Student: Vescan Catalin

**Group: 30235**

Table of Contents

1. Requirements Analysis 3

1.1 Assignment Specification 3

1.2 Functional Requirements 3

1.3 Non-functional Requirements 4

2. Use-Case Model 4

3. System Architectural Design 4

4. UML Sequence Diagrams 10

5. Class Design 11

6. Data Model 13

7. System Testing 14

8. Bibliography 14

1. Requirements Analysis

# Assignment Specification

[Application description]

Use Java/C# API to design and implement a client-server application for managing online show visualization such as movies, theatre performances and sport events. The application has three types of users: the basic user, the premium user and an administrator.

The basic user can perform the following operations:

* Search show, select a show and view details of a show
* View history of all shows he has seen
* Give a rating to the show
* Add a comment to the show

The premium user can perform the following operations:

* All operations from basic user
* Recommend a show to a friend or a group of friends who also have accounts on the site and are premium users (the recommendation will also appear as a notification on the friends page)
* Add interests in a show he wants to see when it will be uploaded on the site and receive notification from application that the show was uploaded so that he can watch it

The administrator can perform the following operations:

* CRUD on shows (for ex. movie information: name, description, actors, release date, imdb rating).
* CRUD on user accounts.

In addition, when a new show is uploaded on the site and there are users interested in that show the application should inform all the interested users about that show by sending them an update about the show and let them know they can watch it.

# Functional Requirements

*[Present the functional requirements]*

* Search show, select a show and view details of a show
* View history of all shows he has seen
* Give a rating to the show
* Add a comment to the show
* All operations from basic user
* Recommend a show to a friend or a group of friends who also have accounts on the site and are premium users (the recommendation will also appear as a notification on the friends page)
* Add interests in a show he wants to see when it will be uploaded on the site and receive notification from application that the show was uploaded so that he can watch it
* CRUD on shows (for ex. movie information: name, description, actors, release date, imdb rating).
* CRUD on user accounts.

# Non-functional Requirements

*[Discuss the non-functional requirements for the system]*

Security sistem for this application is not integrated, and database doesn’t run on multiple servers.

2. Use-Case Model

*[Create the use-case diagrams and provide one use-case description (according to the format below).*

*Use-Case description format:*

*Use case: <use case goal>*

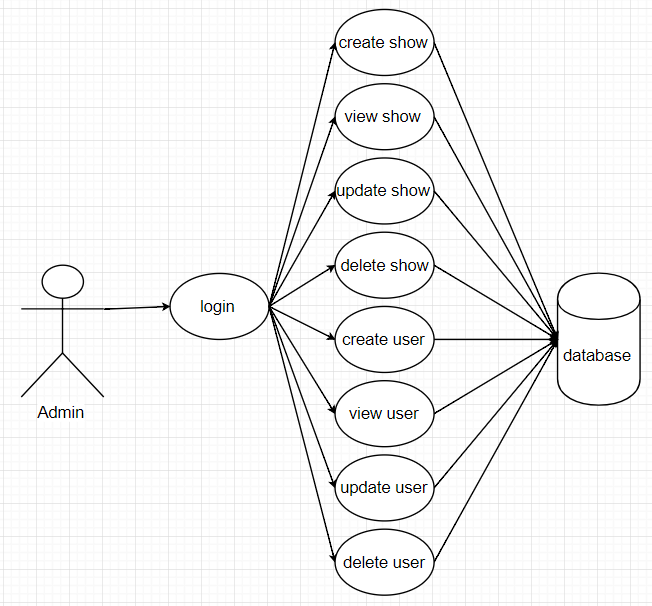
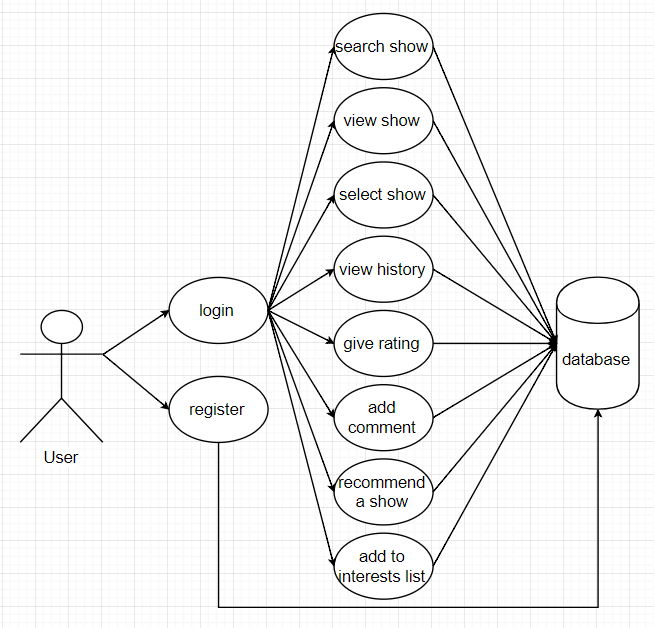
*Level: <one of: summary level, user-goal level, sub-function>*

