**Training Report**

**On**

**“Box Office App”**

**Submitted in partial fulfilment of the requirements for the award of the degree of**

**Master of Computer Application (Batch 2022 - 2024)**



# By

**Kajal Pundir**

# Roll No. 1322705

**Under the Supervision of Mr. Gulbir Singh**

# M.M. Institute of Computer Technology & Business Management Maharishi Markandeshwar (Deemed to be University), Mullana (Ambala)

**(NACC accredited Grade ‘A++’ University)**

# August 2023

**DECLARATION**

I hereby declare that the work which is being presented in this report entitled “Box Office App”, in partial fulfilment of the requirement for the award of the degree of **MASTER OF COMPUTER APPLICATIONS** submitted at M. M. Institute of Computer Technology & Business Management, **Maharishi Markandeshwar (Deemed to be University), Mullana, Ambala** is an authentic work done by me during a period form **15/06/2023** to **27/072023** under the Guidance of Tejbir Sir **(Internal Supervisor appointed by the Institute).**

The work presented in the report has not been submitted by me for the award of any other degree of this or any other Institute/University.

### Signature

**Name of the Candidate Roll No:**

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

**Date: Signature**

### Place: Name of the Supervisor

**Designation**



**Kajal Pundir**

has successfully completed a 6-week online training on **React**. The training consisted of Introduction, Tic Tac Toe Game, Box Office App, Chat Application, Custom Backend, and Assignment & Summary modules.

Kajal scored 94% marks in the final assessment and is a top performer in the training.

We wish Kajal all the best for future endeavours.

Date of certification: 2023-07-26 Certificate no. : dl7f2m667ie For certificate authentication, please visit [https://trainings.internshala.com/verify\_certificate](https://trainings.internshala.com/verify_certificate?certificate_number=dl7f2m667ie)

# ABSTRACT

Abstraction is a fundamental concept in computer science and software engineering that involves simplifying complex systems by focusing on the essential characteristics and hiding unnecessary details. It is a crucial aspect of designing and building software applications and enables developers to manage the complexity of large-scale systems effectively. Abstraction promotes code readability, maintainability, and reusability. It also reduces the complexity of understanding and modifying code, making it easier for multiple developers to collaborate on large projects.

# List of diagrams

1. Level 0 Data Flow Diagram: Page 18
2. Level 1 Data Flow Diagram: Page 19
3. API Request Response Flow Diagram: Page 20

# List of Abbreviations

1. API - Application Programming Interface
2. CPU - Central Processing Unit
3. DFD - Data Flow Diagram
4. ERD - Entity-Relationship Diagram
5. GUI - Graphical User Interface
6. IoT - Internet of Things
7. RAM - Random Access Memory
8. UI - User Interface
9. URL - Uniform Resource Locator
10. UX - User Experience

**INDEX**

|  |  |  |
| --- | --- | --- |
| **S.no** | **Contents** | **Page No.** |
| 1. | **Introduction**   1. Organization Profile 2. Introduction of Project 3. Problem definition 4. Limitation of existing system 5. Objective of the project | **1 - 8** |
| 2. | **System Analysis**   1. Feasible Study 2. Hardware and software requirements | **9 – 11** |
| 3. | **Development Environment**  a) Introduction of technology used | **12 – 13** |
| 4. | **System Design**  a) Module description | **14 - 15** |
| 5. | **Implementation and Testing**   1. Implementation and Testing 2. Testing | **16 – 43** |
| 6. | **Conclusion** | **44** |
| 7. | **Bibiliography** | **45** |

# Introduction

## Organization profile Introduction:

The organization where I completed my MCA project is **Internshala** is TechSolutions Pvt. Ltd. Located in the bustling city of TechTown, TechSolutions is a leading IT company that specializes in providing innovative software solutions to clients across various industries.

## History and Background:

### 2010: How it all started

Sarvesh, our founder & CEO, started Internshala as a blog with a mission to bring a culture of meaningful internships in India. And for the first two years, he hired only virtual interns.

### 2013: The website launch

After building a small team, we then launched our website with just one goal - to equip every student in India with their dream internship. And we did it all for free.

### 2015: Internshala app launch

The next big step could not have been anything other than launching our very own Android app, bringing Internshala in the ‘hands’ of the students.

### 2016: The beginning of Internshala Trainings

After many successful years as an internship platform, our motivation to upskill the students increased, and that is when we kickstarted a journey with Internshala Trainings.

### 2020: When we launched Fresher Jobs

With Fresher jobs, we embarked on a journey filled with newer challenges, which allowed us to provide better opportunities to graduates with 0-2 years of experience.

### 2021: The launch of Placement Guarantee courses:

With an insight that more than 90% of the graduates in India start their careers with a job that pays less than 3LPA, we came up with Placement Guarantee courses to help the students start their careers in their dream profiles.

### Mission and Vision:

Internshala is a dot com business with the heart of dot org.

We are a technology company on a mission to equip students with relevant skills & practical exposure to help them get the best possible start to their careers. Imagine a world full of freedom and possibilities. A world where you can discover your passion and turn it into your career. A world where you graduate fully assured, confident, and prepared to stake a claim on your place in the world.

### Organizational Structure:

Internshala follows a well-defined organizational structure. The executive team includes the CEO, CTO, and heads of various departments such as Development, Quality Assurance, Design, and Project Management. The organization encourages a collaborative work environment that fosters creativity and teamwork.

### Overall Impression:

My experience with Internshala has been highly rewarding. The organization's professionalism, emphasis on quality, and supportive work culture have contributed to a positive learning environment during my MCA project.

## Introduction of project

**Project Name: Box Office App**

The Box Office App is a web application designed to provide users with up-to-date information about the latest movies, box office rankings, and celebrity gossips. The primary goal of this project is to create an intuitive and user-friendly platform that allows movie enthusiasts to stay informed about the latest happenings in the entertainment industry.

I created a Box Office App that displays data about movies, shows, and actors.

The Bollywood Box Office App has been introduced for the web users. With the help of this app, you can now enjoy daily Box Office shows, movies as well as about of actors and actresses.

You can fine box office leaders, pictures, videos, behind the scenes, interviews and official trailers!

Easily follow your favorite actors, directors, movies, view trailers and read the gossip behind the scenes!

Main Features:

* 1. No setup or learning time required. Fire the app for the first time and immediately receive the latest movies.
  2. Box office charts, including latest hits
  3. An awesome widget
  4. Latest movie trailers
  5. Catch up on your own schedule. You can choose your own time-frame with one tap – latest, day or week.

## Problem definition

The problem at hand is the lack of an efficient and user-friendly platform that provides movie enthusiasts with up-to-date information about the latest movies, box office rankings, and celebrity gossips. The traditional system of relying on physical media sources, such as newspapers and television, to access entertainment news has several limitations and does not meet the demands of modern users who seek convenience, interactivity, and real-time updates.

The main challenges and limitations of the existing system are as follows:

**Limited Accessibility:** The traditional system restricts access to information only to those who actively seek it through specific media channels, leaving out a vast audience looking for real-time entertainment updates.

**Time-Consuming:** The process of gathering information from various sources manually can be time-consuming and inconvenient for users.

**Lack of Personalization:** The traditional system does not offer personalized experiences tailored to individual user preferences, leading to a generic and less engaging user experience.

**Outdated Information:** Information provided by traditional media sources may not always be up-to-date, and users may miss out on the latest movie releases and entertainment news.

**Geographical Constraints:** Different regions may have varying coverage of movies and entertainment news, leading to inconsistent and limited access to relevant content.

