

MDB CLONE

**A PROJECT REPORT
for
Mini Project (KCA353)
Session (2023-24)**

Submitted by

**MohdNazim
2200290140094**

**Submitted in partial fulfilment of the
Requirements for the Degree of**

MASTER OF COMPUTER APPLICATION

**Under the Supervision of
Dr. Amit Kumar Goyal
Assistant Professor**



Submitted to

**DEPARTMENT OF COMPUTER APPLICATIONS
KIET Group of Institutions, Ghaziabad
Uttar Pradesh-201206**

JAN 2024

[Type text]

CERTIFICATE

Certified that **Mohd Nazim(2200290140094)** has carried out the project work having “**IMDB CLONE**” (**Mini Project-KCA353**) for **Master of Computer Application** from Dr. A.P.J. Abdul Kalam Technical University (AKTU) (formerly UPTU), Lucknow under my supervision. The project report embodies original work, and studies are carried out by the student himself/herself and the contents of the project report do not form the basis for the award of any other degree to the candidate or to anybody else from this or any other University/Institution.

Date:

Mohd Nazim(2200290140094)

This is to certify that the above statement made by the candidate is correct to the best of my knowledge.

Date:

Dr. Amit Kumar Goyal
Assistant Professor
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

Dr. Arun Tripathi
Head of Department
Department of Computer Applications
KIET Group of Institutions, Ghaziabad

ABSTRACT

In the ever-evolving world of entertainment, the need for a comprehensive and user-friendly movie database is paramount. IMDb Clone emerges as a solution to this demand, harnessing the power of cutting-edge technologies, with a primary focus on React, to offer users an immersive and efficient movie browsing experience.

This IMDb Clone aims to replicate the essence of the renowned IMDb platform while incorporating modern technologies to enhance user engagement and satisfaction. React, a highly versatile and widely adopted JavaScript library, plays a pivotal role in the development of this platform. React's component-based architecture empowers the IMDb Clone to deliver dynamic, responsive, and interactive user interfaces, ensuring seamless navigation and exploration of the vast world of cinema.

Key features of the IMDb Clone include a robust movie database, user-friendly search functionality, detailed movie information, user reviews and ratings, personalized user profiles, and a recommendation engine. React's efficiency in managing state and rendering complex UI components enables the IMDb Clone to provide users with real-time updates and personalized content recommendations, thereby enhancing their overall experience.

Furthermore, the IMDb Clone leverages other modern technologies such as GraphQL for efficient data retrieval, responsive design principles for optimal performance on various devices, and cloud-based storage for scalability and data redundancy. These technologies collectively contribute to the IMDb Clone's ability to provide a seamless, reliable, and feature-rich movie database platform.

In summary, IMDb Clone exemplifies the fusion of classic movie database functionality with contemporary web technologies. Its utilization of React as a core technology empowers the platform to deliver an engaging and user-friendly experience to movie enthusiasts and industry professionals alike.

ACKNOWLEDGEMENTS

Success in life is never attained single-handedly. My deepest gratitude goes to my project supervisor, **Dr. Amit Kumar Goyal** for his guidance, help, and encouragement throughout my project work. Their enlightening ideas, comments, and suggestions. Words are not enough to express my gratitude to Dr. Arun Kumar Tripathi, Professor and Head, Department of Computer Applications, for his insightful comments and administrative help on various occasions. Fortunately, I have many understanding friends, who have helped me a lot on many critical conditions. Finally, my sincere thanks go to my family members and all those who have directly and indirectly provided me with moral support and other kind of help. Without their support, completion of this work would not have been possible in time. They keep my life filled with enjoyment and happiness.

Mohd Nazim

TABLE OF CONTENTS

| | Page Number |
|---|-------------|
| Certificate | |
| Abstract | |
| Acknowledgement | |
| Table of Contents | |
| List of Abbreviations | |
| List of Tables | |
| List of Figures | |
| Chapter 1 - Introduction | 04-07 |
| 1.1 Project description | 04 |
| 1.2 Project Scope | 04 |
| 1.3 Hardware / Software used in Project | 06 |
| Chapter 2 Feasibility Study | 08 |
| Chapter 3 Database Design | 09-10 |
| 3.1 E-R Diagram | 09 |
| 3.2 Flow Chart | 09 |
| 3.3 Use Case Diagram | 10 |
| Chapter 4 Coding | |
| Module wise code | |
| Chapter 5 Proposed Time Duration | |
| References | |

CHAPTER 1

INTRODUCTION

1.1 PROJECT DESCRIPTION

In a world where the entertainment industry constantly evolves, there exists a growing need for a platform that not only catalogues the vast array of films, television shows, and celebrities but also provides an immersive and user-friendly experience for cinephiles and casual viewers alike. Enter IMDB Clone, a cutting-edge application leveraging the power of React, the frontrunner in modern web development technologies, to transform the way we engage with entertainment data.

