**Vasantdada Patil Pratishthan's College of**

**Engineering & Visual Arts, Sion Mumbai - 400 022**

**A**

**MINI PROJECT REPORT**

**ON**

“Movies DB”

Submitted in partial fulfilment of the requirements of the degree

**BACHELOR OF ENGINEERING IN COMPUTER ENGINEERING**

Submitted By:

**Prathamesh Malekar (VU1F2021127)**

**Sahil Dhuri (VU1F2021129)**

**Mayuresh Ovhal (VU12021130)**

**Mohit Parmar (VU1F2021140)**

**Kinjal Dhumal(VU1F2021136)**

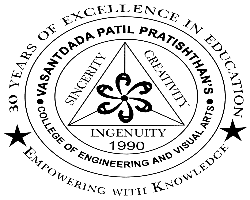
Under the guidance of

**Prof. Vinod Alone**

**Department of Computer Engineering**

**Vasantdada Patil Pratishthan's College of Engineering &**

**Visual Arts, Sion Mumbai - 400 022**



**University of Mumbai**

**(AY 2021-22)**

**CERTIFICATE**

This is to certify that the Mini Project entitled “**Movies DB”** is a Bonafide work of **Prathamesh Malekar (VU1F2021127) , Sahil Dhuri(VU1F2021129) , Mayuresh Ovhal(VU1F2021130), Mohit Parmar (VU1F2021140), Kinjal Dhumal (VU1F2021136)** submitted to the University of Mumbai in partial fulfilment of the requirement for the award of the degree of **“Bachelor of Engineering”** in **“Computer Engineering”.**

### **Prof. Vinod Alone**

Supervisor

###### **Prof. Mahavir Devmane** **Prof. Alam Shaikh**

Head of Department Principal

# **INSTITUTE VISION & MISSION**

**VISION:**

To educate, encourage and explore students by facilitating conducive environment to achieve professional goals

**MISSION:**

To kindle the zeal among the student and promote their quest for academic excellence with industry interaction to enhance their career opportunities

# **COMPUTER ENGINEERING DEPARTMENT**

**VISION:**

To inculcate skills for overall development of students to be a leader in the world of computer engineering and contributes in favour of society.

**MISSION:**

1. To provide students with a fundamental knowledge of theory, practical and problem-solving skills with an exposure to emerging technologies.
2. Provide platform for overall growth and adapting challenges in rapidly changing technology.
3. To produce globally competent computer professionals with moral values and leadership abilities for sustainable development of the society.

## PROGRAM EDUCATIONAL OBJECTIVES (PEO's)

1. To create graduates with sound fundamental knowledge of computer engineering & enhance their skillset towards emerging technologies.
2. To inculcate the skills among students to formulate, analyze and propose the solutions to engineering problems.
3. To make the students aware of professional ethics of the software industry.
4. To motivate the students for life-long self-learning.

## PROGRAM SPECIFIC OUTCOMES (PSOs)

1. Graduates of programme will be able to provide effective and efficient real time solutions using practical knowledge in Computer Engineering domain.
2. Graduate of programme will be able to use engineering practices, strategies and tactics for the development, operation and maintenance of software system.

## PROGRAM OUTCOMES (POs)

1. **Engineering knowledge:** Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.

1. **Problem analysis:** Identify, formulate complex engineering problems reaching substantiated conclusions using principles of Computer Engineering.

1. **Design / development of solutions:** Design / develop solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the society.

1. **Conduct investigations of complex problems:** Use knowledge for the design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.

1. **Modern tool usage**: Create, select and apply appropriate techniques and modern engineering tools, including predictions and modelling to complex engineering activities with an understanding of the limitations.

1. **The engineer and society**: Apply the knowledge to assess social issues and the responsibilities relevant to engineering practices.

1. **Environment and sustainability:** Understand the impact of the professional engineering solutions in social and environmental contexts, and demonstrate the knowledge for sustainable development.

1. **Ethics:** Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.

1. **Individual and team work:** Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.

1. **Communication:** Communicate effectively such as, being able to comprehend and write effective reports and design documentation, make effective presentations.

1. **Project management and finance:** Demonstrate knowledge and understanding of the engineering and management skills and apply these skills to manage projects effectively.

