

Movie Rating System

SUBMITTED BY

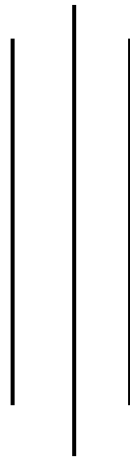
Aayush Bajracharya (12302/20)

Pratiksha Gartaula (12319/20)

Bidhata Upreti (12312/20)

Abhishek Upreti (12305/20)

Hetauda School of Management & Social Sciences



A Report Submitted to

Er. Sameer Gautam

Bachelor of Information Management (7th semester)

Table of Content

Movie Rating System	4
1. Introduction	4
2. Objectives	4
3. Actors	4
4. Objectives/Goals of Each Actor	4
5. Interactions and Missions to Achieve Goals	5
5.1 Users' Interactions:	5
5.2 Admin's Interactions:	5
5.3 Moderator's Interactions:	6
6. Use Case Scenarios	8
Finding Conceptual Classes and Description Classes	10
7. Identify Nouns and Nouns Phrase in textural description of the domain.	10
8. Domain Model	10
9. Class Descriptions and Class Diagram	12
System Sequence Diagram	14
10. Sequence Diagram	14
Activity Diagram of Role-Specific User Interactions and System Workflows	18
11. Activity Diagram	18
UML State Diagrams and Modeling	20
12. State Diagram	20
13. Deployment Diagram	21

Table of Figure

Figure 1: Usecase Diagram	7
Figure 2: Domain Model before Refining.....	10
Figure 3: Refined Class Diagram.....	12
Figure 4: Sequence Diagram in User Perspective.....	14
Figure 5: Sequence Diagram in Admin Perspective	15
Figure 6: Sequence Diagram in Moderator Perspective.....	16
Figure 7; Activity Diagram	18
Figure 8: State Diagram	20
Figure 9: Deployment Diagram	22

Movie Rating System

1. Introduction

This study examines the design and development of a Movie Rating System as a case study to better understand the principles and applications of Object-Oriented Analysis and Design. The system is a working example of how to apply OOAD concepts such as system requirements identification, use case design, and class and interaction diagram implementation. This project provides students and learners with hands-on experience with OOAD methodologies while developing a system that manages movie data, allows user interactions for rating and reviewing, and includes administrative features. This study uses the Movie Rating System to connect theoretical concepts with real-world applications, emphasising the importance of structured design approaches in software development.

2. Objectives

- To explore and apply core OOAD concepts such as encapsulation, inheritance, polymorphism, and abstraction in designing and analysing a real-world system.
- To study the processes of requirements gathering, identifying objects, and defining relationships, which are essential for designing

3. Actors

- a) **Users:** General audience members who interact with the system to rate movies and leave comments.
- b) **Admin:** Manages the system's movie database, user reviews, and generates analytical reports.
- c) **Moderator:** Monitors user-generated content to ensure compliance with platform guidelines.

4. Objectives/Goals of Each Actor

- a) **Users:**
 - **Rate Movies:** Users can provide ratings for movies, which help other users gauge the movie's popularity and quality.
 - **View Ratings:** Users can see aggregated ratings and reviews of movies.
 - **Comment:** Users can leave textual feedback on movies.
 - **Edit/Remove Comments:** Users can modify or delete their own comments if needed.

b) Admin:

- **Manage Movies:** Add, edit, and update the list of movies, including details like title, release date, cast, synopsis, and poster.
- **Generate Reports:** Create detailed reports on user activity, movie ratings, and comments for analytical purposes.
- **Moderate User Reviews:** Approve or reject user-submitted reviews to ensure quality and relevance.

c) Moderator:

- **Reviewing Content:** Verify the appropriateness of user-generated content (ratings and comments).
- **Monitor Inappropriate Comments:** Identify and take actions such as deleting offensive comments or banning users violating platform policies.

5. Interactions and Missions to Achieve Goals

5.1 Users' Interactions:

a) Login:

- Users authenticate themselves through credentials to access the system's features.

b) Rate Movies:

- Select a movie from the database.
- Provide a rating using a predefined scale (e.g., 1 to 5 stars).

c) Comment on Movies:

- Navigate to the movie's review section.
- Leave a textual comment with or without rating.

d) Edit/Delete Comments:

- Locate existing comments in their profile.
- Modify or delete comments using provided options.

5.2 Admin's Interactions:

a) Manage Movies:

- **Login Admin Panel:** Securely access the admin dashboard.
- **Approve User Reviews:** Moderate and approve reviews flagged for quality.
- **Access Movies Management Section:** View and manage the movie list.
- **Add New Movies:** Add information for upcoming or newly released movies.

- **Edit Existing Movies:** Update or modify movie details.
- b) **Generate Reports:**
- **Login:** Authenticate using admin credentials.
 - **Access Report Section:** Use dashboard tools to access report generation features.
 - **Select Report Type:** Choose between reports like "Top Rated Movies," "Most Commented Movies," or "User Activity Trends."
 - **Export Report:** Save the report in formats like PDF or Excel for further use.
- c) **View Ratings:**
- Login to the admin panel.
 - Access a dashboard detailing aggregated ratings and individual user feedback.

5.3 Moderator's Interactions:

- a) **Reviewing:**
- **Log in to the Moderation Platform:** Authenticate using moderator credentials.
 - **Access Review Queue:** View flagged content or new submissions requiring moderation.
 - **Select Content to Review:** Analyse individual ratings or comments.
 - **Make a Decision:** Decide to approve or reject content based on quality and relevance.
- b) **Monitor Inappropriate Comments:**
- **Log In:** Access the moderation panel securely.
 - **Take Action:**
 - Delete offensive or inappropriate comments.
 - Ban users for repeated violations.

