Movie List Documentation

Overview

This web application allows users to browse movies, watch trailers, and actively contribute by writing, editing, and deleting movie reviews. It leverages a modern tech stack, featuring React for the frontend, Spring Boot 3.2.0 with Java 17 for the backend, and MongoDB Atlas for efficient data storage.

Backend

- Responsible for handling business logic, data processing, and interacting with databases.
- Implements the server-side logic and exposes APIs to be consumed by the frontend.
- Independent of the frontend implementation and can be developed, deployed, and maintained separately.

Frontend

- Focuses on the user interface (UI) and user experience (UX).
- Consumes APIs provided by the backend to retrieve and send data.
- Operates independently of the backend, allowing for flexibility in frontend development and potential reuse of the backend with other frontends.

Technologies

Backend: Spring Boot 3.2.0, Java 17, MongoDB
Frontend: React

Main Functions

Browse Movies: Displays a curated list of movies from MongoDB.
Watch Trailers: Allows users to watch movie trailers within the application.
Review Section: Each movie has a dedicated review section for users to Add , Update and Delete Reviews.
User Authentication: Provides secure user registration and login functionality.

Backend

Setup

1. Clone the Backend Repository

To get started with the project, you'll need to clone the backend repository. Make sure you have Git installed on your system. Open a terminal or command prompt and run the following command:

```
git clone https://github.com/ravindub/MovieList.git
```

2. Install Dependencies

Before running the Spring Boot application, you need to install its dependencies. Ensure that you have JDK 17 (Java Development Kit) installed on your machine. You can download JDK 17 from the official Oracle website or use a package manager if you're on a Unix-based system.

Download JDK 17

Next, set up IntelliJ IDEA, a powerful IDE for Java development. You can download the Community Edition, which is free to use.

Download IntelliJ IDEA

Once JDK 17 and IntelliJ IDEA are installed, open the project in IntelliJ IDEA, and it will automatically prompt you to install any required dependencies. If you're using Maven, IntelliJ will download the necessary dependencies for you.

3. Set Up Your MongoDB Database

3.1 Set Up MongoDB Atlas

Create a New Project:

- Log in to MongoDB Atlas.
- Create a new project by clicking on the "New Project" button.

Create Deployment:

- Within your project, click on "Build a Cluster" to create a new deployment.
- Choose your preferred cloud provider, region, and cluster tier. For development purposes, the free tier is usually sufficient.

Create Username and Password:

- In the "Security" tab, click on "Database Access" and then "Add New Database User."
- Create a new username and password for accessing your database. Remember to save these credentials for later use.

Configure IP Address Access:

- Still in the "Security" tab, click on "Network Access" and then "Add IP Address."
- Add 0.0.0.0/0 to allow connections from any IP address. This is suitable for development but consider restricting access in a production environment.

Connect and Copy Connection String:

- Navigate to the "Clusters" tab, find your cluster, and click on "Connect."
- Choose "Connect your application" and copy the connection string. Replace <password> with the password you created earlier.

3.2 Connect with MongoDB Compass

Install MongoDB Compass:

- Download and install MongoDB Compass on your local machine.

Connect to Database:

- Open MongoDB Compass and click on "New Connection."
- Paste the connection string you copied earlier. MongoDB Compass will automatically parse the details.
- Click "Connect" to establish a connection to your MongoDB Atlas cluster.

Create a Database and import Data:

- Create a new database.
- Create a new collection named ‘movies’.
- Go to the collection and import data from `data/movies.json` file.

4. Configure Environment Variables

sensitive information like connection string, should be provided as environment variables. So create a `.env` file in the `src/main/resources` folder and add values according to the following structure.

```
MONGO_DATABASE=
MONGO_USER=
MONGO_PASSWORD=
MONGO_CLUSTER=
```

Now you can run the project in IntelliJ IDEA.