1. **Life-long learning:** Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

# **Mini Project Approval**

##### This Mini Project entitled “**Movies DB**” by **Prathamesh Malekar(VU1F2021127), Sahil Dhuri (VU1F2021129), Mayuresh Ovhal(VU1F2021130), Mohit Parmar(VU1F2021140), Kinjal Dhumal (VU1F2021136)** is approved for the degree of **Bachelor of Engineering** in **Computer Engineering.**

## Examiners

**1………………………………………**

(Internal Examiner Name & Sign)

**2…………………………………………**

(External Examiner name & Sign)

Date:

Place:

**Index**

|  |  |
| --- | --- |
| **Sr.No** | **Contents** |
| 1 | Introduction to Javascript |
| 1.1 | Explanation about Mini Project |
| 1.2 | List of Javascript Concepts/Frameworks used in Mini Project |
| 1.3 | Explanation of Each & Every Javascript Concept/Framework used in Mini Project |
| 1.4 | Requirements |
| 1.5 | Objectives |
| 2 | Coding/Implementation |
| 3 | Outputs |
| 4 | Conclusion |
| 5 | References |

**1. Introduction to Javascript**

JavaScript is a cross-platform, object-oriented scripting language used to make webpages interactive (e.g., having complex animations, clickable buttons, popup menus, etc.). There are also more advanced server side versions of JavaScript such as Node.js, which allow you to add more functionality to a website than downloading files (such as realtime collaboration between multiple computers). Inside a host environment (for example, a web browser), JavaScript can be connected to the objects of its environment to provide programmatic control over them.

# **Features of Javascript:**

This object-centered script language most commonly used for designing web pages which are a standalone language developed in Netscape. It gives the user extra control over the browser with potential in the creation of new functions in scripts. This scripting language features case sensitive input with the detection of the user’s browser and operating system. JavaScript is mostly used by validation in client edge technology

1. Object-Centered Script Language
2. Client edge Technology
3. Validation of User’s Input
4. Else and If Statement
5. Platform Independent
6. Generating HTML content
7. Light Weight
8. Case sensitive
9. Dynamic Typing
10. Default Parameters
11. Numeric separators
12. Async Processing

**Application:**

1. Movies DB is used for seeing details of the movies/series which one might be interested in.
2. **It's used for information purpose.**

**The Advantages of Javascript are**

1) Speed  
Since JavaScript is an**‘interpreted’** language, it reduces the time required by other programming languages like [**Java**](https://data-flair.training/blogs/java-tutorial/) for compilation. JavaScript is also a client-side script, speeding up the execution of the program as it saves the time required to connect to the server.

2) Simplicity  
JavaScript is easy to understand and learn. The structure is simple for the users as well as the developers. It is also very feasible to implement, saving developers a lot of money for developing dynamic content for the web.

3) Popularity  
Since all modern browsers support JavaScript, it is seen almost everywhere. All the famous companies use JavaScript as a tool including *Google, Amazon, PayPal, etc.*

4) Interoperability  
JavaScript works perfect with other programming languages and therefore numerous developers prefer it in developing many applications. We can embed it into any webpage or inside the script of another programming language.

5) Server Load  
As JavaScript operates on the client-side, data validation is possible on the browser itself rather than sending it off to the server. In case of any discrepancy, the whole website needs not to be reloaded. The browser updates only the selected segment of the page.

6) Rich Interfaces  
JavaScript provides various interfaces to developers for creating catchy webpages. Drag and drop components or sliders may give a rich interface to the webpages. This leads to improved user-interactivity on the webpage.

7) Extended Functionality  
Third-party add-ons likeGreasemonkey (a Mozilla Firefox extension) allow the developers to add snippets of predefined code in their code to save time and money. These add-ons help the developers build [JavaScript applications](https://data-flair.training/blogs/javascript-uses/) a lot faster and with much more ease than other programming languages

8) Versatility

JavaScript is now capable of front-end as well as back-end development. Back-end development uses NodeJS while many libraries help in front-end development like [AngularJS](https://data-flair.training/blogs/angularjs-tutorials-home/)*,* **ReactJS***,* etc

**The Disadvantages of Javascript** **are**

#### 1) Client-side Security Client-side Security. Since the JavaScript code is viewable to the user, others may use it for malicious purposes. These practices may include using the source code without authentication. Also, it is very easy to place some code into the site that compromises the security of data over the website.

2) Browser Support  
The browser interprets JavaScript differently in different browsers. Thus, the code must be run on various platforms before publishing. The older browsers don’t support some new functions and we need to check them as well.

#### 3) Lack of Debugging Facility

#### Though some HTML editors support debugging, it is not as efficient as other editors like C/[C++](https://en.wikipedia.org/wiki/C%2B%2B) editors. Also, as the browser doesn’t show any error, it is difficult for the developer to detect the problem

#### 4) Single Inheritance

#### JavaScript only supports single inheritance and not multiple inheritance. Some programs may require this object-oriented language characteristic.

#### 5) Rendering Stopped

A single code error can stop the rendering of the entire JavaScript code on the website. To the user, it looks as if JavaScript was not present. However, the browsers are extremely tolerant of these errors.

**1.1 Explanation about Mini Project**

**Movies DB**

Movies DB is a website which is used for information purpose about movies.

