MoviePocket Developer documentation

Introduction

MoviePocket is movie management system that will help you to orginize your movie collection and share it with other movie lovers. The more percise description of the service you can find in Documentation/ section:)

Explore world of movies with MoviePocket and create your personal unforgettable experience collecting sharing movies between users.

To start develop

Introduction

MoviePocket is movie management system that will help you to orginize your movie collection and share it with other movie lovers. The more percise description of the service you can find in Documentation/ section:)

Explore world of movies with MoviePocket and create your personal unforgettable experience collecting sharing movies between users.

Backend Java Spring

You will need:

- Java(at least 1.8), **Spring boot** version at least 2.7.10 and Spring 5.X bounded with Maven for backend
  - Cookie based authentication by Spring Security 2.7
  - ORM: Hibernate v 5. + JPA
  - port used for local: *8080*

API Documentation

Info about API endpoints, request/response examples, and any required headers generated by Swageer UI and can be found by `{hostname}/swagger-ui.html`

MVP pattern is used.

Swagger.png ### Database Schema

MySql is used. DB script can be found in DbScript

DB schema DB schema 1

DB schema 2

Authentication

Cookie based authentication is implemented. Cookie expiration set to -1 and can be changed in application.properties file

Error Handling

Most errors are handled by ResponseEntity and Http status codes responses Main codes used are: 1. `200 Ok` 2. `401 Unauthorized` 3. `403 Forbidden` 4. `404 Not found` 5. `500 Server error`

Frontend React.js

Project Structure

Describe the project directory structure and organization.

Component Documentation

Document each React component, including its purpose, props, and usage.

- Component Buttons
  - DislikedMovieButton

- Purpose: The DislikedMovieButton component is a reusable React component designed to represent a button that allows users to dislike a movie. It interacts with the server to check the current disliked state of the movie and enables users to toggle the dislike state by clicking the button.
- Props:
  - idMovie (Required, Number): The unique identifier of the movie for which the dislike button is being displayed.
  - className (Optional, String): Additional CSS class name(s) that can be provided to customize the styling of the button.
- Usage: the DislikedMovieButton is used within a MovieDetails component, where idMovie is set to the unique identifier of the movie being displayed. The optional className prop is used to customize the styling of the button.

This component utilizes the AuthContext to check if the user is logged in before allowing them to dislike a movie. When the button is clicked, it toggles the dislike state and sends a request to the server to update the dislike status.

- FavoriteMovieButton - Purpose: The FavoriteMovieButton component is a reusable React component designed to represent a button that allows users to mark a movie as a favorite. It interacts with the server to check the current favorite state of the movie and enables users to toggle the favorite state by clicking the button. - Props: - idMovie (Required, String): The unique identifier of the movie for which the favorite button is being displayed. - className (Optional, String): Additional CSS class name(s) that can be provided to customize the styling of the button. - Usage: the FavoriteMovieButton is used within a MovieDetails component, where idMovie is set to the unique identifier of the movie being displayed. The optional className prop is used to customize the styling of the button.

- RatingComponent - Purpose: The RatingComponent is designed to provide a user interface for selecting a rating using star icons. It uses state to track the selected rating and updates the display dynamically. - Props: rating: A state variable that holds the selected rating. It is initially set to null and gets updated when a star is clicked.

- ToWatchMovieButton - Purpose: The ToWatchMovieButton component serves as a button that allows users to add or remove movies from their “To Watch” list. It displays a tooltip and changes its appearance based on whether the movie is in the “To Watch” list or not. - Props: - toWatch: A state variable that holds whether the movie is in the “To Watch” list. It is initially set to false and gets updated when the user clicks the button. - isLoggedIn: The component uses the AuthContext to check if the user is logged in.

- WatchMovieButton.js - Purpose: The WatchMovieButton component serves as a button that allows users to mark a movie as watched. It displays a tooltip and changes its appearance based on whether the movie has been watched or not. - Props: - watched: A state variable that holds whether the movie has been watched. It is initially set to false and gets updated when the user clicks the button. - isHovered: A state variable that holds whether the button is being hovered. It is used to determine the button’s appearance during hover. - isClicked: A state variable that holds whether the button has been clicked. It is used to determine the button's appearance after being clicked.

- Component Lists
    - ListOfFilms
    - SingleList
    - UserLikedList
- Component navbar
    - logoBar
    - logOutComponent
    - navBar - navBrand
    - navList
    - searchComponent
    - userBarComponent
- Component pagination
    - Pagination
- Component poster
    - MoviePoster
    - Poster
- Component review
    - CreateReviewForm.js
    - LikeReviewButton.js
    - ReviewDeleteButton.js
    - SingleReview.js
- Component spinners
    - Spinner
- Component top
    - Top
- Component Layout

API Integration

Service is using external TMDB APi for several operations such as poster and movie info fetching etc.

Mobile App Android

General

Android 10+ should be used for running

App uses mainly MVC pattern and uses both backend api endpoints and TMDB API.

Authentication

Authentication flow is depends on Backend part as it uses backend api see Backend (Java Spring)

Deployment

Backend and Frontend Deployment

Both front and back end are deployed on remote linux server provided by UAM WMI

Server information and connection credentials can be found Here #### Mobile App Deployment

Android app is deployed to Google Play

Contributing

Version Control

Backend: https://github.com/prymakD/MoviePocket

Frontend: https://github.com/EkintoQ/MoviePocket-Frontend

Android: https://github.com/AntonPazniak/MoviePocketAndroid/tree/master

Testing

Document the testing strategy for backend, frontend, and mobile app components.

Changelog

v. 1.0.0

License

Proprietary license

Please add header to created files `/** * Copyright (C) {MoviePocket} - All Rights Reserved * * This source code is protected under international copyright law.`