API Endpoints

1. Retrieve the List of Movies

- **Endpoint:** GET /api/v1/movies
- **Description:** Retrieve the list of movies available in the database.
- **Parameters**: None
- **Response:**
    - 200 OK
    - 404 Not Found

2. Get Details of a Specific Movie

- **Endpoint:** GET /api/v1/movies/{imdbId}
- **Description:** Get detailed information about a specific movie identified by its IMDb ID.
- **Parameters:** {imdbId}: IMDb ID of the movie (e.g., tt1234567).
- **Response:**
    - 200 OK
    - 404 Not Found
- **Notes:** IMDb ID is a unique identifier for movies.

3. Add a New Review for a Movie

- **Endpoint:** POST /api/v1/reviews
- **Description:** Add a new review for a movie.
- **Parameters:** None
- **JSON Body:**review body and imdb id

```
{
   "reviewBody": "Nice Movie",
   "imdbId": "tt3915174"
}
```

- **Response:**
    - 201 Created
    - 400 Bad Request
    - 404 Not Found

4. Update a Movie Review

- **Endpoint:** PUT /api/v1/reviews/{id}
- **Description:** Update an existing movie review.
- **Parameters:** {id}: ID of the review to be updated.
- **JSON Body:** updated review body
- **Response:**
    - 200 OK
    - 400 Bad Request

    - 404 Not Found

5. Delete a Movie Review

- **Endpoint:** DELETE /api/v1/reviews/{id}
- **Description:** Delete an existing movie review.
- **Parameters:** {id}: ID of the review to be deleted.
- **Response**:
    - 204 No Content
    - 404 Not Found

6. Register a New User

- **Endpoint:** POST /api/v1/users/register
- **Description:** Register a new user.
- **Parameters:** None
- **JSON Body:**
    - username: User's chosen username.
    - password: User's chosen password.
- **Response:**
    - 201 Created
    - 400 Bad Request

7. Log in an Existing User

- **Endpoint:** POST /api/v1/users/login
- **Description:** Log in an existing user.
- **Parameters:** None
- **JSON Body:**
    - username: User's username.
    - password: User's password.
- Response:
    - 200 OK
    - 401 Unauthorized

Dependencies

1. Spring Boot Starter Parent

The Spring Boot Starter Parent provides a set of common configurations and dependencies for Spring Boot projects.

```
<parent>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-parent</artifactId>
    <version>3.2.0</version>
    <relativePath/> <!-- lookup parent from repository -->
```

```
</parent>
```

2. MongoDB Starter

The MongoDB Starter enables Spring Data MongoDB for your project, allowing seamless integration with MongoDB databases.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-data-mongodb</artifactId>
</dependency>
```

3. Spring Boot Starter Web

The Spring Boot Starter Web dependency includes everything needed to build a web-based application using Spring Boot.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
```

4. Spring Boot DevTools

Spring Boot DevTools provides development-time features to enhance productivity, such as automatic restarts and enhanced debugging.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-devtools</artifactId>
    <scope>runtime</scope>
    <optional>true</optional>
</dependency>
```

5. Lombok

Lombok is a library that simplifies Java code by providing annotations to generate boilerplate code during compilation.

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <optional>true</optional>
</dependency>
```

6. Spring Boot Starter Test

The Spring Boot Starter Test dependency includes testing libraries and annotations for testing Spring Boot applications.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-test</artifactId>
    <scope>test</scope>
</dependency>
```

7. Spring Boot Starter Security

The Spring Boot Starter Security dependency includes Spring Security for securing your Spring Boot application.

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-security</artifactId>
</dependency>
```

8. Spring Dotenv

Spring Dotenv is used for loading environment variables from a .env file into Spring's Environment.

```
<dependency>
    <groupId>me.paulschwarz</groupId>
    <artifactId>spring-dotenv</artifactId>
    <version>4.0.0</version>
</dependency>
```

