OULU UNIVERSITY OF
APPLIED SCIENCES

Ujjawal Shrestha

IMPLEMENTING API IN REACTJS

IMPLEMENTING API IN REACTJS

Ujjawal Shrestha
Bachelor's Thesis
Autumn 2020
Information Technology
Oulu University of Applied Sciences

ABSTRACT

Oulu University of Applied Sciences
Degree Programme, Information Technology

Author: Ujjawal Shrestha
Title of the bachelor's thesis: Implementing API in ReactJS.
Supervisor: Veijo Väisänen
Term and year of completion: Autumn 2020 Number of pages: 43

The aim of this thesis was to implement the movie database API (Application Programming Interface) in ReactJS and develop a website from the retrieved data. The website needed to provide information about movies, television shows and celebrities which were already stored in the TMDb (The Movie Database) API. The website had to be user-friendly, responsive, interactive and informative.

During the thesis, Axios was used to retrieve data of movies, television shows and celebrities from the movie database API and ReactJS of version 16.13.1 or above along with Redux was used to display that data to the user in the front-end. Material-UI and Ant design along with Bootstrap and SCSS (Sassy Cascading Style Sheets) were used to design the website. Node.js, Express and mLab were used in the backend to develop an API. Google Chrome and Opera were the primary testing browsers. GitHub was used to store the code and Netlify was used for hosting the website.

As a result, the website is responsive, attractive, interactive, informative, and user-friendly. The website can be used by anyone who seeks for the information about movies, television shows or celebrities.

Keywords: ReactJS, API, Redux, User Interface, Node.js, Express, MongoDB, Material-UI, Axios

PREFACE

The basis for this thesis originally stemmed from my interest to learn and implement an API in ReactJS along with Redux. The development of the website and the writing of this thesis took place in Oulu. The development of the website would not have been possible without the API provided by The Movie Database. Therefore, I would like to show my gratitude to The Movie Database.

In truth, I could not have achieved my goal to develop the website without a strong support group. I would like to thank my thesis supervisor Veijo Väisänen and my language teacher Kaija Posio for their constant support and guidance. I would also like to thank my brother Prajwal Shrestha, who has provided me with his patient advice and guidance throughout the development of the website.

Oulu, 31.08.2020
Ujjawal Shrestha

CONTENTS

VOCABULARY

API	Application Programming Interface
CORS	Cross-Origin Resource Sharing
CSS	Cascading Style Sheets
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
JS	JavaScript
JSON	JavaScript Object Notation
NPM	Node Package Manager
SASS	Syntactically Awesome Style Sheets
SCSS	Sassy Cascading Style Sheets
TMDb	The Movie Database
TV	Television
UI	User Interface
URL	Uniform Resource Locator
VS Code	Visual Studio Code
XML	Extensible Markeup Language

1 INTRODUCTION

This thesis is related to the development of the website that could be used by the users to get information about movies, TV (television) shows, and celebrities. The website was a self-learning project developed by the author, to get a hands-on experience of using APIs (Application Programming Interface) in ReactJS along with Redux.

There were two possible ways to retrieve the data from the API. One was by using the built-in fetch function of JavaScript and the other was by using the Axios library. Axios had features, such as Axios progress bar and an interceptor, to run before any of our HTTP (Hypertext Transfer Protocol) request, which was lacking in a built-in fetch function. Thus, Axios was used as a tool to send a request to the API and retrieve data as a response from the API. [1.]

The development was done by using ReactJS along with Redux to display retrieved data on the frontend with the UI library: Material-UI and Ant design to design the website. Bootstrap and SCSS (Sassy Cascading Style Sheets) were also used to animate the website and to make it interactive, responsive, and attractive. The website also uses libraries, such as react-helmet, react-js-pagination, react-slick, react-photoswipe-2, and react-router-dom, to make the content on the website more dynamic. The components of the website are reusable. Backend, using Node.js, Express.js, and mLab, was also implemented in the website for the users to mark their favorite movies or TV shows and to display them in their dashboard of the website. Moreover, the website has a responsive view which can be viewed in large, medium, or small screen devices as required by developers and users. Furthermore, the website also has an authentication system.