* The objective is to give a detailed overview of a movie/series
* It was a frontend based project where popular javascript framework React.js was used for all the logic that was built on the frontend
* Two API’s were used in this project known as TMDB API and OMDB API
* One was used to fetch movies based on users request from the search bar and one is used to display top grossing movies/series.

**1.2 List of Javascript Frameworks used in Mini Project**

**React.js**

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

**API**

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

There are two different API’s called in this application. TMDB and OMDB API.

TMDB API:

This API is used to fetch movies/series based on their popularity and filter them by their genre etc.

OMDB API:

The OMDb API is a RESTful web service to obtain movie information, all content and images on the site are contributed and maintained by our users.

**1.3 Explanation of Each & Every Javascript Framework/Concept used in Mini Project**

**React.js**

React makes it painless to create interactive UIs. Design simple views for each state in your application, and React will efficiently update and render just the right components when your data changes.

Component Based:

Build encapsulated components that manage their own state, then compose them to make complex UIs.

Since component logic is written in JavaScript instead of templates, you can easily pass rich data through your app and keep state out of the DOM

React don’t make assumptions about the rest of your technology stack, so you can develop new features in React without rewriting existing code.

**API**

API is the acronym for Application Programming Interface, which is a software intermediary that allows two applications to talk to each other. Each time you use an app like Facebook, send an instant message, or check the weather on your phone, you're using an API.

**Example of an API:**

When you use an application on your mobile phone, the application connects to the Internet and sends data to a server. The server then retrieves that data, interprets it, performs the necessary actions and sends it back to your phone. The application then interprets that data and presents you with the information you wanted in a readable way. This is what an API is - all of this happens via API.

To explain this better, let us take a familiar example.

Imagine you’re sitting at a table in a restaurant with a menu of choices to order from. The kitchen is the part of the “system” that will prepare your order. What is missing is the critical link to communicate your order to the kitchen and deliver your food back to your table. That’s where the waiter or API comes in. The waiter is the messenger – or API – that takes your request or order and tells the kitchen – the system – what to do. Then the waiter delivers the response back to you; in this case, it is the food.

There are two different API’s called in this application. TMDB and OMDB API.

TMDB API:

This API is used to fetch movies/series based on their popularity and filter them by their genre etc.

You may learn more this API on their site:

https://developers.themoviedb.org/3/getting-started/introduction

OMDB API:

The OMDb API is a RESTful web service to obtain movie information, all content and images on the site are contributed and maintained by our users.

This API is used to fetch on users request or whatever he/she search for

You may learn more this API on their site:

https://www.omdbapi.com/

**1.4 Requirements**

* **SOFTWARE REQUIREMENTS:**
* Windows Operating System
* Pycharm Community Edition 2021.2.1
* Python SDK(3.10 or higher)
* **HARDWARE REQUIREMENTS:**
* 2 GB RAM
* Minimum 50MB Hard Disk Space
* Any Dual or Four core Processor

**1.5 Objective**

Movies DB is a website which holds information about movies which you want to watch/know

* The objective is to give a detailed overview of a movie/series
* It holds information like when was it released, what is the imdb rating, number of nominations and awards etc.
* Website can also display movies/series that are currently popular/trending worldwide.
* It can filter movies based on the genre of the movie.

**2.Coding/Implementation**

**Context.js:**

import React, { useContext, useState } from "react";

import { useEffect } from "react";

const AppContext = React.createContext();

export const AppProvider = ({children}) => {

    const [movies, setMovies] = useState([]);

    const [value, setValue] = useState('one piece');

    const [moviesActive, setMoviesActive] = useState(true);

    const [genresList, setGenresList] = useState([]);

    const [movieId, setMovieId] = useState(0);

    const [loaded, setLoaded] = useState(true);

    const [pages, setPages] = useState(0);

    const [details, setDetails] = useState([{}]);

    const [sidebar, setSidebar] = useState(false);

    const [tempMovies, setTempMovies] = useState([]);

    const newApi = async () => {

        setLoaded(false)

        const url = `https://www.omdbapi.com/?s=${value? value.length >= 3? value:'one piece':'one piece'}&apikey=${process.env.REACT\_APP\_OMDB\_API\_KEY}`

        const response = await fetch(url);

        const data = await response.json();

        setMovies(data.Search)

        setLoaded(true)

    }

    // genres fetch

    const genreUrl = `https://api.themoviedb.org/3/genre/movie/list?api\_key=${process.env.REACT\_APP\_MOVIE\_DB\_API\_KEY}&language=en-US`

    const fetchGenres = async() => {

        const response = await fetch(genreUrl);

        const data = await response.json();

        const {genres} = data

        setGenresList(genres);