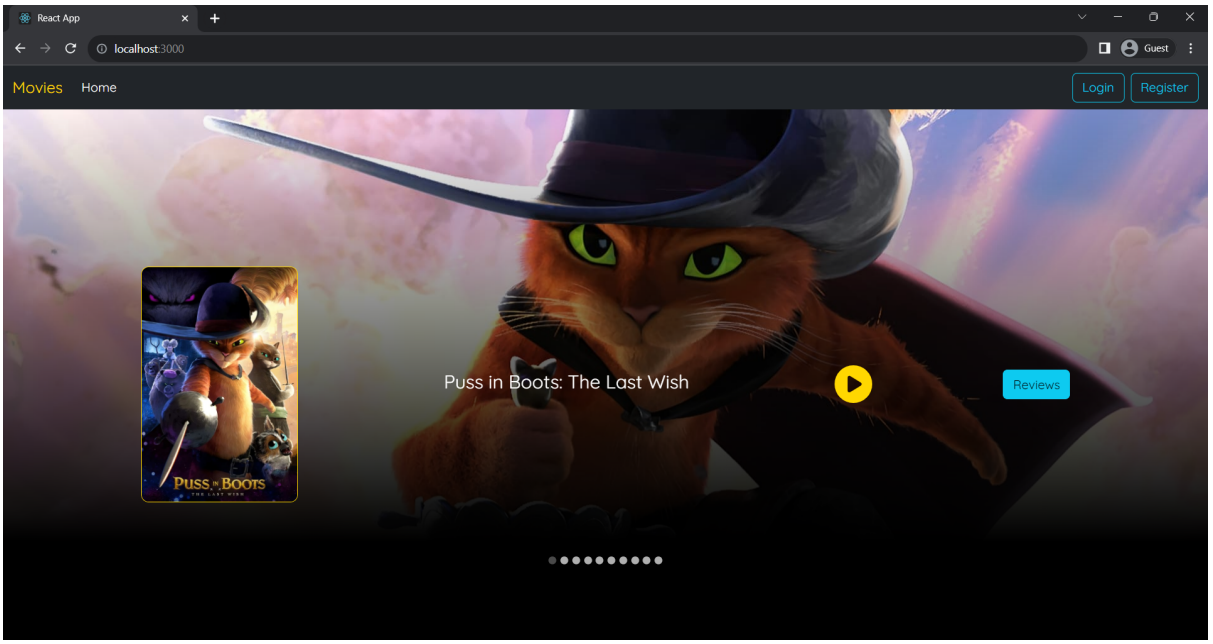
Frontend

Setup

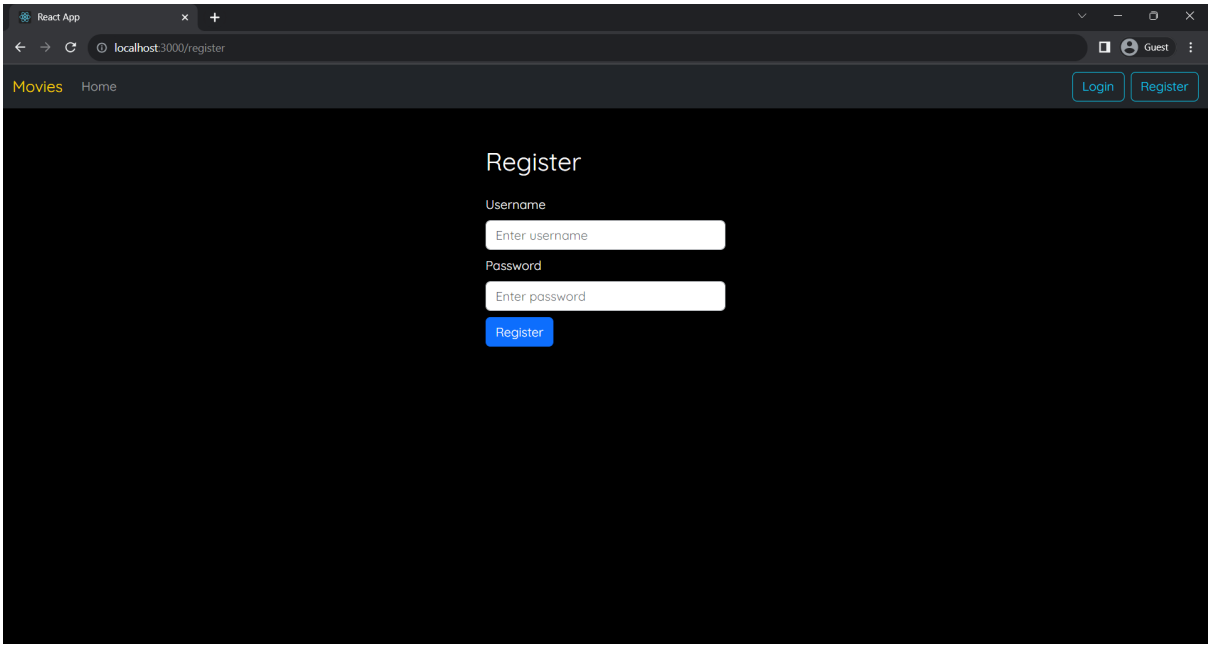
- Navigate to the Frontend: `cd movies-front-end`
- Install Dependencies: `npm install`
- Run the Frontend: `npm start`