- i. Front End Technology:**
 - React JS
- ii. Back-End Technology:**
 - Node JS
 - Express JS
- iii. Database:**
 - MongoDB

1.2 PROJECT SCOPE

IMDB Clone is not just another movie database; it's a technological marvel built upon the foundation of React, a robust JavaScript library known for its dynamic and interactive user interfaces. This revolutionary platform aims to redefine the way we explore, discover, and interact with the world of cinema. By harnessing the capabilities of React, IMDB Clone brings a fresh, responsive, and feature-rich experience to users seeking information and entertainment insights

In this era of digital innovation, where streaming services, movie releases, and celebrity updates are at our fingertips, IMDB Clone rises to the occasion by seamlessly integrating modern technologies to meet the ever-growing demands of movie enthusiasts and industry professionals. This platform represents not just a database but a community-driven hub, catering to the diverse interests of entertainment aficionados around the globe.

Join us on a journey through the intricacies of IMDB Clone, as we delve deeper into the technologies underpinning this groundbreaking application and discover how React elevates it to a league of its own. Explore a world where movie magic meets technological excellence, and the IMDB Clone experience awaits you

1.3 HARDWARE/SOFTWARE USED

1.3.1 Hardware Requirements:

| | |
|-----------|----------------------|
| Processor | i3 processor or more |
| RAM | 4GB or more |
| Hard Disk | 40GB OR MORE SSD |
| Window | Window 8 or higher |

1.3.2 Software Requirements:

| | |
|----------|---------|
| Database | MongoDB |
| Server | Node Js |
| IDE | Vs Code |

CHAPTER 2

FEASIBILITY STUDY

The advent of the digital age has revolutionized the entertainment industry, reshaping the way people consume and interact with movies and TV shows. One of the most prominent platforms that facilitate this change is IMDb (Internet Movie Database), a comprehensive online database of films, television series, and celebrities. The success and popularity of IMDb have prompted the development of IMDb clone websites and applications that aim to replicate its functionality and provide similar services. This literature review explores the need for IMDb clone technologies, with a focus on the use of React, a popular JavaScript library for building user interfaces.

1. Evolution of IMDb and Its Impact:

The IMDb platform has played a pivotal role in the entertainment industry by providing information, ratings, reviews, and user-generated content related to movies and TV shows. IMDb's vast and easily accessible database has become a primary source of information for both casual viewers and industry professionals. The need for IMDb clone technologies arises from the desire to replicate IMDb's success and offer similar services to a broader audience.

2. IMDb Clone Platforms and Technologies:

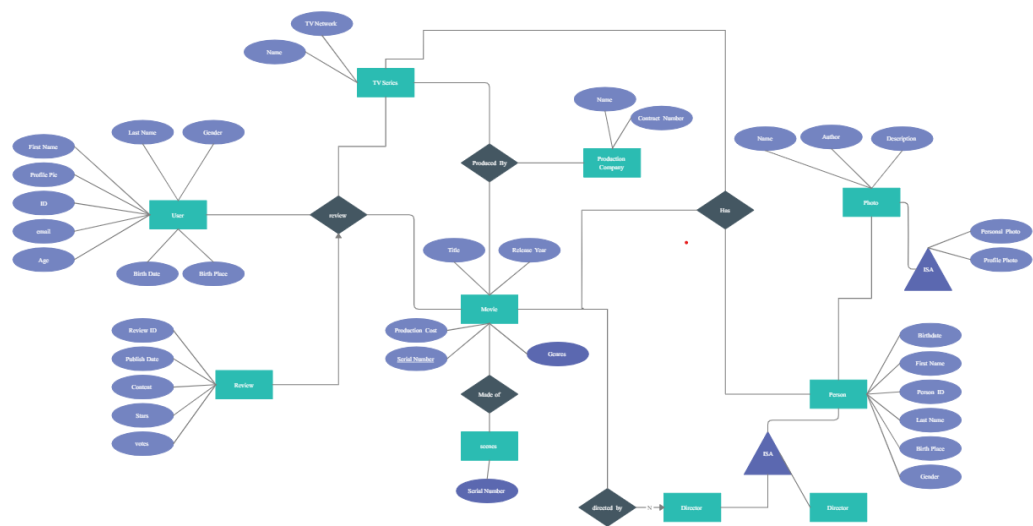
To create IMDb clone websites and applications, developers have employed various technologies and frameworks. React, a JavaScript library maintained by Facebook, has gained significant popularity due to its flexibility, performance, and robust ecosystem. React's component-based architecture, virtual DOM, and state management capabilities make it an attractive choice for building user interfaces in IMDb clone projects.

CHAPTER 3

DATABASE DESIGN

3.1 DATABASE TABLES

3.2 E-R Diagram



3.2 Data Flow Diagram

[Type text]

The context level data flow diagram (DFD) is describing the whole system. The (o) level DFD describe the all-user module who operate the system. Below data flow diagram of Art Gallery website.

