



**UNIVERSITATEA  
TEHNICĂ  
DIN CLUJ-NAPOCA**

---

**Cinema Web Application**

*Software Design*

---

Author: Radu Vlad

Group: 30236

FACULTATEA DE AUTOMATICA  
SI CALCULATOARE

29 03 2022

# Contents

<b>1</b>	<b>Deliverable 1</b>	<b>2</b>
1.1	Project Specification	2
1.2	Functional Requirements	2
1.3	Use Case Model 1	2
1.3.1	Use Case Identification	2
1.3.2	UML Use Case Diagrams	4
1.4	Supplementary Specification	4
1.4.1	Non-functional Requirements	4
1.4.2	Design Constraints	4
1.5	Glossary	5
<b>2</b>	<b>Deliverable 2</b>	<b>5</b>
2.1	Domain Model	5
2.2	Architectural Design	6
2.2.1	Conceptual Architecture	6
2.2.2	Package Design	7
2.2.3	Component and Deployment Diagram	8
<b>3</b>	<b>Deliverable 3</b>	<b>8</b>
3.1	Design Model	8
3.1.1	Dynamic Behavior	8
3.1.2	Class Diagram	8
3.2	Data Model	9
<b>4</b>	<b>System Testing</b>	<b>9</b>
<b>5</b>	<b>Future Improvements</b>	<b>9</b>
<b>6</b>	<b>Conclusion</b>	<b>9</b>
<b>7</b>	<b>Bibilography</b>	<b>9</b>

# 1 Deliverable 1

## 1.1 Project Specification

My project is an online cinema application designed to allow users to browse movies, view ratings and reviews from others, and create their own reviews and personal favorites list. The application is similar to IMDb in its functionality and user experience.

## 1.2 Functional Requirements

### **User Registration and Login:**

The application should provide a user registration process where users can create an account. Users should be able to log in to the application using their registered email and password. The application should have proper security measures in place to protect user data.

### **Movie Management:**

The application should allow administrators to add, delete, and update movie details such as the title, release date, genre, and plot summary. A movie can have multiple actors, so the application should allow administrators to associate one or more actors with a movie.

### **Actor Management:**

The application should allow administrators to add, delete, and update actor details such as name, date of birth, and nationality. The application should allow users to view all movies in which a particular actor has acted.

### **Rating Management:**

The application should allow users to rate a movie on a scale of 1 to 5. Users should be able to see the average rating of a movie based on all the ratings submitted by different users. The application should allow administrators to delete inappropriate or spam ratings.

### **User Profile:**

The application should provide users with a profile page that displays their personal information, including their name and email address. Users should be able to update their profile information, change their password, and delete their account.

## 1.3 Use Case Model 1

### 1.3.1 Use Case Identification

1. Search for movies by multiple criteria
  - Use-Case: Search for movies
  - Level: User Goal
  - Primary Actor: Regular User
  - Main success scenario: The user searches for movies using multiple criteria and the application returns a list of movies that match the search criteria.
  - Extensions: If the search fails to return any results, the application throws a custom exception to inform the user.
2. Search for actors by multiple criteria
  - Use-Case: Search for actors

- Level: User Goal
  - Primary Actor: Regular User
  - Main success scenario: The user searches for actors using multiple criteria and the application returns a list of actors that match the search criteria.
  - Extensions: If the search fails to return any results, the application throws a custom exception to inform the user.
3. Add a movie to the database
- Use-Case: Add a movie
  - Level: User Goal
  - Primary Actor: Administrator
  - Main success scenario: The administrator adds a new movie to the database, including details such as title, director, cast, and genre.
  - Extensions: If the movie already exists in the database, the application displays an error message and does not allow the administrator to add it again.
4. Delete a movie from the database
- Use-Case: Delete a movie
  - Level: User Goal
  - Primary Actor: Administrator
  - Main success scenario: The administrator deletes a movie from the database, removing all associated data such as ratings and reviews.
  - Extensions: If the movie does not exist in the database, the application displays an error message and does not allow the administrator to delete it.
5. View movie details and reviews
- Use-Case: View movie details
  - Level: User Goal
  - Primary Actor: Regular User
  - Main success scenario: The user selects a movie from the search results and views its details, including title, director, cast, genre, release date, and average rating. The user can also read reviews from other users.
  - Extensions: If there are no reviews for the movie, the application displays a message indicating that there are no reviews available.