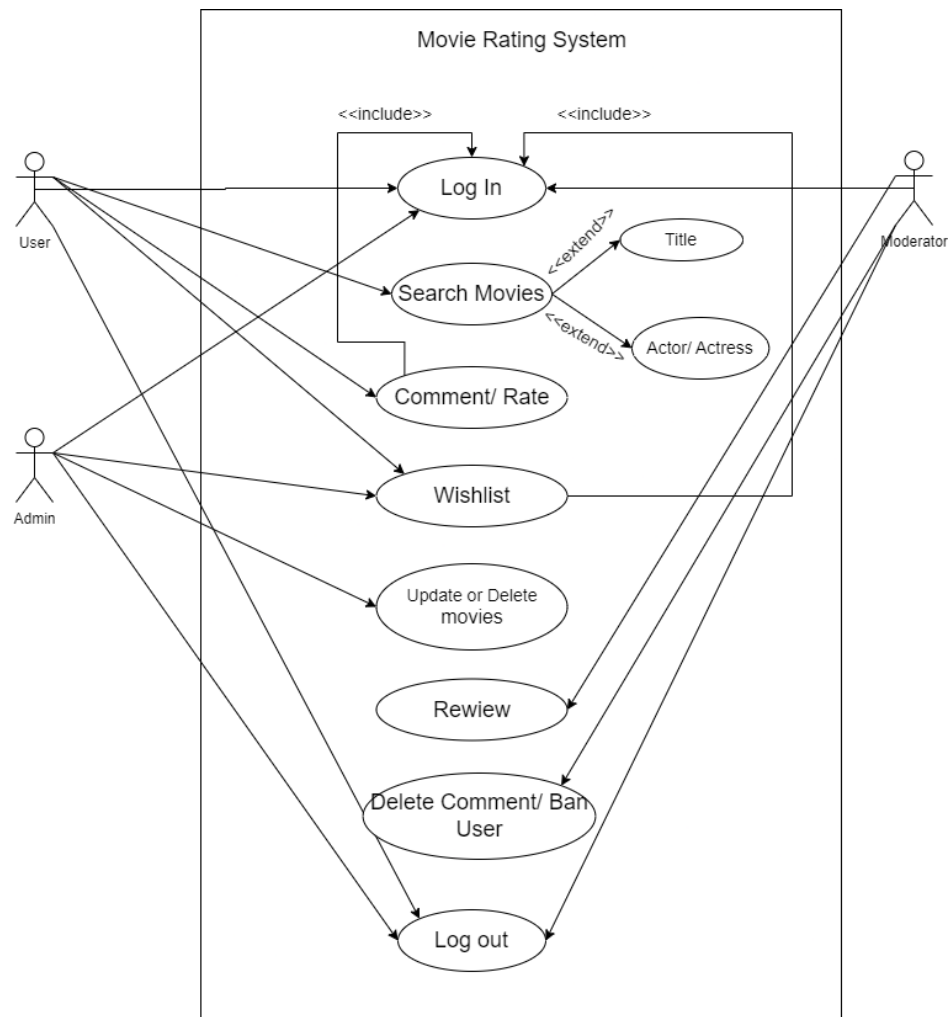


Figure 1: Usecase Diagram

6. Use Case Scenarios

a) Use case :- Log-In

Actors:-User, Admin, moderator.

Types:-Primary.

Descriptions:-

This use case describes the process by which registered actors (Users, Admins, and Moderators) log in to the site to access their respective dashboards and perform their tasks. Each actor has different roles and permissions, which determine the functionalities available to them upon successful login.

- The actor navigates to the login page of the website.
- The actor enters their username and password in the provided fields.
- The actor clicks the "Login" button.
- The system verifies the credentials:
- If the credentials are valid:
 - The system identifies the role of the actor (User, Admin, or Moderator).
 - The system transfers the actor to their respective dashboard.
- If the credentials are invalid: Error message will be display “Failed to Login”.

b) Use Case: Rating

Actors: User, Admin, Moderator

Types: Primary

Descriptions: This use case describes the process by which registered actors (Users, Admins, and Moderators) can rate content on the website. Each actor has different roles and permissions that determine the functionalities available to them upon successful rating submission.

- The actor navigates to the content they wish to rate (e.g., an article, product, or service).
- The actor selects a rating option (e.g., stars, thumbs up/down, numeric score) provided by the system.
- The actor clicks the "Submit Rating" button.
- The system verifies the rating input:
- If the rating input is valid:
 - The system records the rating in the database.
 - The system updates the overall rating for the content based on the new input.
 - The system displays a confirmation message, "Rating Submitted Successfully."

- If the rating input is invalid (e.g., out of range, missing):
 - The system displays an error message, "Failed to Submit Rating."

c) **Use Case: Comment**

Actors: User, Admin, Moderator

Types: Primary

Descriptions: This use case describes the process by which registered actors (Users, Admins, and Moderators) can comment on content on the website. Each actor has different roles and permissions that determine the functionalities available to them upon successful comment submission.

- The actor navigates to the content they wish to comment on (e.g., an article, product, or service).
- The actor enters their comment in the provided comment field.
- The actor clicks the "Submit Comment" button.
- The system verifies the comment input:
 - If the comment input is valid:
 - The system records the comment in the database.
 - The system displays the comment under the respective content.
 - The system displays a confirmation message, "Comment Submitted Successfully."
 - If the comment input is invalid (e.g., empty, contains prohibited words):
 - The system displays an error message, "Failed to Submit Comment."

d) **Also, Note:**

- Users can typically comment on content multiple times.
- Admins and Moderators may have additional permissions to edit, approve, or delete comments if necessary.