Screenshots

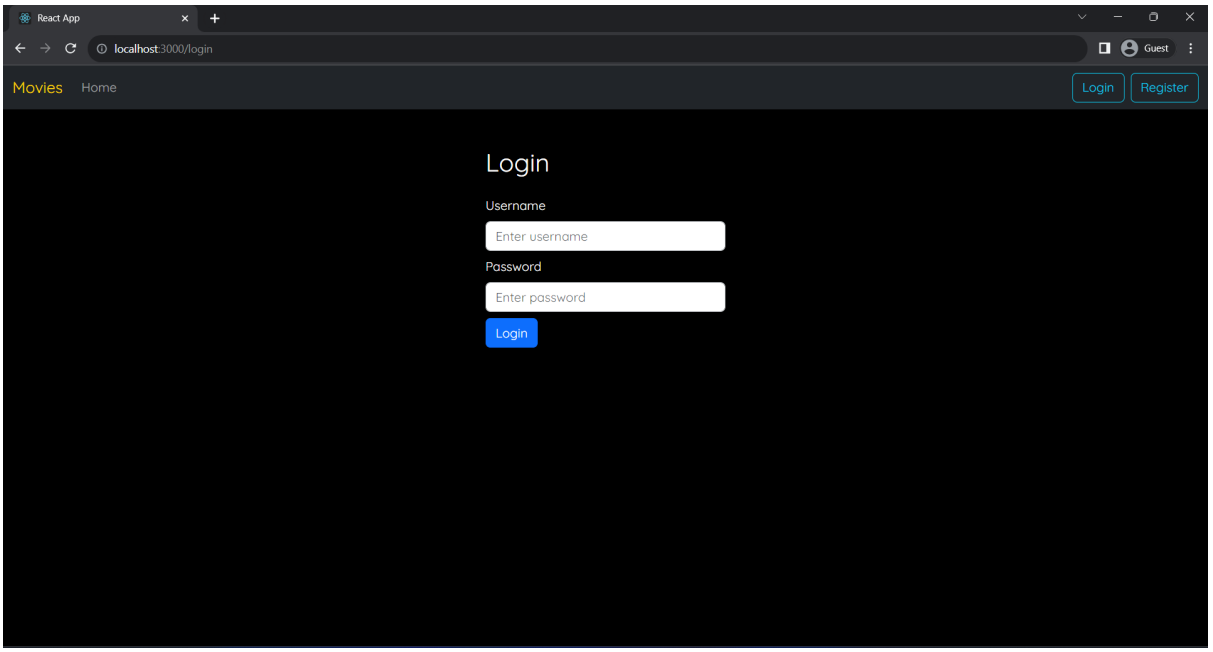
Home page



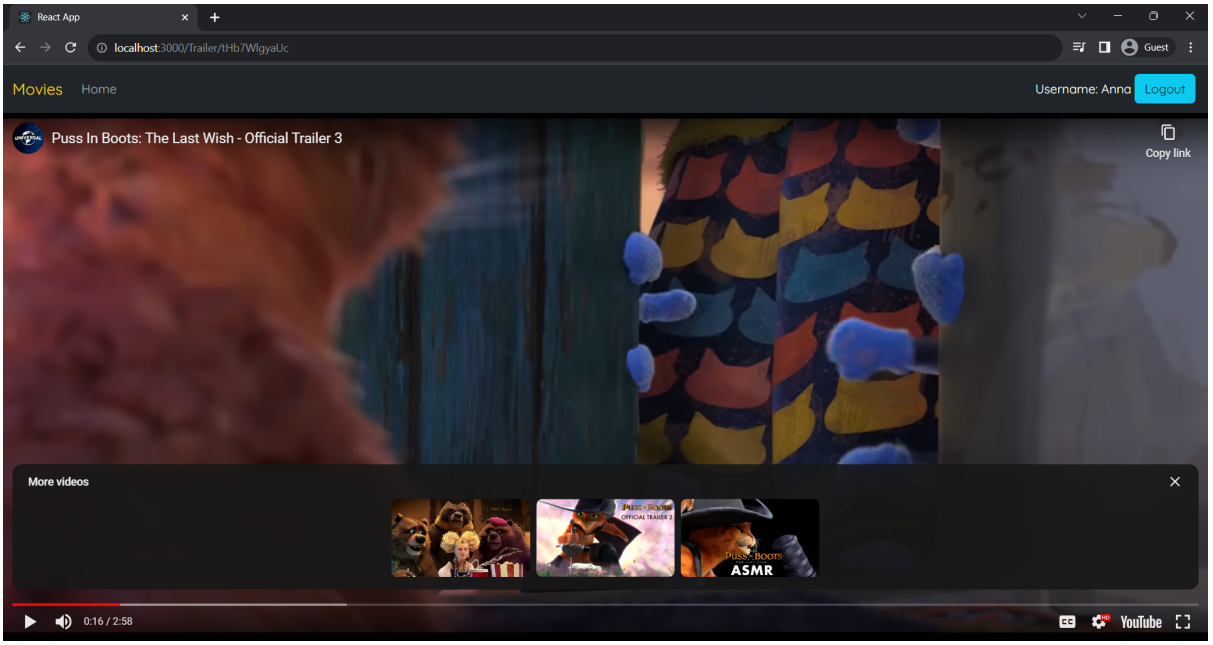
Registration Page



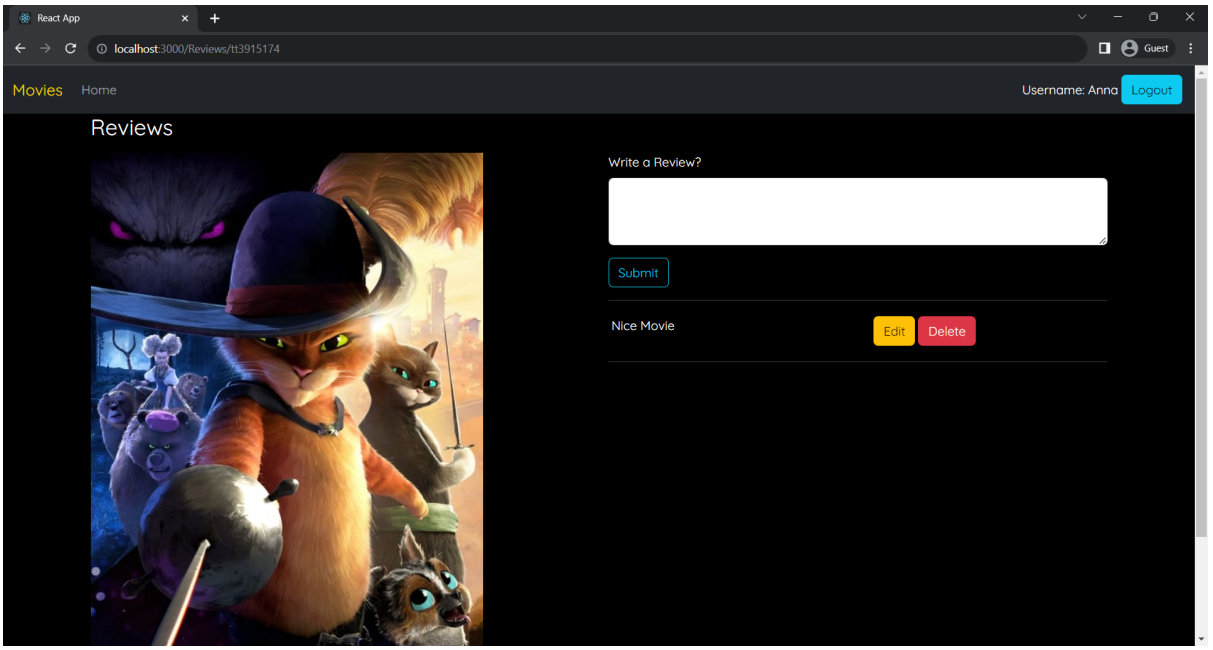
Login Page