3.3 Use Case Diagram

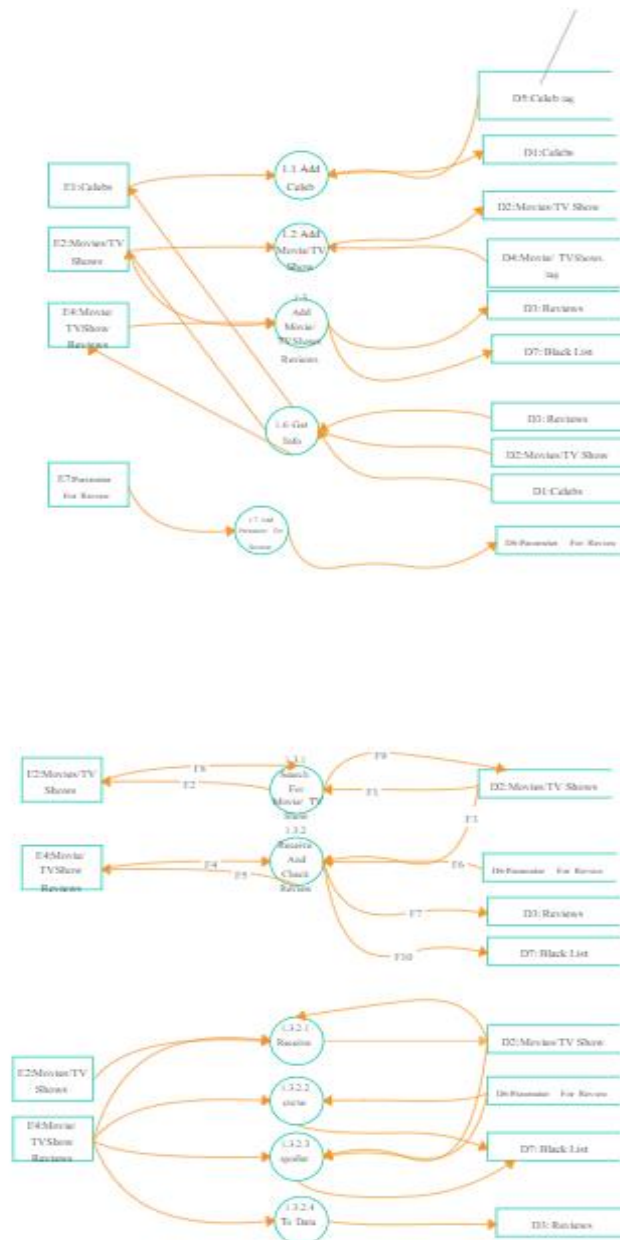
A use case diagram is a type of diagram in the Unified Modelling Language (UML) that is used to visualize and describe the functional requirements of a system from an external user's perspective. It provides a high-level view of how users interact with a system and the various functionalities or use cases the system offers in response to those interactions.

Use case diagrams are particularly useful for:

- Communicating the system's functionality and behaviour to stakeholders in a visual and understandable way.
- Capturing and documenting high-level user requirements.
- Identifying system boundaries and external interactions.
- modelling how different use cases relate to each other.

They are a valuable tool in the early stages of software development for understanding and discussing the functional aspects of a system before diving into more detailed design and implementation phases.