Finding Conceptual Classes and Description Classes

7. Identify Nouns and Nouns Phrase in textural description of the domain.

Key nouns and phrases identified in the description include:

- **User:** Represents the end-user interacting with the system.
- **Moderator:** Represents a user role that reviews and monitors inappropriate content.
- **Admin:** A user role responsible for managing movies and generating reports.
- **Movies:** The primary entity being rated, commented on, and managed.
- **Wish list:** Optional feature allowing users to save movies for later reference.
- **Genre:** Categorization of movies, such as action, comedy, drama, etc.
- **Review:** Represents textual feedback left by users.
- **Action:** Operations such as approving, rejecting, deleting, or banning performed by moderators/admins.
- **Rating:** Numerical or star-based evaluation provided by users for movies.

8. Domain Model

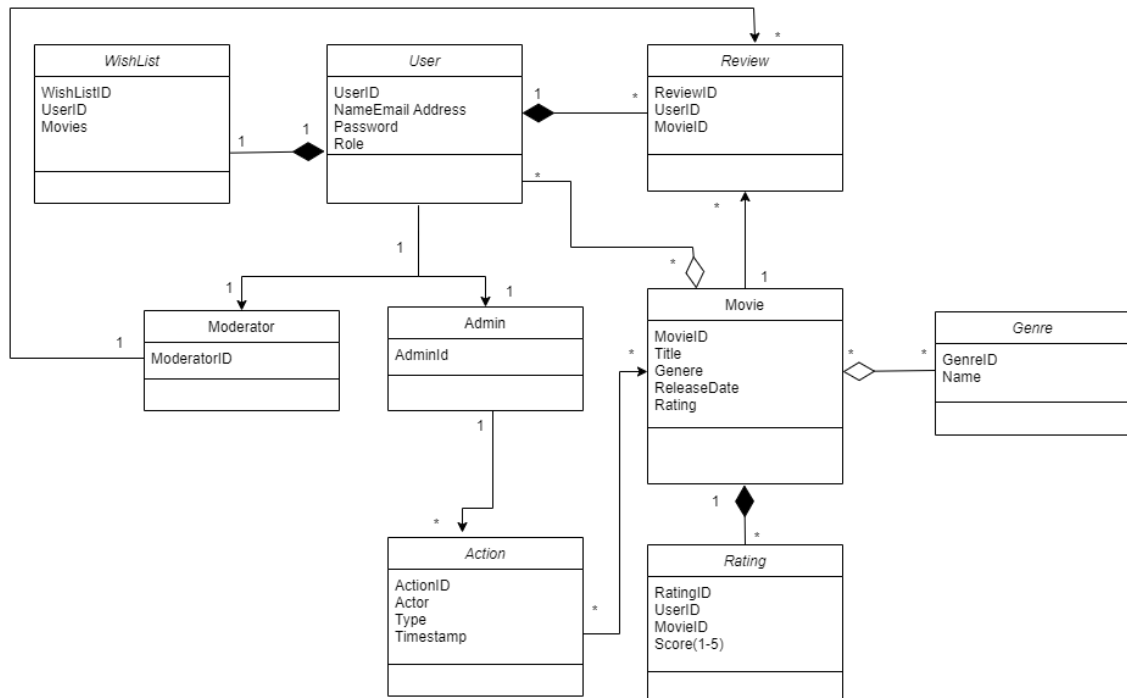


Figure 2: Domain Model before Refining

a) Generalizations

Generalization is a type of relationship where one class inherits the properties and behavior of another class. In this context:

- **Admin and Moderator** are specialized types of **User**.
 - This means both Admin and Moderator inherit attributes (e.g., UserID, Name, Email, Password, Role) and operations (e.g., Login()) from the User class.
 - Admin and Moderator also have their own additional attributes and operations, specific to their roles (e.g., AddMovie() for Admin, ReviewContent() for Moderator).

This allows reusability and avoids duplicating common properties and methods in multiple classes. It also reflects the hierarchy in the system.

b) Associations

Associations define the interactions between different classes. Here's how the associations are set:

- **User interacts with Movies:**
 - Users can view details of movies, rate movies, and add them to their wish list.
 - This relationship indicates that a User can perform actions on Movies.
- **User interacts with Reviews:**
 - Users can write, edit, and delete reviews for movies.
 - This relationship links the Review class to both User and Movies.
- **User interacts with Ratings:**
 - Users can provide numerical ratings (1–5) for movies.
 - The Rating class stores the user's score for each movie.
- **User interacts with Wish list:**
 - Users can create a wish list of movies they plan to watch.
 - Wish list belongs to a user and contains multiple movies.

c) Multiplicity

Multiplicity specifies how many instances of one class can be associated with a single instance of another class. In this system:

- **A movie can have multiple reviews, ratings, and genres:**
 - **Reviews:** A single movie can have multiple user reviews (one-to-many relationship).
 - **Ratings:** A movie can have multiple ratings from different users. The average of these ratings can be displayed as the movie's overall score.

- **Genres:** A movie can belong to one or more genres (e.g., Action, Drama).
- **A user can have multiple ratings, reviews, and movies in their wish list:**
 - **Ratings:** A single user can rate multiple movies.
 - **Reviews:** A user can write multiple reviews, one for each movie they interact with.
 - **Wish list:** A user can have a collection of movies they want to watch.

These multiplicities represent the flexibility and scalability of the system, accommodating multiple users, movies, and interactions.

