**Movie review platform**

**Adinai Iuldasheva s20218**

1. **User requirements**

We want to have a platform where users can review movies and create their own collection of favorite movies.

Our main entity is a basic user who has created a profile on the platform and has information containing username, email and password.

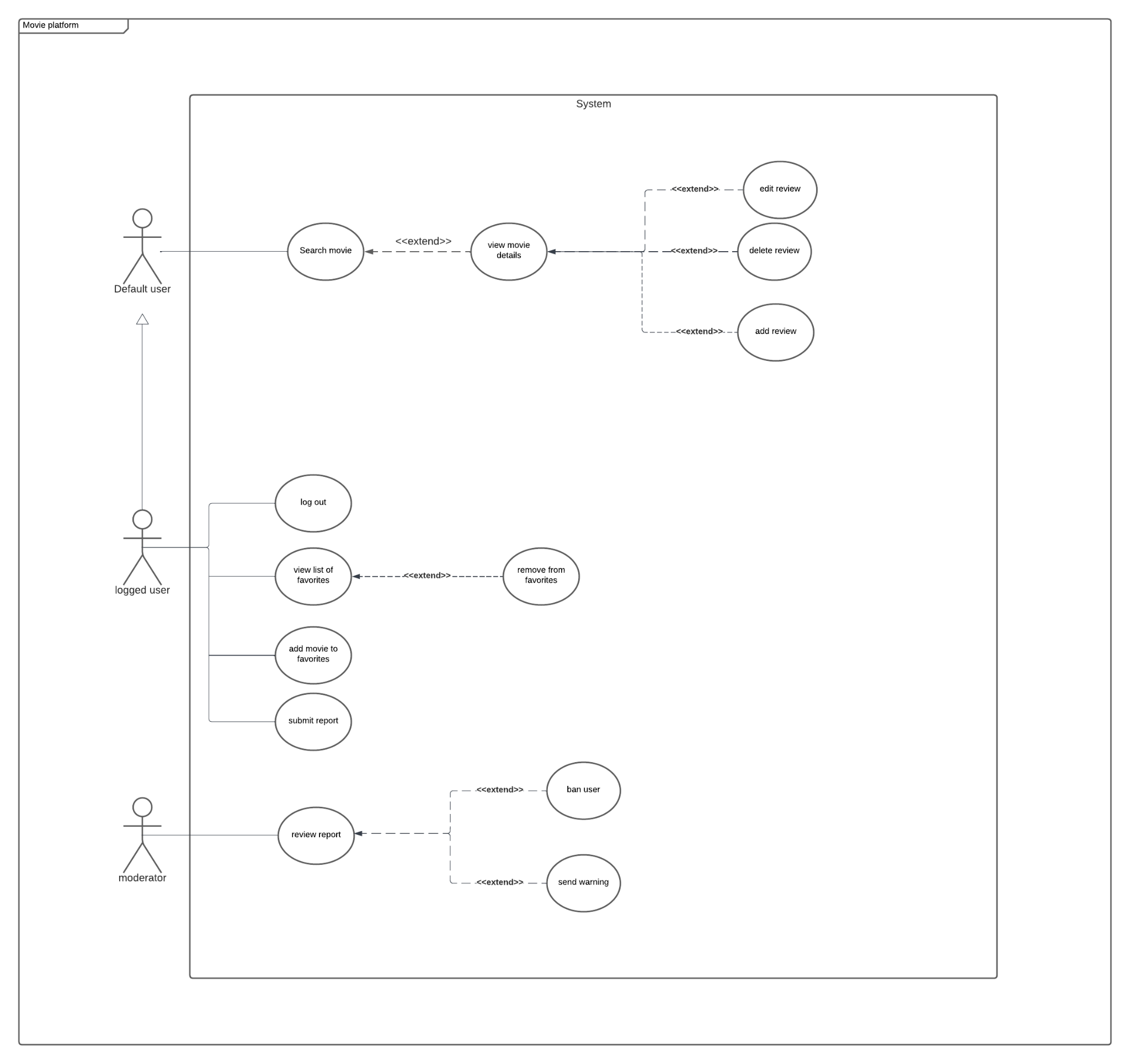
We expect to have two types of users: standard user and moderator of the website. Standard user is a logged in user, who can add reviews to the movies, edit own reviews or delete them later. The reviews will consist of the rating, taken from the specified set of grades (1-5), and a comment. Standard users will also have the ability to add a movie to the favorites list, and remove it from there. Apart from that standard users are expected to be able to report certain behavior on the website as well as get reported. In the report the standard user has to provide the id of the reported user and a short description of the issue. Those reports will later be reviewed by the second type of users: moderator. Based on the report moderator decides what to do: he either can ban the reported user or issue a warning.

Important note: user with the same username and email can’t be moderator and standard user at the same time. If a particular user has been granted a moderator role, he/she has to first change their profile type in order to have access to functionalities of a standard user. And vice versa. In short one username and email - one profile type.