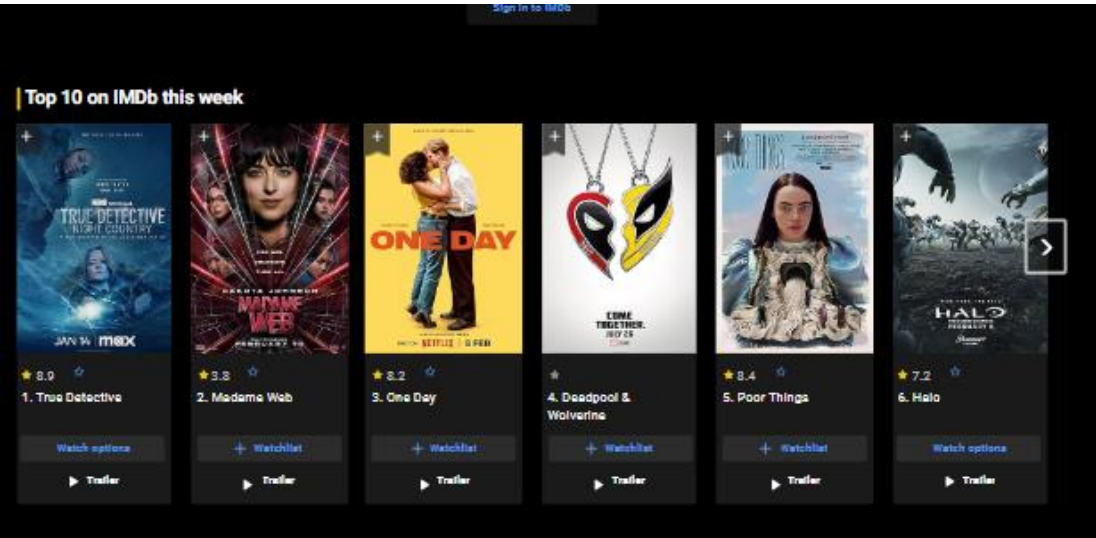
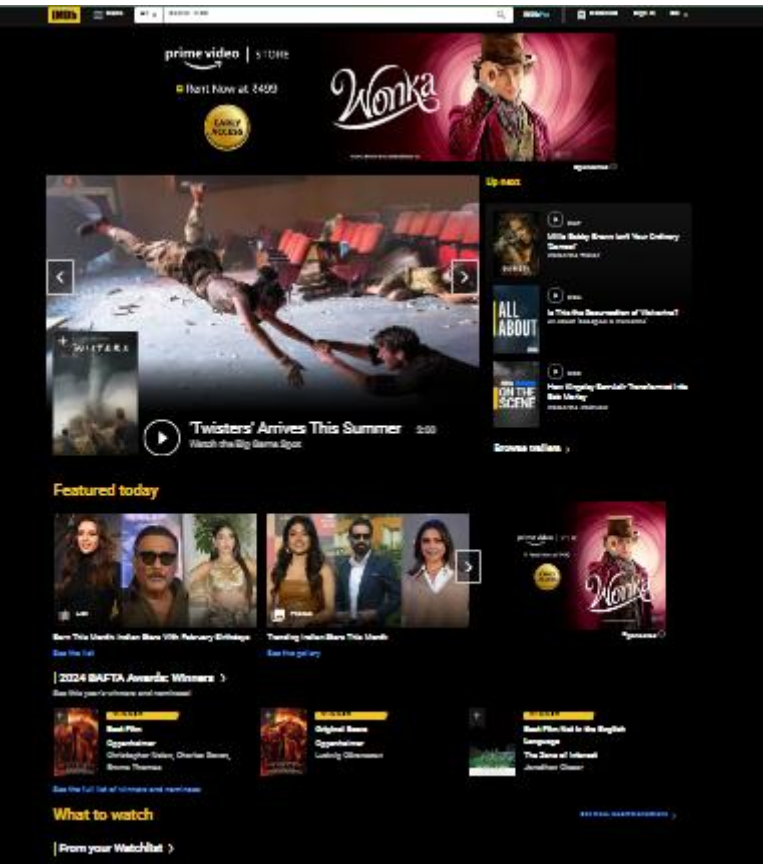
[Type text]



[Type text]

CHAPTER 4

SCREENSHORTS




[Type text]

More to watch
IMDb helps you select the perfect next show or movie to watch.

[Watch Guide](#)[Most Popular](#)

Exclusive videos

IMDb Originals
Celebrity Interviews, trending entertainment stories, and expert analysis




2:20

My Black Is...

Omarl Hardwick, Meagan Good & More Define Their Blackness

[Watch now](#)




1:55

**BLACK FASHION
IN FILM & TV HISTORY**

Celebrating Black Fashion in Film History

[Watch now](#)



5:57

**ALL ABOUT
HORROR IN 2024**

What Horrors Await in 2024?

[Watch now](#)

Explore what's streaming

CHAPTER 5

CODING

Package.json

```
// App.js
import React from 'react';
import Header from './components/Header';
import MovieList from './components/MovieList';
import Footer from './components/Footer';
import './App.css';

function App() {
  return (
    <div className="App">
      <Header />
      <main>
        <MovieList />
      </main>
      <Footer />
    </div>
  );
}

export default App;    ]
},
  "browserslist": {
    "production": [
      ">0.2%",
      "not dead",
      "notop_mini all"
    ],
    "development": [
      "last 1 chrome version",
      "last 1 firefox version",
      "last 1 safari version"
    ]
  },
  "devDependencies": {
```

[Type text]

```
    "tailwindcss": "^3.2.7"  
  }  
}
```

Index.js

```
// Header.js  
import React from 'react';  
  
function Header() {  
  return (  
    <header>  
      <h1>IMDb Clone</h1>  
      { /* Add navigation links here */ }  
    </header>  
  );  
}
```

[Type text]

```
export default Header;    <Toaster/>
    </BrowserRouter>
);
```

App.js

```
// MovieCard.js
import React from 'react';

function MovieCard({ movie }) {
  return (
    <div className="MovieCard">
      <imgsrc={movie.posterUrl} alt={movie.title} />
      <div className="details">
        <h3>{movie.title}</h3>
        <p>{movie.description}</p>
      </div>
    </div>
  );
}
```


[Type text]

```
        { /* Add more movie details here */ }
    </div>
</div>
    );
}

export default MovieCard;
```

App.css

[Type text]

```
/* App.css */
.App {
  text-align: center;
}

/* MovieList.css */
.MovieList {
  padding: 20px;
}

.movie-grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(250px, 1fr));
  gap: 20px;
}

.MovieCard {
  border: 1px solid #ccc;
  border-radius: 5px;
  overflow: hidden;
  box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
}

.MovieCardimg {
  width: 100%;
  height: auto;
}

.MovieCard .details {
  padding: 10px;
}

.MovieCard h3 {
  margin-top: 0;
}

.MovieCard p {
  margin-bottom: 0;
}
```