9. Class Descriptions and Class Diagram

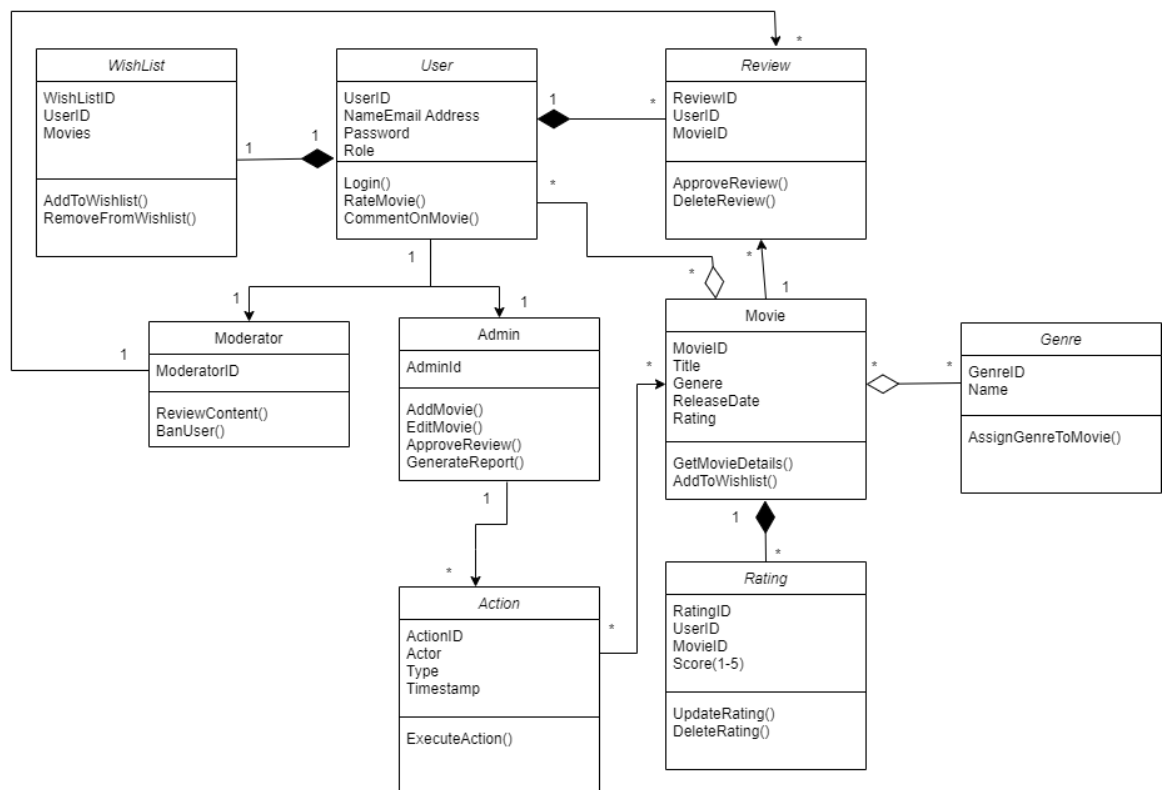


Figure 3: Refined Class Diagram

This class diagram shows a Movie Rating System, with clear relationships between entities, as well as key attributes and operations. Here is an explanation of each class and its relationships based on the diagram provided:

a) Generalization Relationships:

- The Admin and Moderator classes inherit from the User class, meaning they share common attributes and operations while having specialized functionalities like managing movies or moderating content.

- b) User Interactions:
 - A User can interact with Movies, Review, Rating, and Wishlist. They can rate movies, leave comments (reviews), and maintain a personalized list of favorite movies (wishlist).
- c) Movies and Genres:
 - Movies are linked to multiple Genres in a many-to-many relationship, allowing a movie to belong to several genres and a genre to include multiple movies.
- d) User Contributions:
 - A User can create multiple Reviews and Ratings for different movies. Each Review includes a comment with a timestamp, while Rating represents a score (e.g., 1–5).
- e) Admin and Moderator Actions:
 - Admin manages movies (e.g., adding, editing) and generates reports. Both Admin and Moderator can perform Actions like approving/rejecting reviews, banning users, and deleting inappropriate content.
- f) Wish list Management:
 - Each User has a Wish list (one-to-one relationship) that can contain multiple Movies (one-to-many relationship), allowing users to save their favorite or intended-to-watch movies.
- g) Multiplicity in Relationships:
 - A movie can have multiple Reviews and Ratings (one-to-many relationships).
 - A User can contribute multiple reviews and ratings (one-to-many).
 - Admin and Moderator can perform multiple Actions on various system entities.
- h) Action Tracking:
 - The Action class records operations performed by Admin or Moderator, including details such as the type of action (approve, reject, ban) and the timestamp.

System Sequence Diagram

10. Sequence Diagram

The sequence diagram, also known as an event diagram, shows how messages move through the system. It facilitates the visualization of various dynamic situations. It depicts any two lifelines' communication as a time-ordered series of events, indicating that these lifelines participated during the runtime. In UML, a vertical bar represents the lifeline, while a vertical dotted line that crosses the bottom of the page represents the message flow. It includes both branching and iterations.