    }

    // fetch single movie details

    const fetchTvdbSeries = async (id) => {

                const url = `https://api.themoviedb.org/3/find/${id}?api\_key=${process.env.REACT\_APP\_MOVIE\_DB\_API\_KEY}&language=en-US&external\_source=tvdb\_id`

                const response = await fetch(url);

                const data = await response.json()

                console.log(data);

                if (data.tv\_results || data.tv\_episode\_results){

                    setDetails(data.tv\_results || data.tv\_episode\_results);

                }

            }

    const fetchDetails = async (title, type, id) => {

        // setLoaded(false);

        const url = `https://www.omdbapi.com/?t=${title}&type=${type}&apikey=${process.env.REACT\_APP\_OMDB\_API\_KEY}`

        const response = await fetch(url)

        const data = await response.json();

        if (data.Error){

            console.log(data);

            fetchTvdbSeries(id);

        }

        else{

            setDetails([data]);

        }

        // setLoaded(true);

    }

    const fetchPopularMovies = async (pageNo) => {

        // setLoaded(false);

        let url = `https://api.themoviedb.org/3/discover/movie?api\_key=${process.env.REACT\_APP\_MOVIE\_DB\_API\_KEY}&language=en-US&sort\_by=popularity.desc&include\_adult=false&include\_video=false&page=${pageNo}&with\_watch\_monetization\_types=flatrate`

        let pagesList = [];

        const response = await fetch(url)

        const data = await response.json()

        setMovies(data.results);

        setPages(() => {

            for (let i = 1; i <= 500; i++){

                pagesList = [...pagesList, i]

            }

            return pagesList;

        });

        // setLoaded(true)

    }

    const fetchPopularSeries = async (pageNo) => {

        // setLoaded(false)

        const url = `https://api.themoviedb.org/3/discover/tv?api\_key=${process.env.REACT\_APP\_MOVIE\_DB\_API\_KEY}&language=en-US&sort\_by=popularity.desc&include\_adult=false&include\_video=false&page=${pageNo}&with\_watch\_monetization\_types=flatrate`;

        const response = await fetch(url);

        const data = await response.json();

        setMovies(data.results)

        // setLoaded(true)

    }

    useEffect(()=> {

        fetchGenres();

        console.log('useEffect called new API');

        newApi();

    }, [value])

    return <AppContext.Provider value={{tempMovies, setTempMovies, fetchPopularMovies, fetchPopularSeries, fetchDetails, fetchGenres, newApi, sidebar, setSidebar, moviesActive, setMoviesActive, details, setDetails, pages, setPages, loaded, setLoaded, movieId, setMovieId, genresList, movies, setValue, value, setMovies}}>{children}</AppContext.Provider>

}

export const useGlobalContext = () =>{

    return useContext(AppContext)

}

**Navbar.js:**

import {BiCameraMovie} from "react-icons/bi"

import {BsSearch} from 'react-icons/bs'

import { useGlobalContext } from "./context"

import { Link } from "react-router-dom"

import {GiHamburgerMenu} from 'react-icons/gi'

export const Navbar = () => {

    const {value, setValue, sidebar, setSidebar, newApi} = useGlobalContext();

    return <nav className={`${sidebar?'side-bar nav':'nav'}`}>

        <div onClick={() => setSidebar(true)} className="sidebar-btn">

            <GiHamburgerMenu className="ham" />

        </div>

        <div className="logo-container">

            <Link onClick={() => newApi()} to="/"><BiCameraMovie className="logo" /></Link>

            <h3 className="logo-title">Movies DB</h3>

        </div>

        <form action="/" onSubmit={(e) => {

            e.preventDefault();

            newApi();

            }} className="search-bar-form">

            <input onChange={(e) => setValue(e.target.value)} value={value} type="text" className="search-bar" name="search-bar" id="search" />

            <Link to="/"><button type="submit" className='btn-w-o-back'><BsSearch type="submit" className="search-icon" /></button></Link>

        </form>

        <div className="links-container">

            <ul className="nav-links">

                <Link key={'home'} onClick={() => newApi()} className="links" to="/"><li>Home</li></Link>

                <Link key="popularity" className="links" to="/popularity/1"><li>Popularity</li></Link>

                <Link key="Genre" className="links" to="/genre/Action/28/1"><li>Genre</li></Link>

            </ul>

        </div>

    </nav>

}

**Movies.js:**

import { useGlobalContext } from "./context";

import { Link } from "react-router-dom";

export const Movies = () => {

    const {movies, genresList} = useGlobalContext();

    return <section className="container">

        <section className="movies-section">

            <section className="movies-container">

                {

                movies?movies.map((movie, index) => {

                    const {Poster, Title, Type, Year, imdbID} = movie

                    return (

                    <div className="movie-details" key={index}>

                    <Link className="movies-link" to={`${Type}/details/${Title}/${imdbID}/`}>

                        <article onMouseOut={(e) => {e.currentTarget.lastElementChild.classList.add('hidden')}} onMouseOver={(e) => {e.currentTarget.lastElementChild.classList.remove("hidden")}} key={index} className="movie-card">