Home.js

[Type text]

```
// MovieList.js

import React, { useState, useEffect } from 'react';
import MovieCard from './MovieCard';
import './MovieList.css';

function MovieList() {
  const [movies, setMovies] = useState([]);

  useEffect(() => {
    // Fetch movies from backend API
    fetch('/api/movies')
      .then(response => response.json())
      .then(data => setMovies(data))
      .catch(error => console.error('Error fetching movies:', error));
  }, []);

  return (
    <div className="MovieList">
      <h2>Popular Movies</h2>
      <div className="movie-grid">
        {movies.map(movie => (
          <MovieCard key={movie.id} movie={movie} />
        ))}
      </div>
    </div>
  );
}
```

[Type text]

```
    );  
  }  
  
  export default MovieList;
```

LoginForm.jsx

```
import React from "react";  
import { useState } from "react";  
import { toast } from "react-hot-toast";  
import { AiOutlineEyeInvisible, AiOutlineEye } from "react-icons/ai";  
import { Link, useNavigate } from "react-router-dom";  
  
const LoginForm = (props) => {  
  const setIsLoggedIn = props.setIsLoggedIn;  
  
  const navigate = useNavigate();  
  
  const [showPassword, setShowPassword] = useState(false);  
  
  const [formData, setFormData] = useState({  
    email: "",  
    password: "",  
  });  
  
  function changeHandler(event) {  
    setFormData([  
      (prev) => [  
        {  
          ...prev,  
          [event.target.name]: event.target.value,  
        },  
      ],  
    ],  
  );  
}
```

[Type text]

```
    ]);
  }

  function submitHandler(e) {
    e.preventDefault();
    setIsLoggedIn(true);
    toast.success("Login Success");
    navigate("/dashboard");
  }

  return (
    <form
      onSubmit={submitHandler}
      className="flex flex-col w-full gap-y-4 mt-6"
    >
      <label className="w-full">
        <p className="text-[0.875rem] text-richblack-5 mb-1 leading-[1.375rem]">
          Email Address
          <sup className="text-pink-200">*</sup>
        </p>

        <input
          type="email"
          required
          value={formData.email}
          placeholder="Enter your email address"
          onChange={changeHandler}
          name="email"
          className="bg-richblack-800 rounded-[0.75rem] w-full p-[12px] text-richblack-5"
        />
      </label>

      <label className="w-full relative">
        <p className="text-[0.875rem] text-richblack-5 mb-1 leading-[1.375rem]">
          Password
          <sup className="text-pink-200">*</sup>
        </p>

        <input
```

[Type text]

```

        type={showPassword ? "text" : "password"}
        required
        value={formData.password}
        placeholder="Enter Password"
        onChange={changeHandler}
        name="password"
        className="bg-richblack-800 rounded-[0.75rem]
w-full p-[12px] text-richblack-5"
      />

      <span onClick={()
=>setShowPassword(!showPassword)} className="absolute right-3 top-
[38px] cursor-pointer ">
        {showPassword
?<AiOutlineEyeInvisible fontSize={24} fill='#AFB2BF' /> :
<AiOutlineEye fontSize={24} fill='#AFB2BF' />}
      </span>

      <Link to="#">
        <p className="text-xs mt-1 text-blue-100 max-
w-max ml-auto">Forgot Password</p>
      </Link>
    </label>

    <button className="bg-yellow-50 py-[8px] px-[12px]
rounded-[8px] mt-6 font-medium text-richblack-900">Sign
in</button>
  </form>
);
};

export default LoginForm;
```

[Type text]

SignupForm.jsx

```
import React from "react";
import { useState } from "react";
import { toast } from "react-hot-toast";
import { AiOutlineEyeInvisible, AiOutlineEye } from "react-icons/ai";
import { useNavigate } from "react-router-dom";

const SignupForm = (props) => {
  const setIsLoggedIn = props.setIsLoggedIn;

  const navigate = useNavigate();

  const [showCreatePass, setShowCreatePass] = useState(false);
  const [showConfirmPass, setShowConfirmPass] = useState(false);
  const [accountType, setAccountType] = useState("student");

  const [formData, setFormData] = useState({
    firstName: "",
    lastName: "",
    email: "",
    password: "",
    confirmPassword: "",
  });

  function changeHandler(event) {
    setFormData({
```

[Type text]

```
(prev) => [
  {
    ...prev,
    [event.target.name]: event.target.value,
  },
],
]);
}

function submitHandler(event) {
  event.preventDefault();
  if(formData.password !== formData.confirmPassword) {
    toast.error("Passwords do not match");
    return;
  }

  setIsLoggedIn(true);
  toast.success("Account Create");
  const accountData = {
    ...formData,
  };
  console.log(accountData);

  navigate("/dashboard");
}

return (
  <div>
    <div className="flex bg-richblack-800 p-1 gap-x-1 rounded-
full max-w-max">
      <button
        onClick={() => setAccountType("Student")}
        className={` ${
          accountType === "student"
            ? "bg-richblack-900 text-richblack-5"
            : "bg-transparent text-richblack-200 "
        } py-2 px-5 rounded-full transition-all`>
        >
        User
      </button>

      <button
```