**Limited Interaction:** The traditional system lacks interactive features, such as leaving comments, sharing opinions, or engaging in discussions with other users about movies and entertainment.

**Difficulty in Searching:** Finding specific information about a particular movie or celebrity can be challenging when relying on physical newspapers or television.

**Environmental Impact:** The use of physical media sources, such as newspapers, contributes to paper waste and has a negative impact on the environment.

To address these limitations and challenges, the proposed solution is to develop a Box Office App using React.js and API integration. This app will serve as a one-stop platform that provides users with real-time and personalized updates on the latest movies, box office rankings, and celebrity gossips.

By leveraging the power of React.js and integrating with a reliable API, the app will offer a responsive, intuitive, and engaging user interface, empowering users to access the information they seek easily and conveniently. The app will also allow users to interact with the content, leave reviews, create watchlists, and share their opinions, enhancing their overall entertainment experience.

Through the Box Office App, users will have a comprehensive, up-to-date, and interactive platform for all their movie-related and entertainment news needs.

## Limitation of existing system

The existing system, which we assume to be the traditional or non-digital approach to accessing box office and entertainment information, has several limitations. Some of the key limitations include:

**Limited Accessibility:** In the traditional system, accessing real-time information about the latest movies, box office rankings, and celebrity gossips often requires physically visiting a movie theater, reading newspapers, or relying on entertainment news channels. This restricts access to information and updates only to those who actively seek them through traditional media channels.

**Time-Consuming:** Gathering information about multiple movies, their showtimes, box office rankings, and celebrity news can be time-consuming when relying on manual methods. Users need to browse through different sources to find the desired information.

**Lack of Personalization:** The traditional system does not offer personalized experiences. Users cannot create watchlists, receive personalized recommendations, or leave reviews and ratings based on their preferences.

**Limited Real-Time Updates:** Information in newspapers or television might not always be up-to-date. Movie releases, box office rankings, and celebrity gossips change rapidly, and the traditional system may not provide the most current information.

**Geographical Constraints:** Traditional media sources may have regional variations and might not cover all movie releases and entertainment news from different parts of the world.

**No Interaction or Engagement:** The traditional system does not allow for user interactions or engagement. Users cannot comment on news articles, share opinions, or participate in discussions related to movies and entertainment.

**Inability to Search Easily:** Finding specific information about a particular movie or celebrity might be challenging, especially when relying on physical newspapers or TV channels.

**Limited Content Depth:** Traditional media often has space and time constraints, which limit the amount of content that can be covered for each movie or celebrity. Users may not get a comprehensive view of the subject of interest.

**Environmental Impact:** Printing newspapers and distributing physical copies contribute to paper waste and have an environmental impact.

**Advertising Overload:** Traditional media sources might be filled with advertisements, making it difficult for users to find the relevant information they seek.

**Lack of Interactivity:** Users cannot watch movie trailers or teaser clips directly from traditional media sources.

By developing a Box Office App with React.js and API integration, many of these limitations can be overcome. The app can provide a centralized platform for real-time and personalized movie and entertainment information, allowing users to stay updated, engaged, and informed with ease and convenience.

## Objective of the project

Indian file industry makes over 1000 movies annually of which a substantial portion is produced by the popular Hindi film industry known as Bollywood. The paper studies so Hindi movies from 2013 to 2017.

### The key objectives of the Box Office App project are as follows:

* To provide users with a visually appealing and responsive web interface.
* To fetch and display the latest movie information from a reliable API source.
* To display real-time box office rankings and updates.
* To incorporate a gossip section to keep users entertained with the latest celebrity news.

Movies are selected on basis of profit percentage earned from box office collections. The data is acquired from [www.boxofficeindia.com](http://www.boxofficeindia.com/) as the website has been used by various research agencies and trade industries to present their reports published as follows:

FICCI KPMG 2010 (Back in the spotlight);

PWC India Entertainment and Media Outlook 2011; FICCI frames 2015 (#shootingforthestars);

FICCI KMG 2017 (Media for the masses; the promise unfolds); FICCI-EY 2018 (Re-imagining India’s M&F sector).

OTT stands for “Over The Top” and refers to any streaming service that delivers content over the internet.

OTT is earned through these advertisements. Overall, the business model on OTT is very simple. First, the platform spends money to make or buy its content, and then the content is sold by charging a charge from the audience or users.

How do OTT platforms make money?

These platforms earn through the Ad revenue model, where they charge brands and companies for posting ads on their platforms. In this model, the OTT platform offers both free and paid subscriptions to its users and generates revenue through both models. This platform is followed by Hotstar plus and Zee5

# System Analysis

## Feasible study

A feasibility study is an important assessment conducted at the early stages of a project to determine whether the proposed project is viable, practical, and achievable. It helps stakeholders make informed decisions about whether to proceed with the project or not. In the context of the Box Office App project, a feasibility study would be conducted to evaluate the project's viability from various perspectives. Here are the key aspects that would be assessed in a feasibility study for the Box Office App:

### Technical Feasibility:

Assess the technical capabilities and expertise required to develop the app with React.js and integrate with the chosen API.

Evaluate if the required technologies and tools are available and feasible for the development team to use.

Determine if the team has the necessary skills and experience to handle the project's technical challenges.

### Economic Feasibility:

Estimate the overall project cost, including development, API usage, hosting, maintenance, and any licensing fees.

Compare the projected costs with the expected benefits and potential revenue generation to determine if the app is financially viable.

### Operational Feasibility:

Evaluate how the Box Office App would fit into the existing operational processes and infrastructure (if applicable).

Identify any potential operational disruptions and assess the ease of integrating the app into the current system.

### Legal and Regulatory Feasibility:

Identify any legal or regulatory requirements that need to be addressed, such as data privacy laws or licensing agreements with API providers.

Ensure that the app complies with all applicable laws and regulations.

### Schedule Feasibility:

Assess the project timeline and deadlines to determine if the app can be developed within the required timeframe.

Identify any potential risks or bottlenecks that could affect the project's completion timeline.

### Market and User Feasibility:

Analyze the target market and user base for the Box Office App.

Conduct market research to understand the demand for such an app and potential competition.

Identify user needs and preferences to ensure that the app meets their expectations.

### Risk Assessment:

Identify potential risks and uncertainties that could impact the success of the project. Develop mitigation strategies to address and minimize these risks.

### Resource Feasibility:

Evaluate the availability of resources, including manpower, expertise, and technology, to complete the project successfully.

Based on the findings of the feasibility study, stakeholders can make an informed decision about whether to proceed with the development of the Box Office App or consider alternative approaches. It also helps in planning the project more effectively and mitigating potential risks that may arise during development and deployment.

## Hardware and software requirements

Hardware requirements refer to the specific physical components and specifications needed to run software applications effectively on a computer or electronic device. These requirements ensure that the hardware can support the software's functionality, performance, and stability. Hardware requirements can vary based on the type of software and its complexity.

Software requirements refer to the specific software components and configurations needed to run an application or system on a computer or electronic device. These requirements ensure that the software is compatible with the underlying operating system and dependencies. Software requirements can vary based on the nature of the application.

### HARDWARE REQUIREMENTS:

* + Processor : Pentium IV or higher
  + Hard Disk : 1 TB.
  + Monitor : 14’ Colour Monitor.
  + Mouse : Optical Mouse.
  + Ram : 4GB.