                            <img className="movie-poster" src={Poster} alt={Title} />

                            <div className={`movie-info links ${window.innerWidth > 886 && 'hidden'}`}>

                                <h3 className="font-weight">{Title}</h3>

                                <h4 className="font-weight">Release Year : {Year}</h4>

                            </div>

                        </article>

                    </Link>

                    </div>

                    )

                }):<h2 className="danger-heading">Nothing to display</h2>

                }

            </section>

        </section>

        <article className="genres-list">

            <h4 className="align-text">Genres</h4>

            <article className="genres-boxes">

            <article className="genre-box-1">

                {genresList.map((genre)=>{

                    const {name, id} = genre

                    return <Link key={id} className='links' to={`/genre/${name}/${id}/1`}><span className="movies-heading" onClick={() => console.log(id)}>{name}</span></Link>

                })}

            </article>

            </article>

        </article>

    </section>

}

**Popularity.js:**

import { useGlobalContext } from "./context"

import { Link, useParams } from "react-router-dom";

import { useEffect } from "react";

export const Popularity = () => {

    const { movies, fetchPopularMovies, fetchPopularSeries, pages, moviesActive, setMoviesActive } = useGlobalContext();

    const {currentPage} = useParams();

    useEffect(() => {

        if (moviesActive){

            fetchPopularMovies(currentPage);

        }

        else{

            fetchPopularSeries(currentPage)

        }

        console.log('useeffect called popularity');

    }, [currentPage, moviesActive])

    return <section className="popularity-container">

        <section className="btn-container">

            <button onClick={() => {

                setMoviesActive(true);

                fetchPopularMovies(currentPage);

                }} className="btn-popularity"><span className="center-margin">Movies</span> <div className={moviesActive?`underline underline-active`:`underline`}></div></button>

            <button onClick={() => {

                setMoviesActive(false);

                fetchPopularSeries(currentPage)

            }} className="btn-popularity"><span className="center-margin">Series</span> <div className= {moviesActive?`underline`:`underline underline-active`}></div></button>

        </section>

        <section className="movies-container container-organized">

                {

                movies?movies.map((movie, index) => {

                    const {poster\_path, title, release\_date, id, name, first\_air\_date} = movie

                    return (

                    <div className="movie-details" key={id}>

                    <Link to={`/${(first\_air\_date && 'series') || (release\_date && 'movie')}/details/${title || name}/${id}`}>

                        <article onMouseOut={(e) => {e.currentTarget.lastElementChild.classList.add('hidden')}} onMouseOver={(e) => {e.currentTarget.lastElementChild.classList.remove("hidden")}} key={index} className="movie-card">

                            <img className="movie-poster" src={`https://www.themoviedb.org/t/p/w220\_and\_h330\_face/${poster\_path}`} alt={title} />

                            <div className={`movie-info ${window.innerWidth > 600 && 'hidden'}`}>

                                <h3 className="font-weight">{name || title}</h3>

                                <h4 className="font-weight">Release Date : {release\_date || first\_air\_date}</h4>

                            </div>

                        </article>

                    </Link>

                    </div>

                    )

                }):<h2 className="danger-heading">Nothing to display</h2>

                }

            </section>

        <section className="pages-btn-container">

            {pages? pages.map((page, index) => {

                index = index + 1

                if (pages.length > 6){

                    if (index <= 3){

                        return <Link key={index} className="required-width" to={`/popularity/${index}`}><button onClick={() => {

                            if (moviesActive)

                            fetchPopularMovies(index)

                            else

                            fetchPopularSeries(index)

                        }} className={`${currentPage == index?`pagination-btn current-page-btn`:`pagination-btn`}`}>{index}</button></Link>

                    }

                    if (currentPage - index <= 2 && currentPage - index >= 1){

                            return <Link key={index} className="required-width" to={`/popularity/${index}`}><button onClick={() => {

                                if (moviesActive)

                            fetchPopularMovies(index)

                            else

                            fetchPopularSeries(index)

                            }} className="pagination-btn">{page}</button></Link>

                        }

                    if (currentPage == index){

                        return <Link key={index} className="required-width" to={`/popularity/${index}`}><button onClick={() => {

                            if (moviesActive)

                            fetchPopularMovies(page)

                            else

                            fetchPopularSeries(page)

                        }} className="pagination-btn current-page-btn">{page}</button></Link>

                    }

                    if (index > currentPage && index - currentPage <= 2){

                            return <Link key={index} className="required-width" to={`/popularity/${index}`}><button onClick={() => {