This thesis provides detailed information about the technologies used during the development of the website. It also provides information about the implementation process of the website.

2 FRONT-END

Front-end, also known as client-side, refers to a part of the website that users can visualize and interact with. It is the interior design of the website that is visible to users. This section includes the explanation of the technologies that were used in the front-end.

2.1 HTML

The Hypertext Markup Language, HTML in short, is the structure of a webpage. Tags are used to control the structure of a webpage and are enclosed using angle brackets. For example: <html>. Everything the user sees in a browser is within a tag. HTML tags such as “html”, “head”, “title”, “body”, are the tags that every webpage should have. Additionally, tags, such as <a>, <p>, and <h1>, can contain other tags as their child-elements. However, the <img /> tag, as shown below (FIGURE 1), does not contain child-elements and shows the content directly to the webpage. With a “.htm” or “.html” file extension, the HTML documents are saved.

```
<!DOCTYPE html>
<html>
  <head>
    <title>This is title tag</title>
  </head>
  <body>
    <h1>This is heading tag.</h1>
    <p>This is paragraph tag.</p>
    <a href="tmdb.com">This is anchor tag.</a>
    <img src="./assets/logo.png" alt="logo" />
  </body>
</html>
```

FIGURE 1. An example of a basic HTML document.

In FIGURE 1, the basic example of the HTML document with various tags is shown.

2.2 SCSS

Sassy Cascading Style Sheet, SCSS in short, is a commonly used syntax of SASS (Syntactically Awesome Style Sheets). SCSS is a superset of CSS (Cascading Style Sheets), which means that it is very similar to CSS. It can be used to style the HTML element. It uses ".scss" as a file extension. [2.]

```scss
$text-color: #000;

body {
  color: $text-color;
}
@mixin flex-center {
  @include flex-basic;
  font-size: 10px;
}
```

FIGURE 2. An example of basic SCSS syntax.

FIGURE 2 shown above represents the basic SCSS syntax which consists of a selector, declaration, property, value, variable, function. The selector is the HTML element ("body" in this case), which is to be styled. There can be one or more than one declaration that is separated by a semi-colon (;) and wrapped by curly braces. In SCSS, the declaration contains a property and value or variable which are separated by a colon (:). Mixin is a function that contains a set of styles that can be used for the other selector as well.

2.3 ReactJS

ReactJS, also known as React or React.js, is an open-source JavaScript library created in March of 2013 by Facebook for building a UI (User Interface). It is used to develop and operate the dynamic UI of the webpages that have high incoming traffic. ReactJS strives to provide speed, simplicity, and scalability. ReactJS uses components, states, and props to render, update, and re-render contents on the webpages. It is used to develop a single page application with the help of many different reusable components. [3.]

```
class HelloMessage extends React.Component {
  render() {
    return (
      <div>
        Hello {this.props.name}
      </div>
    );
  }
}
```

FIGURE 3. An example of React Component. [4]

FIGURE 3 shown above shows an example of a basic React Component. The “HelloMessage” is the JavaScript class that extends the React.Component class and is exported to use in other parent or child components. The render function inside the component returns what the component should show on the pages. The component can be accessed several times as a child component in any other render function of the component by declaring it as: <HelloMessage />. While rendering the class, props (“name” in this case) can be passed with different values to show on the webpage. Furthermore, components also have states, which are specific to the component that can be used to update the component with new data.

```
const HelloMessage = ({ name }) => {
  return <div>Hello {name}</div>;
};
```

FIGURE 4. An example of a functional React Component.

FIGURE 4 shown above shows an example of a basic functional React Component. The above component is called a functional component because it is a JavaScript arrow function. It is similar to a class component and props can also be passed from the parent component. The component is rendered by declaring it as: <HelloMessage />. The difference between a function component and a

class component is that a class component can have states as well as props but, a functional component can only have props.