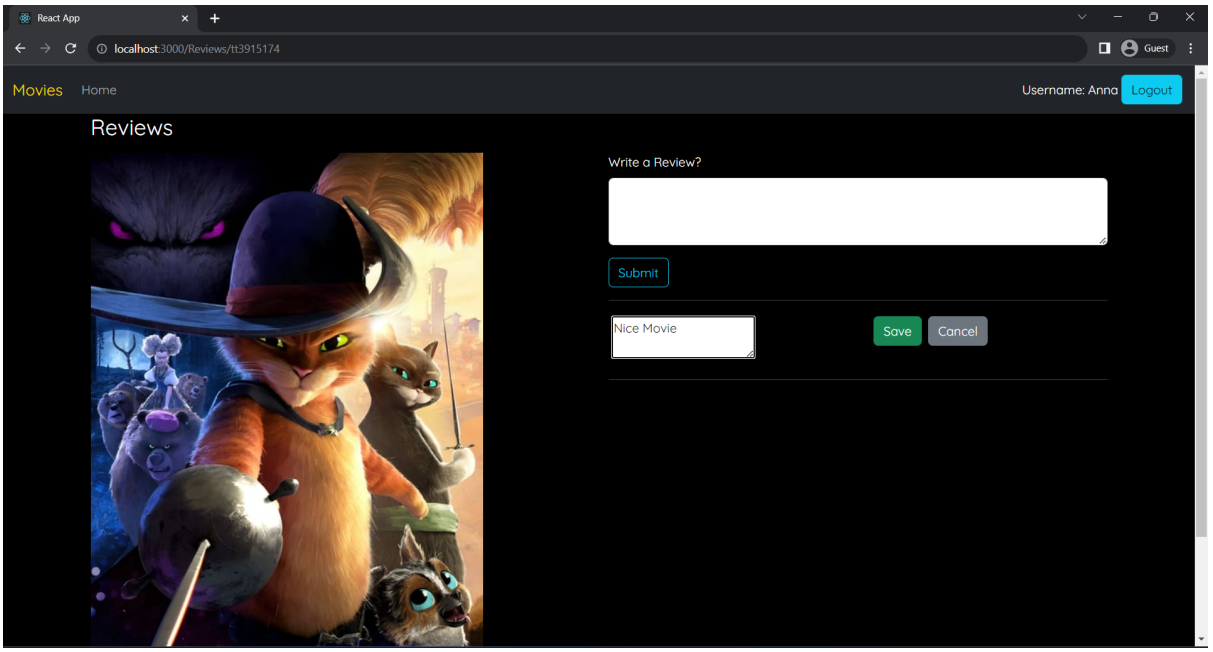


Movie Trailer Page



Review Page


React App
localhost:3000/Reviews/tt3915174
Guest
Movies
Home
Username: Anna
Logout
Reviews
Write a Review?
Submit
Nice Movie
Edit
Delete


React App
localhost:3000/Reviews/tt3915174
Guest
Movies
Home
Username: Anna
Logout
Reviews
Write a Review?
Submit
Nice Movie
Save
Cancel