[Type text]

```
onClick={() => setAccountType("instructor")}
className={` ${
  accountType === "instructor"
    ? "bg-richblack-900 text-richblack-5"
    : "bg-transparent text-richblack-200 "
} py-2 px-5 rounded-full transition-all`}
>
  Adminstrator
</button>
</div>

<form onSubmit={handleSubmit}>
  <div className="flex gap-x-4">
    <label htmlFor="" className="w-full">
      <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        First Name <sup className="text-pink-200">*</sup>
      </p>
      <input
        type="text"
        required
        placeholder="Enter First Name"
        onChange={handleChange}
        value={FormData.firstName}
        name="firstName"
        className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
      />
    </label>

    <label htmlFor="" className="w-full">
      <p className="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        Last Name <sup className="text-pink-200">*</sup>
      </p>
      <input
        type="text"
        required
        placeholder="Enter Last Name"
        onChange={handleChange}
        value={FormData.lastName}
        name="lastName"
```

[Type text]

```
        className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />
</label>
</div>

<labelhtmlFor=""className="w-full">
    <pclassName="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
        Email Address
        <supclassName="text-pink-200">*</sup>
    </p>

    <input
        type="email"
        required
        placeholder="Enter your email address"
        value={formData.email}
        onChange={changeHandler}
        className="bg-richblack-800 rounded-[0.75rem] w-full
p-[12px] text-richblack-5"
        name="email"
    />
</label>

<divclassName="flex gap-x-4">
    <labelhtmlFor="w-full relative">
        <pclassName="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
            Create Password
            <supclassName="text-pink-200">*</sup>
        </p>

        <input
            type={showCreatePass ? "text" : "password"}
            required
            placeholder="Enter Password"
            onChange={changeHandler}
            value={formData.password}
            name="password"
            className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
```

[Type text]

```

    />
    { /* <span
      onClick={() => setShowCreatePass(!showCreatePass)}
      className="absolute right-3 top-[38px] cursor-
pointer z-10"
    >
      {showCreatePass ? (
        <AiOutlineEyeInvisible fontSize={24}
fill="#AFB2BF" />
      ) : (
        <AiOutlineEyefontSize={24} fill="#AFB2BF" />
      )}
    </span> */}
  </label>

  <labelhtmlFor="" className="w-full relative">
    <pclassName="text-[0.875rem] text-richblack-5 mb-1
leading-[1.375rem]">
      Confirm Password
      <supclassName="text-pink-200">*</sup>
    </p>

    <input
      type={showConfirmPass ? "text" : "password"}
      required
      placeholder="Confirm Password"
      onChange={changeHandler}
      value={formData.confirmPassword}
      name="confirmPassword"
      className="bg-richblack-800 rounded-[0.75rem] w-
full p-[12px] text-richblack-5"
    />

    <span
      onClick={() => setShowConfirmPass(!showConfirmPass)}
      className="absolute right-3 top-[38px] cursor-
pointer z-10"
    >
      {showConfirmPass ? (
        <AiOutlineEyeInvisiblefontSize={24}fill="#AFB2BF"
/>
      ) : (
```

[Type text]

```
        <AiOutlineEyefontSize={24}fill="#AFB2BF"/>
      )}
    </span>
  </label>
</div>

    <buttonclassName="bg-yellow-50 py-[8px] px-[12px]
rounded-[8px] mt-6 font-medium text-richblack-900 w-full">
      Create Account
    </button>
  </form>
</div>
);
};

exportdefaultSignupForm;
```

[Type text]

Login.jsx

```
import React from 'react'
import loginImg from '../assets/login.png';
import Template from './Template';

function Login({ setIsLoggedIn }) {
  return (
    <Template
      title="Welcome Back"
      desc1="Discover the best source for royalty-free images."
      image={loginImg}
      formType="login"
      setIsLoggedIn={setIsLoggedIn}
    />
  );
}

export default Login
```

[Type text]

Signup.jsx

```
import React from 'react'
import signupImg from '../assets/signup.png';
import Template from './Template';

function Signup({ setIsLoggedIn }) {
  return (
    <Template
      title="Join the ArtGallery"
      desc1="To Search Unlimitd Image."
      image={signupImg}
      formType="signup"
      setIsLoggedIn={setIsLoggedIn}
    />
  );
}

export default Signup
```

[Type text]

Navbar.jsx

```
import React from "react";
import logo from "../assets/Logo1.png";
import { Link } from "react-router-dom";
import { toast } from "react-hot-toast";

const Navbar = (props) => {
  const isLoggedIn = props.isLoggedIn;
  const setIsLoggedIn = props.setIsLoggedIn;
  return (
    <div className="flex justify-between items-center w-11/12 max-w-[1160px] py-4 mx-auto">
      <Link to="/">
        <img src={logo} height={32} width={180} loading="lazy" alt="Logo" />
      </Link>

      <nav>
        <ul className="flex gap-x-6 text-richblack-100">
          <li>
            <Link to="/">Home</Link>
          </li>
          <li>
            <Link to="/About">About</Link>
          </li>
          <li>
            <Link to="/Contact">Contact</Link>
          </li>
        </ul>
      </nav>

      { /* Button - Login = Signup = Logout = Dashboard */ }
```

