

# **PRODUCTION GUIDE DOCUMENT**

## **Movies Project**

Swati Khandare  
([swati.khandare@epita.fr](mailto:swati.khandare@epita.fr))

## Table of Contents

Roles.....	3
User Role.....	3
Compilation and Installation Steps .....	3
Execution.....	7
API testing.....	12

## Roles

### User Role

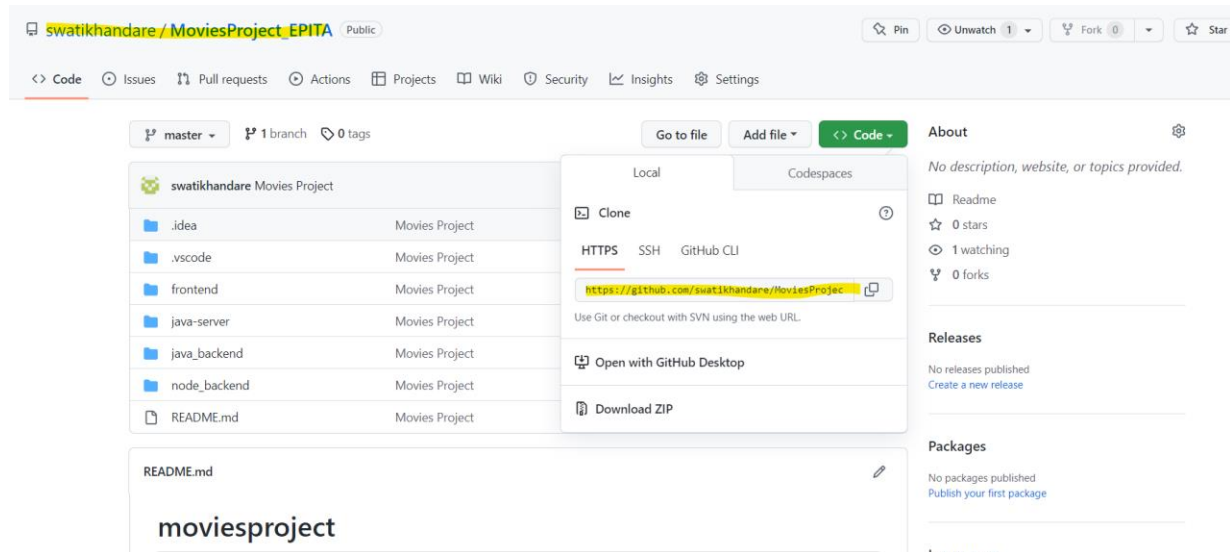
1. Create User Login
2. After login, below options are displayed:
  - HomePage Trailer
  - Continue Watching
  - Latest Movies
  - Recommended Movies
  - Top 10 Most Popular Movies
  - Top 10 Most Rated Movies

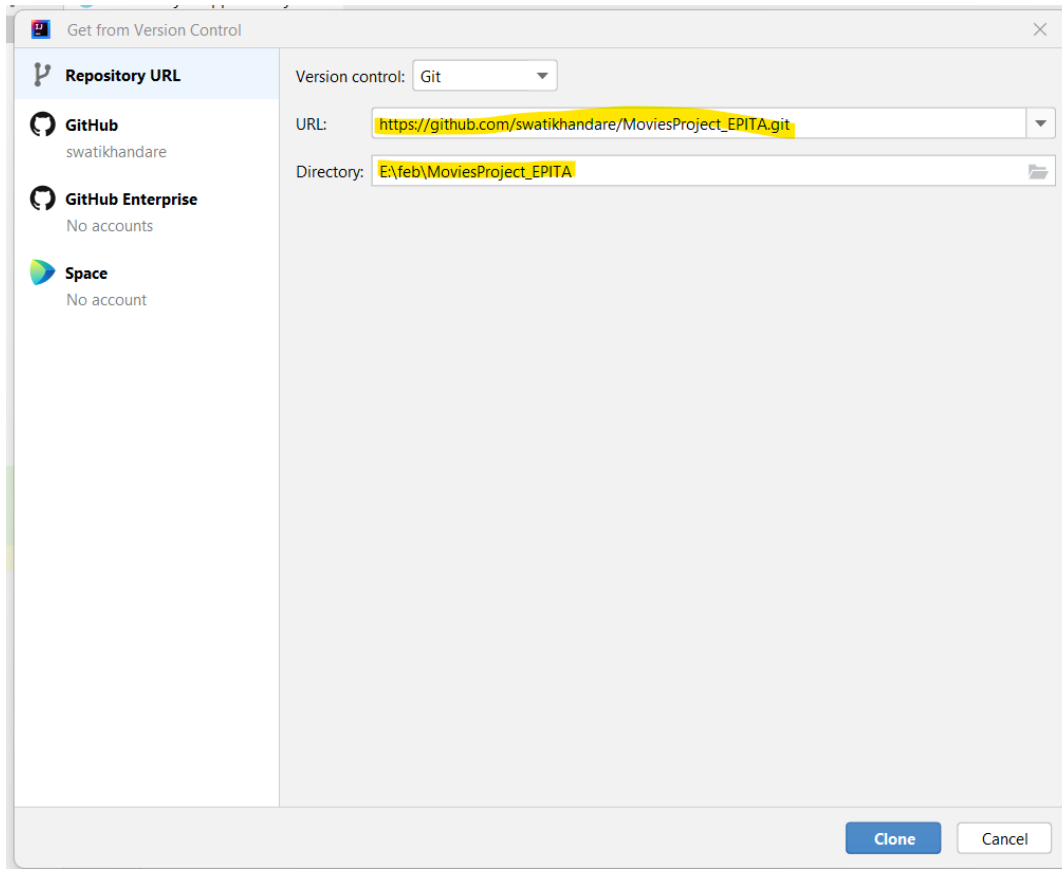
Click on play button to watch the trailer!