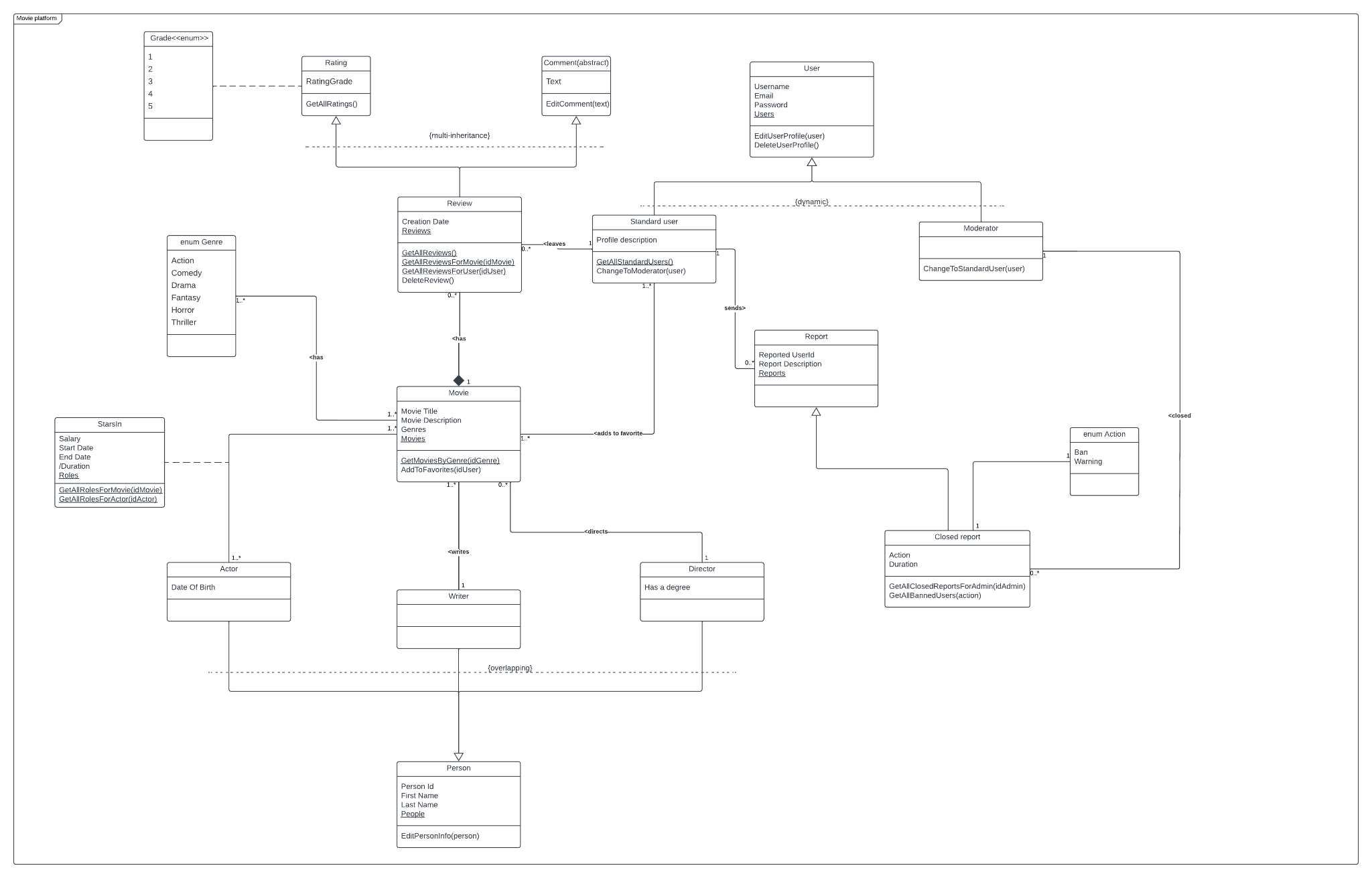
Another important entity we would like to see is movie, which will contain the most important information about the selected movie like the title, the set of roles and actors playing those roles, the director and writer and genre(s) also taken from a specified set of genres (Action, Comedy, Drama, Fantasy, Horror, Thriller). Actors, directors and writers are based on one main entity - person, apart from some additional information for each of these roles, e.g. date of birth, degree.

Important note: one person can be the director, writer and actor of the movie at the same time.Or any other combination of those roles.