[Type text]

```
<div className="flex items-center gap-x-4 text-richblack-100">
  {!isLoggedIn&& (
    <Link to="/login">
      <button className="bg-richblack-800 py-[8px] px-[12px] rounded-[8px] border border-richblack-700">
        Login
      </button>
    </Link>
  )}
  {!isLoggedIn&& (
    <Link to="/signup">
      <button className="bg-richblack-800 py-[8px] px-[12px] rounded-[8px] border border-richblack-700">
        Sign Up
      </button>
    </Link>
  )}
  {isLoggedIn&& (
    <Link to="/">
      <button
        onClick={() => {
          setIsLoggedIn(false);
          toast.success("Logout Sucessfully");
        }}
        className="bg-richblack-800 py-[8px] px-[12px] rounded-[8px] border border-richblack-700"
      >
        Log Out
      </button>
    </Link>
  )}
  {isLoggedIn&& (
    <Link to="/dashboard">
      <button className="bg-richblack-800 py-[8px] px-[12px] rounded-[8px] border border-richblack-700">
        Profile
      </button>
    </Link>
  )}
</div>
</div>
```


[Type text]

```
);  
};
```

```
export default Navbar;
```

Template.jsx

```
import React from "react";  
import frame from "../assets/frame.png";  
import SignupForm from "./SignupForm.jsx";  
import LoginForm from "./LoginForm.jsx";  
import { FcGoogle } from "react-icons/fc";  
  
const Template = ({ title, desc1, desc2, image, formType, setIsLoggedIn })  
=> {  
  return (  
    <div className="flex w-11/12 max-w-[1160px] py-12 mx-auto gap-y-0  
gap-x-12 justify-between">  
      <div className="w-11/12 max-w-[450px] mx-0 text-white">  
        <h1 className="text-richblack-5 font-semibold text-[1.875rem]  
leading-[2.375rem]">{title}</h1>  
        <p className="text-[1.125rem] mt-4 leading-[1.625rem]">  
          <span className="text-richblack-100">{desc1}</span>  
          <span className="text-blue-100 italic">{desc2}</span>  
        </p>  
  
        {formType === "signup"  
? <SignupForm setIsLoggedIn={setIsLoggedIn}/>  
: <LoginForm setIsLoggedIn={setIsLoggedIn}/>}  
  
        <div className="flex w-full items-center my-4 gap-x-2">  
          <div className="h-[1px] w-full bg-richblack-700"></div>  
          <p className="text-richblack-700 font-medium leading-  
[1.375rem]">OR</p>  
          <div className="h-[1px] w-full bg-richblack-700"></div>  
        </div>  
  
        <button className="w-full flex items-center justify-center  
rounded-[8px] font-medium text-richblack-100 border-richblack-700  
border px-[12px] py-[8px] gap-x-2 mt-6">  
          <FcGoogle/>  
        </button>  
      </div>  
    </div>  
  );  
}
```

[Type text]

```
        <p>Sign Up with Google</p>
    </button>
</div>

<div className="relative w-11/12 max-w-[450px]">
    <img
        src={frame}
        alt="patter"
        width={558}
        height={504}
        loading="lazy"

    />
    <img
        src={image}
        alt="patter"
        width={558}
        height={504}
        loading="lazy"
        className="absolute -top-4 right-4 "
    />
</div>
</div>
);
};

export default Template;
```

[Type text]

Search.jsx

```
import React, {Component} from 'react';
import axios from 'axios';
import ImageResults from "../pages/imageResults";
class Search extends Component {
  state = {
    searchText: '',
    apiUrl: 'https://pixabay.com/api',
    apiKey: '17241914-90da7b93c0ccceb734849dcd1',
    images: []
  };
  onTextChange = (e) => {
    const val = e.target.value;
    this.setState({[e.target.name]: val}, () => {
      if (val === '') {
        this.setState({images: []});
      } else {
        axios
          .get(
            `${this.state.apiUrl}/?key=${this.state.apiKey}&q=${this.state.searchText}&image_type=photo&safesearch=true`
          )
          .then(res => this.setState({images: res.data.hits}))
          .catch(err => console.log(err));
      }
    });
  };
};
```

[Type text]

```
render(){
  console.log(this.state.images);
  return(
    <div>
      <input type="text"
        style=
          {{backgroundColor: 'white',
            color:
              'Black',
            marginLeft:570,
            marginTop:50,
            paddingTop:0,
            paddingLeft:70,
            fontSize:30,
            borderTopStyle:"hidden",
            borderRightStyle:"hidden",
            borderLeftStyle:"hidden",
            outline:"none",
            borderBottomStyle:"groove",
          }}
        placeholder="Search for images"
        name="searchText"
        value={this.state.searchText}
        onChange={this.onTextChange}
      />
    <br/>
    {this.state.images.length>0?(<ImageResults images={this.state.images}/>):null}
    </div>
  )
}
}

export default Search;
```

