block → No changes made.   **Alternative Flow 2: Already Blocked**   1. At step 5: If the user is already blocked 2. The system may show *“This user is already blocked.”* or disable the action.   **Alternative Flow 3: Block Failure**   1. At step 5: If the block request fails due to network/server issue 2. Show message: *“Failed to block user. Please try again.”*.   **Alternative Flow 4: Not Logged In**   1. At step 2: If the user is not logged in 2. System prompts login before allowing block action.   **Alternative Flow 5: Unblock a Previously Blocked User**   1. The user encounters a previously blocked user (e.g., in a comment section, review list, or suggestion list). 2. The system displays limited information about the blocked user (e.g., name and avatar only). 3. The user right-clicks (or uses a dropdown/menu) on the blocked user's avatar or username. 4. The user selects **"**Unblock User" from the context menu. 5. The system prompts a confirmation dialog: “Are you sure you want to unblock this user?” 6. The user confirms the action 7. The system removes the block and displays a success message: “User has been unblocked successfully.” |
| Pre-conditions | The user is logged in.  The target user is visible somewhere in the UI (e.g., profile, comment, follower list). |
| Post-conditions | The target user is blocked by the current user.  Future interactions are restricted:   * Blocked users can’t view the blocker's profile. * Can’t comment, react, or follow the blocker. * Past interactions may be hidden or filtered (optional depending on policy). |

## Use-case: Follow a user

| Use case Name | Follow user |
| --- | --- |
| Brief description | This use case describes how a user can follow another user to receive updates or content recommendations based on that user’s activity. The action can be performed from any location where another user's presence is visible (e.g., profile page, comment, review, or suggestion list). |
| Actors | User |
| Basic Flow | 1. The user sees another user’s profile or username/avatar on any page (e.g., review, comment, profile suggestions). 2. The user right-clicks (or clicks on an options icon) next to the username/avatar. 3. The user selects “Follow” from the context menu. 4. The system processes the request and updates the following status. 5. The UI reflects the updated state (e.g., button changes to “Following”). 6. The system stores the following relationship and may begin updating the feed or suggestions accordingly. |
| Alternative Flows | **Alternative Flow 1: Already Following the User**   1. The system shows the status as “Following” instead of “Follow.” 2. Optionally, the user can unfollow from the same menu.   **Alternative Flow 2: Attempt to Follow a Blocked User**   1. If the target user is currently blocked by the acting user: 2. The system displays: *“You must unblock this user before following them.”*   **Alternative Flow 3: Attempt to Follow After Being Blocked by Target**   1. If the target user has blocked the acting user: 2. The following action is denied. 3. The system shows a message: *“You cannot follow this user.”*   **Alternative Flow 4: Follow Request Failure**   1. If the follow action fails due to network or server issues 2. System displays: *“Failed to follow user. Please try again later.”*   **Alternative Flow 5: Unfollow a Previously Followed User**   1. The user clicks the “Following” status or accesses the same menu. 2. The user selects **“Unfollow”**. 3. System updates and confirms the unfollow action. |
| Pre-conditions | The acting user is logged in.  The acting user is not currently blocked by the target user. |
| Post-conditions | The following relationship is recorded.  The system may adjust the user’s feed, suggestion system, or notification preferences accordingly. |