Based on NPM (Node Package Manager) downloads data shown below in FIGURE 5, ReactJS is one of the most popular JavaScript frameworks used to design UIs.

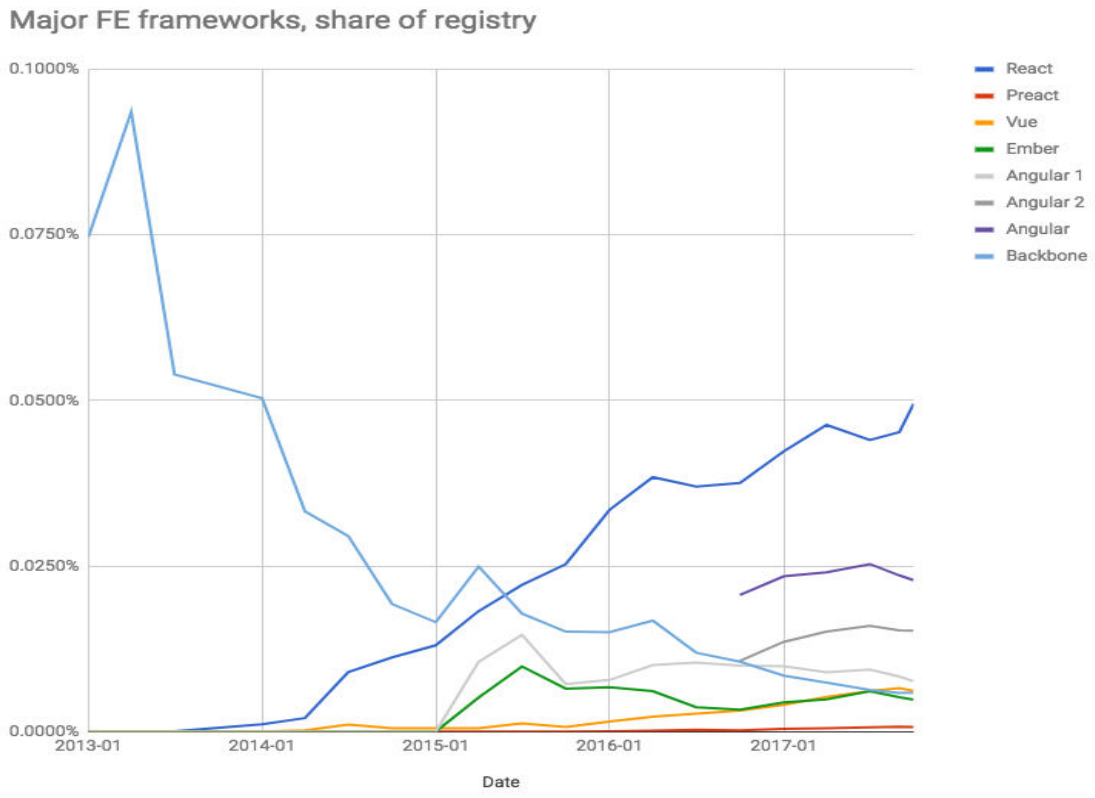

FIGURE 5. A popularity data of JavaScript Frameworks. [5]

2.4 Redux

Created by Dan Abramov and Andrew Clark in 2015, Redux is an open-source JavaScript library used for managing the state of an application. It is most commonly used with ReactJS than any other JavaScript Frameworks, such as AngularJS, Vue.js, Ember, Polymer. A single immutable state tree is formed to maintain all the state of an application. Predictability of outcome, organized codes, server rendering, maintainability, developer tools, and ease of testing are some of the benefits of using Redux. The creation of Redux was inspired by

Facebook's Flux Architecture and influenced by the functional programming language Elm. [6.]

Store, Action, and Reducer are three fundamental principles of Redux. The store makes it easier to inspect or debug an application by storing all the states of an application into a single tree. Actions are just a plain object that centralizes all the changes and ensures that views and network callbacks are not writing directly to the state. When the action is executed, the reducer changes the previous state according to the action and returns to the next state. [7.]