## Compilation and Installation Steps

1. Copy the URL link and clone the project in IntelliJ.

URL: [https://github.com/swatikhandare/MoviesProject\\_EPITA.git](https://github.com/swatikhandare/MoviesProject_EPITA.git)





2. Start the PostgreSQL and create the database using the details from “java\_backend/src/main/resources/application.properties” file.
3. Create MongoDB cluster and collection with user and password on MongoDB Atlas
4. Create a file “.env” at path “node\_backend” and add the MonogoDB details (for reference you check file “node\_backend/.env.example”)
5. Run the SpringBootApplication at path “java\_backend/src/main/java/fr/epita/java\_backend/MoviesProjectApplication.java” to start the application.

The screenshot shows the VS Code editor with the file `MoviesProjectApplication.java` open. The file content is as follows:

```

1 package fr.epita.java_backend;
2 import ...
3
4
5 1 usage
6 @SpringBootApplication
7 public class MoviesProjectApplication {
8     public static void main(String[] args) { SpringApplication.run(MoviesProjectApplication.class, args); }
9 }
10
11
12

```

The Run console at the bottom displays the following logs:

```

2023-02-17 21:49:17.088 INFO 31696 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat initialized with port(s): 8080 (http)
2023-02-17 21:49:17.101 INFO 31696 --- [main] o.apache.catalina.core.StandardService : Starting service [Tomcat]
2023-02-17 21:49:17.102 INFO 31696 --- [main] org.apache.catalina.core.StandardEngine : Starting Servlet engine: [Apache Tomcat/9.0.64]
2023-02-17 21:49:17.538 INFO 31696 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-02-17 21:49:17.539 INFO 31696 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 2062 ms
2023-02-17 21:49:17.989 INFO 31696 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [name: default]
2023-02-17 21:49:18.083 INFO 31696 --- [main] org.hibernate.Version : HHH000412: Hibernate ORM core version 5.6.14.Final
2023-02-17 21:49:18.360 INFO 31696 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.1.2.Final}
2023-02-17 21:49:18.502 INFO 31696 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-02-17 21:49:18.817 INFO 31696 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-02-17 21:49:18.848 INFO 31696 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2023-02-17 21:49:19.763 INFO 31696 --- [main] o.h.e.t.j.p.i.JtaPlatformInitiator : HHH000490: Using JtaPlatform implementation: [org.hibernate.engine.transaction.jta.platform.internal.NoJtaPlatform]
2023-02-17 21:49:19.772 INFO 31696 --- [main] j.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
2023-02-17 21:49:20.426 WARN 31696 --- [main] JpaBaseConfiguration$JpaWebConfiguration : spring.jpa.open-in-view is enabled by default. Therefore, database queries may run in the application context.
2023-02-17 21:49:21.504 INFO 31696 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-02-17 21:49:21.521 INFO 31696 --- [main] f.e.j.MoviesProjectApplication : Started MoviesProjectApplication in 6.816 seconds (JVM running for 7.7)

```

6. Open the project in VSCode.

7. Navigate to folder “node\_backend” and execute command “npm install”.

The screenshot shows the VS Code editor with the `README.md` file open. The file content is as follows:

```

1 # MoviesProject
2 Movies Project is developed during SE 3rd semester using React, Node, Express, MongoDB, Java and PostgreSQL
3
4 Technical Specification Document contains the software and hardware requirement for the project.
5
6 Production document contains the installation steps and steps to execute the project.

```

The terminal at the bottom shows the following commands and output:

```

PS D:\S3\MoviesProject_EPITA> cd .\node_backend\
PS D:\S3\MoviesProject_EPITA\node_backend> npm install
[#####] | IdealTree:mongoose: [#####] IdealTree:node_modules/mongoose Completed in 1312ms

```

8. Execute command “npm run dev” to start the backend server.

```

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

●
● added 206 packages, and audited 207 packages in 18s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
PS D:\S3\MoviesProject_EPITA\node_backend> npm run dev

○ > server@1.0.0 dev
> nodemon index.js

[nodemon] 2.0.20
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
MongoDB Connected...
(node:2644) [MONGODB] DeprecationWarning: Mongoose: the `strictQuery` option will be switched back to `false` by default
in Mongoose 7. Use `mongoose.set('strictQuery', false);` if you want to prepare for this change. Or use `mongoose.set('s
trictQuery', true);` to suppress this warning.
(Use `node --trace-deprecation ...` to show where the warning was created)
Server running on port 3001

```

## 9. Navigate to folder “frontend” and execute command “npm install”.

```

EXPLORER
> OPEN EDITORS
MOVIESPROJECT_EPITA
> .idea
> .vscode
> frontend
> public
> src
> .gitignore
() package.json
> java_backend
> node_backend
> testAPI_postman_collection
① README.md

README.md
1 # MoviesProject
2 Movies Project is developed during SE 3rd semester using React, Node, Express, MongoDB, Java and PostgreSQL
3
4 Technical Specification Document contains the software and hardware requirement for the project.
5
6 Production document contains the installation steps and steps to execute the project.

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

● PS D:\S3\MoviesProject_EPITA> cd .\node_backend\
● PS D:\S3\MoviesProject_EPITA\node_backend> npm install

added 206 packages, and audited 207 packages in 18s

16 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
○ PS D:\S3\MoviesProject_EPITA\node_backend>

```

## 10. Execute command “npm run start” to start the front end.

```

TERMINAL

○ PS D:\S3\MoviesProject_EPITA\frontend> npm run start

```

SOLE TERMINAL

Compiled successfully!