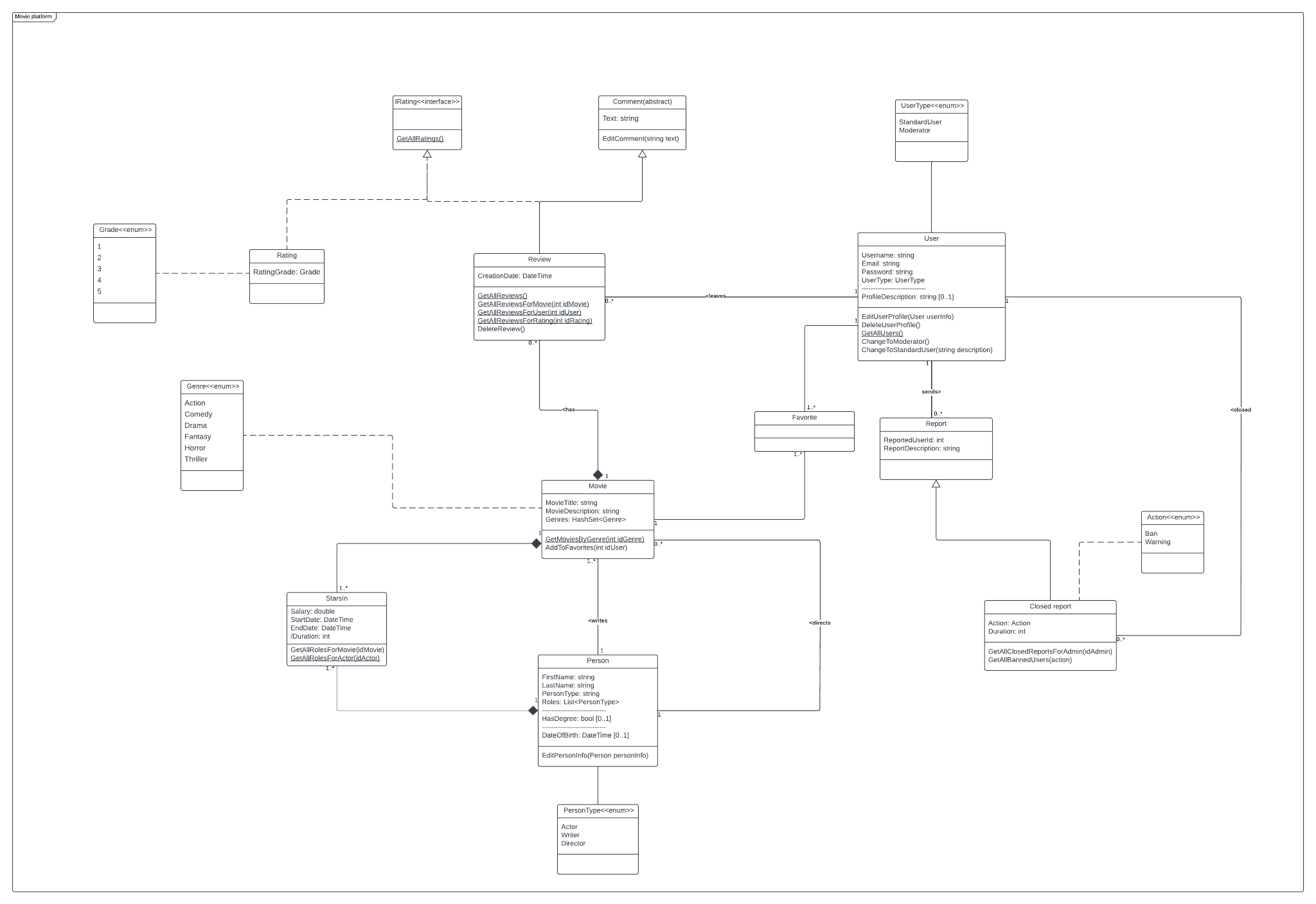
1. **Use case diagram**

****

1. **Analytical class diagram**

****

1. **Design class diagram**

****

1. **Use case description**

**Add review**

* 1. **Actors**

Logged in user

* 1. **Purpose and context**

Logged in user wants to review the movie

* 1. **Dependences**
     1. **Included use-cases**

None

* + 1. **Extended use-cases**

None

* 1. **Assumptions and pre-conditions**

User is logged in and viewing the list of movies

* 1. **Basic flow of event**

1. User picks a movie from the list
2. System shows movie details including all previous reviews for this movie
3. User clicks “Add review” button
4. System displays submit form
5. User picks the grade (1-5)
6. User writes the review
7. User clicks “Submit” button
8. System shows “Your review has been saved” message
   1. **Alternative flows of events**
      1. **User didn’t add rating**

