Project Title : Movie Review System

Due Date: 11/07/2024
Completion Date: 10/07/2024

Github Repository Link : https://github.com/Dineshchowdaryjampala/Movie-Reviews-FSD-Project

Project Summary :

The entertainment industry produces a vast number of movies and TV shows every year, and audiences often struggle to find reliable and comprehensive platforms where they can discover, rate, review, and save their favorite content. Existing platforms may not provide an intuitive user experience or might lack essential features for detailed reviews and personalized content management. This project aims to address these gaps by developing a sleek and modern web application that allows users to seamlessly search, view, review, and rate movies and TV shows, with a focus on an engaging and user-friendly interface.

Key Features :

- Search: Find your favorite movies and TV shows quickly and easily.
- Detailed Information: Access comprehensive information about movies and TV shows, including ratings and reviews.
- Reviews and Ratings: Users can submit their own reviews and rate movies and TV shows.
- Favorites: Save favorite movies and shows for easy access.
- Sleek UI/UX: An intuitive and visually appealing user interface for a better user experience.

Tech Stack Details:

Front-end:

- **React:** Building user interfaces with React.
- **Material UI:** A popular React UI framework for creating visually appealing interfaces.
- **SwiperJS:** A swiper/slider component for images and other content.
- **React Router v6:** For handling navigation within the app.
- **Formik:** For building and validating forms in React.
- **Yup:** A schema validation library for form validation.
- **Axios:** For making HTTP requests to APIs.

Back-end:

- **Express:** A fast and minimalist web application framework for Node.js.
- **Express Validator:** Middleware for validation and sanitation of incoming data.
- **Jsonwebtoken:** For handling user authentication and authorization.
- **Mongoose:** Elegant MongoDB object modeling for Node.js.
- **Axios:** For making HTTP requests from the server.

Database:

- MongoDB

External API:

- TMDB

Deployment:

- Vercel

Work Contribution:

NAME	ROLE	ITEMS RESPONSIBLE FOR
B Satish	Frontend Developer	Frontend Development, Interface Design, Page Development, Documentation
AK Charith Kumar Reddy	Frontend Developer	Frontend Development, Component Design, Layout, Configurations, Documentation
J Dinesh Chowdary	Backend Developer	Backend Development, API Integration (tmdb), Documentation
NSL Karthikeya Reddy	Backend Developer	Backend Development, Database Integration (mongodb), Vercel Deployment, Documentation

Contents

1.0 DESIGN

1.1 PROBLEM DEFINITION

In today's digital age, internet evaluations have become an important resource for customers making purchase decisions, particularly in the entertainment business. Movies, being a major segment of the entertainment industry, rely significantly on reviews to attract spectators. A strong movie review system will help potential viewers make informed decisions about a film's quality. This paper describes the issue specification for designing a movie review system.

The movie business is now struggling to efficiently manage and present user evaluations. Existing systems frequently lack key features like user identification, review moderation, and real-time changes. Furthermore, the integration of front-end and back-end technologies is often inadequate, leading in unsatisfactory user experiences. A Java Full Stack solution will address these difficulties by using Java's powerful backend capabilities and incorporating cutting-edge front-end frameworks.

The primary objective of this project is to develop a comprehensive movie review system that allows users to submit, view, and manage movie reviews. The system will be built using a Java Full Stack approach, ensuring a seamless integration between the front-end and back-end components. Key features will include user authentication, review moderation, real-time updates, and an intuitive user interface.

1.2 SOLUTION SPECIFICATION

Algorithm: User Registration and Login

Algorithm RegisterUser(username, password, email):

INPUT: username, password, email

OUTPUT: registration status (success/failure)

Begin

Check if username or email already exists in database

IF exists THEN

RETURN "Failure: Username or email already exists"

ELSE

Hash the password

Create a new user object with username, hashed password, and email

Save the new user object to the database

RETURN "Success: User registered successfully"

End

Algorithm LoginUser(username, password):

INPUT: username, password

OUTPUT: login status (success/failure) and user token

Begin

Retrieve user object from database using username

IF user object exists THEN

Compare the hashed password with the stored hashed password

IF passwords match THEN

```
            Generate a JWT token for the user
            RETURN "Success: Login successful", token
        ELSE
            RETURN "Failure: Incorrect password"
    ELSE
        RETURN "Failure: User does not exist"
End
```

Algorithm: Movie Search and Display

```
Algorithm SearchMovies(query):
    INPUT: query (search term)
    OUTPUT: list of movies

    Begin
        Make an API call to TMDB with the search query
        Retrieve the search results from TMDB API
        IF results found THEN
            Parse and format the results
            RETURN list of formatted movies
        ELSE
            RETURN "No movies found"
    End
```

Algorithm: Submit Review

```
Algorithm SubmitReview(userId, movieId, reviewText, rating):
    INPUT: userId, movieId, reviewText, rating
    OUTPUT: submission status (success/failure)
```

```
Begin
    Retrieve user object from database using userId
    Retrieve movie object from database using movieId
    IF both user and movie exist THEN
        Create a new review object with userId, movieId, reviewText, and rating
        Save the new review object to the database
        RETURN "Success: Review submitted successfully"
    ELSE
        RETURN "Failure: User or movie does not exist"
End
```

Algorithm: Display Reviews

```
Algorithm DisplayReviews(movieId):
    INPUT: movieId
    OUTPUT: list of reviews

    Begin
        Retrieve all review objects from database using movieId
        IF reviews exist THEN
            Parse and format the reviews
            RETURN list of formatted reviews
        ELSE
            RETURN "No reviews found"
    End
```