                                if (moviesActive)

                            fetchPopularMovies(index)

                            else

                            fetchPopularSeries(index)

                            }} className="pagination-btn">{index}</button></Link>

                        }

                    if (pages.length - index <= 2){

                        return <><div className="hidden"></div> <Link key={index} className="required-width" to={`/popularity/${index}`}><button onClick={() => {

                            if (moviesActive)

                            fetchPopularMovies(index)

                            else

                            fetchPopularSeries(index)

                        }} className={`${currentPage === index?`pagination-btn current-page-btn`:'pagination-btn'}`}>{index}</button></Link></>

                    }

                    else{

                        return <div className="hidden" key={index}>.</div>

                    }

                }

                else{

                    return <Link key={index} className="required-width" to={`/popularity/${index}`}><button onClick={() => {

                        if (moviesActive)

                            fetchPopularMovies(index)

                            else

                            fetchPopularSeries(index)

                    }} className={`${currentPage == index?`pagination-btn current-page-btn`:'pagination-btn'}`}>{index}</button></Link>

                }

            }) : <div className="hidden"></div> }

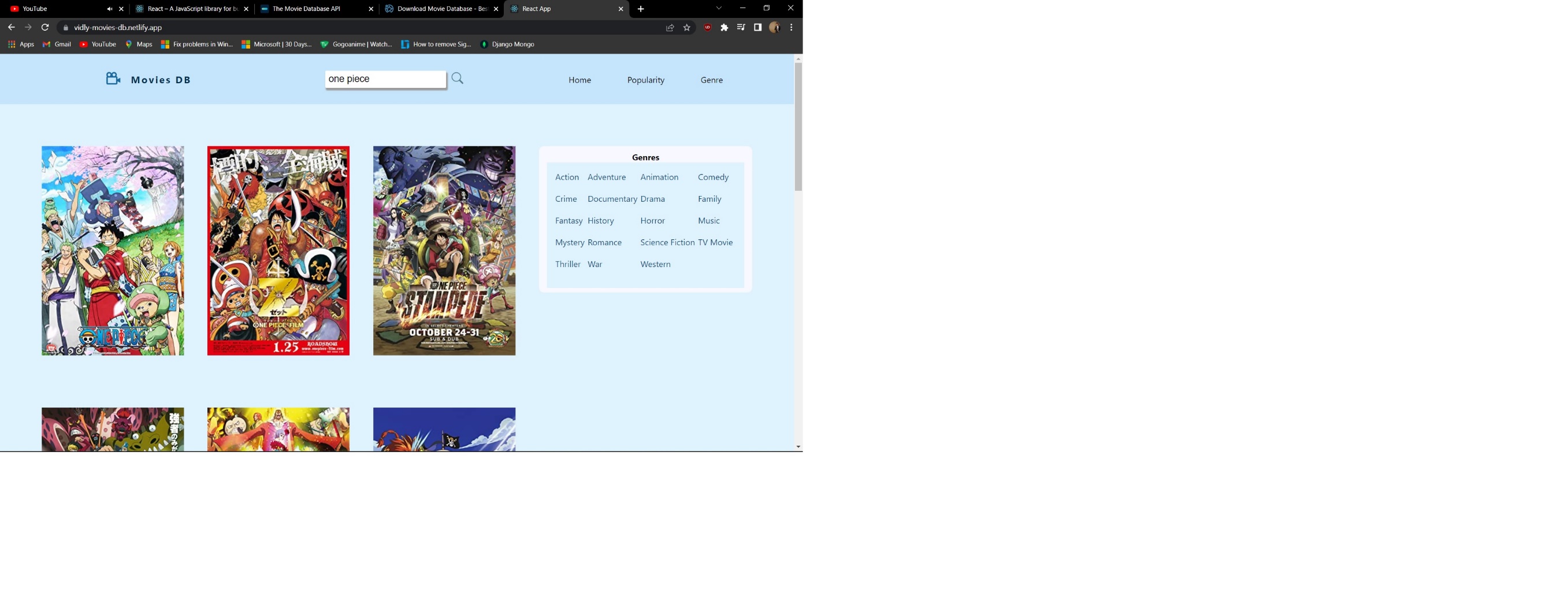
        </section>

    </section>