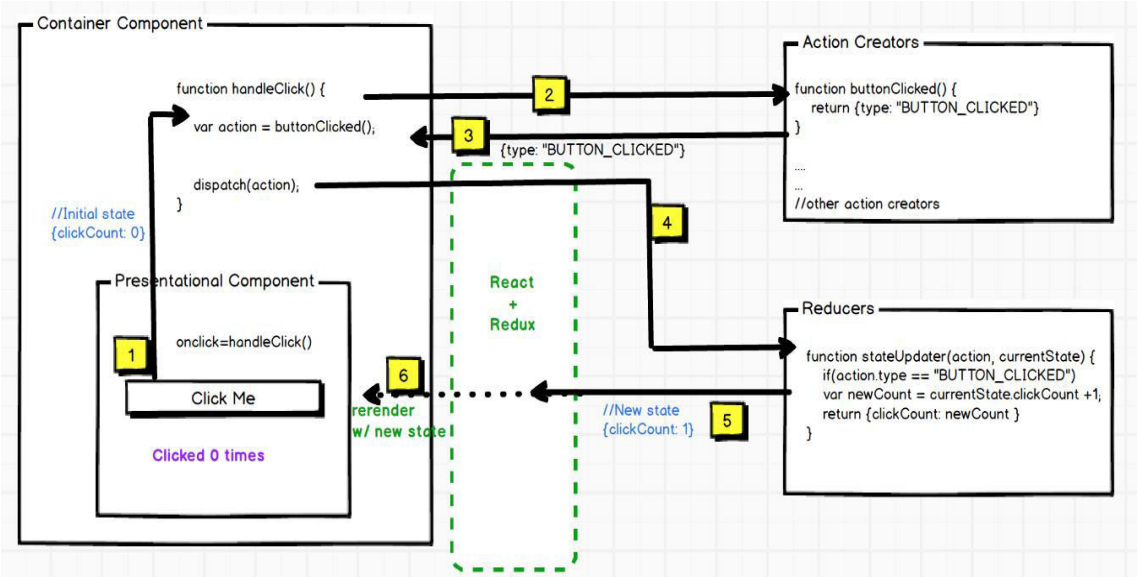

FIGURE 6. The React Redux Life Cycle. [8]

FIGURE 6 shown above shows the life cycle of React-Redux.

2.4.1 React-Redux

When ReactJS and Redux are used together, it is vital to connect both libraries using React-Redux to transfer data to each other. React-Redux is the official React UI connecting library for Redux, which helps to create small functions by breaking down the large functions. React-Redux optimizes the performance by implementing the optimization system which helps ReactJS to re-render the components only when needed. It allows React components to read data from a Redux store, and dispatch actions to the store to update data in the store. [9.]

2.4.2 Redux-Thunk

Thunk is a function that is used to delay the dispatch of an action. It is also used to dispatch the action only if the particular conditions are met. It wraps an expression to delay its evaluation. Redux-Thunk was used as a middleware to return a function instead of action through the action creator. The store methods “dispatch” and “getState” are received as parameters by the inner function. [10.]

```
import { createStore, applyMiddleware } from 'redux';
import thunk from 'redux-thunk';
import rootReducer from './reducers';

const store = createStore(rootReducer, applyMiddleware(thunk));
```

FIGURE 7. An example of enabling Redux-Thunk.

FIGURE 7 shown above is an example of the way to enable the Redux-Thunk, using the “applyMiddleware” function, in an application where Redux is being used to manage the state of an application.

2.5 Other Libraries (front-end)

The website was built using React as a primary tool for front-end. However, it would have been time-consuming and unpleasant for the author to develop it using React alone. To make the development of the website fast, seamless, and pleasant, some additional React libraries were used. Those libraries were installed in the project using an “npm install --save library-name” command line, where library-name is replaced with the name of the library that the author was installing.