Algorithm: Display Favourites

Algorithm DisplayFavorites(userId):

INPUT: userId

OUTPUT: list of favorite movies

Begin

Retrieve user object from database using userId

IF user exists THEN

Retrieve the list of favorite movieIds from user object

Retrieve movie objects from database using the list of favorite movieIds

Parse and format the movie objects

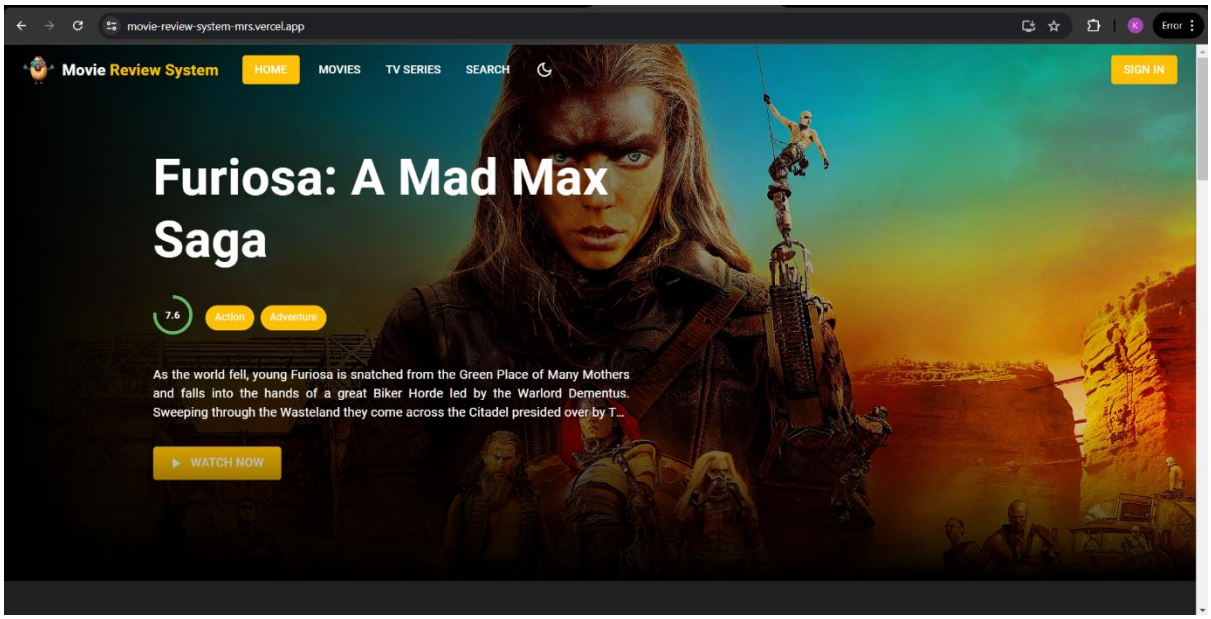
RETURN list of formatted favorite movies