1. System shows “Rating is required” message

2. User is returned to the step 5 of basic flow

* + 1. **User didn’t write review**

1. System shows “Review is required” message

2. User is returned to the step 6 of basic flow

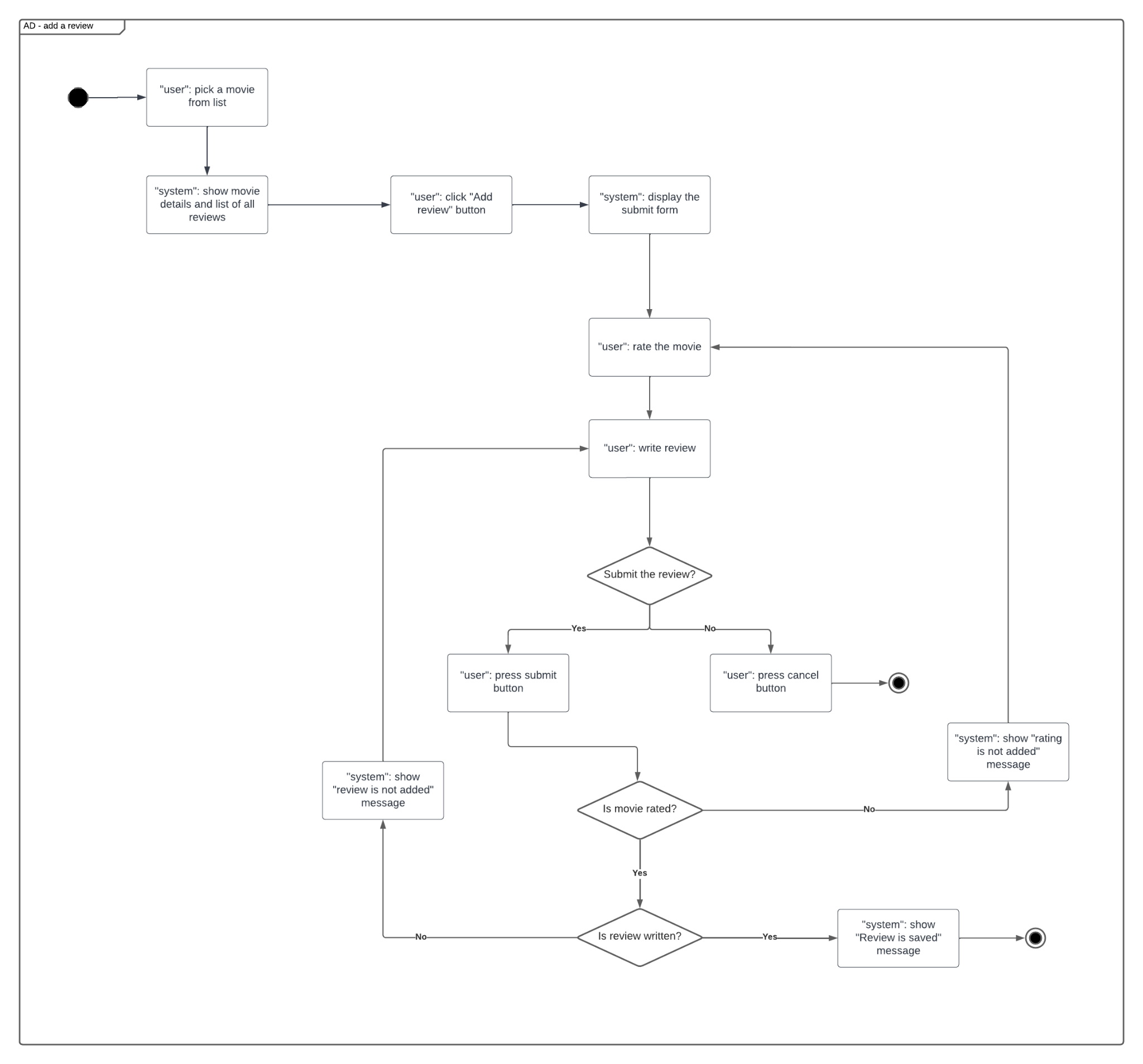
* + 1. **User clicks “Cancel” button**

1. End of flow

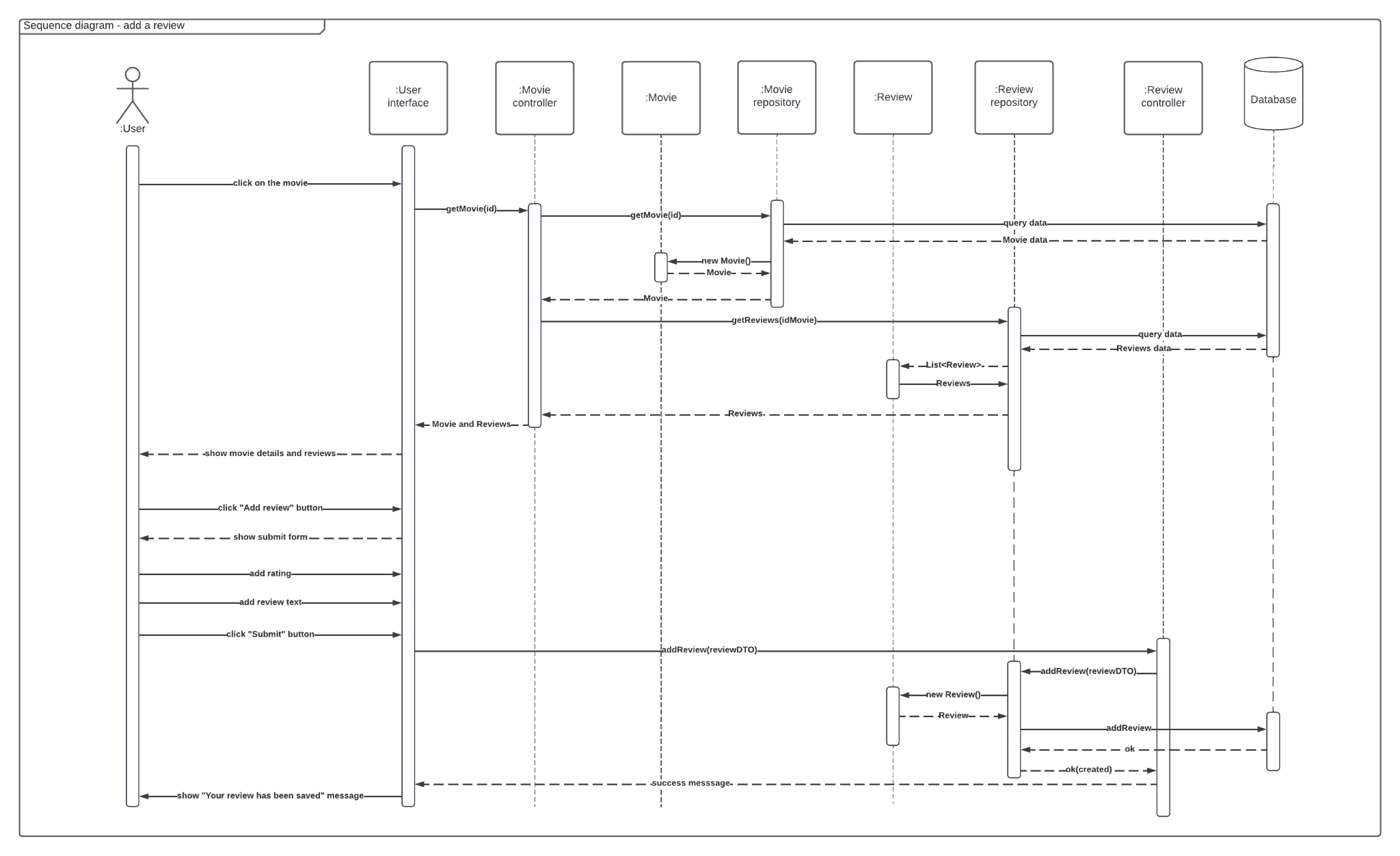
**Post condition**

The review for the movie has been added and user returned to the page with movie details