### SOFTWARE REQUIREMENTS:

* + Operating system : Windows 10.
  + Coding Language : JavaScript.
  + Front-End : Html, CSS, React JS.
  + Designing : Html, CSS, .
  + Framework : React.
  + API : <https://www.tvmaze.com/api>

# Development Environment

## a) Introduction of technology used

### HTML 5:

HTML is a markup language used for structuring and presenting content on the World Wide Web. It is the fifth and current version of the HTML standard.HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves and rationalizes the markup available for documents, and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications, because it includes features designed with low-powered devices in mind.

### CSS 3:

Cascading Style Sheets, fondly referred to as CSS, is a simple design language intended to simplify the process of making web pages presentable.

CSS handles the look and feel part of a web page. Using CSS, you can control the color of the text, the style of fonts, the spacing between paragraphs, how columns are sized and laid out, what background images or colors are used, layout designs variations in display for different devices and screen sizes as well as a variety of other effects.

### Bootstrap:

Bootstrap is a free and open-source front end development framework for the creation of websites and web apps. The Bootstrap framework is built on HTML, CSS, and JavaScript (JS) to facilitate the development of responsive, mobile-first sites and apps. Responsive design makes it possible for a web page or app to detect the visitor’s screen size and orientation and automatically adapt the display accordingly; the mobile first approach assumes that smartphones, tablets, and task-specific Mobile apps are employees' primary tools for getting work done and addresses the requirements of those technologies in design.

### React.js:

React.js is a widely-used JavaScript library for building user interfaces. It enables developers to create reusable UI components that efficiently update and render changes without a full-page refresh. In the Box Office App project, React.js is the core technology used for creating the front-end user interface. Its component-based architecture allows for

a modular and organized approach to designing the app's various pages, enhancing code maintainability and reusability.

### React Router:

React Router is a popular routing library for React.js applications. It allows developers to implement client-side routing, enabling the app to have multiple views and pages without the need for full-page reloads. In the Box Office App, React Router is utilized to handle navigation between different sections of the app, such as the Home Page, Movie/Show Details Page, Actors/Stars Page, etc. It enhances user experience by providing seamless and smooth transitions between different views.

### API:

An API (Application Programming Interface) is a set of rules and protocols that enables

different software applications to communicate and interact with each other. In the context of the Box Office App, the application integrates with a reliable movie database API. The API provides access to real-time movie data, including details about movies, shows, actors, and box office rankings. By leveraging an API, the app can dynamically fetch and display up-to- date information to users.

### Node.js:

Node.js is a powerful and popular open-source JavaScript runtime environment that allows developers to execute server-side code using JavaScript. In the Box Office App project, Node.js is used (optional) to set up a backend server for handling user authentication, managing sessions, and serving API requests. Its non-blocking and event- driven architecture makes it suitable for building scalable and efficient web applications.

By leveraging these technologies in the Box Office App, developers can create a robust, interactive, and responsive application that offers real-time movie data and entertainment updates to users. The combination of Node.js, API integration, React Router, and React.js enables the app to provide an engaging and seamless user experience, making it an effective platform for movie enthusiasts to explore and stay informed about the latest entertainment content.

### System Design

**a) Modular description**

### Home Page (Landing Page):

The Home Page serves as the landing page of the Box Office App. It welcomes users with an attractive and intuitive interface that provides a glimpse of the latest and popular movies, shows, and celebrity news. The page features eye-catching banners, carousel sliders, and quick access buttons to various sections of the app. Users can navigate seamlessly from the Home Page to explore movies, shows, actors, and other exciting content.

### Movies/Shows Page:

The Movies/Shows Page presents users with an extensive collection of the latest movies and TV shows. Users can browse through a well-organized grid view displaying movie/show posters, titles, release dates, and ratings. An advanced search bar allows users to find specific movies or shows by genre, title, or cast. Clicking on a movie/show opens its dedicated details page.

### Movie/Show Details Page:

The Movie/Show Details Page provides comprehensive information about a selected movie or TV show. It includes the movie/show poster, title, release date, genre, rating, synopsis, and a list of cast members. Users can also watch trailers and view user reviews and ratings for the movie/show. Additionally, they can add the movie/show to their watchlist or share it on social media.

### Actors/Stars Page:

The Actors/Stars Page showcases a list of popular actors and celebrities from the entertainment industry. Users can explore their favorite actors' profiles, including their photos, biography, filmography, and social media links. The page may also feature a search option to find actors quickly.

### Starred Page:

The Starred Page provides a personalized experience for users. It displays a list of movies and shows that users have added to their watchlist or marked as favorites. This page

helps users keep track of their preferred content and facilitates easy access to their chosen movies/shows.

### About Us:

The About Us section introduces the Box Office App to users and offers insights into its purpose and development. It includes a brief overview of the app's objectives, features, and its commitment to providing users with real-time entertainment updates. The section may also acknowledge the development team and express gratitude to any organizations or APIs that contributed to the app's functionality. Additionally, it may include contact information or links to the app's social media profiles for users to connect and provide feedback.

### Implementation and Testing

1. **Implementation and screenshots**

The implementation of the Box Office App with React.js and API integration involves creating various components, setting up routing, fetching data from the API, and managing the application state (optional). Here's a high-level overview of the steps involved:

### Development Process:

* + Requirement Gathering: Understanding the project scope, user expectations, and defining features.
  + Design and Wireframing: Creating wireframes and design mockups to visualize the application's layout and appearance.
  + Frontend Development: Implementing the user interface using React.js, HTML, and CSS.
  + API Integration: Connecting the application to a reliable movie database API for fetching movie details and gossip information.
  + Backend Development (optional): Setting up a backend server using Node.js for user authentication and other optional features.
  + Testing: Conducting thorough testing to identify and fix bugs and ensure the application functions smoothly.
  + Deployment: Deploying the application to a hosting service to make it accessible to users online.

### Project Setup:

* + Set up a new React.js project using Create React App or your preferred tool.
  + Install necessary dependencies, such as React Router, Redux (optional), Axios (for API requests), etc.

### Flow Charts:

A flow chart is a visual representation of the flow of processes or steps involved in a particular aspect of the project. It uses standardized symbols and arrows to illustrate the sequence and connections between different stages of the process. Flow charts are an effective way to present complex information in a clear and easy-to-understand manner. By including flow charts in the project report, you enhance its visual appeal and provide a clear

representation of the app's processes, making it easier for readers to understand the project's functionality and user interactions.

### Flow charts in project reports serve several purposes:

**Process Visualization:** Flow charts help readers understand the sequence of actions or steps involved in a specific process within the project. This could be the flow of data, the sequence of tasks, decision-making processes, or any other workflow.

**Clarity and Structure:** They provide a structured representation of the project's processes, making it easier for readers to follow the logical progression of events.

**Communication:** Flow charts facilitate effective communication by providing a visual aid that can be easily shared and understood by team members, stakeholders, or readers of the project report.

**Problem-Solving:** They are valuable tools for identifying bottlenecks, inefficiencies, or potential issues in the project's processes, helping project teams to optimize and improve their workflows.

**Documentation:** Flow charts serve as documentation for the project's processes, making it easier to reference and review procedures at a glance.

### Flow Chart Symbols:

Flow charts use specific symbols to represent different elements of a process. Some commonly used symbols include:

**Oval/Circle:** Represents the start or end point of the process.

**Rectangle:** Represents a process step or action.

**Diamond:** Represents a decision point or branching in the process.

**Arrow**: Indicates the direction of flow from one step to another.

**Parallel Lines:** Represents input/output or data storage.

**Connector:** Connects different parts of the flow chart that continue on separate pages.

### Level 0 DFD (Context Diagram):

**Box Office App**

**Starred**

**Search Actors**

**Search Movies**

**Search Shows**

1. **Level 1 DFD:**

**Box Office**

**App**

**Starred Data**

**Starred**

**Actors Data**

**Search Actors**

**Movie Data**

**Search Movies**

**Show Data**

**Search Shows**

### API Request Response Flow:

**API**

Routing

**Show Page**

**Starred Page**

Fetch Data

**Home Page**

Rendering

**Index.js**

**App.js**

**Component Structure:**

* + Create the required components for different sections of the app, such as Home, MovieDetails, BoxOfficeRankings, GossipSection, etc.
  + Implement a navigation bar or sidebar for easy navigation between different sections.