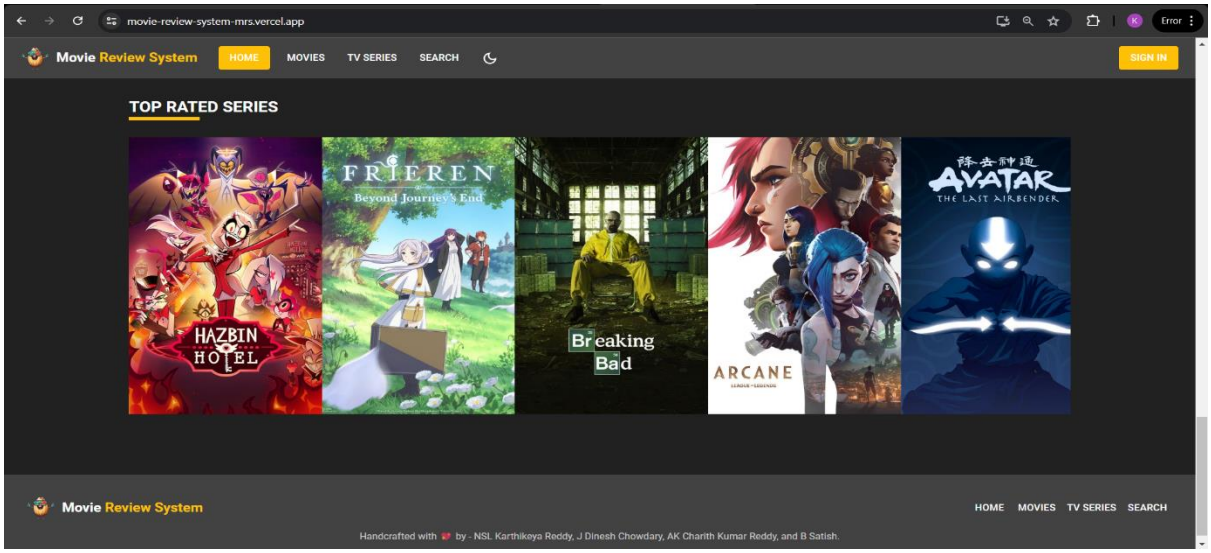
ELSE

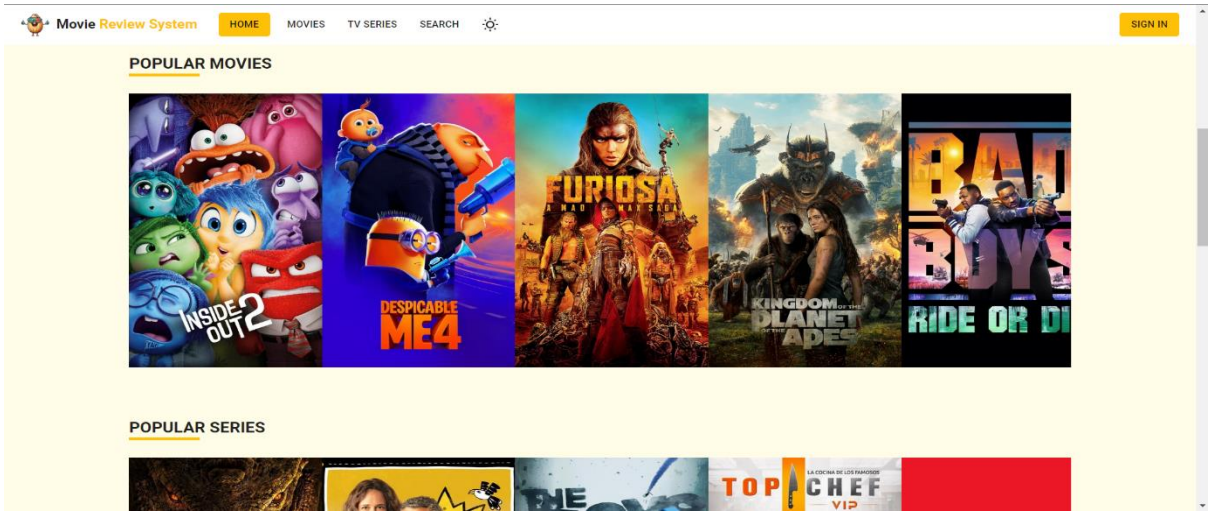
RETURN "Failure: User does not exist"