[Type text]

ImageResults.jsx

```
import React, {Component} from 'react';
import PropTypes from 'prop-types';
import {GridList, GridTile} from 'material-ui/GridList';
import IconButton from 'material-ui/IconButton';
import ZoomIn from 'material-ui/svg-icons/action/zoom-in';
import Dialog from 'material-ui/Dialog';
import FlatButton from 'material-ui/FlatButton';

class ImageResult extends Component {
  state = {
    open: false,
    currentImg: ''
  }
  handleOpen = img => {
    this.setState({open: true, currentImg: img})
  }
  handleClose = () => {
    this.setState({open: false});
  }
  render() {
    {
      let imageList;
      const {images} = this.props

      if (images)
      {
        imageList = (
          <GridList cols={4}>
            { images.map(img => (
              <GridTile
                title={img.tags}

```

[Type text]

```

        key={img.id}
        actionIcon={
          <IconButton onClick={()=>this.handleOp
en(img.largeImageURL)}>
            <ZoomIn color="white"/>
          </IconButton>
        }
      >
        <img src={img.largeImageURL} alt="" />
      </GridTile>
    ))
  }
</GridList>
)
}
else {
  imageUrl=null;
}
const actions=[
  <FlatButton label="Close" primary={true} onClick={this
.handleClose}/>
]
return(
  <div style={{marginLeft:50,marginRight:50,marginTop:20
}}>
    {imageUrl}
    <Dialog
      actions={actions}
      modal={false}
      open={this.state.open}
      onRequestClose={this.handleClose}
    >
      <img src={this.state.currentImg} alt="" style={{width:'1
00%'}} />
    </Dialog>
  </div>
)
}
}
ImageResults.propTypes={
  images:PropTypes.array.isRequired
}
```

[Type text]

```
export default ImageResults;
```

CHAPTER 6

TESTING

6.1 INTRODUCTION

The purpose of testing is to discover errors. Testing is the process of trying to discovery conceivable fault or weakness in a work product. It provides a way to check the functionalities of components, sub-assemblies, and/or a finished product it is the process of exercising software with the intent of ensuring that the software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of tests. Each test type addresses a specific testing requirement.

6.2 TYPES OF TESTING

6.2.1 Unit Testing

Unit testing focuses verification effort on the smallest unit of software design, the module. The unit testing, we have is white box oriented and some modules the steps are conducted in parallel.

6.2.2. Integration Testing

Testing is done for each module. After testing all the modules, the modules are integrated and testing of the final system is done with the test data, specially designed to show that the system will operate successfully in all its aspects conditions. Thus, the system testing is a confirmation that all is correct and an opportunity to show the user that the system works.

The purpose of integration testing is to verify functional, performance and reliability requirements placed on major design items. These "design items", i.e. assemblages (or groups of units), are exercised through their interfaces using black box testing, success and error cases being simulated via appropriate parameter and data inputs. Simulated usage of shared data areas and inter-process communication is tested and individual subsystems are exercised through their input interface.

6.2.3 System Testing

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration-oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

CHAPTER 7

CONCLUSION

The IMDb Clone project successfully replicates the core functionality of the popular Internet Movie Database (IMDb) website. Through the use of modern web development technologies such as React.js for the front end and Node.js with Express.js for the back end, the project provides users with a user-friendly interface to browse, search, rate, and review movies, TV shows, and celebrities.

Throughout the development process, several key objectives were achieved:

1. **User Interface:** A clean and intuitive user interface was designed to enhance the browsing experience for users. Components such as the header, movie lists, and movie cards were implemented to present information in an organized and visually appealing manner.
2. **Database Integration:** The project successfully integrated a MongoDB database to store data related to movies, TV shows, celebrities, user accounts, ratings, and reviews. This allowed for efficient data management and retrieval, enabling seamless interaction with the application.

[Type text]

3. **Functionality:** Core functionalities such as browsing, searching, rating, and reviewing movies and TV shows were implemented effectively. Users can easily navigate through the application, find information about their favorite titles, and engage with other users through ratings and reviews.

4. **Performance:** Efforts were made to optimize the performance of the application, ensuring fast loading times and smooth user interactions. Techniques such as lazy loading and data caching were employed to minimize latency and enhance the overall responsiveness of the application.

5. **Security:** Security measures were implemented to protect user data and ensure the confidentiality and integrity of the system. User authentication and authorization mechanisms were enforced to prevent unauthorized access and maintain the privacy of user accounts.

In summary, the IMDb Clone project serves as a comprehensive platform for users to explore, interact, and engage with their favorite movies, TV shows, and celebrities. While the project achieves its core objectives, there is still room for further enhancements and refinements, such as implementing additional features, improving user experience, and addressing any identified issues or feedback from users.

Overall, the IMDb Clone project demonstrates the feasibility and effectiveness of modern web development technologies in building robust and feature-rich applications that cater to the entertainment needs of users worldwide.

[Type text]

CHAPTER 8

REFERENCES

- www.w3schools.com
- www.geeksforgeeks.org
-