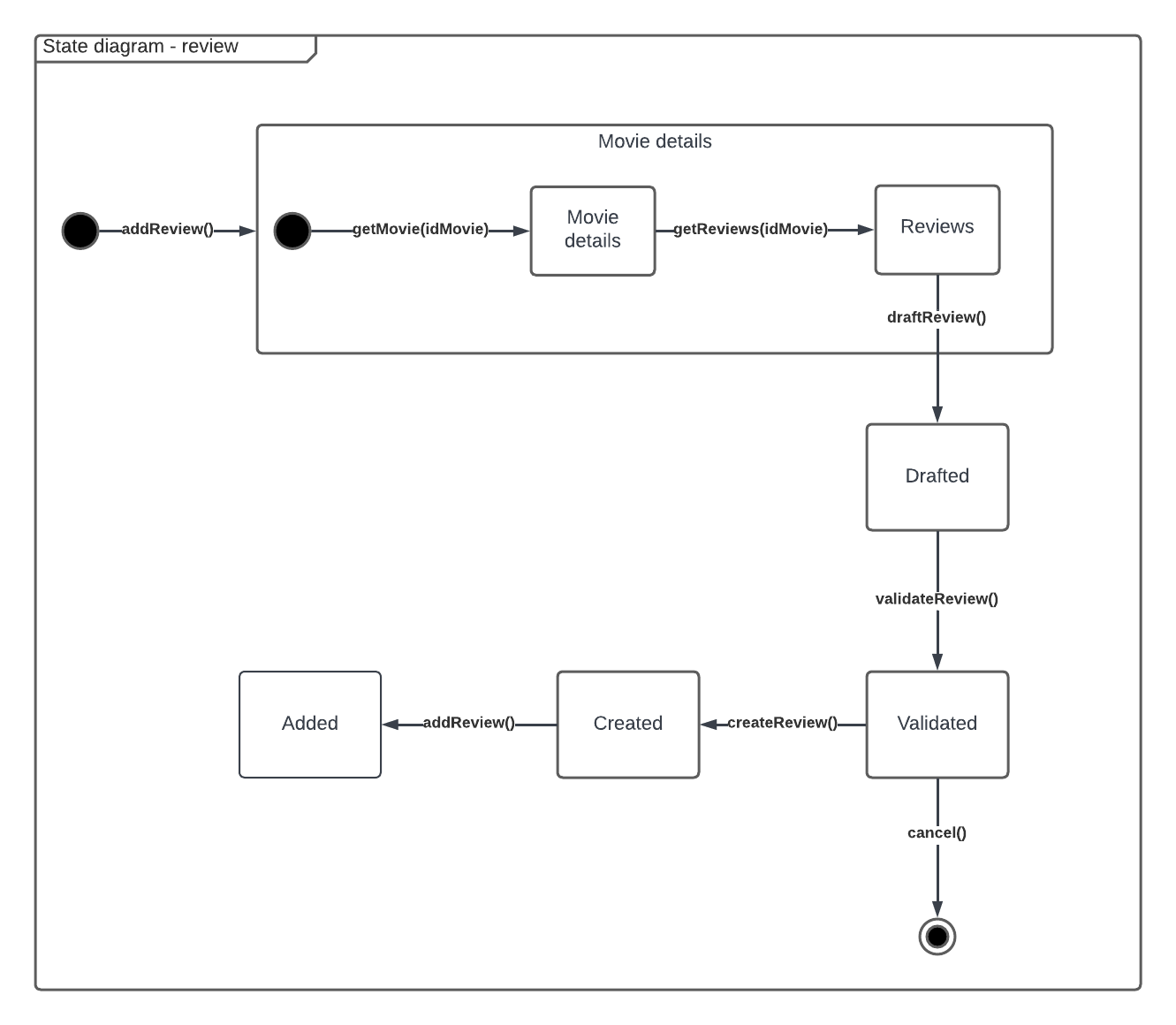
1. **Activity diagram**



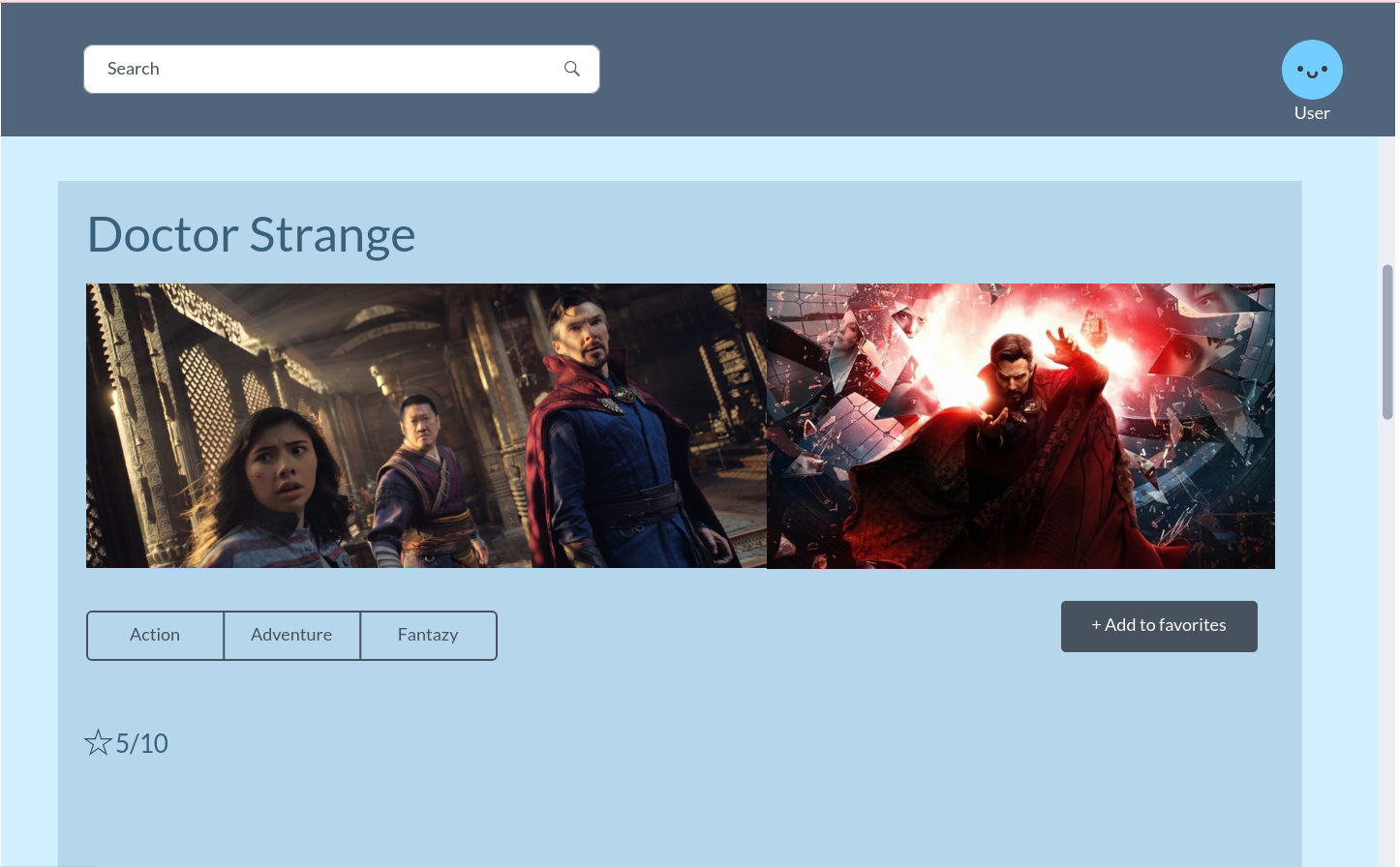