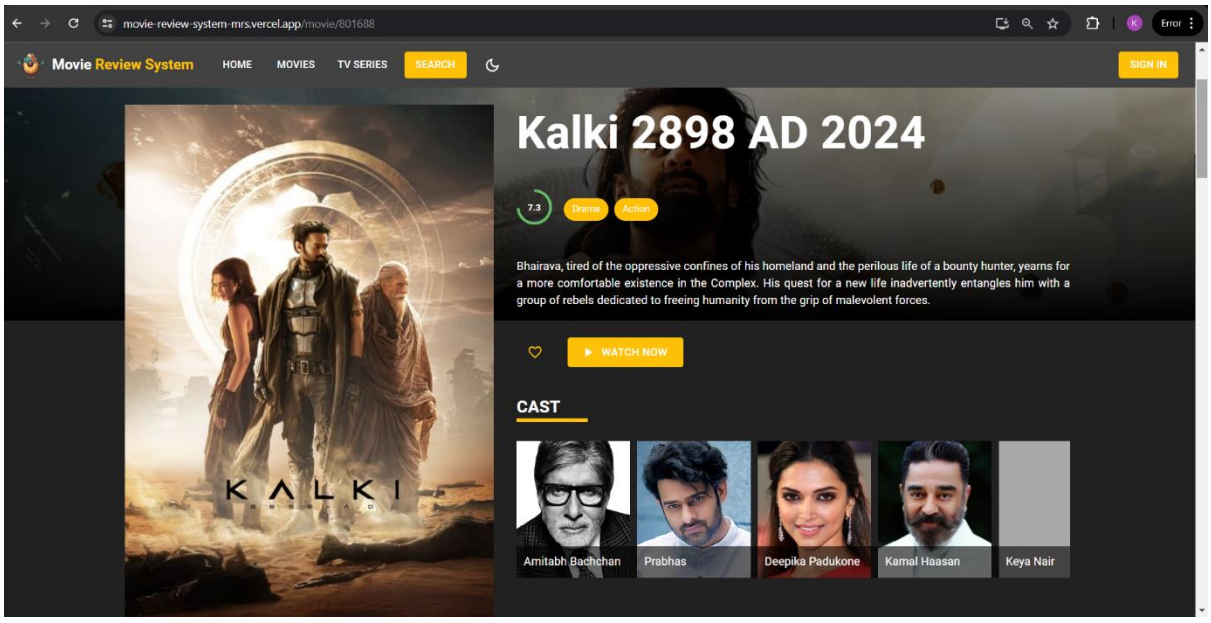
End

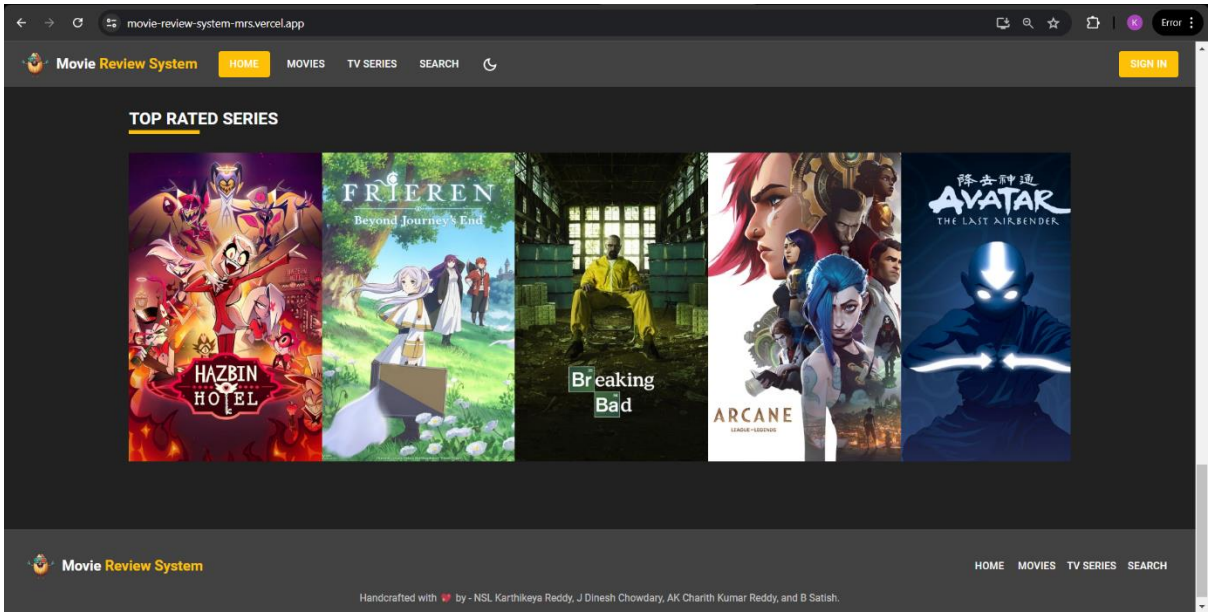
Screen Design:



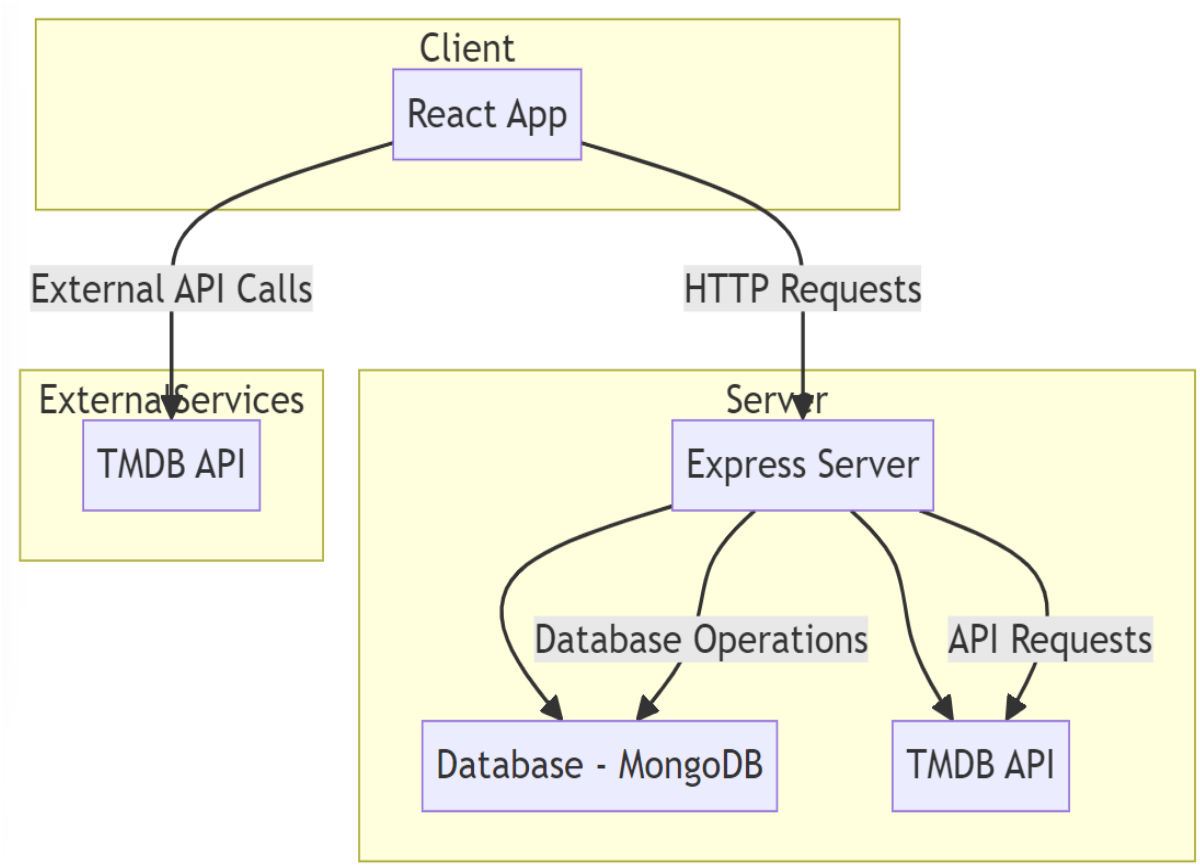

movie-review-system-mrs.vercel.app
Movie Review System
HOME
MOVIES
TV SERIES
SEARCH
SIGN IN
TOP RATED SERIES
HAZBIN HOTEL
FRIEREN
Beyond Journey's End
Breaking Bad
ARCANE
AVATAR
THE LAST AIRBENDER
Movie Review System
HOME MOVIES TV SERIES SEARCH
Handcrafted with by - NSL Karthikeya Reddy, J Dinesh Chowdary, AK Charith Kumar Reddy, and B Satish.