## Use-case: Search

| Use case Name | Search |
| --- | --- |
| Brief description | This use case describes how a user performs a keyword-based search from the search bar, available on most pages (e.g., game page, review page, profile list), to find content that matches their interest — such as games, reviews, or user profiles. |
| Actors | User |
| Basic Flow | 1. The user accesses any page that displays searchable content (e.g., Review page, Game page, etc.). 2. The user clicks on the search bar typically located at the top of the interface. 3. The user types a keyword or phrase (e.g., game title, username, review content). 4. The system performs a search across relevant entities (games, reviews, users, etc.). 5. The system displays a list of matching results, usually grouped or filtered by type. 6. The user can click on a result to navigate to the corresponding page (e.g., game detail, user profile, review). |
| Alternative Flows | **Alternative Flow 1: No Search Results**   1. If no matching items are found: 2. System displays: *“No results found for ‘{query}’.”* 3. Suggests alternative or recent searches, if applicable.   **Alternative Flow 2: Empty Search Query**   1. If the user submits without entering any keyword: 2. System displays: *“Please enter a keyword to search.”* 3. Search is not executed.   **Alternative Flow 3: Search Failure**   1. If the search service fails due to a network or system issue: 2. System displays: *“Unable to search at the moment. Please try again later.”* |
| Pre-conditions | The user has access to the search bar on the current page.  Searchable content exists in the system. |
| Post-conditions | The system returns and displays relevant results.  The user can navigate to detailed pages based on the selected result. |

## Use-case: Manage user

| Use case Name | Manage User |
| --- | --- |
| Brief description | This use case describes how an admin performs moderation and management operations on user accounts, including banning, unbanning, kicking, or deleting users from the platform. |
| Actors | Admin |
| Basic Flow | 1. Admin logs into the admin panel. 2. The system displays the user management interface, listing all registered users. 3. Admin searches or selects a user from the list. 4. The system displays user details and available actions:    1. Ban / Unban User    2. Kick User (force logout)    3. Delete User Account 5. Admin chooses an action (e.g., Ban). 6. The system asks for confirmation and optional reasons. 7. The system performs the action and updates user status. |
| Alternative Flows | **Alternative Flow 1: Cancel Operation**   1. Admin cancels at confirmation step → no changes made.   **Alternative Flow 2: Action Not Available**   1. If the selected action is not applicable (e.g., trying to ban an already banned user), the system disables or hides that option.   **Alternative Flow 3: Action Failed**   1. If the system fails to process the action (e.g., network/server error), it displays an appropriate error message.   **Alternative Flow 4: Undo Action (Unban / Re-activate)**   1. Admin chooses to reverse a previous action, such as unbanning a user or restoring a deleted account (if supported). |
| Pre-conditions | Admin is authenticated and authorized.  User accounts exist. |
| Post-conditions | The user's status is updated in the system.  The affected user may be restricted (ban), logged out (kick), or removed (delete).  The system logs the action for audit purposes. |

## Use-case: Delete review or comment of user

| Use case Name | Delete Review, Delete Comment |
| --- | --- |
| Brief description | This use case allows an admin to remove inappropriate or rule-violating content such as reviews or comments created by users. Admins can access this feature from moderation tools embedded within the content. |
| Actors | Admin |
| Basic Flow | 1. The admin accesses a moderation panel or navigates to a page that contains user-generated content (e.g., reviews or comments). 2. The admin identifies a review or comment that requires deletion (via report flag, keyword, or manual inspection). 3. The admin clicks the "Delete" or "Remove" button associated with that specific review or comment. 4. The system prompts the admin for confirmation. 5. Upon confirmation, the system deletes the selected content. 6. The system updates the UI to reflect that the content has been removed and optionally logs the action. |
| Alternative Flows | **Alternative Flow 1: Cancel Deletion**   1. At step 4, if the admin cancels the confirmation dialog: 2. The deletion is aborted. 3. The content remains unchanged.   **Alternative Flow 2: Content Already Deleted**   1. If the content was deleted before the current admin confirms: 2. *The system notifies: “This content has already been removed.”*   **Alternative Flow 3: Deletion Failure**   1. If the deletion fails due to a system error or permission issue: 2. The system displays: *“Failed to delete content. Please try again.”* |
| Pre-conditions | The admin is authenticated and has proper moderation privileges.  The review or comment exists and is visible to the admin. |
| Post-conditions | The targeted user content is removed from the system.  The system may log this action for future auditing or notify the original content creator (optional). |