### 1.3.2 UML Use Case Diagrams

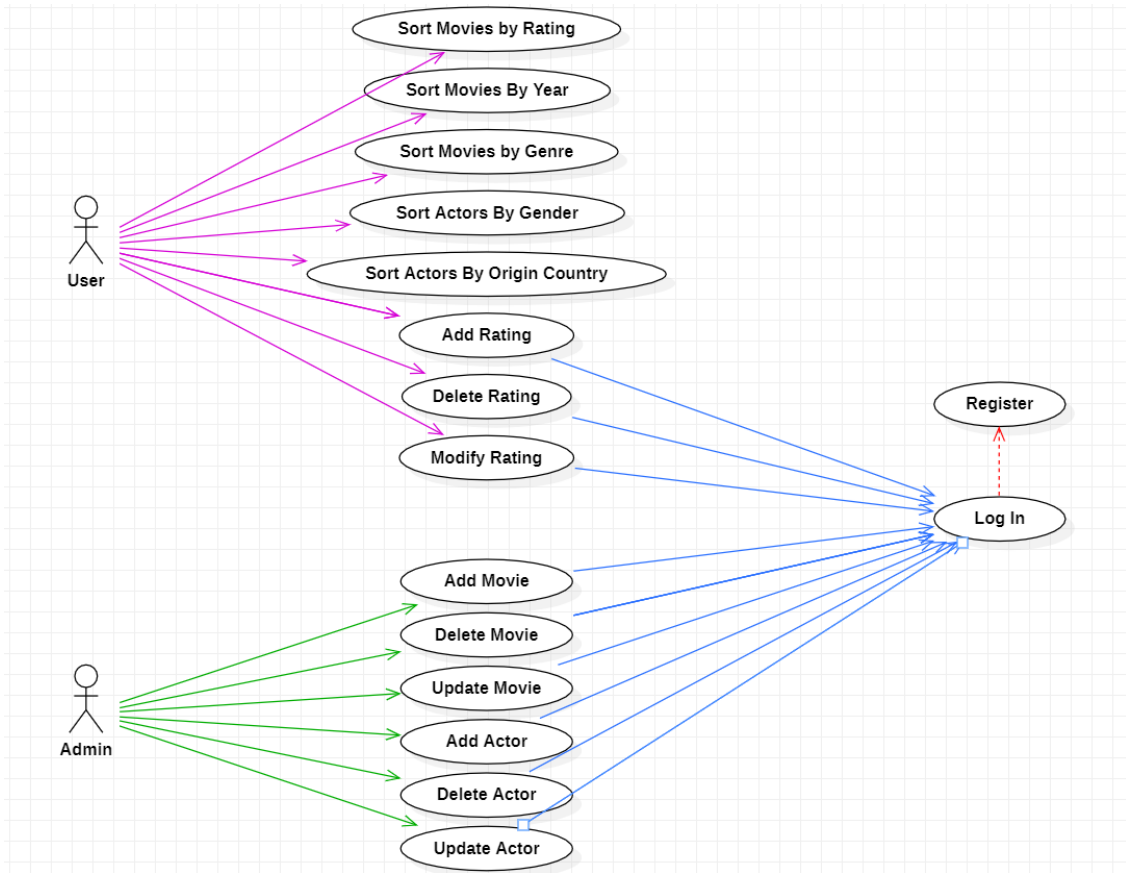


Figure 1: UML Diagram

## 1.4 Supplementary Specification

### 1.4.1 Non-functional Requirements

**Performance:** The application should load quickly and respond to user actions within a reasonable amount of time to provide a smooth user experience.

**Security:** The application should implement proper security measures to protect user data, including password hashing, encryption, and secure communication protocols.

**Scalability:** The application should be designed to handle a large number of users and data without compromising its performance or functionality.

**Accessibility:** The application should be designed with accessibility in mind, ensuring that users with disabilities can use the application with assistive technologies such as screen readers or voice commands.

### 1.4.2 Design Constraints

The application is developed using Java programming language and Spring framework, which is integrated with a MySQL database. The data is fetched and tested using Postman. The application runs on localhost on port 8081. The user interface is created using Angular, ensuring a modern and user-friendly interface. The data security measures are implemented using Spring Boot, providing a secure and reliable platform for data management.