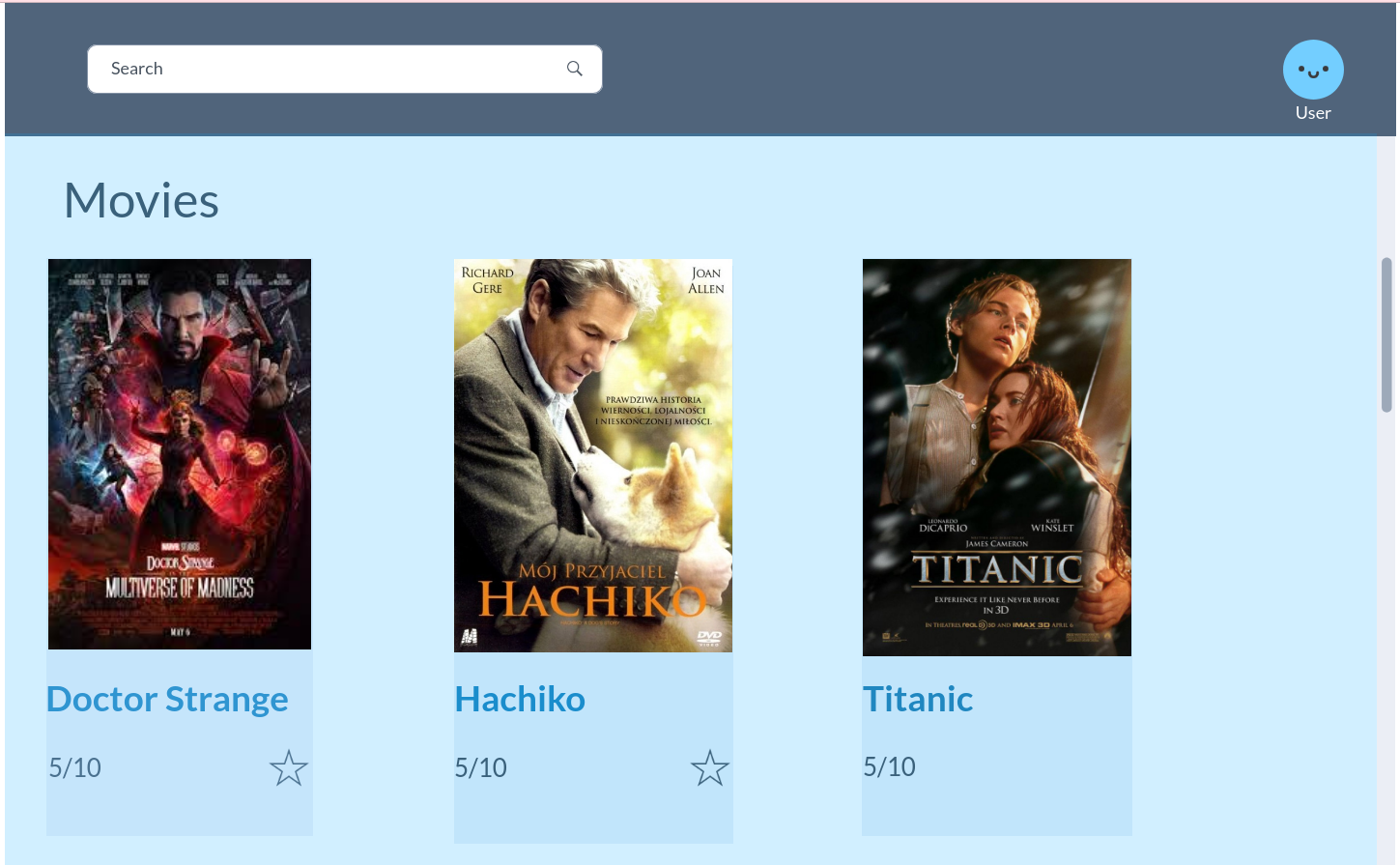
1. **Interaction (sequence) diagram**