You can now view **client** in the browser.

Local: <http://localhost:3000>

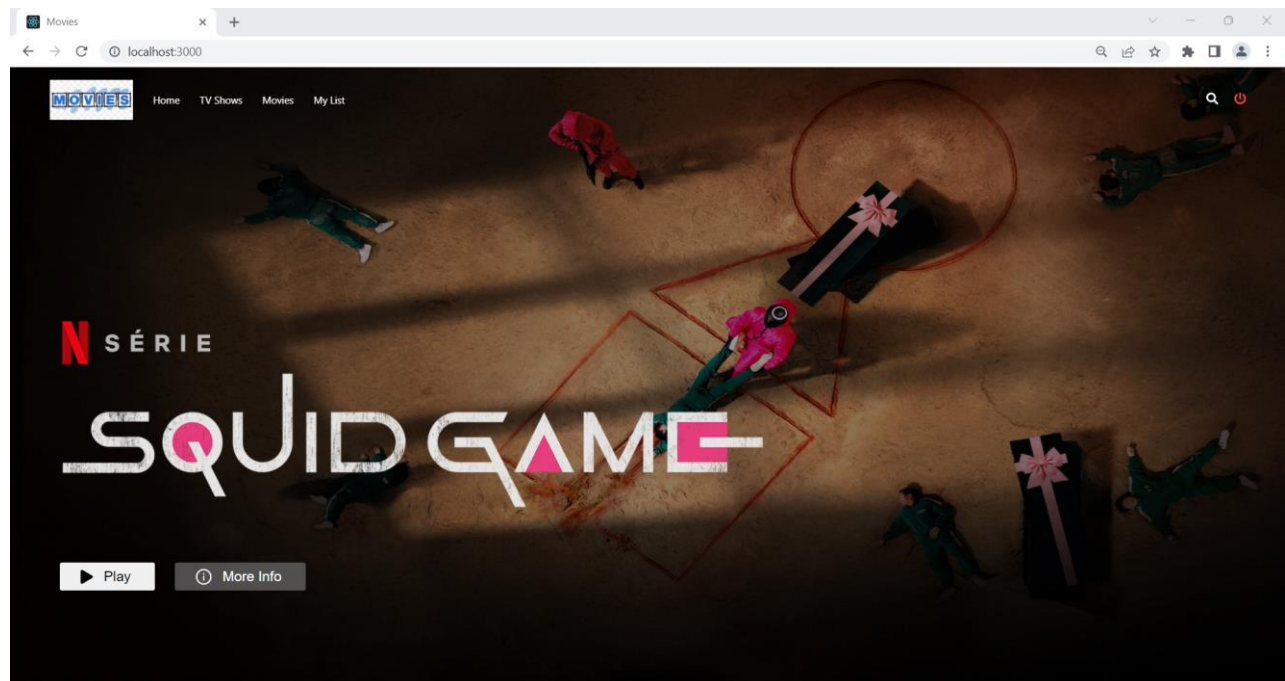
On Your Network: <http://10.188.83.198:3000>

Note that the development build is not optimized.  
To create a production build, use `npm run build`.