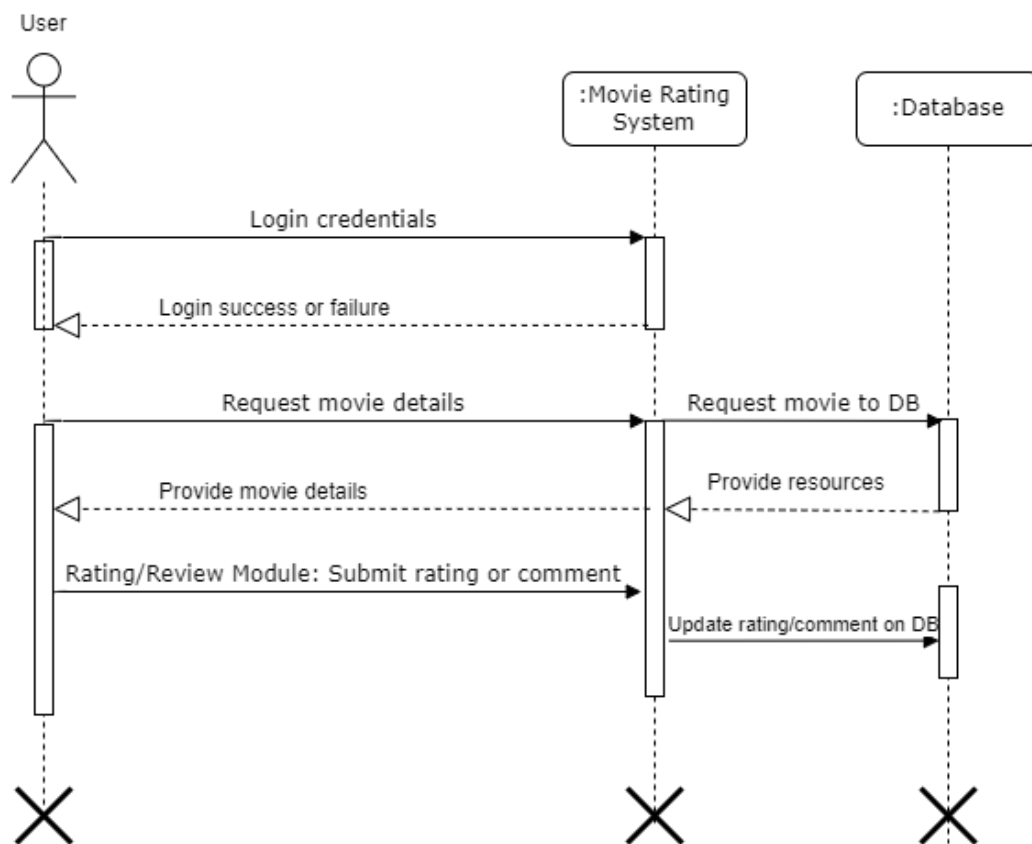


Figure 4: Sequence Diagram in User Perspective

The above Sequence Diagram represents the interaction between a User, a Movie Rating System, and a Database when a user wants to rate or review a film. Here is the breakdown of this Sequence Diagram:

a) Login:

- The user initiates the process by providing their login credentials to the Movie Rating System.
- The system validates the credentials and sends a message back to the user indicating login success or failure.

b) Movie Details Request:

- If the login is successful, the user requests details about a specific movie.
- The Movie Rating System forwards this request to the Database.

c) Movie Details Retrieval:

- The Database processes the request, retrieves the relevant movie details, and sends them back to the Movie Rating System.
- The system then provides the movie details to the user.

d) Rating/Review Submission:

- The user uses the "Rating/Review Module" to submit their rating or comment for the movie.
- The Movie Rating System receives this information and sends it to the Database for storage.

e) Database Update:

- The Database updates its records to include the new rating or comment.

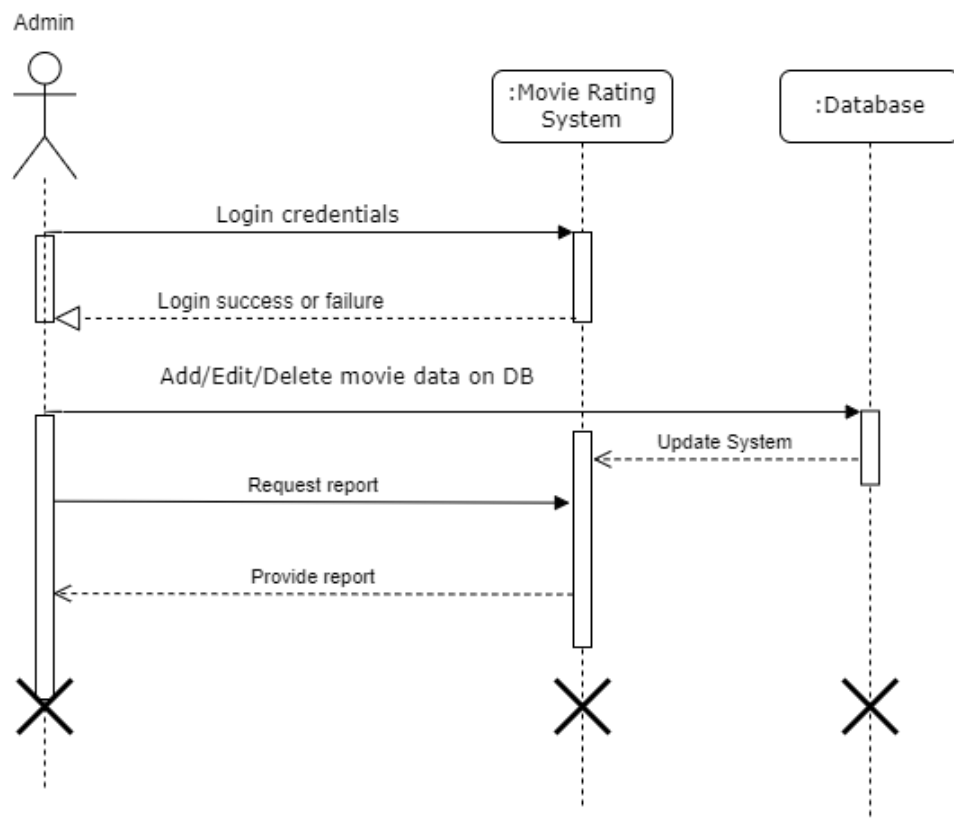


Figure 5: Sequence Diagram in Admin Perspective

The above Sequence Diagram represents the interaction between Admin, a Movie Rating System, and a Database when a user wants to rate or review a film. Here is the breakdown of this Sequence Diagram:

a) Data Management:

- After successful login, the Admin has the authority to add, edit, or delete movie data in the system.
- The system sends a message to the Database to update the movie data accordingly.

b) Report Request:

- The Admin requests a report from the system.

c) Report Generation and Delivery:

- The system processes the report request, generates the report, and sends it back to the Admin.