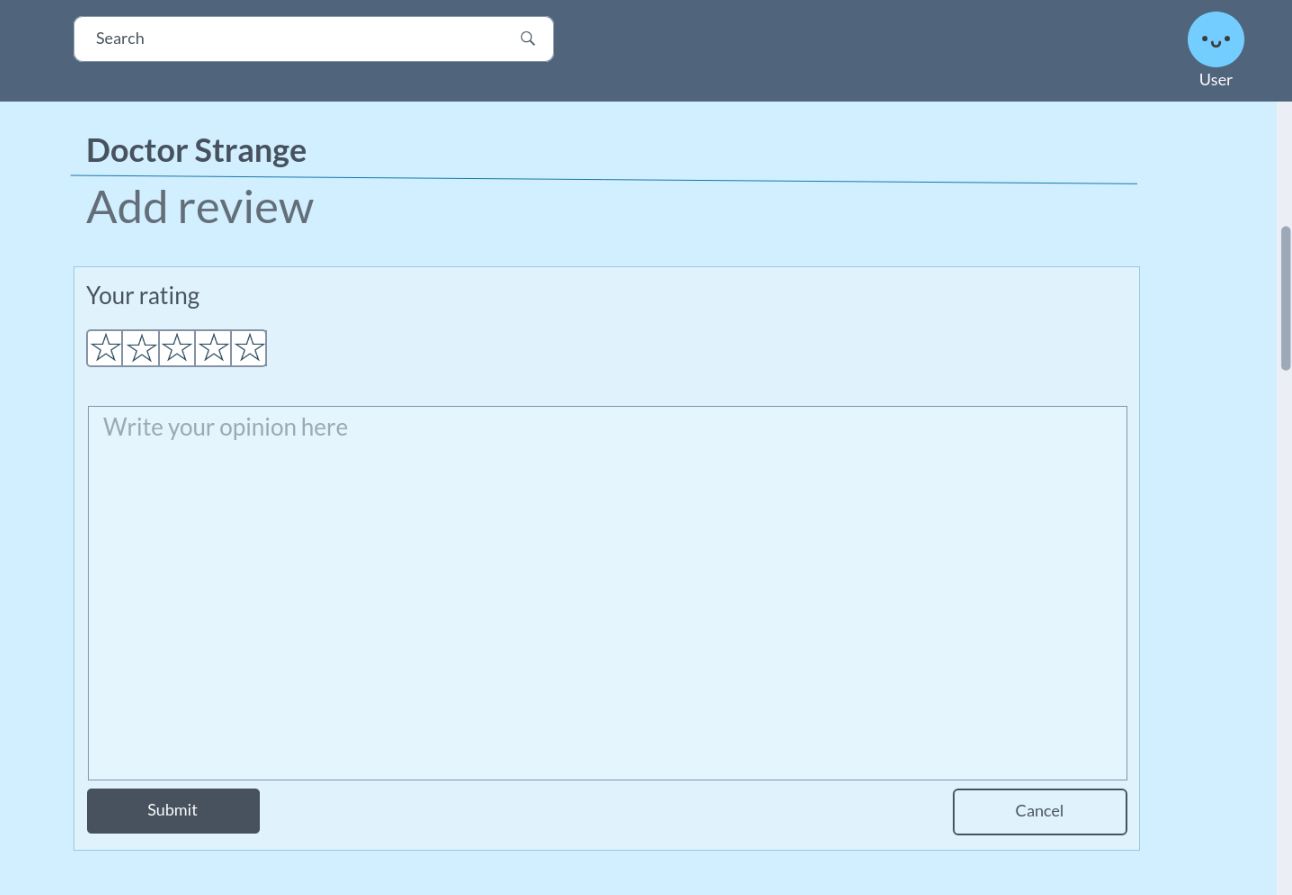
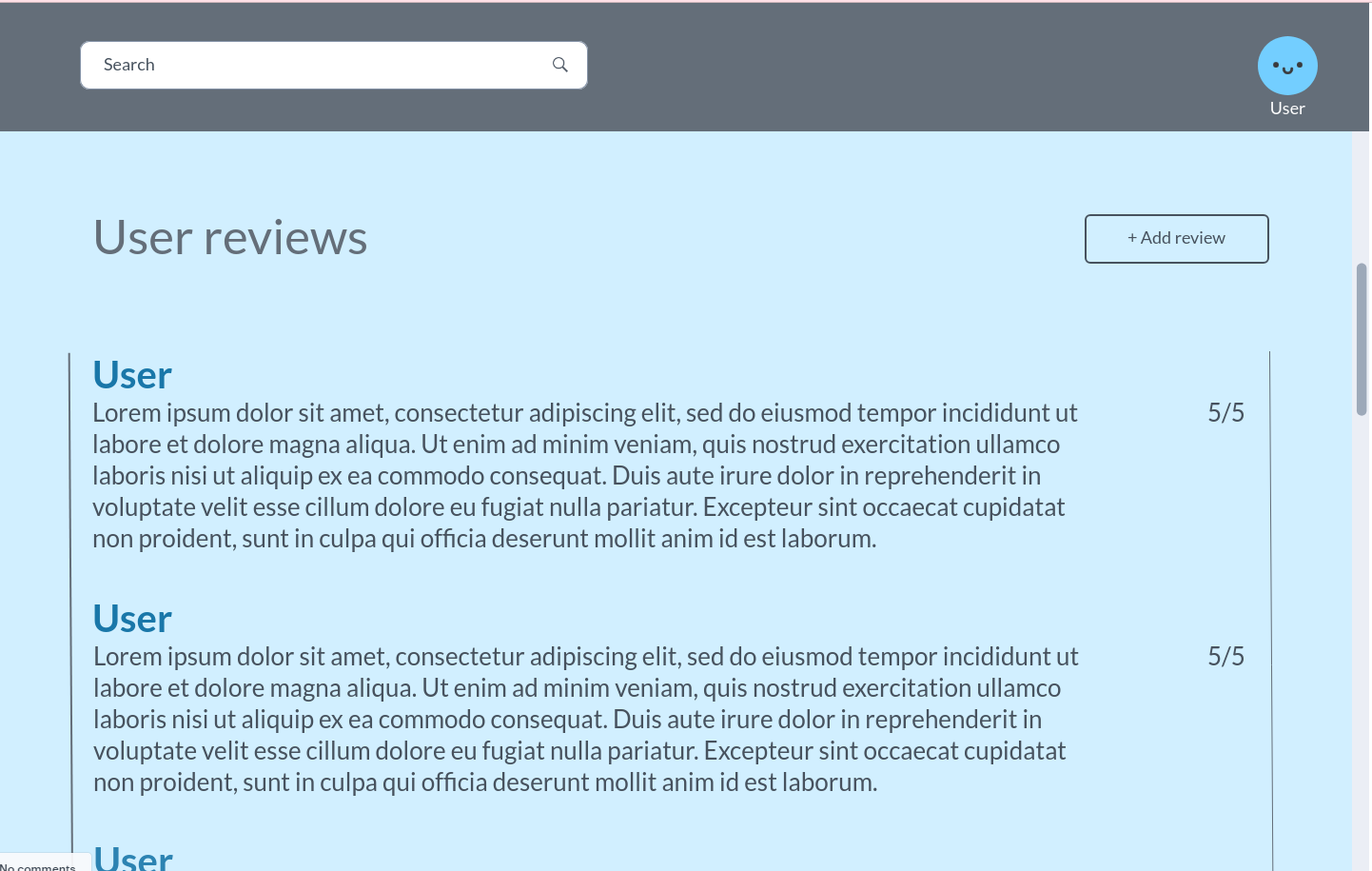
****