## 1.5 Glossary

1. Movie - a feature-length film with a title, release date, and genre.
2. Actor - a person who portrays a character in a movie.
3. Rating - a numerical score (1-5) given by a user to a movie.
4. User - a person who uses the cinema web application.
5. Registration - the process of creating a new user account.
6. Login - the process of accessing the cinema web application with credentials (email and password).
7. Administrator - a user with privileged access to the cinema web application.
8. Database - a structured collection of data that is stored and managed by the cinema web application.
9. API - an interface that allows communication between different software applications.
10. Front-end - the user-facing part of the cinema web application, created using Angular..
11. Back-end - the server-side part of the cinema web application, implemented using Java and Spring.

## 2 Deliverable 2

### 2.1 Domain Model

The cinema application's domain model is a conceptual representation of the key classes and relationships in the application's domain. The model includes six essential classes: Movie, Actor, Rating, Review, User, and Admin, which are crucial in defining the application's functionality.

The Movie class is responsible for representing a movie within the application, and it has the following attributes: `movieId`, `title`, `description`, `releaseDate`, `genre`, and a list of actors who played in it. This class's properties provide all the necessary information about a movie, such as its basic details, genre, and the actors who played in it.

The Actor class represents an actor and has the attributes of `actorId`, `name`, `age`, `nationality`, and a list of movies in which the actor played. This class helps to define the actors' profiles in the application, such as their personal information and the movies they have starred in.

The Rating class represents a rating given to a movie by a user and has the attributes of `ratingId`, `value`, `movie`, and `user`. This class helps users rate movies they have watched and provides a mechanism to track the rating given by a user.

The Review class represents a review made by a user and has the attributes of `reviewId`, `content`, `movie`, and `user`. This class helps users to review movies they have watched and provides a mechanism to track the reviews given by a user.

The User class represents a user of the application and has the attributes of `userId`, `name`, `email`, `password`, and a list of reviews and ratings made by the user. This class helps to define a user's profile and provides all the necessary attributes to enable a user to log in and use the application's features.

The Admin class represents an admin user of the application and has the same attributes as the User class. This class helps to define the admin user's profile, which has the same attributes as a regular user, but with elevated privileges to perform administrative tasks.

