

**Sofia University “St. Kliment Ohridski”**

**Faculty of Mathematics and Informatics**

**Оnline Movie Review System**

**Date: 25.06.2022г. Author: Filip Stoyanov, FN: 81950**

**1. Short project description**

The Internet age changed the film industry. There are so many movies on the world wide web today and therefore we can’t decide which movie is good to watch easily. ***Online Movie Review System (OMRS)*** provides ability for users to add new movies in the system, search movies, review movies and create collections with movies, rate each movie and he/she can review(comment) each movie. This system provides abilities for users to select a good movie to watch.

**2. User Roles**

The main user roles in **Online Movie Review System** are **anonymous user**, **registered user** and **administrator**.

**Anonymous user** can view the information pages and view information about movies in the system (watching trailer, reading comments, reviewing and reading other information about the movies).

**Registered users** can execute all operations that are allowed for anonymous users. Registered users can add new movie in the system, edit and delete their own movies, filter movies by different critera, review and rate movies, create collections with movies and change your profile personal information. Registered users can also add and delete friends (other users in **Online Movie Review System**) and they have opportunity to share collections with their friends.

**Administrator** can manage (**create, edit and delete**) all Registered Users as well as Movies, Reviews, Comments, Notifications and Movie Collections.

**3. Functionality / Scenarios/**

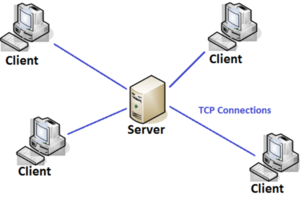
**Online Movie Review System** has the following main use cases:

* **Browse information**:The *User* can browse the information views (Home, Movies, Terms of use, About us) in **Online Moview Review System** and can view comments and reviews about the movie and watch movie trailer.
* **Registration**: Anonymous Usercan register in the system by providing a personal information in three step registration forms. User can go back or go on only with click on animation stepper. By default, all new registered users have **Registered User Role.**
* **Change User Data**: RegisteredUsercan view and edit her/his personal User Data *(first and last name, email, profile image, favorite genres and username.).*
* **Manage Users**: Administrator can manage the User. Administrator can also add, edit and delete movies from system, delete users from system, add and delete movie reviews and comments from system.
* **RegisteredUseror Administrator** can create, edit and delete collection from movies.
* **RegisteredUseror Administrator** can review each movie and add it to one of his/her movie collections.
* **All Users** can filter movies by different criteria (year, rating, producer and etc.)
* **Registered User or Administrator** can add new friend to his/her friend list. In this way he/she receive notifications about new commented and reviewed Movies from his/her Friend. *Registered User* can manage his/her own friend list (can add and delete friends from it).
* **Registered Users** have also opportunity to share collections with your friends. The friends of Registered User can rearrange the movies in order they want (begin with the best one movie from this collection) with drag and drop. Then friends can send rearranged movie list back to movie list creator. Registered User can use this information ( to reorder movies in your movie collection)

**4. Architecture model and used technologies**

**The Online Movie Review System** uses **client-server architecture**. **Client-server architecture** is an architecture of a computer network in which many clients (remote processors) request and receive service from a centralized server (host computer).

**Client-server architecture**



The system is developed as **a *Single Page Application (SPA)*** using ***Angular*** as front-end, **Angular Material** and **Ng Bootstrap** for styling, and ***Node.js + Express*** as backend technologies. Each view will have a distinct URL, and the routing between pages will be done client side using ***Angular Router***. The backend is implemented as a ***REST/JSON API*** using JSON data serialization.

**5. Frontend implementation**

The OMRS frontend is implemented from views, each of them has distinct URL.

The main views are:

* **Home (/home):** Presents the introductory information for the purpose of the system as well as detailed instructions how to start using it. Prominently offers ability to register.
* **Movies (/movies):** Presents movies available. Offers abilities to browse, choose, create, read, update, delete (CRUD) and filter Movies, according to *User's Role*. ( for *Administrators* and *Registered Users* only).
* **About (/about):** Presents information about the *OMRS* project and his owner.
* **Terms of use (/terms):** Define terms of use for OMRS.
* **Collections (/collections):** Presents User’s Movies Collections according to *User's Role. User can also add new Movies in Movie Collections and delete movies from Movie Collections.*
* **User Registration (/registration):** Presents a view allowing the *Anonymous Users* to register in *OMRS*.
* **Login (/login):** Presents a view allowing the users to login.
* **Personal information (/profile):** Presents ability to view and edit personal *User Data*
* **Friends (/friends):** Presents ability to view and edit personal friend list.
* **Rate friend collection** **(/rateCollection):** Presents ability to rate friend’s collection
* **Ratings (/ratings):**  Presents ability to see how your friend rated your collection.