Movie Review System
HOME
MOVIES
TV SERIES
SEARCH
SIGN IN
POPULAR MOVIES
INSIDE OUT 2
DESPICABLE ME4
FURIOSA
KINGDOM OF THE PLANET OF THE APES
BAD BOYS
RIDE OR DI
POPULAR SERIES
TOP CHEF VIP

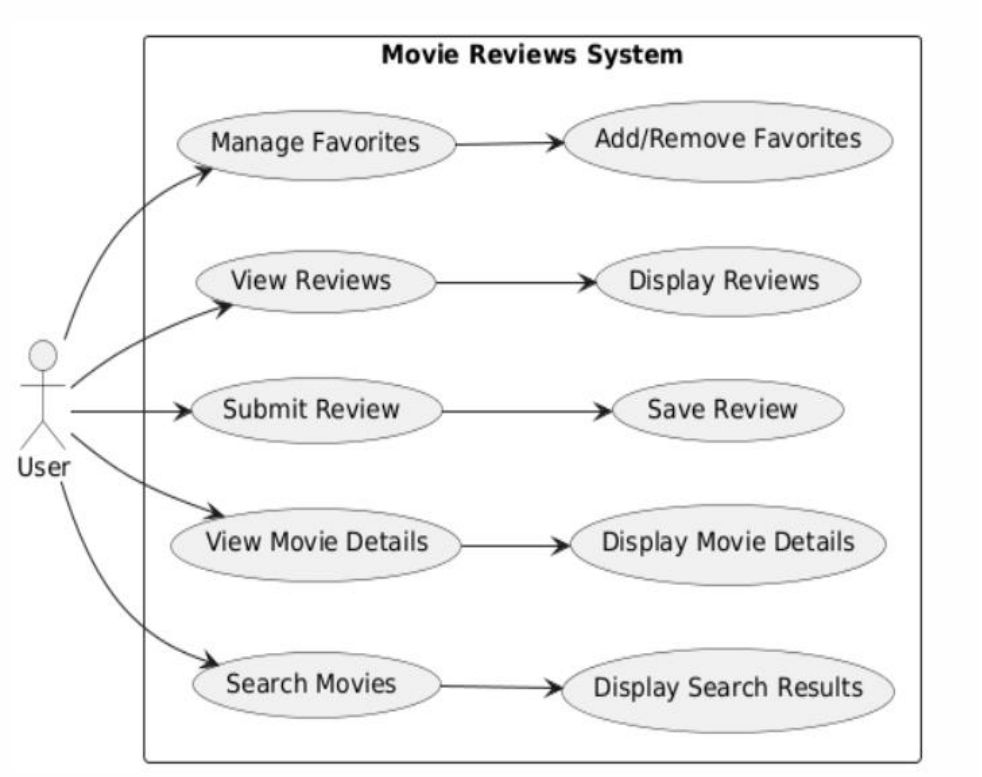

movie-review-system-mrs.vercel.app/movie/801688
Movie Review System
HOME
MOVIES
TV SERIES
SEARCH
SIGN IN
KALKI
Kalki 2898 AD 2024
7.3
Drama
Action
Bhairava, tired of the oppressive confines of his homeland and the perilous life of a bounty hunter, yearns for a more comfortable existence in the Complex. His quest for a new life inadvertently entangles him with a group of rebels dedicated to freeing humanity from the grip of malevolent forces.
WATCH NOW
CAST
Amitabh Bachchan
Prabhas
Deepika Padukone
Kamal Haasan
Keya Nair



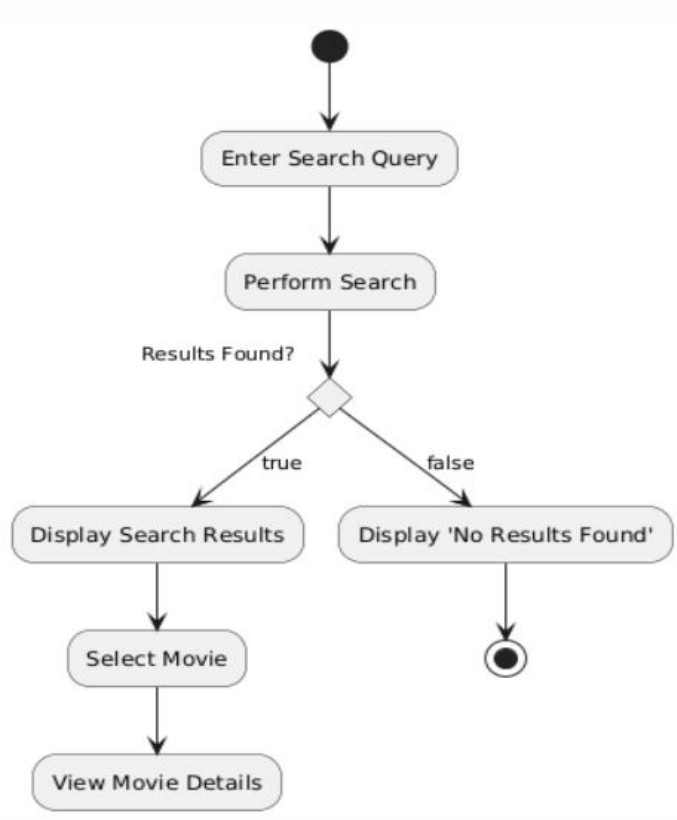
ARCHITECTURE DIAGRAM:



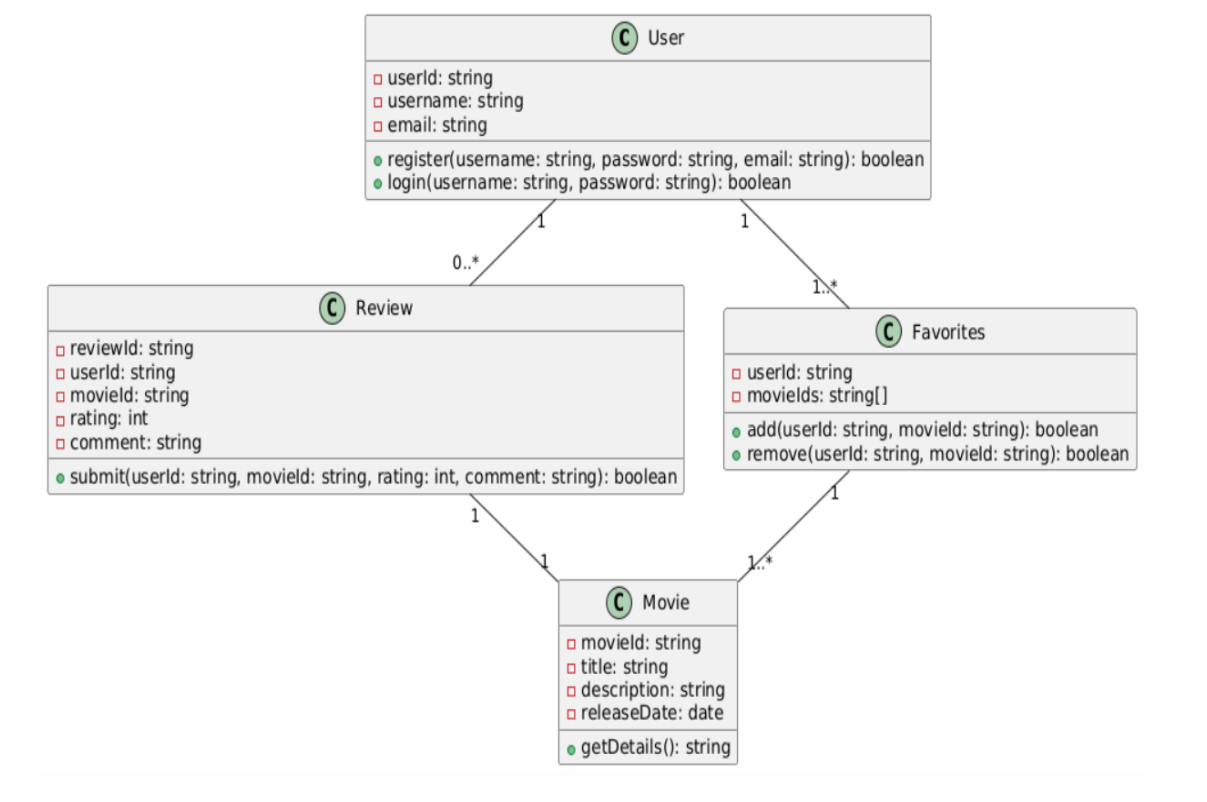
USE CASE DIAGRAM:



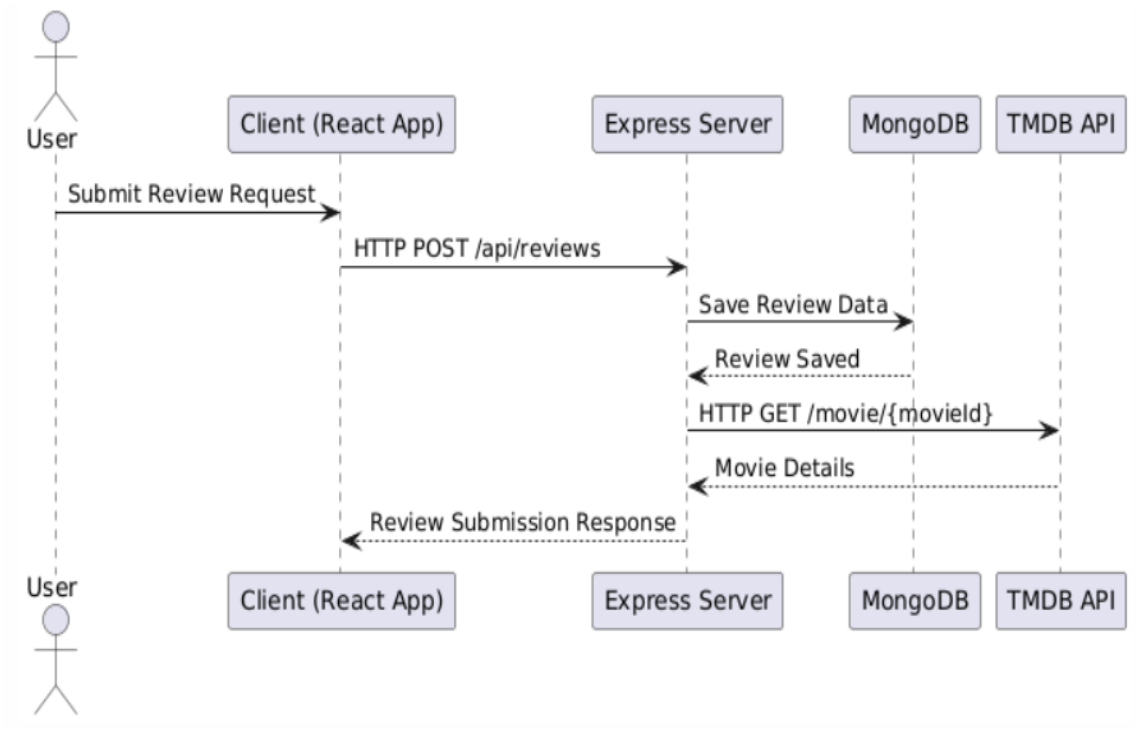
ACTIVY DIAGRAM:



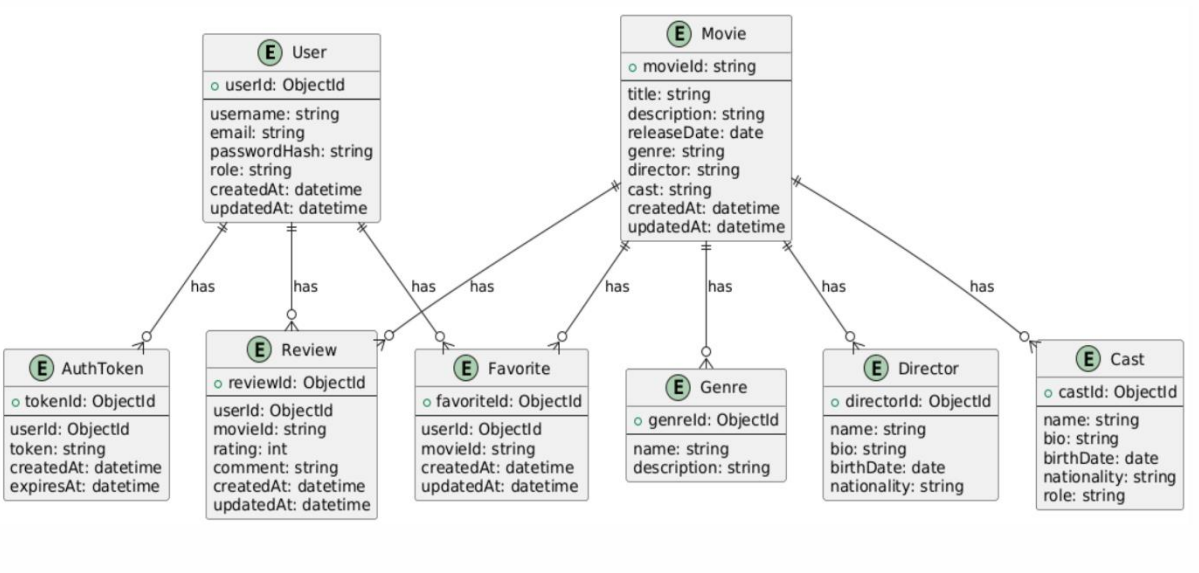
CLASS DIAGRAM:



SEQUENCE DIAGRAM:



DATABASE STRUCTURE DIAGRAM:



2.0 DEVELOP

2.1 IMPLEMENTATION

Variable Names and Types:

Name	Value Type	Initial Value
UserId	string	""
Username	string	""
Password	string	""
Email	string	""
MovieId	string	""
Title	string	""
Description	string	""
ReleaseDate	date	null
Rating	int	0
Comment	string	""
MovieIds	array of strings	[]

Functions:

Name	Values Required	Values Returned	Description (pseudo-code)
register	username, password, email	boolean	Check if username or email exists, then register user.
login	username, password	boolean, token	Authenticate user based on username and password.
getDetails	movieId	string	Fetch and return details of a specific movie.
submitReview	userId, movieId, rating, comment	boolean	Save user's review for a movie into the database.
addFavorite	userId, movieId	boolean	Add a movie to the user's list of favorite movies.
removeFavorite	userId, movieId	boolean	Remove a movie from the user's list of favorites.

Descriptions (Pseudo-Code):

- **register(username, password, email)**:
  Checks if the username or email already exists in the database. If not, registers the user with the provided details.
- **login(username, password)**:
  Authenticates the user based on the provided username and password. Returns a boolean indicating success or failure, along with an authentication token.
- **getDetails(movieId)**:
  Retrieves and returns detailed information about a movie identified by movieId.
- **submitReview(userId, movieId, rating, comment)**:
  Saves a review submitted by a user (userId) for a specific movie (movieId) with a rating and comment.
- **addFavorite(userId, movieId)**:
  Adds a movie (movieId) to the list of favorites for a user (userId).
- **removeFavorite(userId, movieId)**:
  Removes a movie (movieId) from the list of favorites for a user (userId).

These functions outline typical operations in the context of a Movie Reviews System, handling user registration, authentication, movie data retrieval, review submission, and favorite management.

3.0 CONCLUSION

Our Movie Reviews System project successfully addresses the need for a comprehensive and user-friendly platform for movie and TV show enthusiasts to discover, review, and manage their favorite content. By leveraging the MERN stack and integrating the powerful TMDB API, the system provides a seamless experience for users to search for movies, view detailed information, submit reviews, and manage their favorite lists. The sleek and intuitive design ensures that users can navigate the application effortlessly, making it an engaging and efficient tool for accessing entertainment content.