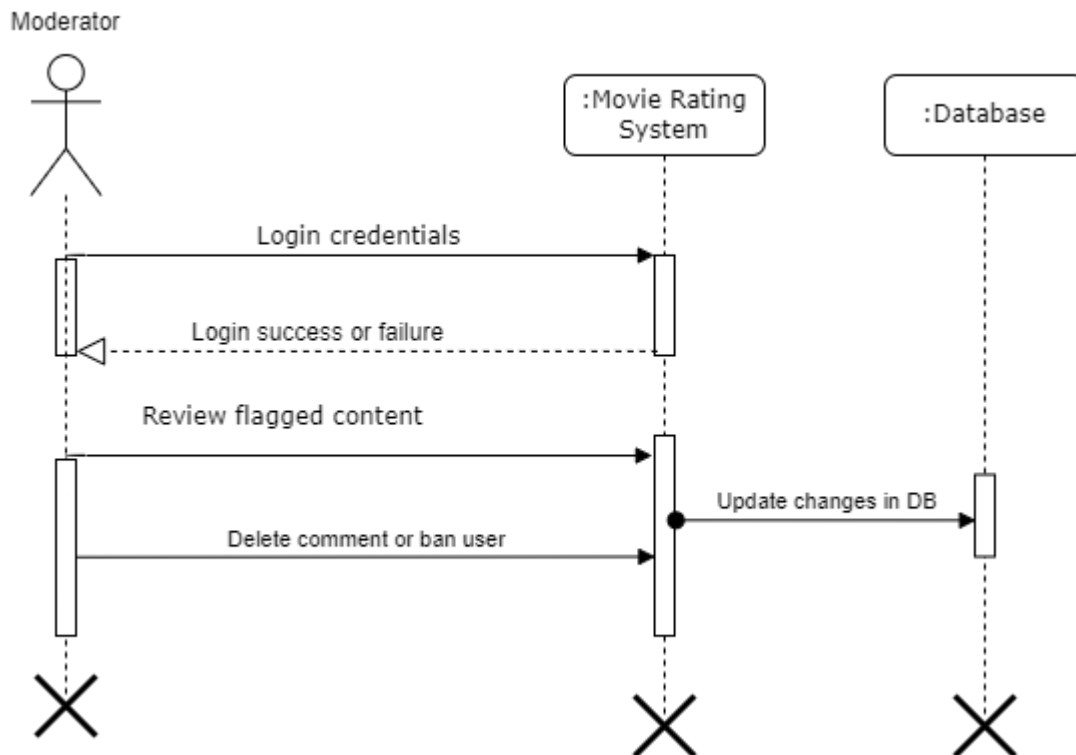


Figure 6: Sequence Diagram in Moderator Perspective

The above Sequence Diagram represents the interaction between Moderator, a Movie Rating System, and a Database within the context of content moderation. Here is the breakdown of this Sequence Diagram:

a) Review Flagged Content:

- After successful login, the Moderator reviews flagged content (e.g., comments, reviews) within the system.

b) Action on Flagged Content:

- Based on the review, the Moderator takes action. This could involve:
- Deleting the flagged comment.
- Banning the user who posted the flagged content.

c) Database Update:

- The system sends a message to the Database to update the relevant records (e.g., delete the comment, ban the user) based on the Moderator's actions.

Activity Diagram of Role-Specific User Interactions and System Workflows

11. Activity Diagram

The activity diagram is used to show the flow of control within a system rather than the implementation. It represents both concurrent and sequential activities.

The activity diagram helps in visualizing the flow from one activity to the next. It focused on the state of flow and the order in which it occurs. The flow can be sequential, branched, or concurrent, and the activity diagram has fork, join, and other features to deal with such flows.