**6. Backend implementation (backend REST API endpoints)**

The **OMRS** backend is implemented from Rest API endpoints, each of them has distinct URL (endpoint).

The main **REST API endpoints** are:

* **Users (*/api/users*):** GET User Data for all users, and POST new User Data (Id is auto-filled by OMRSand modified entity is returned as result from POST request
* **User (*/api/users/{userId}*):** GET, PUT, DELETE UserData for User with specified *userId*, according to restrictions decribed in UCs.
* **Login (*/api/login):***POST User Credentials (e-mail address and password) and receive a valid Security Token to use in subsequent API requests.
* **Movies (/api/movies):** GET *Movies*, and POST new Movies (Id is auto-filled by *OMRS* and modified entity is returned as result from POST request), according to User's Role and identity security restrictions.
* **Movie (/api/movies/{movieId}):** GET, PUT, DELETE Movie from Movies with specified *movieId*.
* **Movies Collections (/api/collections):** GET Movie Collection and POST new Movie Collection (Id is auto-filled by *OMRS* and modified entity is returned as result from POST request).
* **Movie Collection (/api/collections/{collectionId}):** GET, PUT, DELETE Movie Collection for *Movies* with specified *movieId*.
* **Rankings (/api/ranking):** GET Movie lists for User that are send from his/her friend (Id is auto-filled by *OMRS* and modified entity is returned as result from POST request).
* **Ranking (/api/ranking/user/{id}):** GET, PUT, DELETE movie list with specified id that is send from his/her friend.
* **Notifications (/api/notifications):** GET Notifications for User, and POST new Notification (Id is auto-filled by *OMRS* and modified entity is returned as result from POST request).
* **Notification (/api/notification/{notificationId}):** GET, PUT, DELETE Notification with specified *notificationtId.*
* **Comments (/api/comments/):** Get Comments for User, and POST new Comments (Id is auto-filled by *OMRS* and modified entity is returned as result from POST request).
* **Comment (/api/comments/{commentsId}):** GET, PUT, DELETE Comment with specified *commentId.*
* **Preferences (/api/preferences***):* GET rated Movie lists for User (Id is auto-filled by *OMRS* and modified entity is returned as result from POST request).
* **Preference (/api/preferences/{id}):**GET, PUT, DELETE rated movie list with specified movielistId

**7. Set up Online Movie Review System**

Execute following commands in the order they are given:

- **Download** zip archive and extract all files from it

- **Find ‘dump’ folder**, copy the folder and paste it to **/bin** folder in your **MongoDB folder** (for example: C:\Program Files\MongoDB\Server\5.0\bin)

- **Open mongo.exe** terminal from your local MondoDB folder (for example: C:\Program Files\MongoDB\Server\5.0)

- Execute the following command**: ` mongorestore --db MovieSystem dump/MovieSystem`**

- **Go to project folder**

- **Open new terminal** there and install node modules with **`npm install`** or **`yarn`**

- Run **`ng serve`** in the terminal

- **Open second terminal**, go to ‘server’ folder in project (type **`cd ./server`)**, run **`npm install`** and then run **`npm start`**

**8. Upcoming Features on Online Movie Review System**

In future **Online Movie Review System** will have new features and improvements. Some of them are related to better responsive design, features related to log in and registration process (hashing password), implementing web socket (creating chat), adding animations, features related to administrator role activities, other multimedia features and etc.

**References**

1. <https://angular.io/docs> Angular Docs
2. <https://nodejs.org/dist/latest-v16.x/docs/api/> Node.js Docs
3. <https://expressjs.com/> Express web framework
4. <https://material.angular.io/> Angular Material
5. <https://ng-bootstrap.github.io/#/home> Ng bootstrap
6. <https://cio-wiki.org/wiki/Client_Server_Architecture> Client server architecture

**GitHub repository**

<https://github.com/FilipStoyanov/Online-Movie-Review-System>