### Routing:

* + Set up routing using React Router to handle different views/pages of the application.

### API Integration:

* + Integrate the chosen movie database API to fetch movie data and celebrity gossips.

### Redux (Optional):

* + Implement Redux for state management (if required) to handle global app state, such as movie data, user authentication, etc.

### User Interface (UI):

* + Design the user interface with HTML, CSS, and React components to display movie information, gossip, and rankings.
  + Ensure the app is responsive and visually appealing on different screen sizes.

### User Interaction:

* + Implement user interactions, such as movie search, click events, and pagination.

### Authentication (Optional):

* + Implement user authentication if you decide to add personalized features like watchlists and reviews.

### Testing:

* + Conduct various testing types mentioned earlier to ensure the app’s functionality, performance, and security.

### Deployment:

* + Deploy the app to a hosting service or platform to make it accessible online.

### Codes:

**App.jsx**

import { HashRouter, Routes, Route } from 'react-router-dom';

import { QueryClient, QueryClientProvider } from '@tanstack/react-query'; import Home from './pages/Home';

import Starred from './pages/Starred'; import Show from './pages/Show'; import About from './pages/About'

import MainLayout from './components/MainLayout'; import { GlobalTheme } from './theme';

const queryClient = new QueryClient(); function App() {

return (

<QueryClientProvider client={queryClient}>

<GlobalTheme>

<HashRouter>

<Routes>

<Route element={<MainLayout />}>

<Route path="/" element={<Home />} />

<Route path="/starred" element={<Starred />} />

<Route path='about' element={<About />} />

</Route>

<Route path="/show/:showId" element={<Show />} />

<Route path="\*" element={<div>Not found</div>} />

</Routes>

</HashRouter>

</GlobalTheme>

</QueryClientProvider>

);

}

export default App;

### tvmaza.js

const BASE\_URL = 'https://api.tvmaze.com'; const apiGet = async queryString => {

const response = await fetch(`${BASE\_URL}${queryString}`); const body = await response.json();

return body;

};

export const searchForShows = query => apiGet(`/search/shows?q=${query}`); export const searchForPeople = query => apiGet(`/search/people?q=${query}`); export const getShowById = showId =>

apiGet(`/shows/${showId}?embed[]=seasons&embed[]=cast`); export const getShowsByIds = async showIds => {

const promises = showIds.map(showId => apiGet(`/shows/${showId}`)); return Promise.all(promises);

};

### Home.jsx

import { useState } from 'react';

import { useQuery } from '@tanstack/react-query';

import { searchForShows, searchForPeople } from './../api/tvmaze'; import SearchForm from '../components/SearchForm';

import ShowGrid from '../components/shows/ShowGrid'; import ActorsGrid from '../components/actors/ActorsGrid'; import { TextCenter } from '../components/common/TextCenter';

const Home = () => {

const [filter, setFilter] = useState(null);

const { data: apiData, error: apiDataError } = useQuery({ queryKey: ['search', filter],

queryFn: () => filter.searchOption === 'shows'

? searchForShows(filter.q)

: searchForPeople(filter.q), enabled: !!filter, refetchOnWindowFocus: false,

});

const onSearch = async ({ q, searchOption }) => { setFilter({ q, searchOption });

};

const renderApiData = () => { if (apiDataError) {

return <TextCenter>Error occured: {apiDataError.message}</TextCenter>;

}

if (apiData?.length === 0) {

return <TextCenter>No results</TextCenter>;

}

if (apiData) {

return apiData[0].show ? (

<ShowGrid shows={apiData} />

) : (

<ActorsGrid actors={apiData} />

);

}

return null;

};

return (

<div>

<SearchForm onSearch={onSearch} />

);

};s

<div>{renderApiData()}</div>

</div>

export default Home;

### Show.jsx

import { Link, useParams } from 'react-router-dom'; import { useQuery } from '@tanstack/react-query'; import styled from 'styled-components';

import { getShowById } from '../api/tvmaze';

import ShowMainData from '../components/shows/ShowMainData'; import Details from '../components/shows/Details';

import Seasons from '../components/shows/Seasons'; import Cast from '../components/shows/Cast';

import { TextCenter } from '../components/common/TextCenter';

const Show = () => {

const { showId } = useParams();

const { data: showData, error: showError } = useQuery({ queryKey: ['show', showId],

queryFn: () => getShowById(showId), refetchOnWindowFocus: false,

});

if (showError) {

return <TextCenter>We have an error: {showError.message}</TextCenter>;

}

if (showData) { return (

<ShowPageWrapper>

<BackHomeWrapper>

<Link to="/">Go back to home</Link>

</BackHomeWrapper>

<ShowMainData image={showData.image} name={showData.name} rating={showData.rating} summary={showData.summary} genres={showData.genres}

/>

<InfoBlock>

<h2>Details</h2>

<Details status={showData.status}

premiered={showData.premiered} network={showData.network}

/>

</InfoBlock>

<InfoBlock>

<h2>Seasons</h2>

<Seasons seasons={showData.\_embedded.seasons} />

</InfoBlock>

<InfoBlock>

<h2>Cast</h2>

<Cast cast={showData.\_embedded.cast} />

</InfoBlock>

</ShowPageWrapper>

);

}

return <TextCenter>Data is loading</TextCenter>;

};

export default Show;

const BackHomeWrapper = styled.div` margin-bottom: 30px;

text-align: left; a {

padding: 10px;

color: ${({ theme }) => theme.mainColors.dark}; text-decoration: none;

&:hover {

text-decoration: underline;

}

}

`;

const ShowPageWrapper = styled.div` margin: auto;

@media only screen and (min-width: 768px) { max-width: 700px;

}

@media only screen and (min-width: 992px) { max-width: 900px;

}

`;

const InfoBlock = styled.div` margin-bottom: 40px;

h2 {

margin: 0;

margin-bottom: 30px; font-size: 22px;

}

`;

### Starred.jsx

import { useQuery } from '@tanstack/react-query'; import { getShowsByIds } from '../api/tvmaze';

import { useStarredShows } from '../lib/useStarredShows'; import ShowGrid from '../components/shows/ShowGrid'; import { TextCenter } from '../components/common/TextCenter';

const Starred = () => {

const [starredShowsIds] = useStarredShows();

const { data: starredShows, error: starredShowsError } = useQuery({ queryKey: ['starred', starredShowsIds],

queryFn: () => getShowsByIds(starredShowsIds).then(result =>

result.map(show => ({ show }))

),

refetchOnWindowFocus: false,

});

if (starredShows?.length === 0) {

return <TextCenter>No shows were starred</TextCenter>;

}

if (starredShows?.length > 0) {

return <ShowGrid shows={starredShows} />;

}

if (starredShowsError) {

return <TextCenter>Error occured: {starredShowsError.message}</TextCenter>;

}

return <TextCenter>Shows are loading</TextCenter>;

};

export default Starred;