*Primary actor: <a role name for the actor who initiates the use case>*

*Main success scenario: <the steps of the main success scenario from trigger to goal delivery>*

*Extensions: <alternate scenarios of success or failure>*

*]*



An example to undestand this usecase will be presented in next few rows:

In the login frame, the user needs an username and a password for autentification, and he must introduce a valid username (only alphabetical characters), otherwise he will see a warning message to prevent him, and a strongly password.

After login, the user has 3 posibilities to choose: movies, theatres and sports. He must add all the data to complete the text fields and after that he can search show, select a show, view details of a show, view history of all shows he has seen, give a rating to the show, add a comment to the show, and if the user has an active premium account, in addition he can recommend a show to a friend or a group of friends who also have accounts on the site and are premium users (the recommendation will also appear as a notification on the friends page), and add interests in a show he wants to see when it will be uploaded on the site and receive notification from application that the show was uploaded so that he can watch it

3. System Architectural Design

**3.1 Architectural Pattern Description**

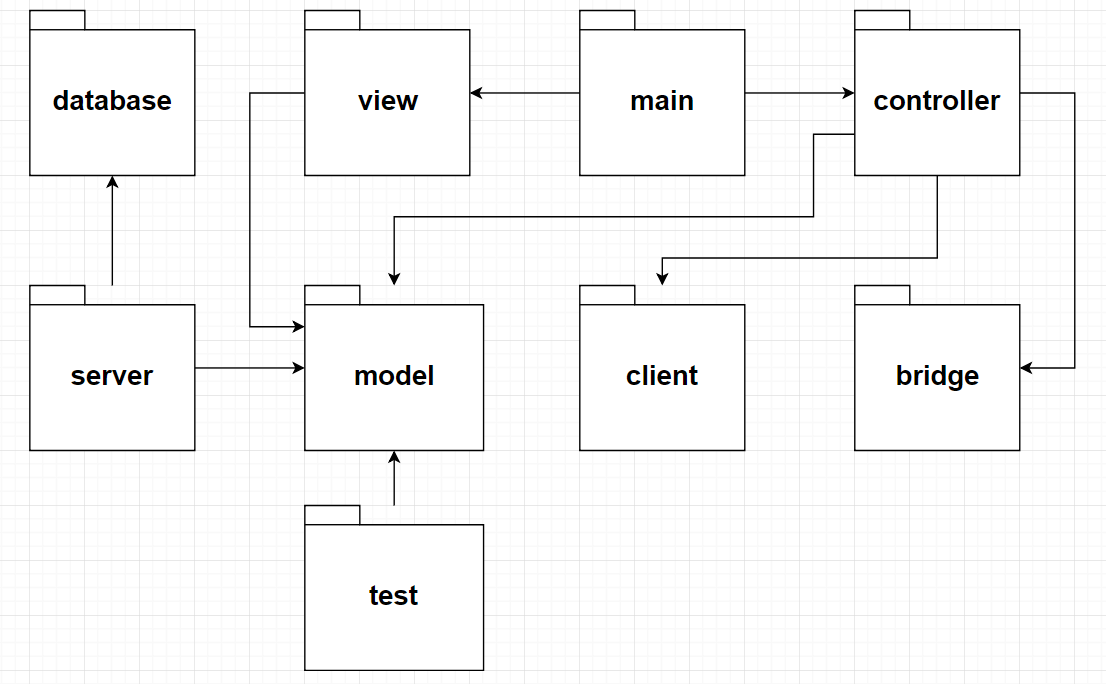
*[Describe briefly the used architectural patterns.]*

In this application I used model-view-controller (MVC) as architectural pattern which is divided in 3 parts, the model part where we have the classes that are mapped on database, the view part where we have the classes that represents the graphic user interface (GUI), and the last one, the controller part where we have the classes which make the connection between view, model and database, and i used client-server architecture.