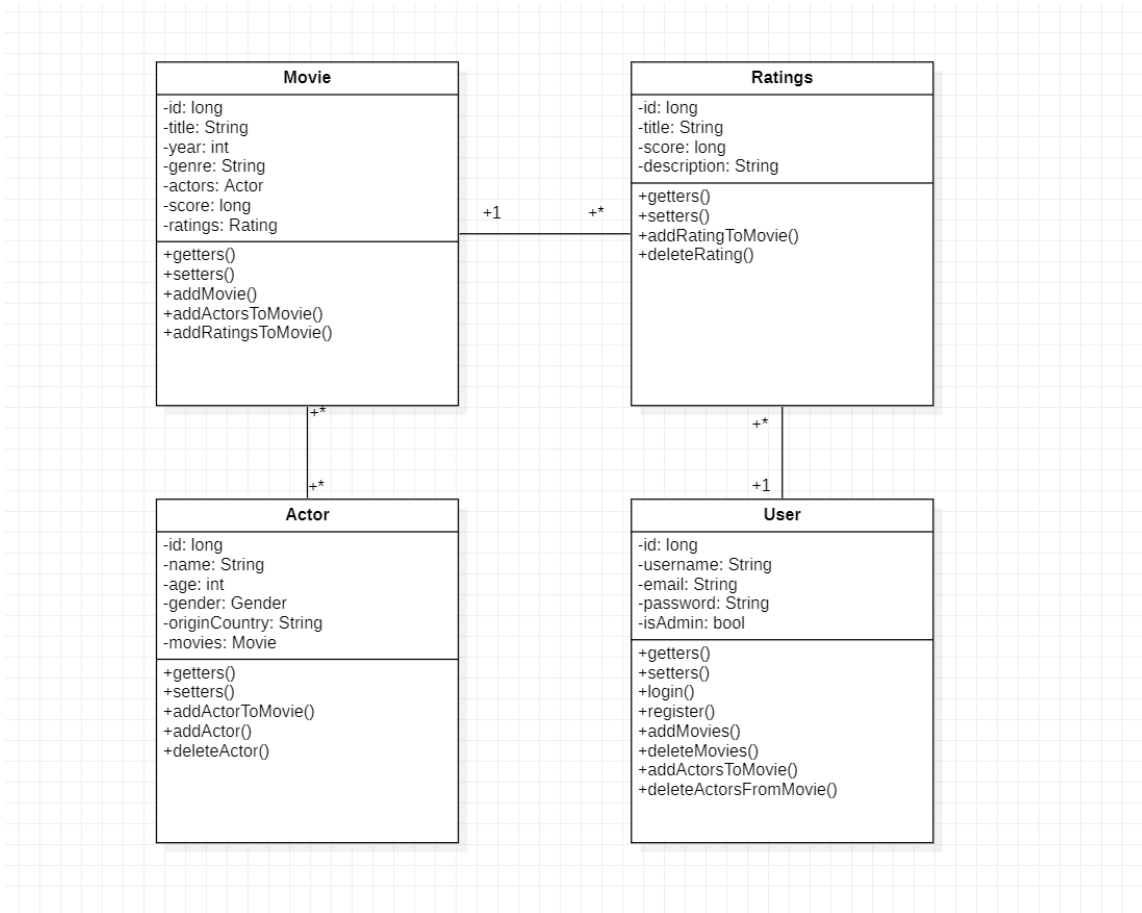


Figure 2: UML Class Diagram

## 2.2 Architectural Design

### 2.2.1 Conceptual Architecture

The application is designed with a Layered architecture pattern. The first layer is the database, which stores the software's information and is accessed through the next layer. This layer includes the programming that enables access to and management of the database, as well as the database technology itself.

The persistence layer ensures data manipulation through interaction with the database, and access to them is made possible through the JpaRepository API. This API allows for CRUD operations to be performed on the data.

The business layer handles data received from the persistence layer and performs any necessary validations. Services required by the application, such as queries, delete, insert and update, are obtained through this layer. The business layer includes the models mentioned in the previous section, as well as Data Transfer Objects (DTOs), which contain only the data required for transmission to the next layer.

The presentation layer is responsible for defining the controllers that facilitate communication with the frontend through the HTTP protocol. It includes the graphical design of the software and manages user interaction. Only UI-specific logic should be found within this layer.