### About.jsx

import React from 'react'

const About = () => { return (

<div>

<h2 style={{textAlign: 'center'}}>About Us</h2>

<p>The Box Office App is a web application designed to provide users with up-to-date information about the latest movies, box office rankings, and celebrity gossips. The primary goal of this project is to create an intuitive and user-friendly platform that allows movie enthusiasts to stay informed about the latest happenings in the entertainment industry.</p>

<ul>

<h3>Future Scope</h3>

<li>Implementing user authentication and personalized user profiles.</li>

<li>Incorporating user-generated content such as movie reviews and ratings.</li>

<li>Expanding the gossip section to cover a wider range of entertainment news.</li>

<li>Integrating social media sharing options to promote content virality.</li>

<li>Collaborating with movie studios for exclusive content and promotions.</li>

</ul>

</div>

)

}

export default About

### MainLayout.jsx

import { Outlet } from 'react-router-dom'; import AppTitle from './AppTitle';

import Navs from './Navs';

const MainLayout = () => { return (

<div>

<AppTitle />

<Navs />

<Outlet />

</div>

);

};

export default MainLayout;

### Navs.jsx

import styled from 'styled-components'; import { NavLink } from 'react-router-dom';

const LINKS = [

{

text: 'Home',

to: '/',

},

{

text: 'Starred', to: '/starred',

},

{

text: 'About',

to:'/about',

},

];

const Navs = () => { return (

<div>

<NavList>

{LINKS.map(item => (

<li key={item.to}>

<LinkStyled to={item.to}>{item.text}</LinkStyled>

</li>

))}

</NavList>

</div>

);

};

export default Navs;

const NavList = styled.ul` display: flex;

justify-content: center; list-style: none; margin: 0 0 30px;

padding: 0; li {

margin: 0 10px;

}

`;

const LinkStyled = styled(NavLink)` display: block;

padding: 3px 15px; position: relative;

text-decoration: none;

color: ${({ theme }) => theme.mainColors.gray}; &.active {

color: ${({ theme }) => theme.mainColors.blue}; &:after {

content: ''; position: absolute; display: block; height: 2px;

left: 0%;

bottom: 0;

background-color: ${({ theme }) => theme.mainColors.blue}; animation: slide-in 0.3s ease-in forwards;

@keyframes slide-in { from {

left: 50%;

width: 0;

}

to {

left: 0%;

width: 100%;

}

}

}

}

`;

### AppTitle.jsx

import styled from 'styled-components'; export default function AppTitle(props) {

const {

title = 'Box Office',

subtitle = 'Are you looking for a movie or an actor?',

} = props;

return (

<TitleWrapper>

<h1>{title}</h1>

<p>{subtitle}</p>

</TitleWrapper>

);

}

const TitleWrapper = styled.div` text-align: center;

margin: 0 0 40px; h1 {

color: ${({ theme }) => theme.mainColors.blue}; letter-spacing: 10px;

text-transform: uppercase; margin: 0 0 10px;

}

p {

color: ${({ theme }) => theme.mainColors.dark}; margin: 0;

}

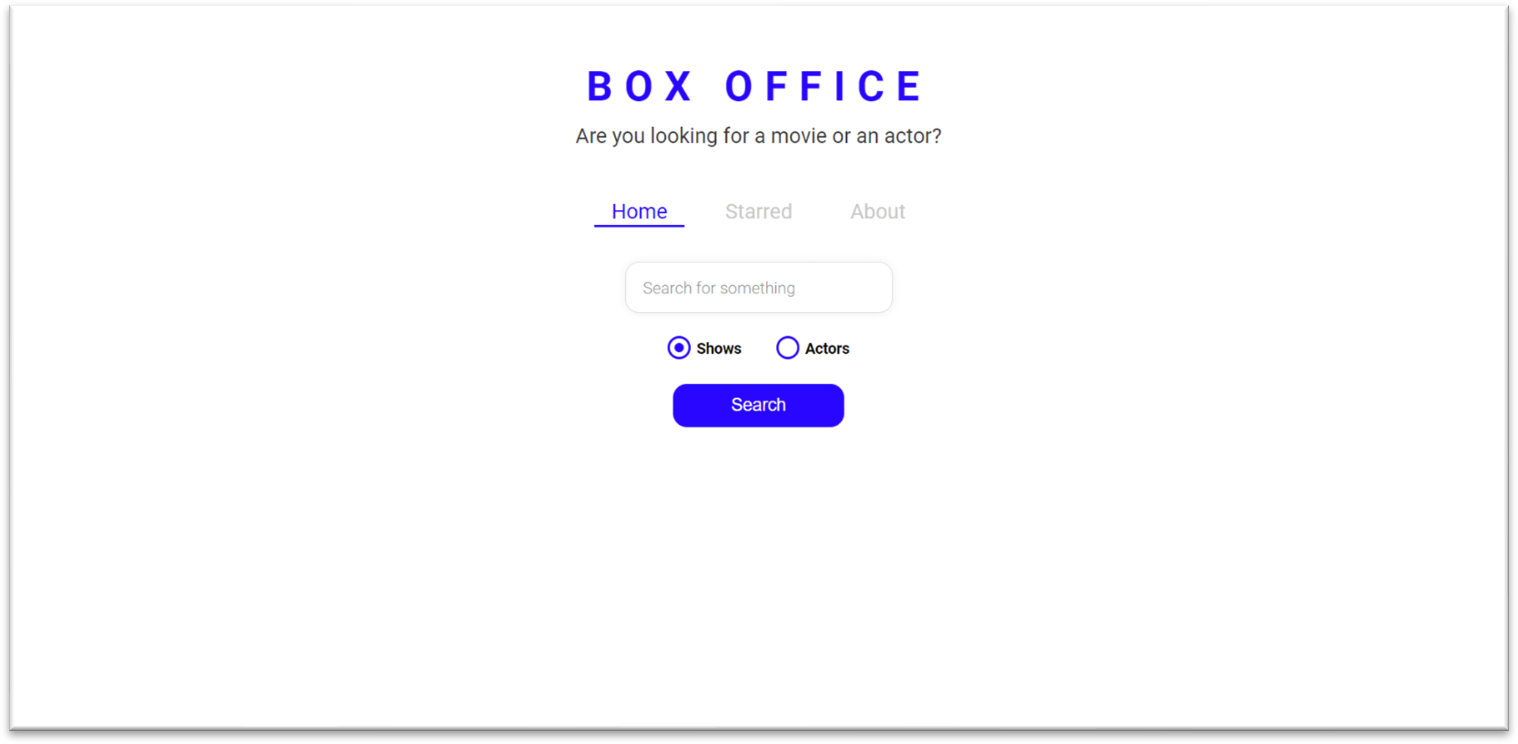
`;