The collaborative efforts from us, team members—NSL Karthikeya Reddy, J Dinesh Chowdary, AK Charith Kumar Reddy, and B Satish have resulted in a robust application that not only meets the project's objectives but also showcases modern web development practices. With a focus on both front-end and back-end functionalities, the project demonstrates the effective use of technologies like React, Express, MongoDB, and Vercel for deployment. The system's ability to handle user authentication, interact with external APIs, and provide a rich user experience underscores its value as a comprehensive solution for movie and TV show reviews and management.

APPENDIX

Code including internal documentation:

Server (index.js) :

```
import express from "express";
import cookieParser from "cookie-parser";
import cors from "cors";
import mongoose from "mongoose";
import dotenv from "dotenv";
import routes from "./src/routes/index.js";

dotenv.config();

const app = express();

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: false }));
app.use(cookieParser());

let isConnected = false;

async function connectToDatabase() {
 if (isConnected) {
  console.log('Using existing database connection');
  return;
 }

 try {
  console.log("Attempting to connect to MongoDB...");
```

```
    await mongoose.connect(process.env.MONGODB_URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
      serverSelectionTimeoutMS: 5000,
      socketTimeoutMS: 45000,
    });
    isConnected = true;
    console.log("MongoDB connected successfully");
  } catch (error) {
    console.error("MongoDB connection error:", error);
    isConnected = false;
    throw error;
  }
}

// Middleware to ensure database connection
app.use(async (req, res, next) => {
  try {
    await connectToDatabase();
    next();
  } catch (error) {
    console.error('Database connection failed', error);
    res.status(503).json({ error: 'Service Unavailable: Database connection failed' });
  }
});

// Routes
app.use("/api/v1", routes);

// Error handling middleware
```

```
app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(500).json({ error: 'Something went wrong!', details: err.message });
});

// Handler for serverless function
const handler = async (req, res) => {
  await app(req, res);
};

export default handler;
```

Server (.env) :

```
MONGODB_URL= connection string
PORT=5000
TOKEN_SECRET_KEY=
TMDB_BASE_URL=https://api.themoviedb.org/3/
TMDB_KEY= api key
```

Client (App.jsx) :

```
import { ThemeProvider } from "@mui/material/styles";
import { useSelector } from "react-redux";
import themeConfigs from "./configs/theme.configs";
import { ToastContainer } from "react-toastify";
import CssBaseline from "@mui/material/CssBaseline";
import { BrowserRouter, Route, Routes } from "react-router-dom";
import MainLayout from "./components/layout/MainLayout";
import routes from "./routes/routes";
import PageWrapper from "./components/common/PageWrapper";
import "react-toastify/dist/ReactToastify.css";
import "swiper/css";
```

```
import "swiper/css/navigation";
import "swiper/css/pagination";

const App = () => {
  const { themeMode } = useSelector((state) => state.themeMode);

  return (
    <ThemeProvider theme={themeConfigs.custom({ mode: themeMode })}>
      {/* config toastify */}
      <ToastContainer
        position="bottom-left"
        autoClose={5000}
        hideProgressBar={false}
        newestOnTop={false}
        closeOnClick
        pauseOnFocusLoss
        pauseOnHover
        theme={themeMode}
      />
      {/* mui reset css */}
      <CssBaseline />

      {/* app routes */}
      <BrowserRouter>
        <Routes>
          <Route path="/" element={<MainLayout />}>
            {routes.map((route, index) => (
              route.index ? (
                <Route
                  index
                  key={index}
```

```
              element={route.state ? (
                <PageWrapper state={route.state}>{route.element}</PageWrapper>
              ) : route.element}
            />
          ) : (
            <Route
              path={route.path}
              key={index}
              element={route.state ? (
                <PageWrapper state={route.state}>{route.element}</PageWrapper>
              ) : route.element}
            />
          )
        ))}
      </Route>
    </Routes>
  </BrowserRouter>
  {/* app routes */}
 </ThemeProvider>
 );
};

export default App;
```

DEVELOPMENT JOURNAL

Date	Comment	Problems / Recommendations
28/06/2024	Initial project setup and repository creation. Team members cloned the repository.	No significant problems.
29/06/2024	Basic React app structure created. Installed necessary dependencies.	Ensure all team members have the same version of Node.js installed.
30/06/2024	Implemented user registration and login functionality.	Faced issues with bcrypt version compatibility. Resolved by updating the package.
01/07/2024	Integrated MongoDB for user data storage. Set up environment variables.	Ensure `.env` file is not pushed to GitHub. Add it to `.gitignore`.
02/07/2024	Designed and implemented the movie search functionality using TMDB API.	API rate limits encountered. Consider implementing caching.
03/07/2024	Developed movie details page and integrated movie data retrieval from TMDB.	Ensure proper error handling for API responses.
04/07/2024	Implemented review submission feature on the movie details page.	Validation errors initially missed. Added express-validator.
05/07/2024	Developed review display functionality and UI improvements on the movie details page.	Ensure consistent styling across different components.
06/07/2024	Implemented user favorites feature, allowing users to add/remove movies from favorites.	Faced issues with updating the user schema. Resolved by schema migration.
07/07/2024	Enhanced UI with Material UI components. Added form validation with Formik and Yup.	Initial learning curve with Material UI. Reviewed documentation.
08/07/2024	Worked on responsive design improvements and tested on different devices.	Minor styling issues on mobile devices. Adjusted breakpoints.
09/07/2024	Deployed the application on Vercel. Finalized documentation and user guide.	Faced deployment configuration issues. Resolved by updating build settings.
10/07/2024	Final testing and bug fixes. Project presentation preparation.	Minor bugs fixed. No major issues.