FIGURE 8 shown below is the list of libraries with their respective versions that were installed in the front-end to support the development of the website. The information about those libraries is described below in this section.

```
"dependencies": {
  "@material-ui/core": "^4.9.14",
  "@material-ui/icons": "^4.9.1",
  "@testing-library/jest-dom": "^4.2.4",
  "@testing-library/react": "^9.3.2",
  "@testing-library/user-event": "^7.1.2",
  "antd": "^4.2.4",
  "axios": "^0.19.2",
  "dotenv": "^8.2.0",
  "node-sass": "^4.14.1",
  "react": "^16.13.1",
  "react-color-extractor": "^1.1.2",
  "react-dom": "^16.13.1",
  "react-helmet": "^6.0.0",
  "react-js-pagination": "^3.0.3",
  "react-photoswipe-2": "^1.3.2",
  "react-redux": "^7.2.0",
  "react-router-dom": "^5.2.0",
  "react-scripts": "3.4.1",
  "react-slick": "^0.26.1",
  "redux": "^4.0.5",
  "redux-thunk": "^2.3.0",
  "slick-carousel": "^1.8.1"
},
```

FIGURE 8. The version of libraries installed in the front-end.

2.5.1 Axios

Axios is a promise-based HTTP client for JavaScript which can be used in both front-end and back-end parts of an application. Axios makes it easier to communicate with Rest API and perform create, read, update, and delete operations. It makes HTTP requests from node.js. It can be used to intercept and transform requests as well as responses. Axios automatically converts the data to JSON (JavaScript Object Notation). [11.]

```
import axios from 'axios';

axios
  .get('https://api.github.com/users/sideshowbarker')
  .then((response) => {
    console.log(response.data);
  })
  .catch((error) => console.log(error));
```

FIGURE 9. An example of sending get request.

FIGURE 9 shown above is an example of sending a get request to an API (https://api.github.com/users/sideshowbarker in this case) using Axios library and then, logging the response data in the console or catching an error.

2.5.2 Material-UI

Material-UI, created in 2014, is an open-source library that implements material design provided by Google. It uses a grid system and is one of the top UI libraries for ReactJS. It provides dynamic styles that are generated at runtime, and have nested themes with intuitive overrides. Because of code splitting, the load time is reduced. The command line “npm install --save @material-ui/core” is used to install the Material-UI library in the project. It can also be installed using the yarn package manager. [12.]

2.5.3 Ant Design

Ant Design, created by Chinese conglomerate Alibaba, is an open-source React UI library that is useful for building elegant user interfaces. According to stars on GitHub, Ant Design is currently in a second position after Material-UI. It is easy-to-use and can be overridden easily. Using the command line “npm install --save antd”, Ant Design can be installed in the project. It can also be installed using the yarn package manager. Additionally, the CSS file of Ant Design must be imported. The import must be done in the “index.js” file to use it across the application. [13.]

2.5.4 React Helmet

React Helmet is a simple component that is used to modify the content of the head section dynamically, on a webpage. Dynamic titles, descriptions, and meta tags, which are helpful for search engine optimization, can be set using React Helmet. The command line “npm install --save react-helmet” is used to install the React Helmet. It can also be installed using the yarn package manager.

FIGURE 10 shown below is an example of usage of React Helmet using the class component of ReactJS to modify the title and meta tag in the head section of a webpage. However, the functional component can also be used while using React Helmet.

```
import { Helmet } from 'react-helmet';

class App extends React.Component {
  render() {
    return (
      <Helmet>
        <title>Updated title</title>
        <meta
          name='description'
          content='Get information about Movies, TV Shows and Celebrities.....'
        />
      </Helmet>
    );
  }
}
```

FIGURE 10. An example of usage of React Helmet.