### Screenshots:

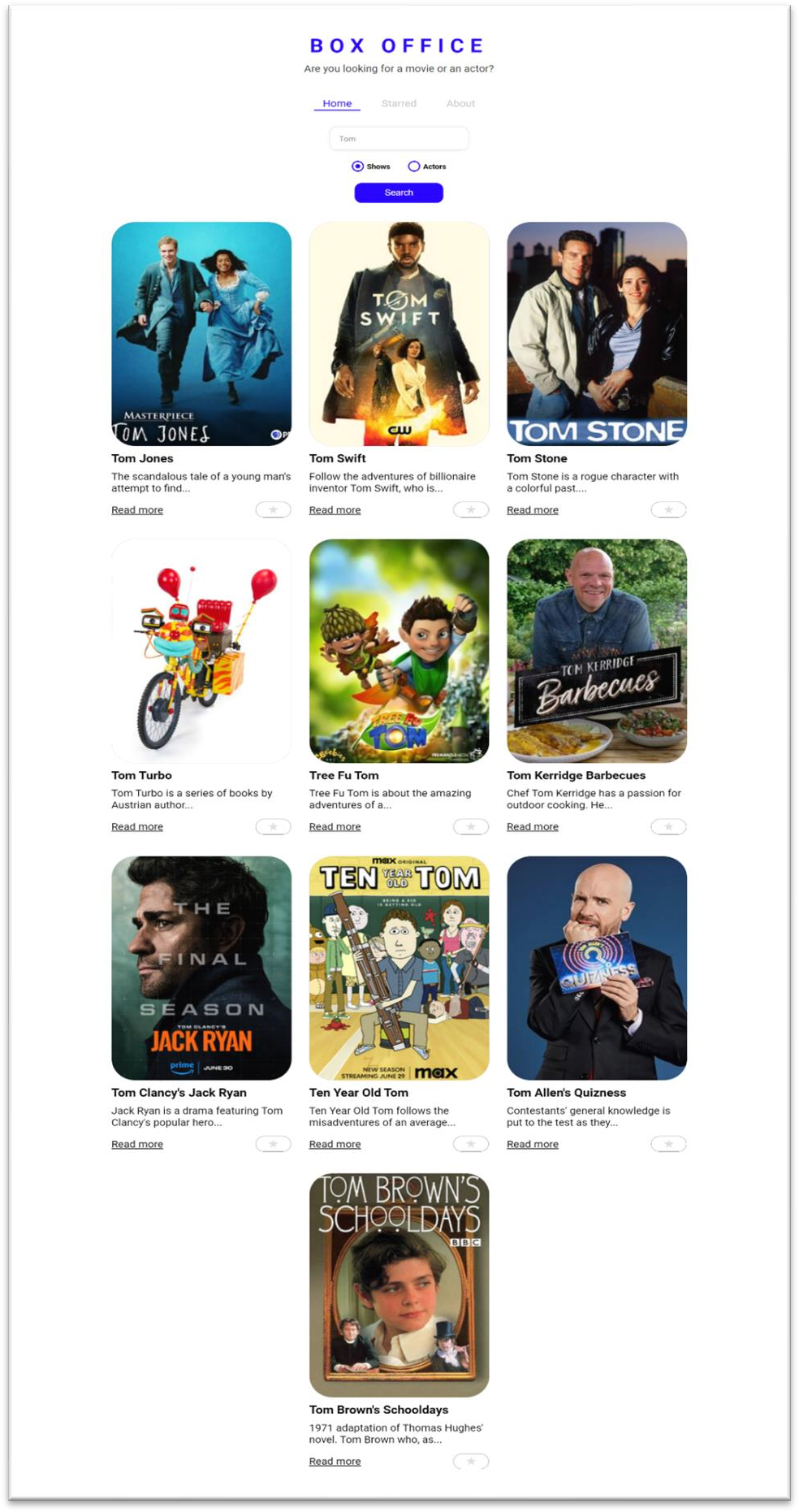
Here are some fictional screenshots to give you an idea of how the Box Office App might look like:

### Home Page:

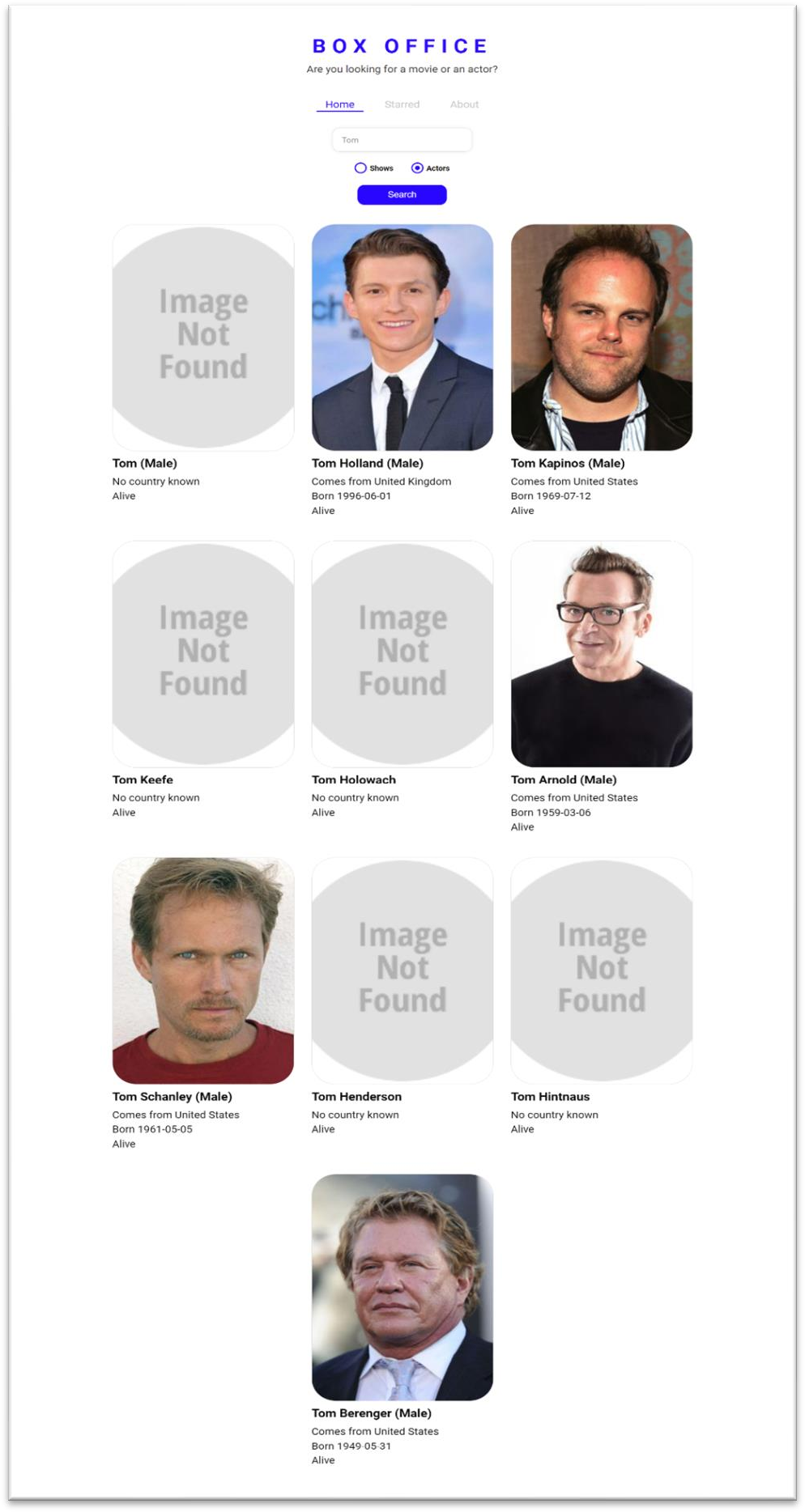
The Home Page serves as the landing page of the Box Office App. It welcomes users with an attractive and intuitive interface that provides a glimpse of the latest and popular movies, shows, and celebrity news. The page features eye-catching banners, carousel sliders, and quick access buttons to various sections of the app. Users can navigate seamlessly from the Home Page to explore movies, shows, actors, and other exciting content.



