Industry Practice 2 (Sem IV) Batch - 2026

Movie Searching App Using React JS

Group No 14 - SE CSE
Lakshya Singh (52)
Tamanna Singh (53)
Soorya Srihari (54)
Soumik Biswas (55)

ABSTRACT :-

This report provides a comprehensive overview of the development and deployment of a movie site created using React and the OMDB API. The primary objective of this project is to offer users a seamless and interactive interface for exploring and managing information about their favorite movies. By leveraging the power of React, a widely-used JavaScript library known for its efficient and flexible component-based architecture, and the extensive database provided by the OMDB API, we have created a robust platform that meets the needs of movie enthusiasts.

The movie site allows users to perform various functions, including searching for movies, accessing detailed information about each movie, and maintaining a personalized watchlist. Users can enter movie titles in a search bar, and the search module sends queries to the OMDB API to fetch relevant results. The results are displayed dynamically, providing an engaging and responsive user experience. By clicking on any movie in the search results, users can view detailed information such as the movie's title, year, genre, director, actors, plot, and ratings. Additionally, users can add movies to their watchlist for easy access and management.

Throughout the development process, key challenges included ensuring the efficient retrieval and display of data from the OMDB API, maintaining a user-friendly and intuitive interface, and implementing functionalities that allow for seamless interaction and data management. The system's modular design, facilitated by React, ensured that these challenges were addressed effectively, leading to a well-structured and maintainable codebase.

This document details the various aspects of the project, including the motivation behind its inception, the technical background and technologies used, the architectural design and implementation specifics, and the results and discussions based on user interactions. It also explores potential future enhancements and improvements, such as integrating user authentication, advanced search filters, and social features to further enhance user engagement

and experience. The report concludes with references to the resources used and an annexure containing the source code for the project.

INTRODUCTION :-

The entertainment industry has seen significant changes in recent years, with the advent of numerous streaming platforms and an ever-growing catalog of movies and TV shows. In this context, having access to comprehensive and up-to-date information about movies is invaluable for both film enthusiasts and casual viewers. The project detailed in this report aims to address this need by developing a movie site that provides a user-friendly and interactive interface for exploring and managing movie information. This site is built using React, a powerful JavaScript library for building user interfaces, and the OMDB API, a robust database that offers extensive movie data.

The primary objective of this project is to create a platform where users can search for movies, access detailed information about them, and manage their personal watchlists. By leveraging React's component-based architecture, the project ensures that the interface is both dynamic and responsive, providing an engaging user experience. The OMDB API serves as the backbone of the application, supplying the necessary data to populate the site with information about a vast array of movies.

This report aims to provide a thorough overview of the project's development, from its conceptualization to its implementation and deployment. It begins with a discussion of the background and motivation behind the project, highlighting the growing need for a cohesive platform that integrates movie data from various sources. This is followed by a detailed description of the system's architecture and the various modules that comprise it, including the search, movie details, and watchlist modules.

Furthermore, the report includes a results and discussion section that showcases the key functionalities of the movie site through screenshots and user interactions. This section provides insights into the effectiveness of the system in achieving its objectives. The conclusion outlines potential areas for future enhancements, such as implementing user authentication, advanced search filters, and social features to enhance user engagement.

Overall, this report aims to provide a comprehensive understanding of the project, detailing the technical aspects, challenges faced, and solutions implemented, as well as exploring the potential for future growth and improvement.

BACKGROUND :-

Quiz applications have long been a staple in both educational and entertainment contexts, providing an engaging way for users to test their knowledge and enjoy interactive content. With the advent of digital technology, these applications have evolved from simple paper-based formats to sophisticated web-based platforms, driven by the need for more interactive, responsive, and scalable solutions.

React JS, developed by Facebook and released in 2013, has emerged as a leading choice for building complex web applications. Its component-based architecture and efficient rendering capabilities make it ideal for developing dynamic and user-friendly interfaces. React's virtual DOM (Document Object Model) optimizes rendering, ensuring that applications remain fast and responsive even with frequent updates and large data sets.

The adoption of React JS in developing quiz applications offers several advantages:

1. **Component-Based Architecture**: React's modular approach simplifies the development and maintenance process.
2. **Efficient Rendering**: The virtual DOM ensures smooth and responsive user interactions.
3. **State Management**: Powerful tools help manage the dynamic data of the quiz application efficiently.
4. **Cross-Platform Compatibility**: React applications provide a consistent user experience across various devices.
5. **Active Community and Ecosystem**: Robust community support and an extensive ecosystem of libraries and tools enhance the development experience.