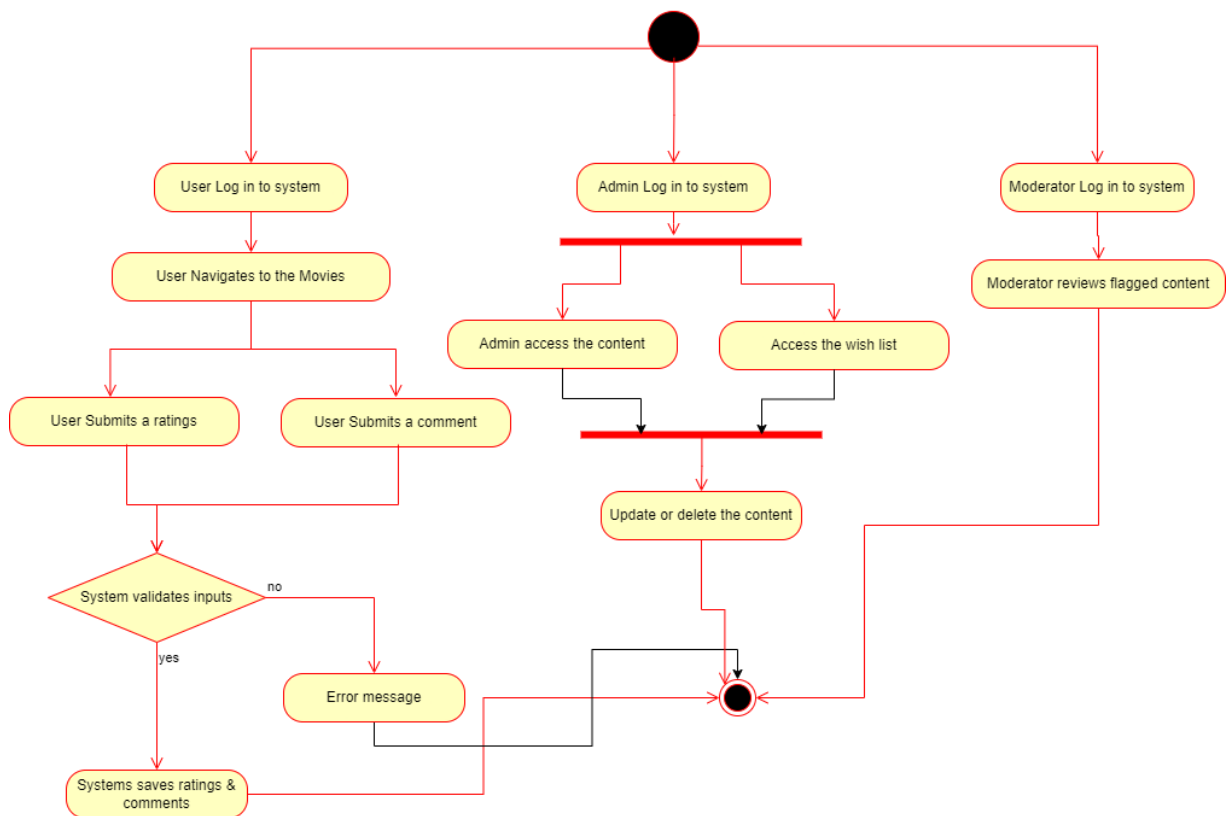



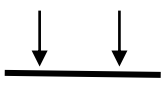
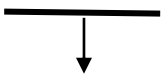




Figure 7; Activity Diagram

The activity diagram represents the workflows for three actors: User, Admin, and Moderator. It illustrates how users log in, navigate to movies, and perform actions like submitting ratings or comments, which are validated by the system. Admins and moderators log in to manage content, review flagged items, and take appropriate actions such as updating, deleting, or banning. Forks and joins demonstrate parallel workflows for these roles, ensuring the system efficiently handles multiple tasks concurrently.

Table 1: Activity Diagram Workflow

Notation	Symbol	Workflow Explanation
Start Node		Represents the starting point of the workflow. All actors (User, Admin, and Moderator) begin their respective tasks here.
Action Node		Denotes the specific action performed by the actor, such as logging in, submitting ratings/comments, or reviewing content.
Decision Node		Indicates a decision point, such as validation of inputs (ratings/comments). Workflow splits based on a condition.
Fork Node		Represents a parallel split in the workflow. Multiple actions (e.g., Admin and Moderator tasks) occur simultaneously.
Join Node		Combines multiple parallel workflows back into a single sequence.
Connector/ Arrow		Represents the flow of control or data between nodes.
End Node		Indicates the termination of the workflow for an actor.

UML State Diagrams and Modeling

12. State Diagram

A state diagram is a behavioral diagram that models an object's dynamic behavior. It visually represents the various states that an object can be in throughout its lifetime, as well as the transitions between those states caused by events. State diagrams can help developers understand object behavior, identify potential issues, and improve the design and maintainability of software systems.

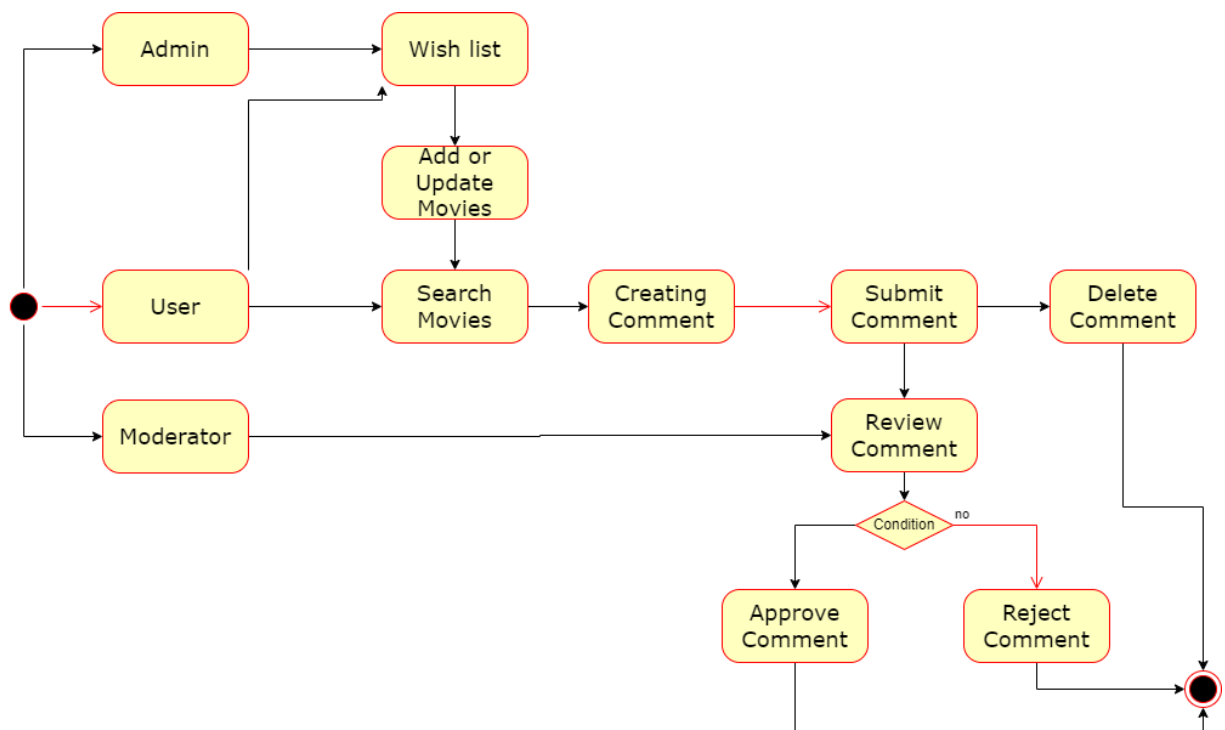


Figure 8: State Diagram

Here's a breakdown of above State diagram:

- Initial State:** The diagram begins with an initial state, represented by a filled circle. This indicates the starting point of the system.
- User Interaction:** From the initial state, the system transitions to the "User" state. This is where the user interacts with the system.
- Search Movies:** The user can then transition to the "Search Movies" state to find movies of interest.
- Creating Comment:** If the user finds a movie they like, they can transition to the "Creating Comment" state to write a comment or review.
- Submit Comment:** Once the comment is written, the user transitions to the "Submit Comment" state to send their comment to the system.