1. **State diagram**

****

1. **GUI design**

****

****

1. **Discussion of design decisions**

Dynamic inheritance between user, standard user and moderator was used because it is required that a user can’t have two user types at once, but be able to change the profile type.

To implement the dynamic inheritance a flattening the class hierarchy will be used. It means all the attributes and methods from inheriting classes will be transferred to the one base class User. This base class will contain one user type enum. There are going to be methods like ChangeToStandardUser and ChangeToModerator which will allow us to change the role of the user. This method of implementation of inheritance is used because those classes don’t have much difference and can be easily stored in one table. Also it will be easier to fetch the data from one table to see if a certain user with a different role already exists.

Overlapping inheritance between person, director, actor and writer classes was used, because it is required that one person can have different responsibilities in the same movie, i.e. one person can be the one who directed and wrote the movie, or directed and acted in the movie etc.

The same flattening the class hierarchy approach will be used for the overlapping inheritance for more or less the same reason as for dynamic. It will be more convenient for getting the data from one base table to see all the person roles assigned to the particular person without joining several tables. The only difference is that in an overlapping inheritance base class Person will contain a list of PersonType enums since one person can have multiple responsibilities in one movie.

Multi-inheritance between review, rating and comment classes was used to add more structure to the review class, so it will be easier to modify one of the parts without affecting other classes(change some constraints, modify the rating system). Also it might be useful in later development where we would like to add separate functionality like adding only rating.

It will be implemented by adding interface IRating, which will be inherited by both review and rating classes. Comment class will be abstract.

The program will be written on c# using MVC pattern and Entity framework code-first approach. For the database SQL Server will be used.

The program will have controllers allowing to exchange requests and responses between server and client. There are going to be repository classes as well, which will contain all necessary methods to interact with the database based on the requests from the client.

To persist the data, we will generate a sql file based on the migration with the starting state of the database, and everytime the state of the database changes (insert, delete, update operations) generated queries are going to be logged into this file.