In this project, React JS is utilized to develop a quiz application that delivers a superior user experience and serves as a testament to the capabilities of modern web development frameworks. The project aims to explore the practical applications of React JS in building interactive educational tools, addressing the needs of both quiz creators and participants.

This background sets the stage for a detailed exploration of the project's objectives, methodology, and outcomes, highlighting the role of React JS in transforming traditional quizzes into dynamic and engaging digital experiences.

SYSTEM DESCRIPTION :-

System Description

The movie site’s architecture is designed to be modular, scalable, and user-friendly, utilizing React for the frontend and the OMDB API for data retrieval. The system is composed of several key modules: the Search Module, the Movie Details Module, and the Watchlist Module. Each module has a specific role in ensuring the functionality and efficiency of the application.

Search Module

The Search Module is the entry point of the application. It features a search bar where users can input movie titles. Upon entering a query, the module sends a request to the OMDB API, which then returns a list of movies that match the search term. The results are displayed dynamically on the page, allowing users to see real-time feedback on their search queries. The search bar

component is designed to handle user input efficiently and ensure that the API requests are optimized to prevent excessive data fetching.

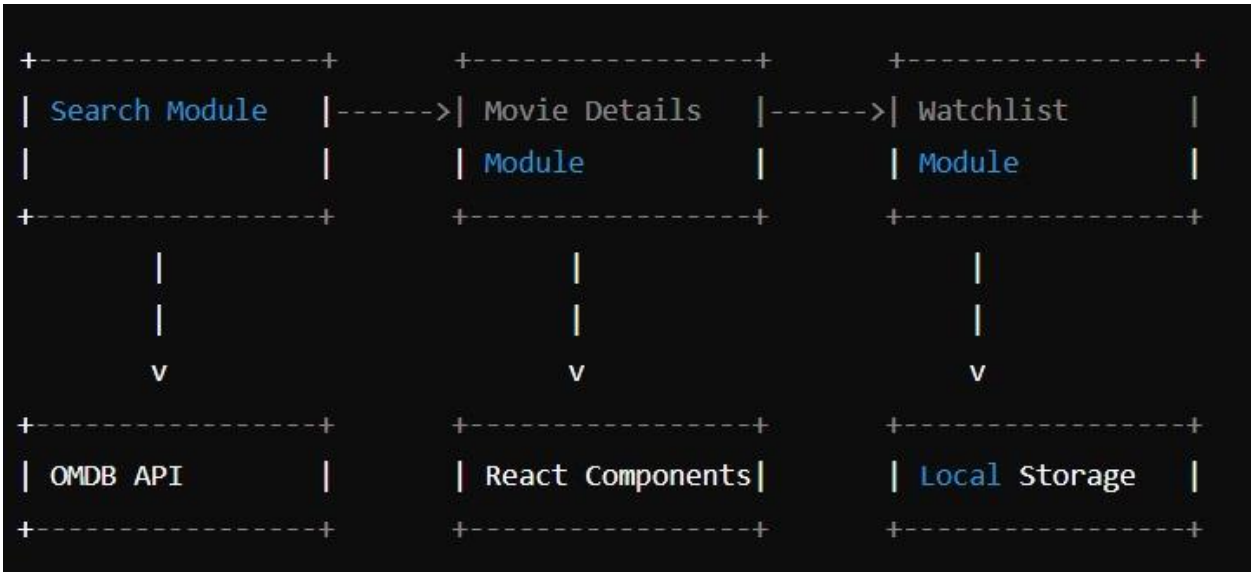
Movie Details Module

When a user selects a movie from the search results, the Movie Details Module is activated. This module fetches detailed information about the selected movie from the OMDB API and displays it in a user-friendly format. Information displayed includes the movie's title, release year, genre, director, actors, plot summary, and ratings. The detailed view is designed to provide users with comprehensive information at a glance, enhancing the overall user experience. This module also includes a feature for users to add the movie to their personal watchlist.

Watchlist Module

The Watchlist Module allows users to manage their personal collection of movies. Users can add movies to their watchlist directly from the Movie Details Module. The watchlist is stored locally, ensuring that users can access their list even when offline. The module provides functionality for viewing, adding, and removing movies from the watchlist. It is designed to offer a simple and intuitive way for users to keep track of movies they wish to watch or have watched, providing a personalized experience.