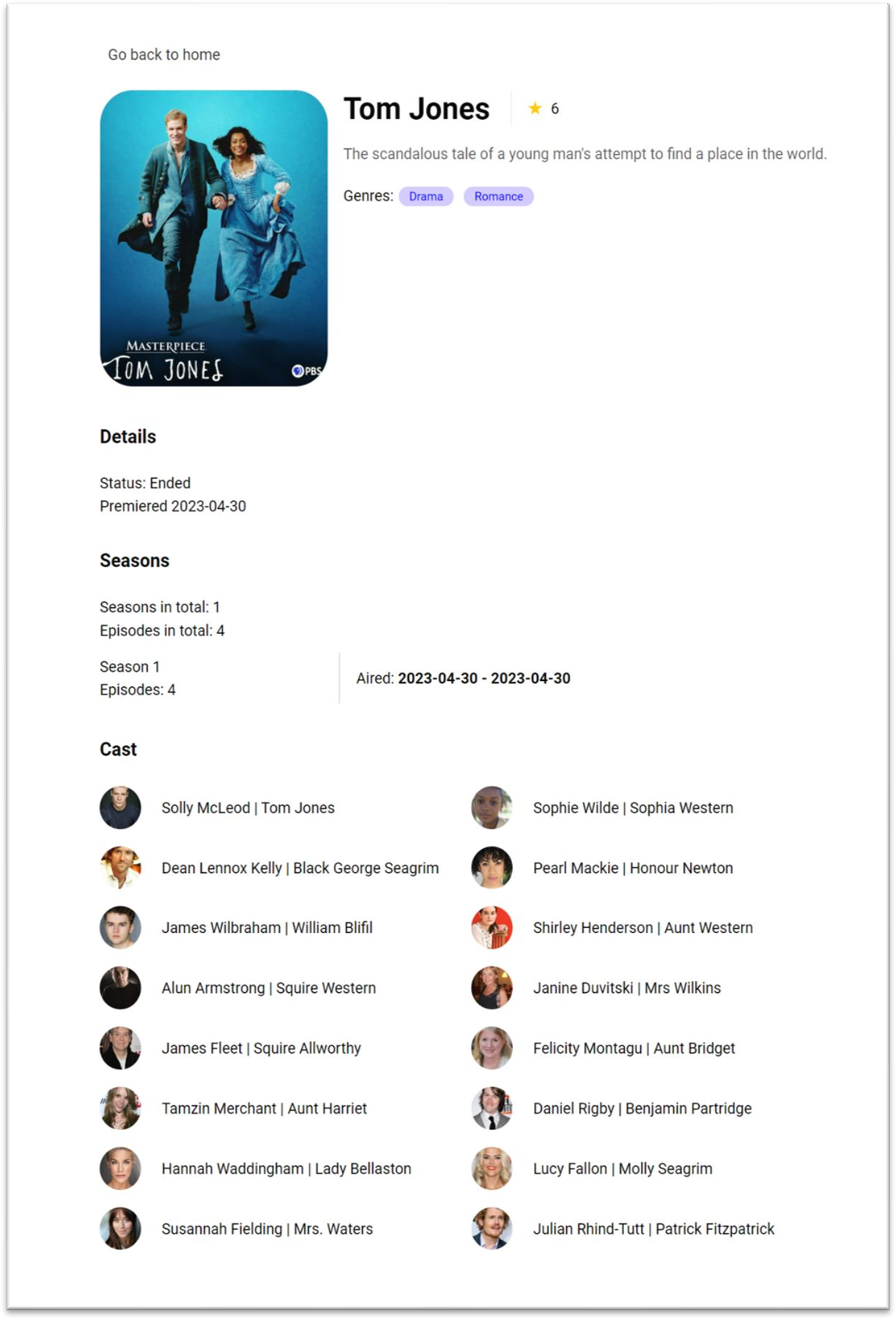
### Movies/Shows Page:



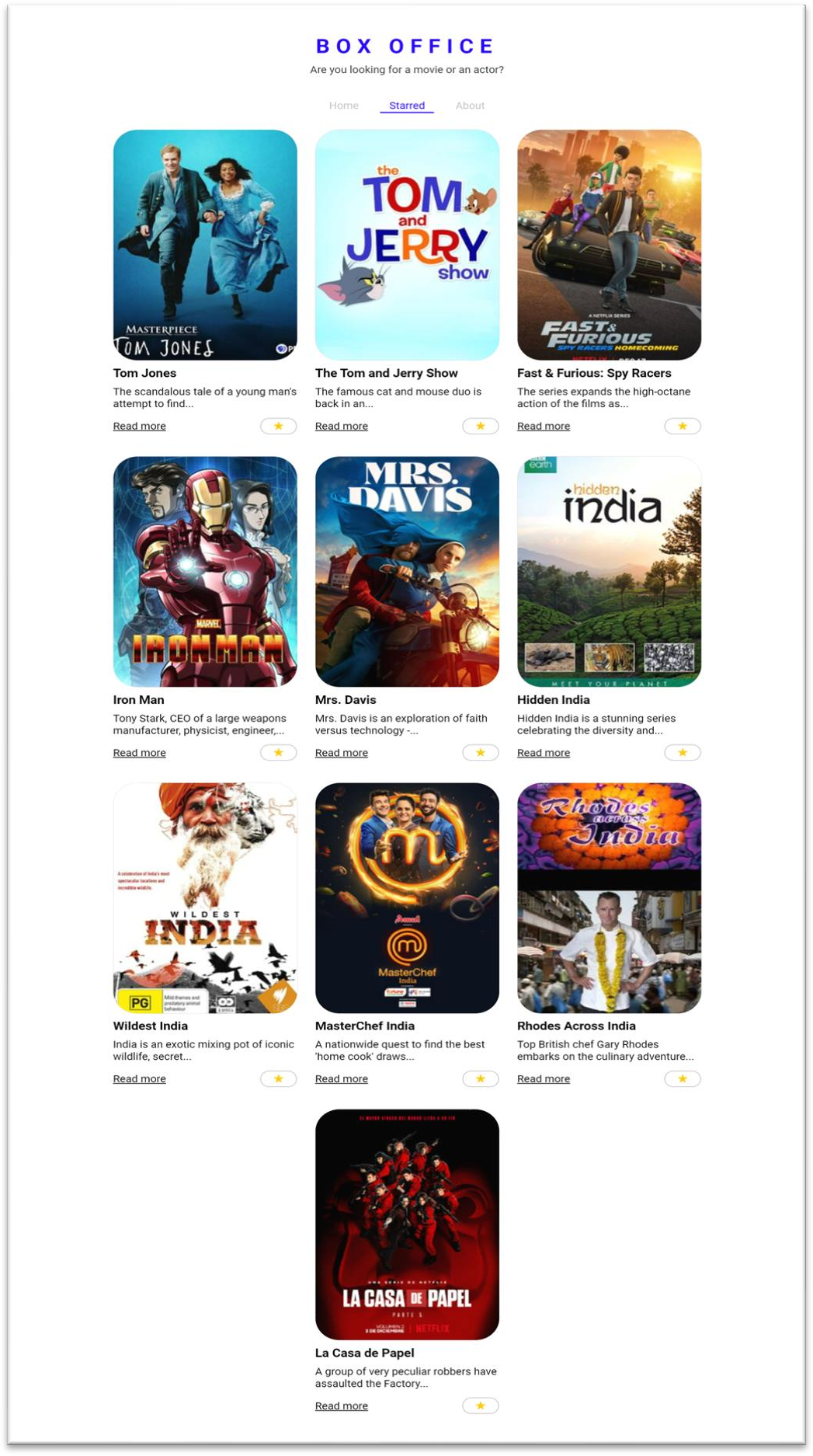
**Actors/Star Page:**



### Movies/Shows Details Page:

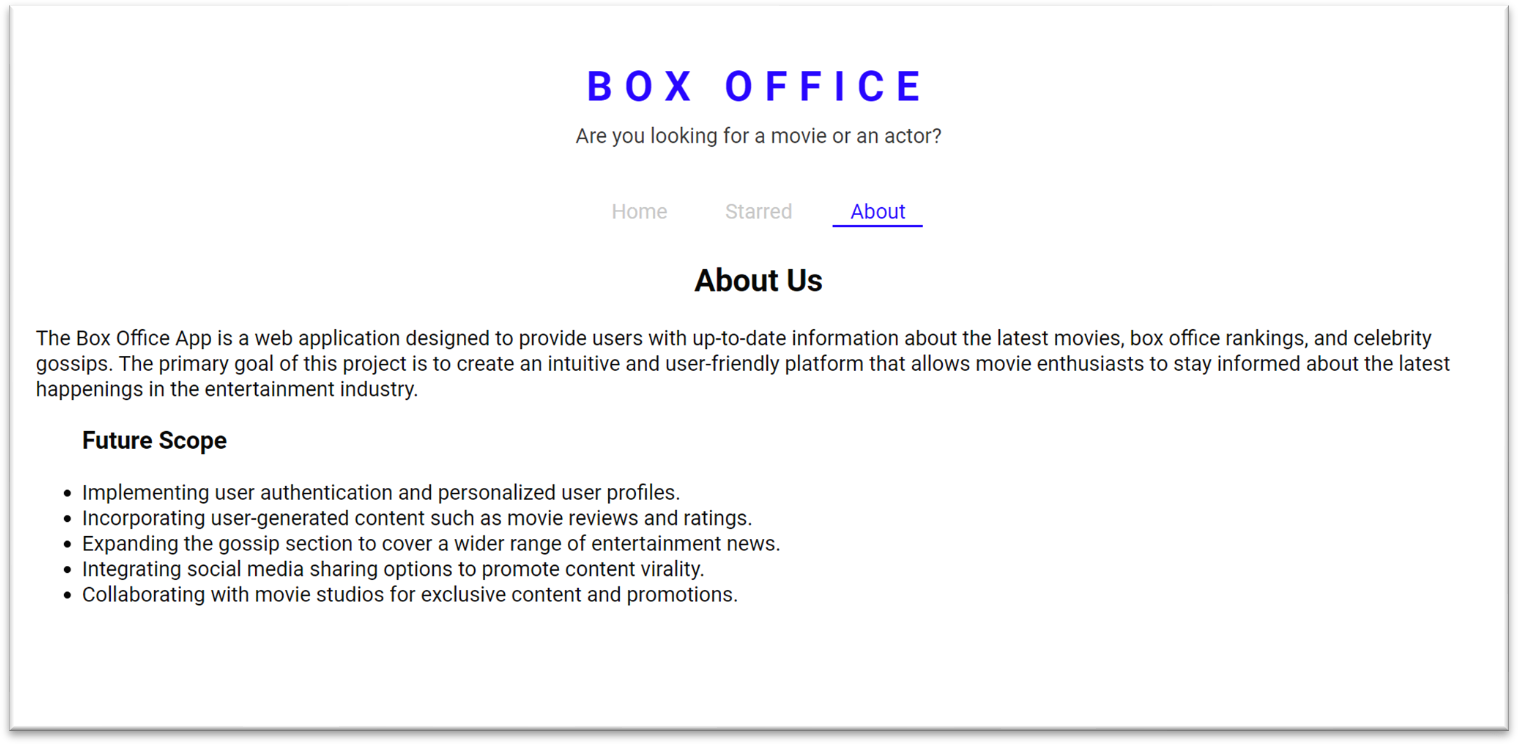


**Starred Movies/Shows Page:**



### About Us Page:

The About Us section introduces the Box Office App to users and offers insights into its purpose and development. It includes a brief overview of the app's objectives, features, and its commitment to providing users with real-time entertainment updates. The section may also acknowledge the development team and express gratitude to any organizations or APIs that contributed to the app's functionality. Additionally, it may include contact information or links to the app's social media profiles for users to connect and provide feedback.



### Testing

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, 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.

### TYPES OF TESTS

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

* Test individual components of the React.js application to ensure they function correctly.
* Use testing frameworks like Jest and Enzyme to write unit tests for components, actions, and reducers.

### Integration testing

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

* Verify the interactions between different components, modules, and APIs within the application.
* Ensure that data flows correctly between various parts of the app.

### Functional test

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted. Invalid Input : identified classes of invalid input must be rejected. Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised. Systems/Procedures : interfacing systems or procedures must be invoked.

* Test the core functionalities of the app, such as movie search, gossip section, and box office rankings.
* Verify that user interactions trigger the correct actions and responses.

### System Test

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.

### White Box Testing

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

### Black Box Testing

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

### Unit Testing

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

### Test strategy and approach

Field testing will be performed manually and functional tests will be written in detail.

### Test objectives

* + All field entries must work properly.
  + Pages must be activated from the identified link.
  + The entry screen, messages and responses must not be delayed.

### Features to be tested

* + Verify that the entries are of the correct format
  + No duplicate entries should be allowed
  + All links should take the user to the correct page.

### Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g., components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results**: All the test cases mentioned above passed successfully. No defects encountered.

### Acceptance Testing

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results**: All the test cases mentioned above passed successfully. No defects encountered.

### Conclusion

The development of the Box Office App using React.js and API integration has been a significant and rewarding endeavor. This project aimed to address the limitations of the traditional entertainment information system and create a user-friendly, real-time platform for movie enthusiasts and entertainment seekers. Through this conclusion, I reflect on the project's accomplishments, challenges faced, key takeaways, and the potential impact of the Box Office App.

### Accomplishments:

The successful completion of the Box Office App demonstrates the effectiveness of using modern web technologies like React.js and API integration to create a feature-rich and dynamic application. The app provides users with a seamless experience to explore the latest movies, TV shows, box office rankings, celebrity gossips, and more. The modular design of the app ensures easy navigation and quick access to desired content. The use of React Router enhances user interaction by enabling client-side routing, reducing page reloads, and improving the overall performance.

### Challenges and Learnings:

During the development journey, several challenges were encountered, including API integration complexities, handling asynchronous data fetching, and managing state effectively. However, these challenges served as valuable learning opportunities. I gained a deeper understanding of React.js concepts, such as component lifecycle, state management, and reusability, which significantly contributed to the app's functionality and performance. The project also taught me the importance of comprehensive testing to identify and address potential issues, ensuring a smooth user experience.

### Future Scope:

* Implementing user authentication and personalized user profiles.
* Incorporating user-generated content such as movie reviews and ratings.
* Expanding the gossip section to cover a wider range of entertainment news.
* Integrating social media sharing options to promote content virality.
* Collaborating with movie studios for exclusive content and promotions.

### Bibliography

React.js Documentation:

Website: <https://reactjs.org/docs/getting-started.html>

React Router Documentation:

Website: <https://reactrouter.com/web/guides/quick-start>

Redux Documentation (Optional, if used for state management): Website: <https://redux.js.org/introduction/getting-started>

Node.js Documentation (Optional, if used for backend server): Website: <https://nodejs.org/en/docs/>

Movie Database API Documentation (The API you integrated for movie data): Website: <https://www.tvmaze.com/api>

Postman Documentation (for API testing, if used):

Website: <https://learning.postman.com/docs/getting-started/introduction/>

Jest Documentation (for testing, if used): Website: <https://jestjs.io/docs/getting-started>