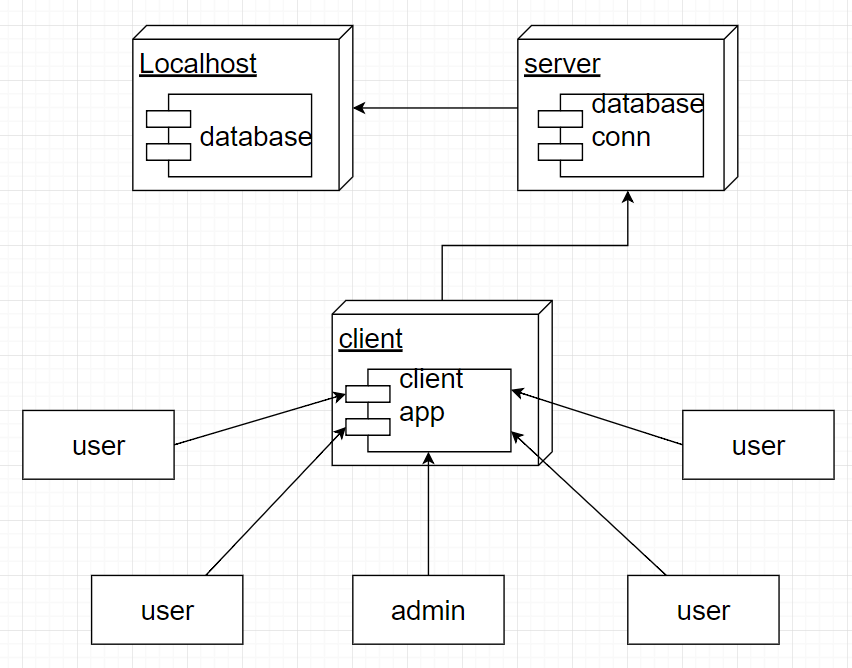
**3.2 Diagrams**

*[Create the system’s conceptual architecture; use architectural patterns and describe how they are applied. Create package, component and deployment diagrams]*