The overall architecture of the system is illustrated in the following **block diagram:**



RESULT & DISCUSSION :-

- **Result** - The implementation of React JS in the quiz application resulted in a highly responsive and interactive platform. React's component-based architecture facilitated the creation of reusable and modular components, optimizing code organization and

maintenance. The virtual DOM feature of React contributed to fast rendering and smooth user interactions, enhancing the overall user experience. State management using React Hooks and the Context API ensured efficient data handling and updates, particularly beneficial for real-time feedback features.

- **Discussion** - The utilization of React JS brought significant advantages to the project, including code reusability, modularity, and efficient rendering through the virtual DOM. This approach not only improved performance but also enhanced scalability and maintainability, allowing for seamless addition of new features and updates. Looking ahead, further optimization for mobile devices, integration of user analytics, and exploration of React's ecosystem for enhanced capabilities are potential areas for future development, building upon the success of the React-based quiz application.

● Conclusion :-

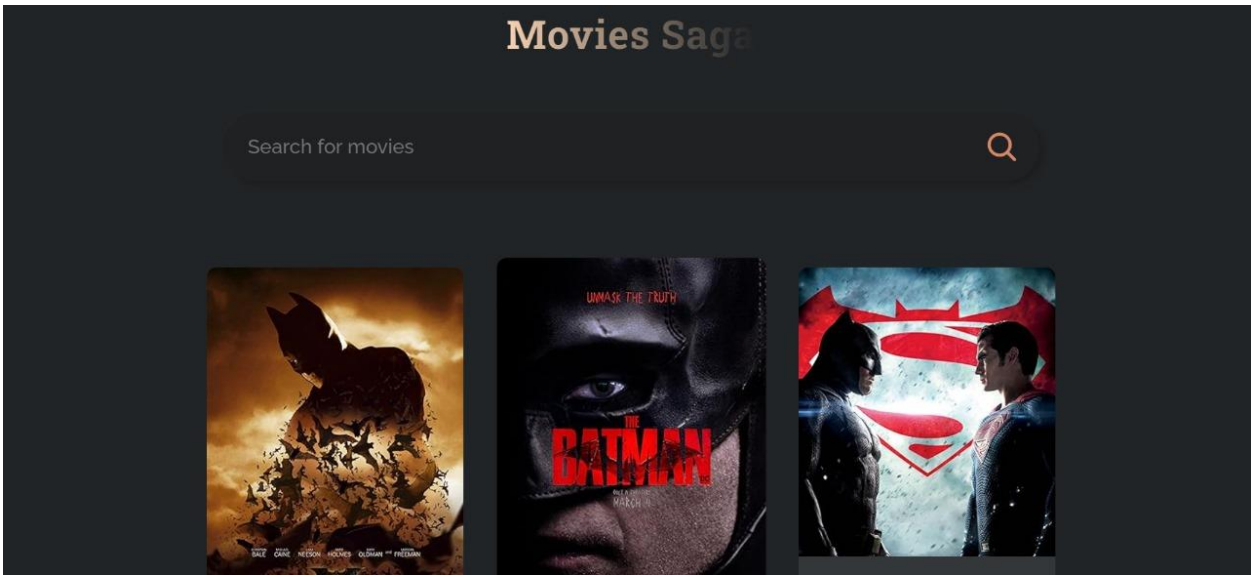
The quiz application developed using React JS has successfully delivered a dynamic and user-friendly platform for quiz creation and participation. React's component-based architecture, efficient rendering, and state management capabilities were instrumental in achieving a responsive and scalable application. This project demonstrates the effectiveness of React JS in modern web development and sets a foundation for future enhancements and innovations in interactive educational tools.