### 2.2.2 Package Design

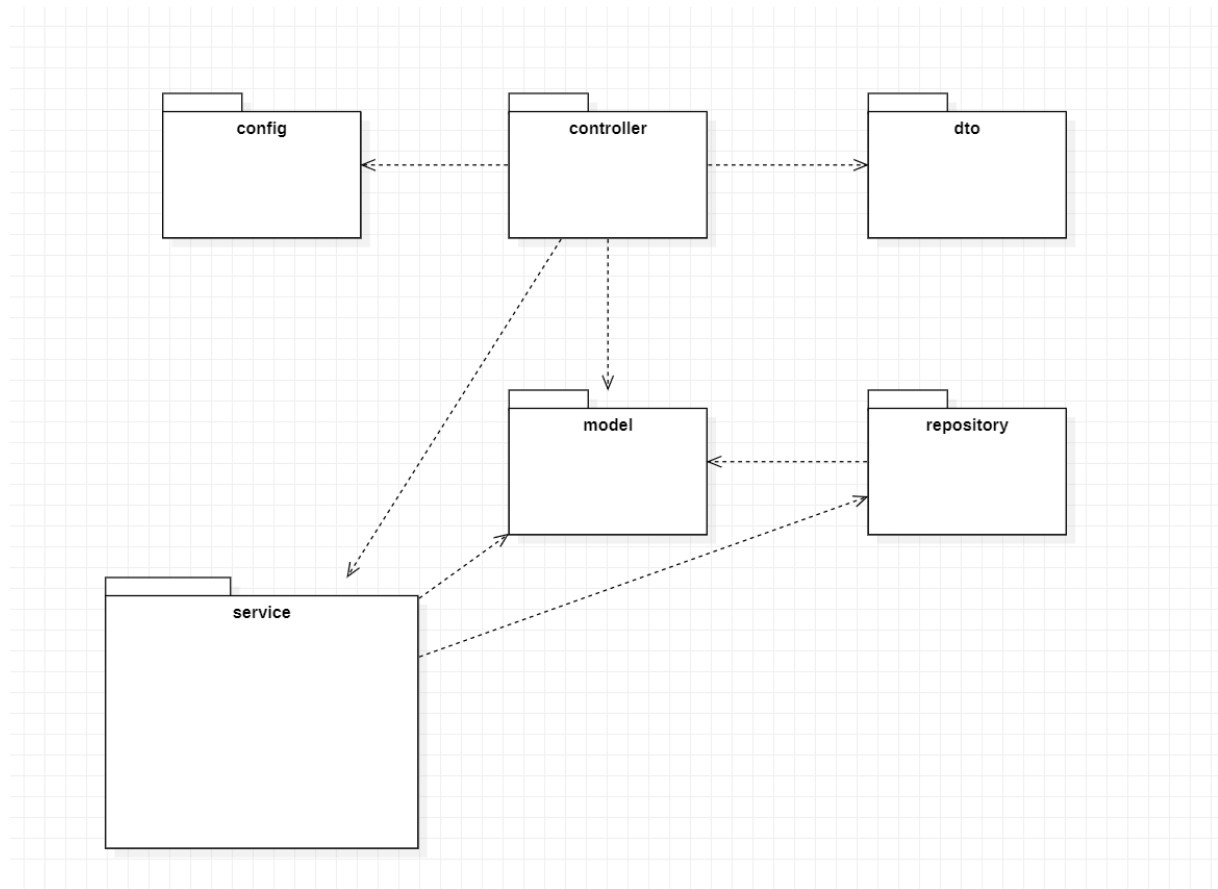


Figure 3: Package Design

The package diagram for the cinema application includes three packages: config, controller, and dto. The config package handles the configuration of the application, while the controller package manages the application logic and user interaction. The dto package contains data transfer objects that reflect the domain model and are used to transmit data between layers.



### 2.2.3 Component and Deployment Diagram

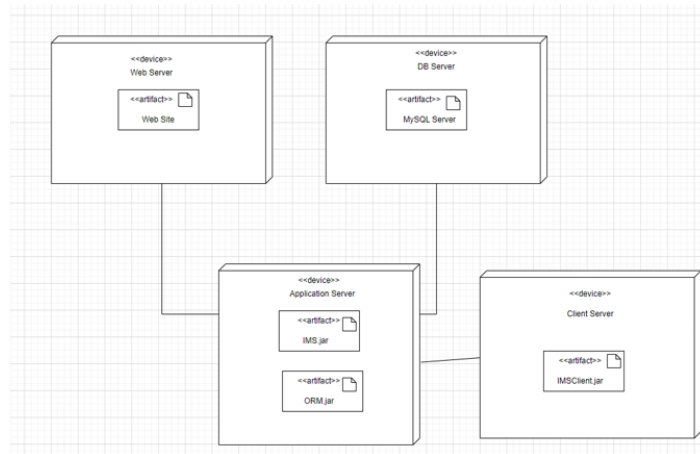


Figure 4: Deployment Diagram

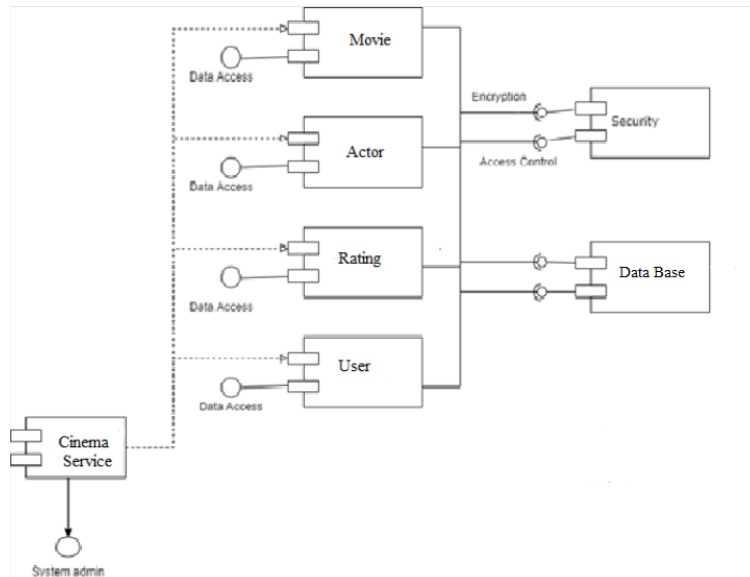


Figure 5: Component Diagram

## 3 Deliverable 3

### 3.1 Design Model

#### 3.1.1 Dynamic Behavior

[Create the interaction diagrams (1 sequence, 1 communication diagrams) for 2 relevant scenarios]

#### 3.1.2 Class Diagram

[Create the UML class diagram; apply GoF patterns and motivate your choice]

### **3.2 Data Model**

[Create the data model for the system.]

## **4 System Testing**

[Describe the testing methides and some test cases.]

## **5 Future Improvements**

[Present some features that apply to the application scope.]

## **6 Conclusion**

## **7 Bibilography**