Package diagram

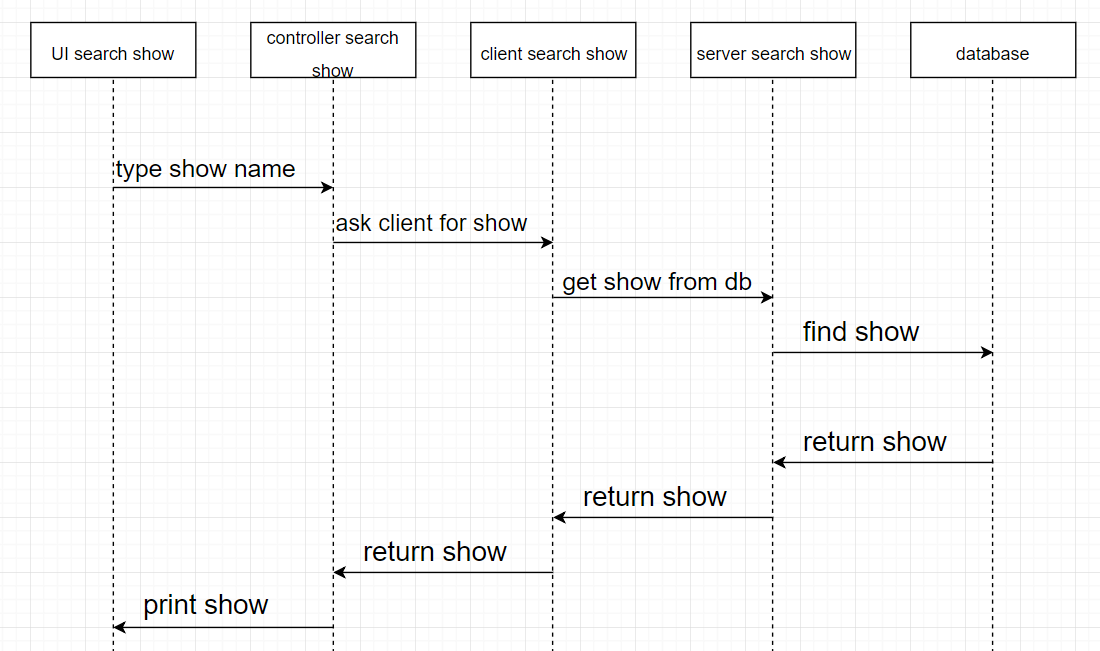


Component & deployment diagram



4. UML Sequence Diagrams

*[Create a sequence diagram for a relevant scenario.]*



5. Class Design

**5.1 Design Patterns Description**

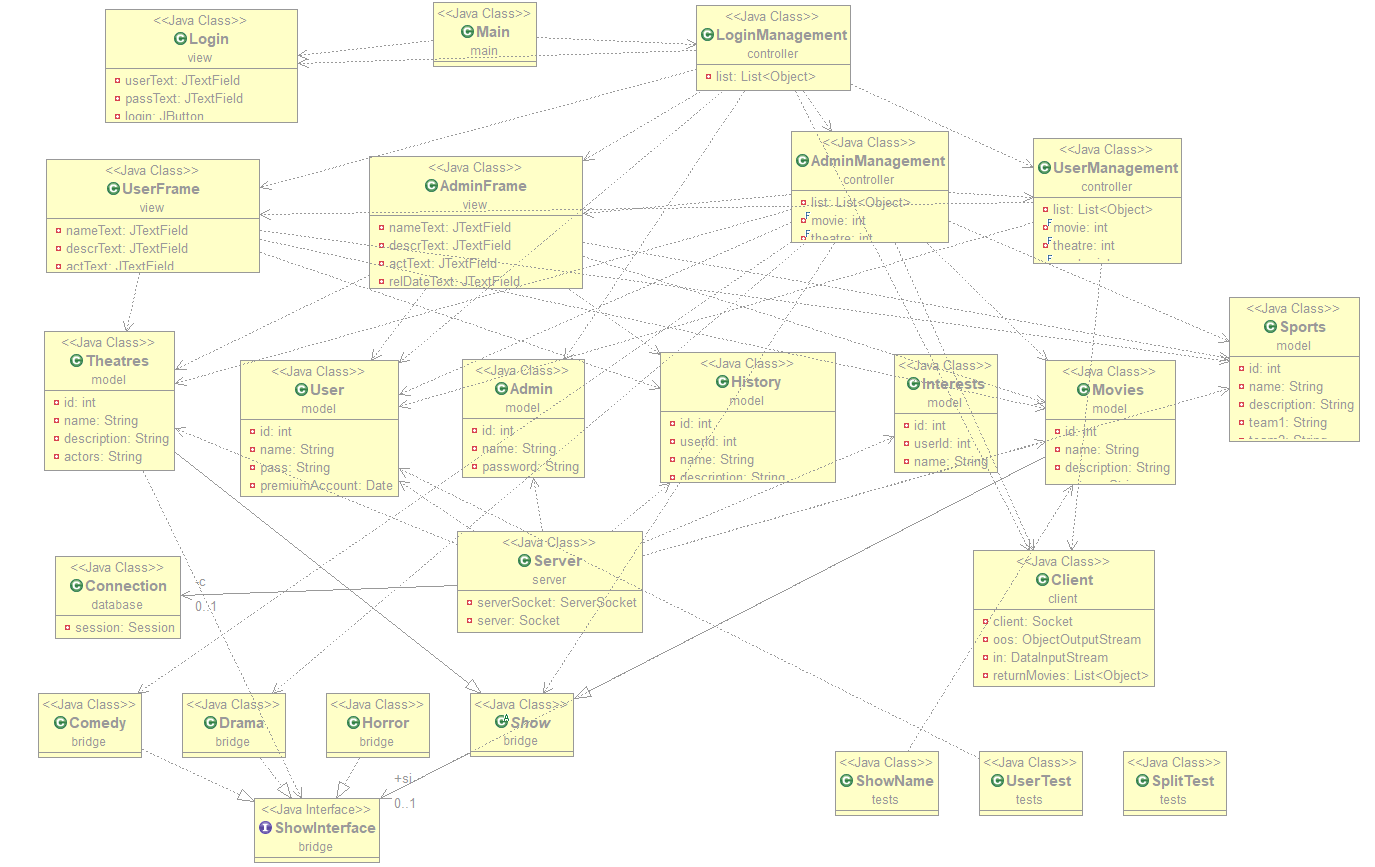
*[Describe briefly the used design patterns.]*

I used Observer as behavioral pattern to notify user when the admin add a movie in database, or when a user recommend a movie to another user.