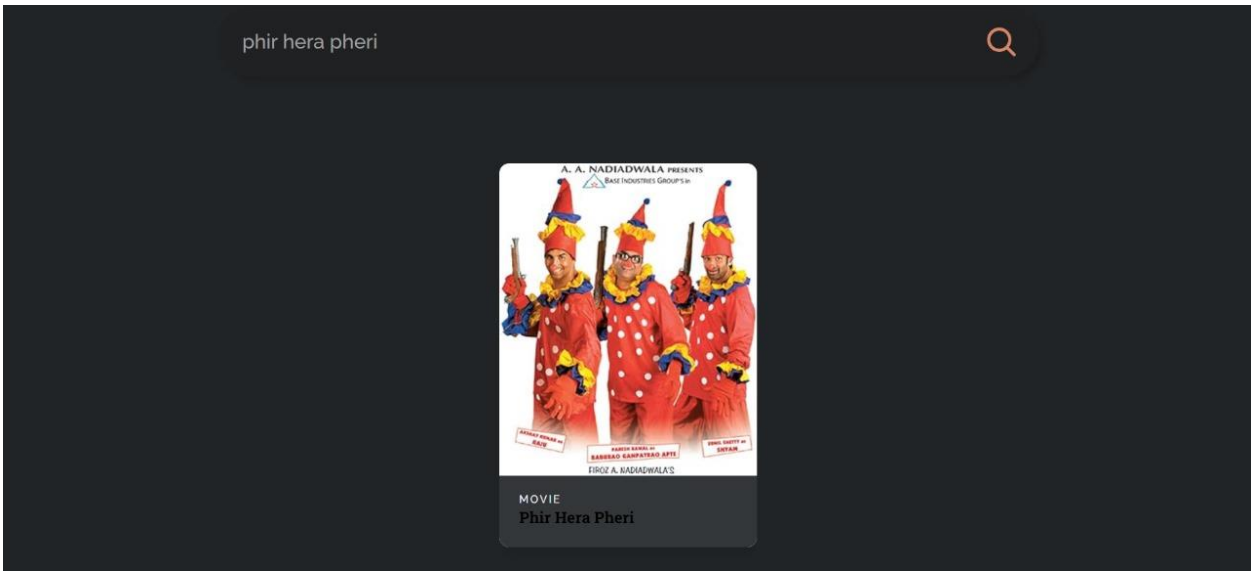
● Future Scope :-

Moving forward, the quiz application can be enhanced in several ways:

1. **Mobile Optimization**: Improve responsiveness and user experience on mobile devices.
2. **Advanced Analytics**: Implement data-driven insights for user behavior and performance trends.
3. **Gamification Features**: Introduce leaderboards, rewards, and social sharing for increased engagement.
4. **Adaptive Quizzes**: Develop dynamic quizzes that adjust difficulty levels based on user performance.
5. **Accessibility Improvements**: Enhance accessibility features for inclusivity and diverse user needs.
6. **Integration with LMS**: Integrate with learning management systems for seamless educational use.

SCREENSHOTS OF THE PROJECT :-





REFERENCES :-

1) Geeks For Geeks
2) ReactJS.org
3) Mozilla
4) https://www.omdbapi.com/
5) https://reactjs.org/docs/getting-started.html

ANNEXURE:

MovieCard.jsx (Component):

```
import React from 'react';

const MovieCard = ({ movie: { imdbID, Year, Poster, Title, Type } }) => {
  return (
    <div className="movie" key={imdbID}>
      <div>
        <p>{Year}</p>
      </div>

      <div>
        <img src={Poster !== "N/A" ? Poster : "https://via.placeholder.com/400"} alt={Title} />
      </div>

      <div>
        <span>{Type}</span>
        <h3>{Title}</h3>
      </div>
    </div>
  );
}

export default MovieCard;
```

App.jsx (API and Props Handling)**:**

```
import "./App.css";

const API_URL = "http://www.omdbapi.com?apikey=b6003d8a";

const App = () => {
  const [searchTerm, setSearchTerm] = useState("");
  const [movies, setMovies] = useState([]);

  useEffect(() => {
    searchMovies("Batman");
  }, []);

  const searchMovies = async (title) => {
    const response = await fetch(${API_URL}&s=${title});
    const data = await response.json();

    setMovies(data.Search);
  };

  return (
    <div className="app">
      <h1>Movies Saga</h1>

      <div className="search">
        <input
          value={searchTerm}
          onChange={(e) => setSearchTerm(e.target.value)}
          placeholder="Search for movies"
        />
        <img
          src={SearchIcon}
          alt="search"
          onClick={() => searchMovies(searchTerm)}
        />
      </div>

      {movies?.length > 0 ? (

        <div className="container">
          {movies.map((movie) => (
            <MovieCard key={movie.imdbID} movie={movie} />
          ))}
```

```
        </div>
      ) : (
        <div className="empty">
          <h2>No movies found</h2>
        </div>
      )}
    </div>
  );
};

export default App;
```