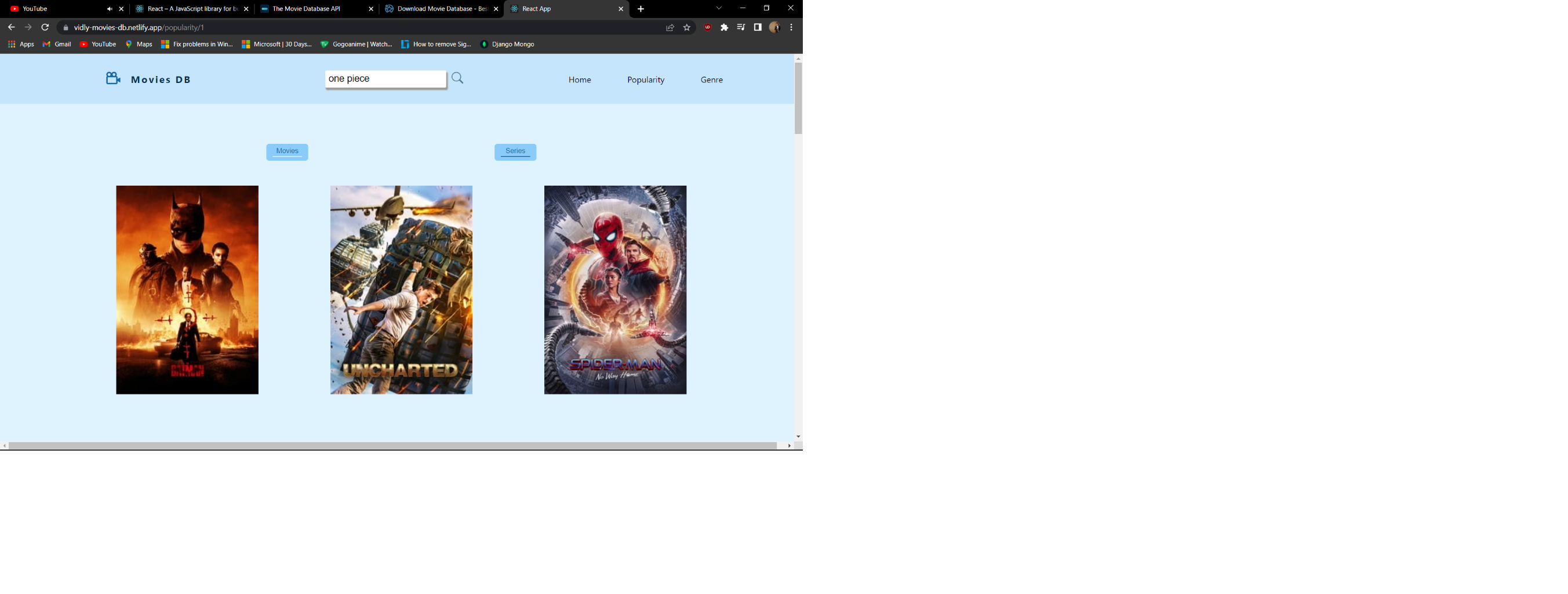
}

**3.Output & Results**

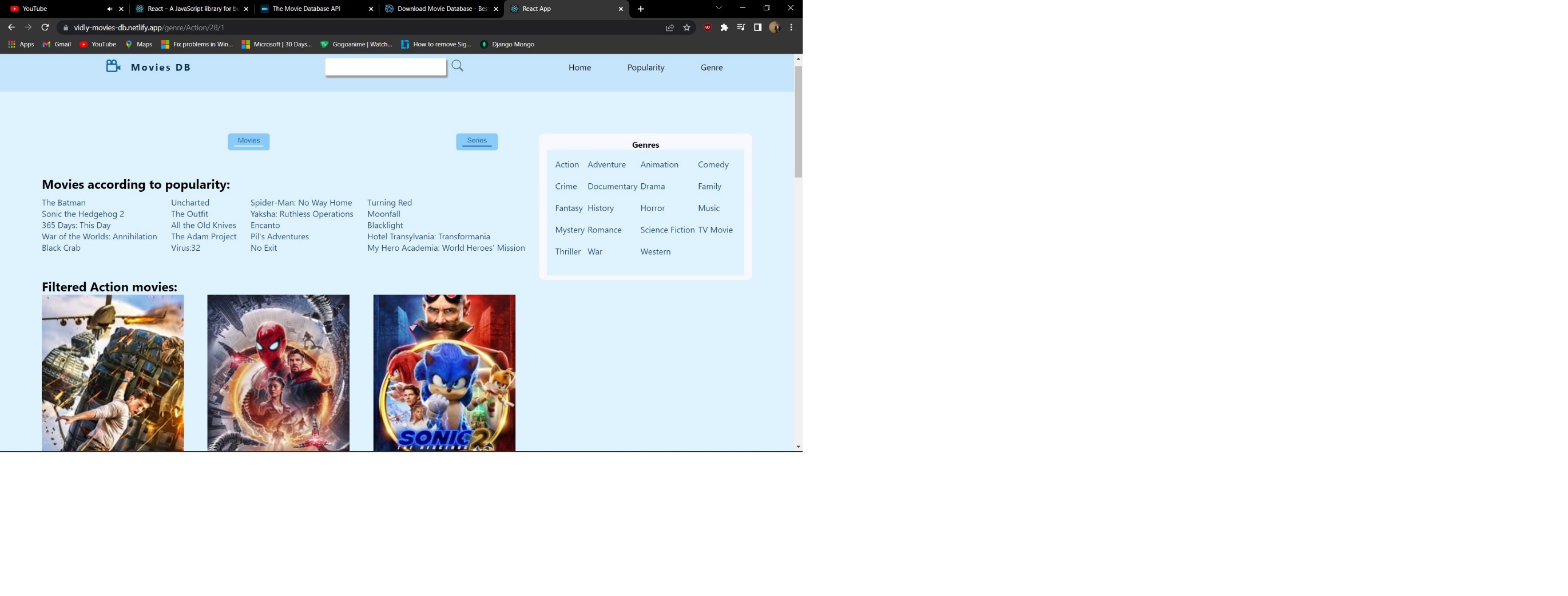
**Home:**



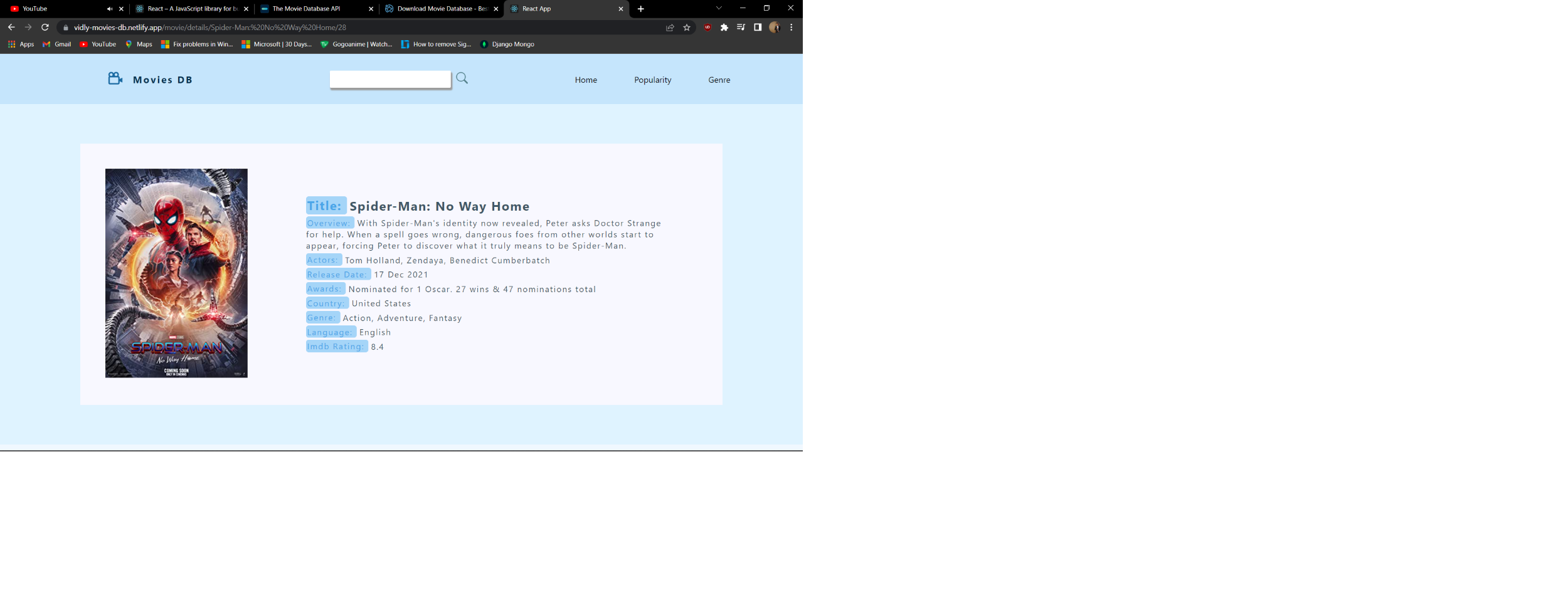
**Popularity:**



**Genre:**



**Movie Details:**



**4.Conclusion**

* The Site is tested and fetched requests a number of times.
* After Testing every page of the website we have verified that it fulfils and exceeds the requirements that were originally set which are described under Project.
* Additional functionality could be added with additional work and some optimization could be implanted though the project is functional and good as is.

**5.References**

[1] <https://github.com/mayuresh1606/react-projects>

[2] <https://reactjs.org/tutorial/tutorial.html>

[3] <https://www.themoviedb.org/documentation/api>

[4] <https://www.omdbapi.com/>