I also used Bridge as design pattern to specify what kind of show was chosen by user.

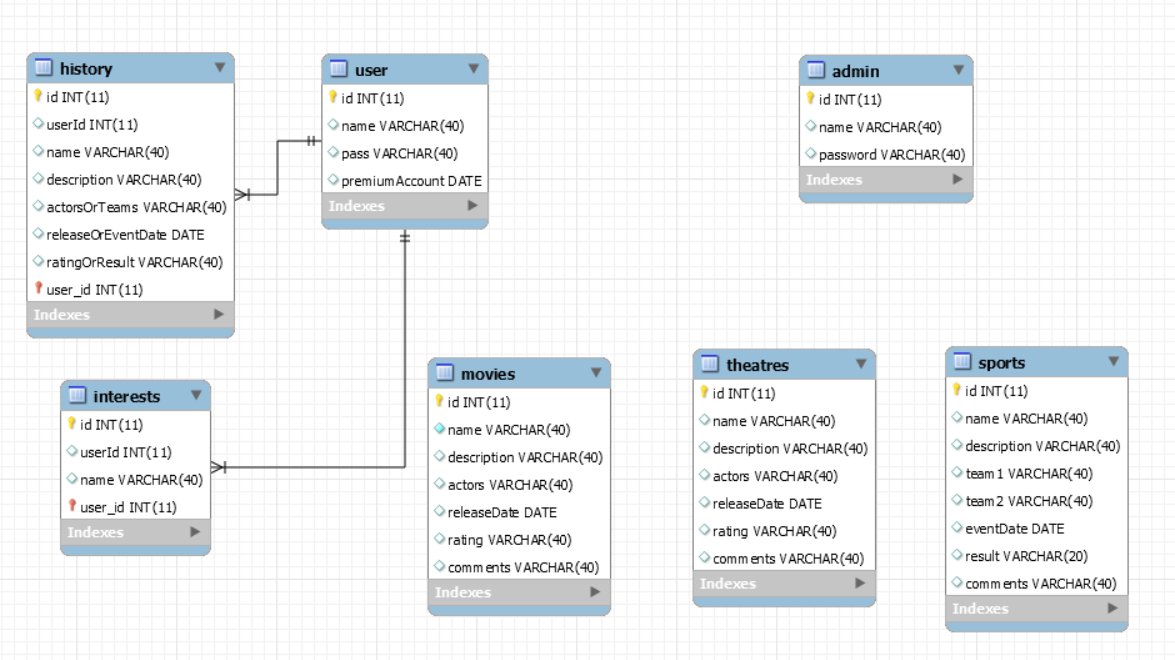
**5.2 UML Class Diagram**

*[Create the UML Class Diagram and highlight and motivate how the design patterns are used.]*



6. Data Model

*[Present the data models used in the system’s implementation.]*



7. System Testing

For application testing I used Junit Test.

8. Bibliography

<https://www.youtube.com/>

<https://stackoverflow.com/>

<https://ro.wikipedia.org/wiki/>

<https://www.tutorialspoint.com/design_pattern/observer_pattern.htm>

<http://download.oracle.com/javase/tutorial/networking/sockets/index.html>

<http://docs.oracle.com/javase/tutorial/uiswing/>

<http://docs.oracle.com/javase/tutorial/jdbc/basics/index.html>

<https://refactoring.guru/design-patterns/bridge>

<https://refactoring.guru/design-patterns/observer>

<https://www.tutorialspoint.com/hibernate/orm_overview.htm>

<https://examples.javacodegeeks.com/enterprise-java/hibernate/hibernate-annotations-example/>

<https://spring.io/docs/reference>