webpack compiled successfully

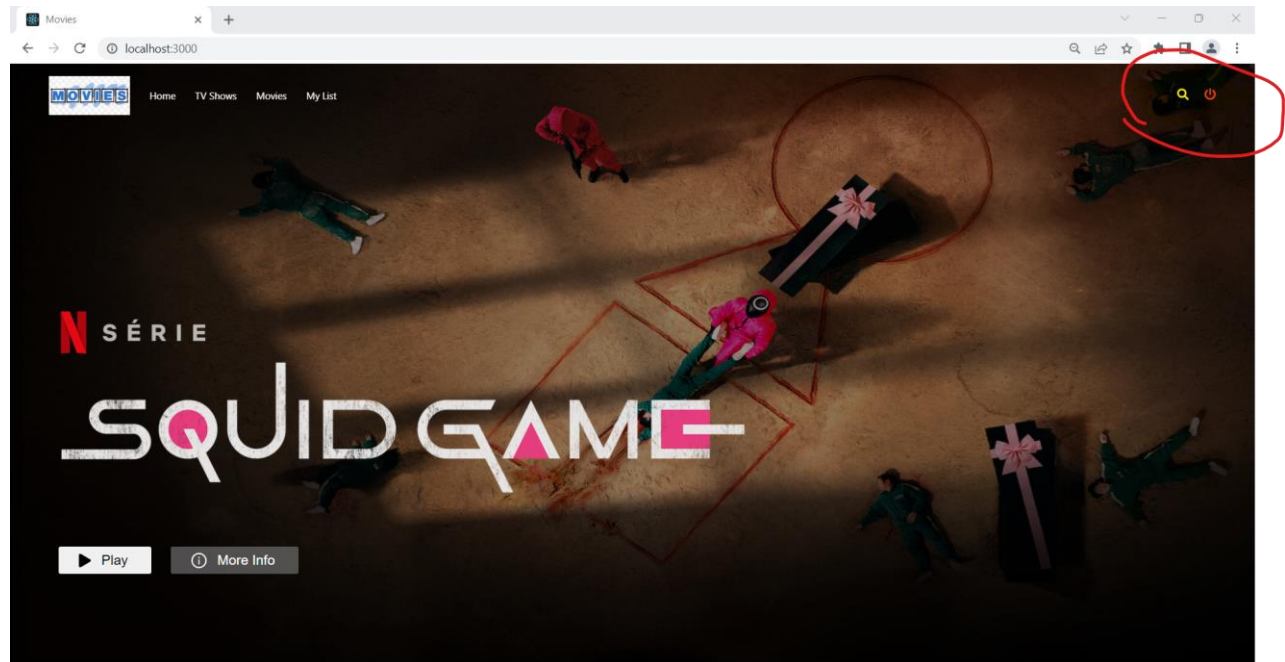
□

11. The application will be launched in your localhost at port 3000!

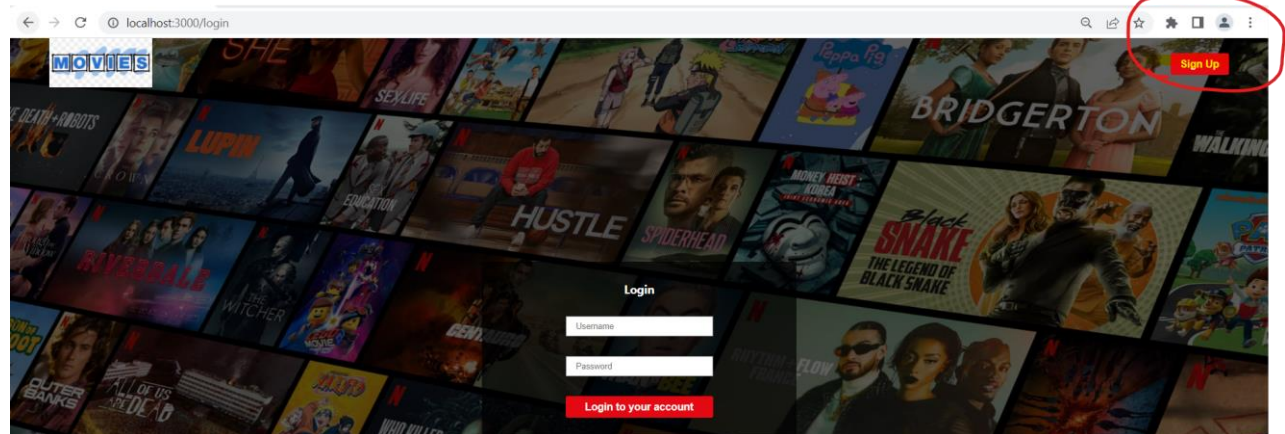


## Execution

### 1. Landing page



## 2. Sign Up





MOVIES

localhost:3000/signup

Sign In

Username

First Name

Last Name

dd- - - - - yyyy

Male

Sign Up

MOVIES

localhost:3000/signup

Sign In

dummyuser

dummy

user

17 - Jun - 2000

Female

Sign Up

MOVIES

localhost:3000/login

Sign Up

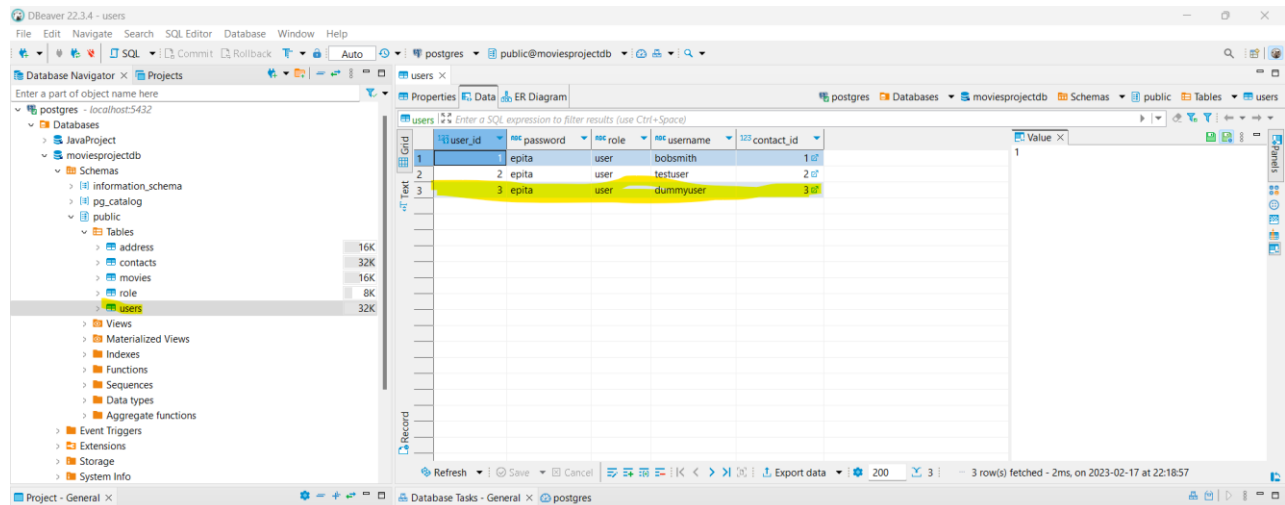
Login

bobsmith

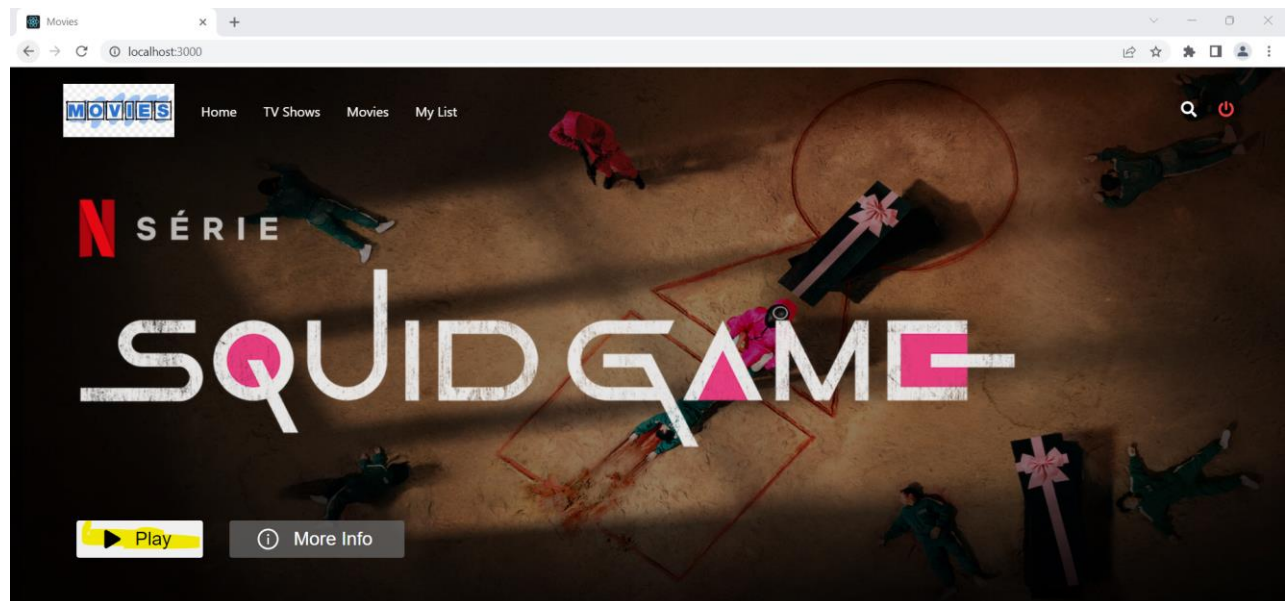
----

Login to your account

### 3. User added in users postgresql db

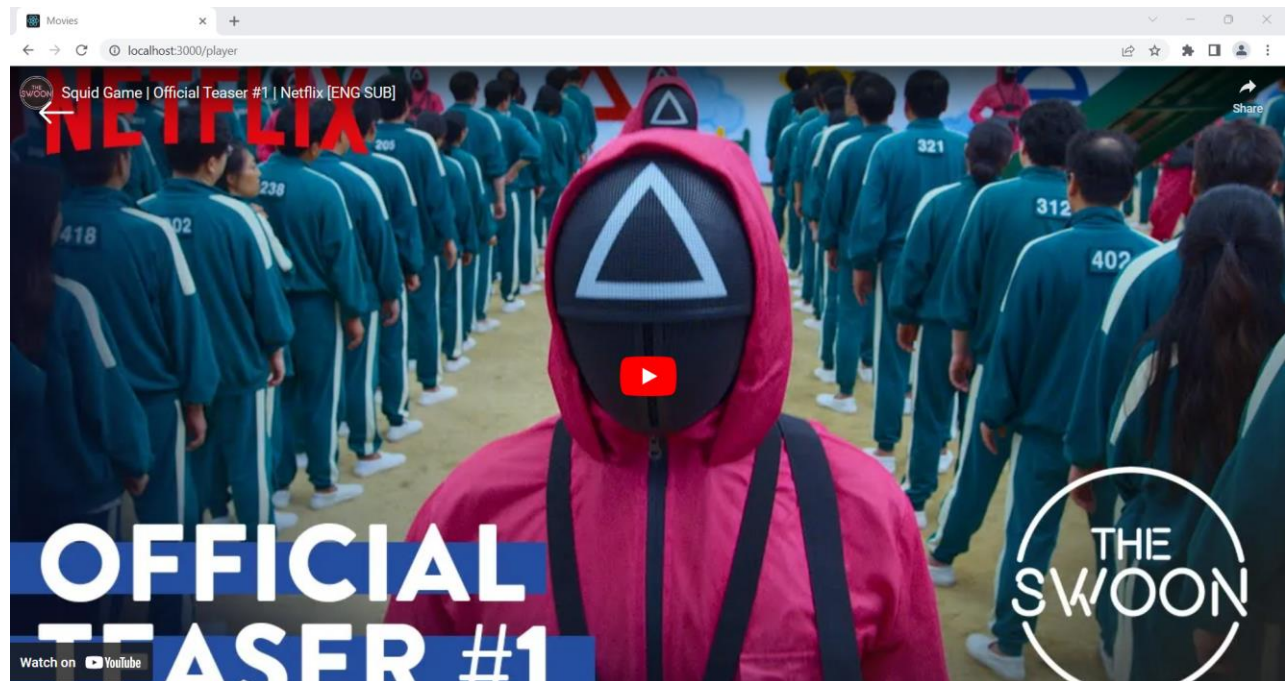


### 4. Sign in with the user created

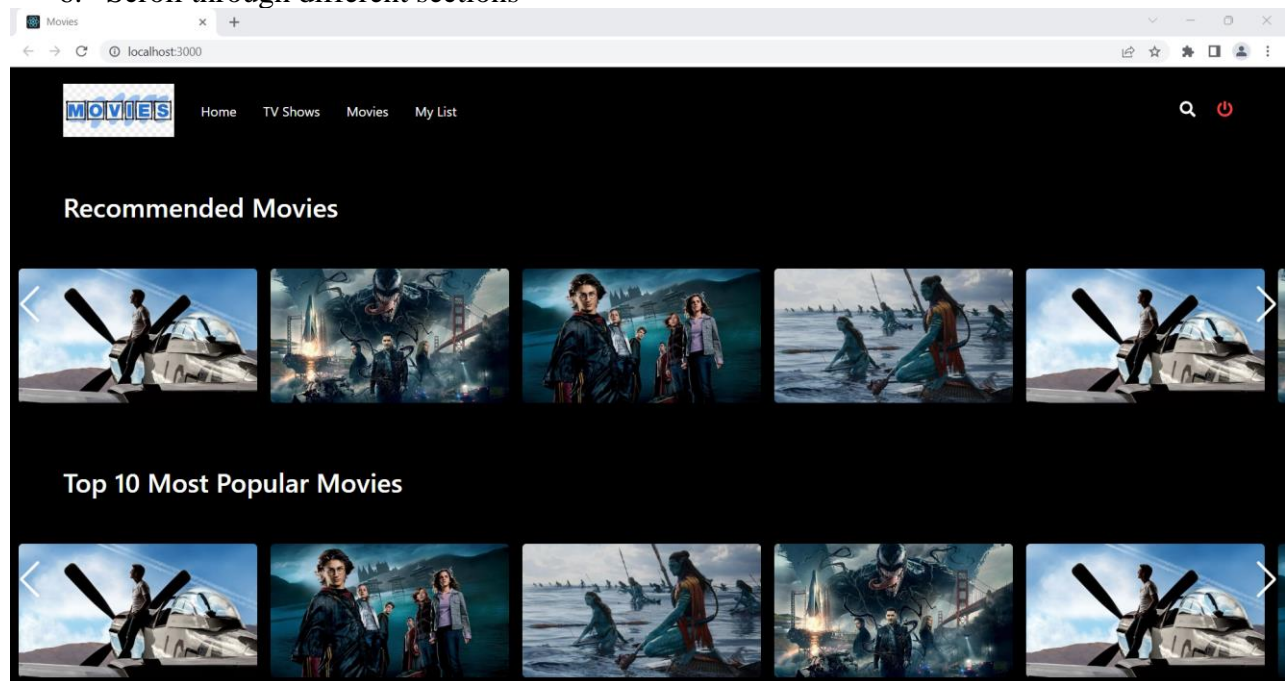


### 5. Click on play to play the Squid Game trailer.

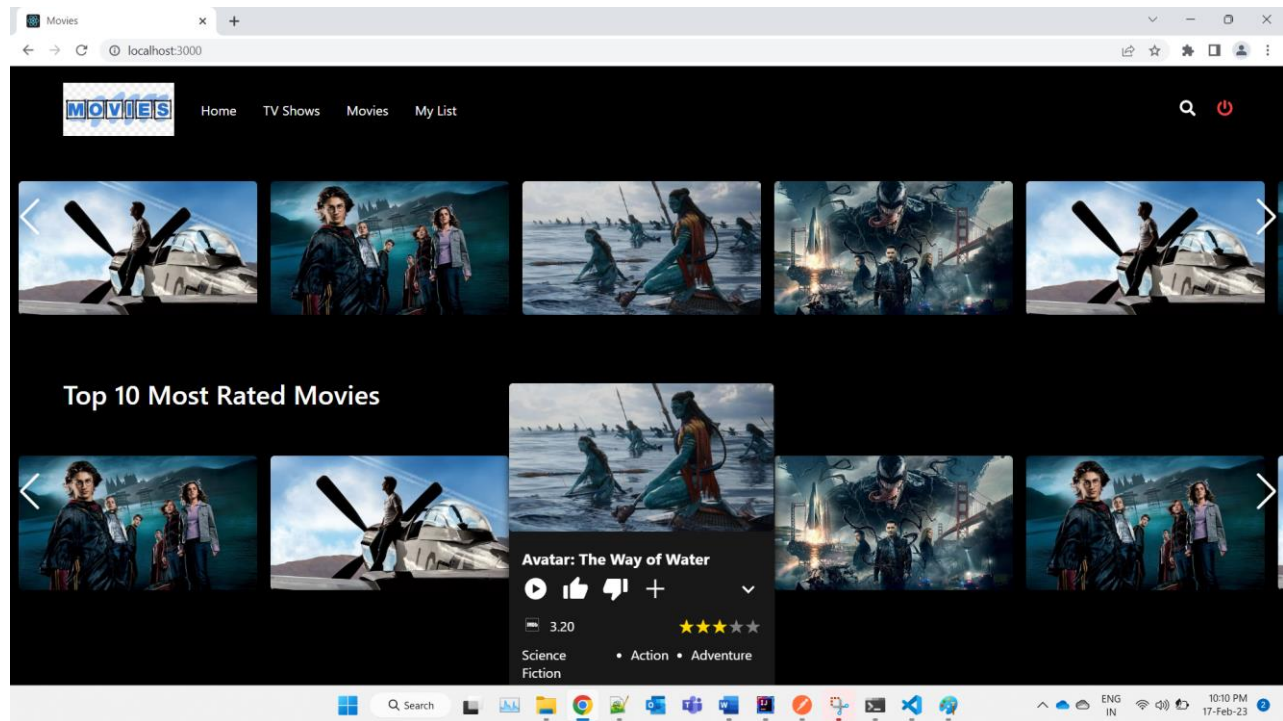




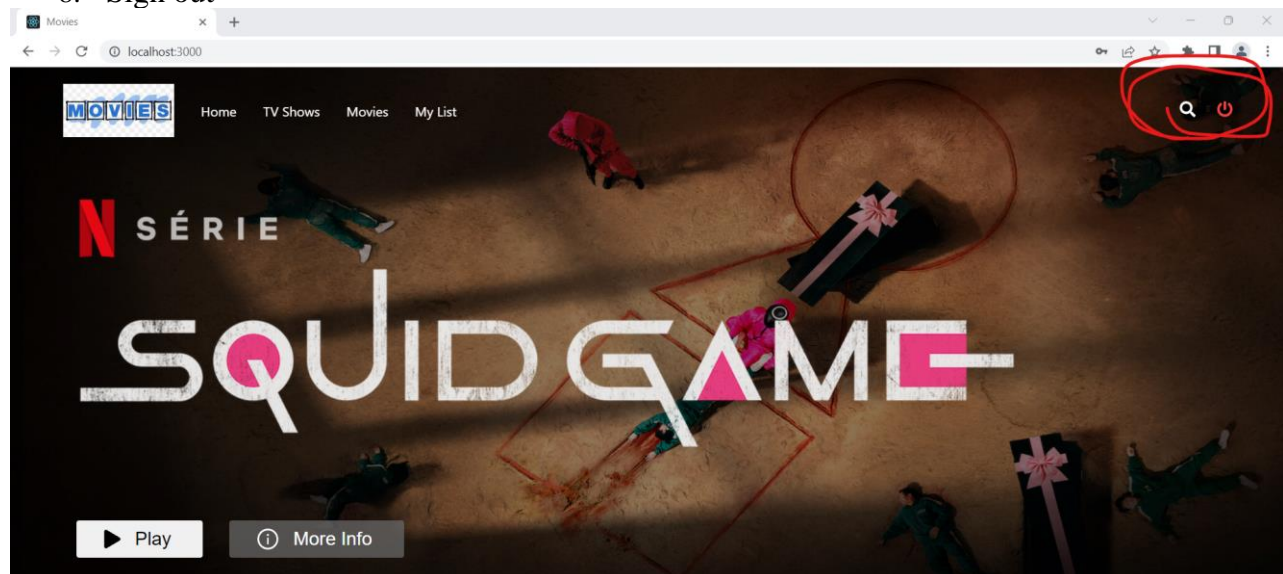
## 6. Scroll through different sections



## 7. Check for the ratings by hovering over a particular movie



## 8. Sign out



## API testing

Import the postman collection from .json