- f) **Moderator Review:** After submission, the comment enters the "Review Comment" state where a moderator reviews it.
- g) **Decision Point:** The moderator then evaluates the comment. If the comment is appropriate and meets the guidelines, the system transitions to the "Approve Comment" state. If the comment is deemed inappropriate, the system transitions to the "Reject Comment" state.
- h) **Comment Disposition:** If the comment is approved, it is likely integrated into the system (not explicitly shown in this diagram). If rejected, the comment may be deleted or hidden from public view.
- i) **Admin Actions:** The diagram also includes states for the "Admin" role. These states allow the administrator to:
 - Add or update movies.
 - Manage user wish lists.

13. Deployment Diagram

The deployment diagram shows the physical hardware on which the program will be executed. It depicts the static deployment view of a system. It is about the nodes and their relationships. It determines how software is deployed on the hardware. It connects the software architecture generated during design to the physical system architecture, where the software will run as a node. Because it involves multiple nodes, the relationship is demonstrated using communication routes.

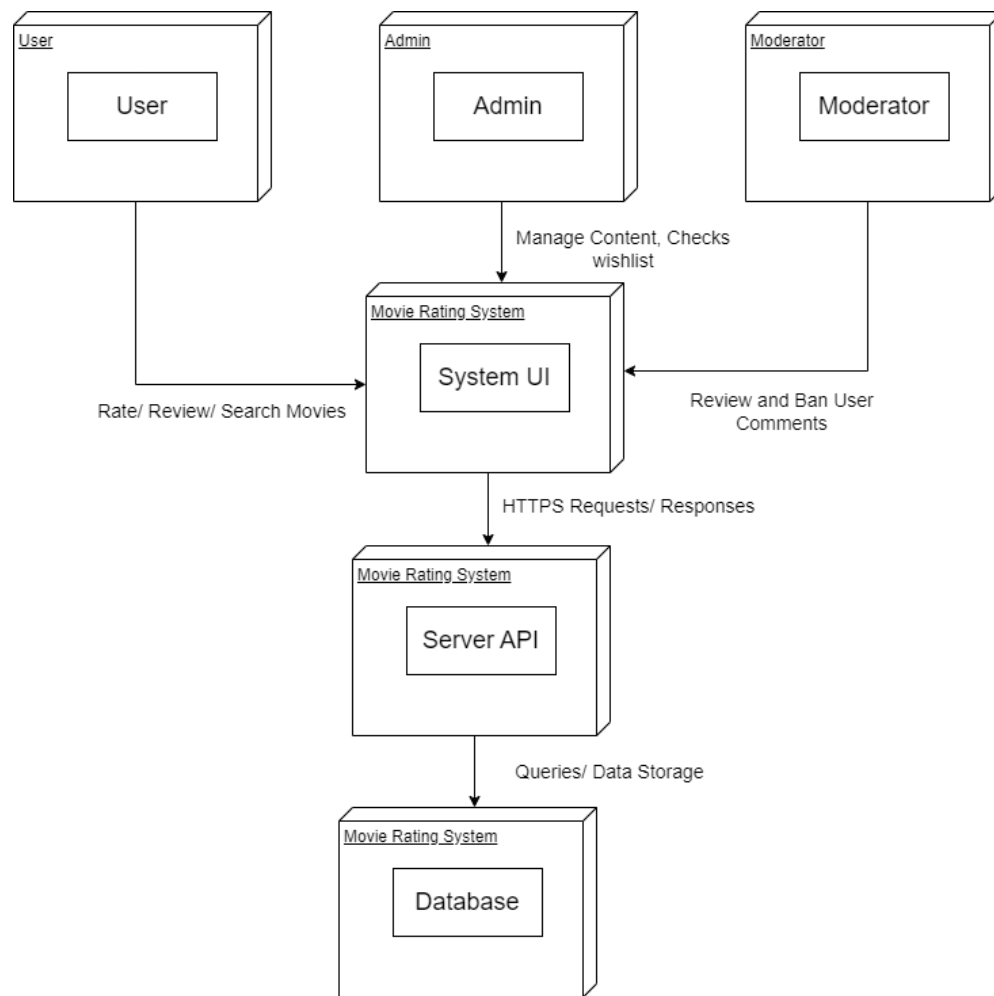


Figure 9: Deployment Diagram

The diagram shows the deployment architecture of a Movie Rating System. It highlights three key roles: user, administrator, and moderator. The User uses the System UI to rate, review, and search for movies. The Admin has the ability to manage content, verify wish lists, and possibly execute more administrative activities. The Moderator is responsible for monitoring and blocking user comments as appropriate. These roles interact with the System UI, which communicates with the Server API using HTTPS requests and answers. The Server API then communicates with the Database, storing and retrieving movie data, user ratings, reviews, and other pertinent information. This layered architecture allows for clear separation of concerns, efficient data management, and a scalable system to accommodate growing user needs and data volumes.