“testAPI\_postman\_collection.postman\_collection.json” to your local postman to test the **java backend** and **node backend** for APIs “api/users”, “api/movies/” and “api/movies/rating/”.

## 1. Register user API

testAPI\_postman\_collection / java\_backend / registerUser

POST `http://localhost:8080/api/users/register`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "username": "testuser",
3   "password": "epita",
4   "role": "user",
5   "contact": {
6     "first_name": "test",
7     "last_name": "user",
8     "date_of_birth": "2000-05-15",
9     "gender": "female",
10    "contact_email": "testuser@gmail.com"
11  }
12 }

```

Status: 201 Created Time: 156 ms Size: 473 B Save Response

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "user_id": 2,
3   "username": "testuser",
4   "password": null,
5   "role": "user",
6   "contact": {
7     "contact_id": 2,
8     "first_name": "test",
9     "last_name": "user",
10    "date_of_birth": "2000-05-15",
11    "gender": "female"
12 }
13 }
14 }

```

## 2. User login API

testAPI\_postman\_collection / java\_backend / userLogin

POST `http://localhost:8080/api/users/login`

Params Authorization Headers (8) Body Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```

1 {
2   "username": "testuser",
3   "password": "epita"
4 }

```

Status: 200 OK Time: 14 ms Size: 468 B Save Response

Body Cookies Headers (8) Test Results

Pretty Raw Preview Visualize JSON

```

1 {
2   "password": null,
3   "role": "user",
4   "contact": {
5     "contact_id": 2,
6     "first_name": "test",
7     "last_name": "user",
8     "date_of_birth": "2000-05-15",
9     "gender": "female",
10    "contact_email": "testuser@gmail.com"
11  }
12 }
13 }
14 }

```

New user is added in the postgresql db table

DBeaver 22.3.4 - users

Database Navigator: public@moviesprojectdb

Table: users

user_id	password	role	username	contact_id
1	epita	user	bobsmith	1
2	epita	user	testuser	2
3	epita	user	dummyuser	3

### 3. Add movie API

Postman: POST addMovie

URL: `http://localhost:3001/api/movies/addMovie`

Method: POST

Body (JSON):

```
{  "themoviedb_movie_id": 585083,  "title": "Hotel Transylvania: Transformania",  "image": "/qBLEWjNVsehJKEJqIigPswyBse.jpg",  "popularity": 709.095,  "release_date": "2022-02-25",  "director": "Derek Drymon",  "genres": [    "Animation",    "Family",    "Fantasy"  ],  "movieDirector": "Derek Drymon",  "popularity": 709.095,  "backdrop_image": "/qBLEWjNVsehJKEJqIigPswyBse.jpg",  "id": "63afe190e349b8c699bb07c7",  "ratings": []}
```

Status: 201 Created | Time: 27 ms | Size: 1.01 KB

### 4. Add movie rating by movie id API



The screenshot shows the Postman interface with a POST request selected. The URL is `http://localhost:3001/api/movies/rating/63efef900349b85699bb07c7`. The request body is a JSON object:

```

1 {
2   "rating": "4.1",
3   "comment": "Good",
4   "userId": 3
5 }

```

The response is a 201 status code, indicating the movie was successfully created. The response body is a JSON object:

```

11 {
12   "movieDirector": "Wesley Wyman",
13   "popularity": 799.095,
14   "backdrop_image": "/qBLKwJNVseh3kEjIgfPswyBse.jpg",
15   "_id": "63efef900349b85699bb07c7",
16   "ratings": [
17     {
18       "rating": "4.1",
19       "comment": "Good",
20       "userId": "3",
21       "_id": "63efef900349b85699bb07c7"
22     }
23   ]
24 }

```

## 5. Add average rating by movie id API

The screenshot shows the Postman interface with a PUT request selected. The URL is `http://localhost:3001/api/movies/rating/updateMovieAverageRating/63efef900349b85699bb07c7`. The request body is a JSON object:

```

1 {
2   "rating": "4.1",
3   "comment": "Good",
4   "userId": 3
5 }

```

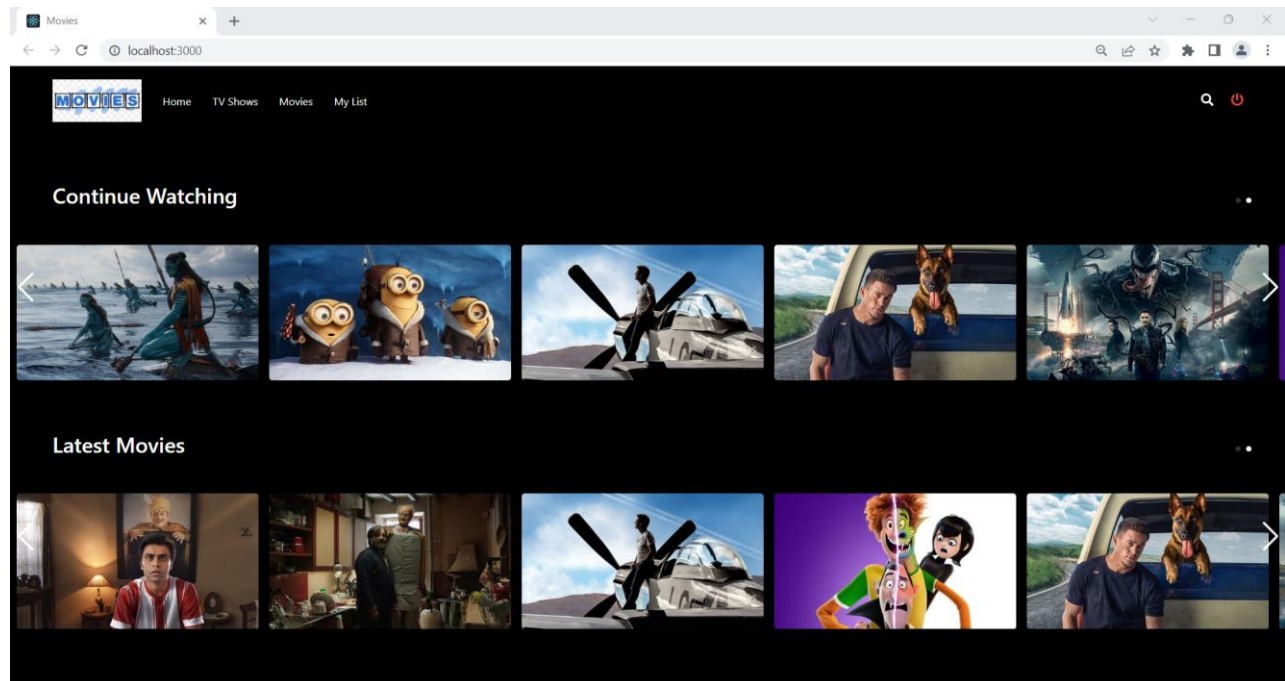
The response is a 201 status code, indicating the movie was successfully created. The response body is a JSON object:

```

20 {
21   "userId": "3",
22   "_id": "63efef900349b85699bb07c7",
23   "rating": "4.1",
24   "comment": "Good",
25   "userId": "3",
26   "_id": "63efef900349b85699bb07c7"
27 }
28 {
29   "rating": "4.1",
30   "comment": "Good",
31   "userId": "3",
32   "_id": "63efef900349b85699bb07c7"
33 }

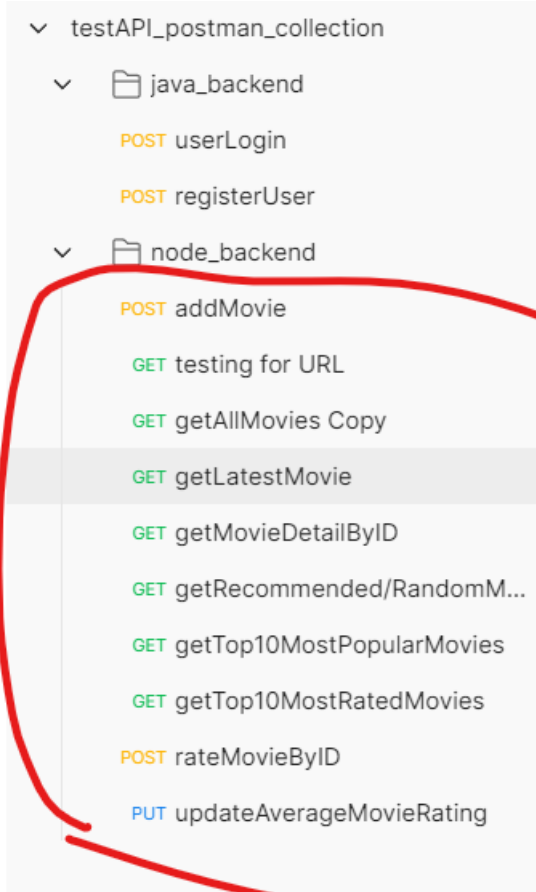
```

Movie is added



6. Other different test APIs for get all movies, get latest movies, get movies details , get top 10 movies etc.





S