2.5.5 React-JS-Pagination

React-JS-Pagination is a react component that helps to render pagination. It does not have any built-in styles.

```
import Pagination from 'react-js-pagination';

class App extends React.Component {
  render() {
    return (
      <Pagination
        activePage={activePageNumber}
        itemsCountPerPage={itemsInOnePage}
        totalItemsCount={totalResults}
        pageRangeDisplayed={pageNumberToDisplay}
        onChange={handlePageChange}
      />
    );
  }
}
```

FIGURE 11. An example of the usage of react-js-pagination.

FIGURE 11 shown above shows the basic usage of react-js-pagination. It shows the active page number, items per page, the number of total items, the range of page number to show, and a function that triggers the change in the page number.

3 APPLICATION PROGRAMMING INTERFACE (API)

The Application Programming Interface, API in short, is an interface or communication protocol between a client and a server. A publicly available web-based API returns data, likely in JSON (JavaScript Object Notation) or XML (Extensible Markup Language). An API is not the database or even the server, it is simply code that governs the access point for the server. The API provides a fast, consistent and reliable way to get third party data. [14.]

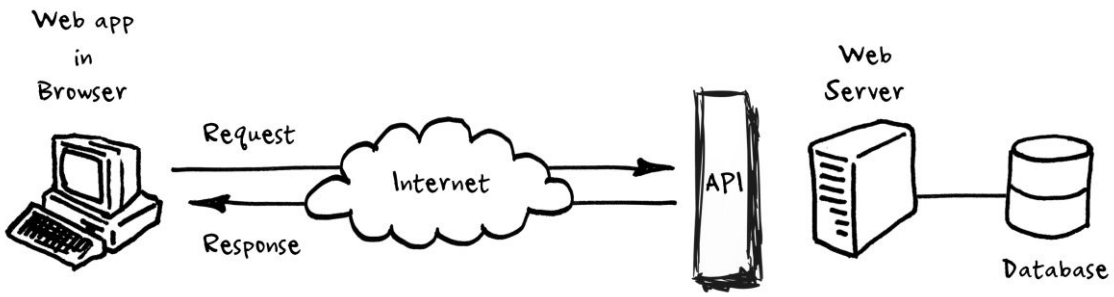

FIGURE 12. API accessing the database through an access point. [14]

FIGURE 12 shows the process where the API receives the request and sends back the response in the JSON or XML format.

The Movie Database (TMDb)

For this thesis, the author has used the API provided by TMDb. For the TMDb API to be used on the website, a person has to log in or sign up on the TMDb website and request an API key. The key is unique and should be sent in every request. There are two different APIs provided by the TMDb that are a Commercial API and a Developers API. The author has used the developer API for this thesis and for implementing it on the website. There are different URLs to request to retrieve different data from the TMDb API. In a single request, the trailers, images, cast, and crew of the movie or TV show can be returned in the JSON format using the “append_to_response” method of the API [15].

https://api.themoviedb.org/3/search/multi?api_key=apikey&language=en-US&query=black&page=1 [16] is used to search and retrieve data for movies, TV shows, and a celebrity in a single request. In this particular case, the query “black” is being searched.

https://api.themoviedb.org/3/discover/movie?api_key=apikey&language=en-US&sort_by=popularity.desc&page=1 [17] is used to sort the list of movies based on their popularity. This API has several sort options. It can be used to retrieve movies by their release dates, keywords, ratings, or certification.

https://api.themoviedb.org/3/movie/movie_id?api_key=apikey&language=en-US&append_to_response=videos,images,credits [18] is used to retrieve details about a movie that has “movie_id” as a unique ID. In this API, the “append_to_response” method has been used to retrieve additional details, such as videos, images, and credits of the movie.

Similarly, instead of a movie, a TV show or person can be used in the API to retrieve the details of the TV show or a celebrity. For a movie and a TV show, almost every query parameter of an API is the same. However, while retrieving details about a celebrity, the “append_to_response” method can be used to retrieve their movie or TV credits, images, and social media ID.

While developing the website, the author has also created the API that is used for authenticating users and saving movies or TV shows in their favorite lists.

4 BACK-END

Back-end, also known as server-side, refers to the part of the website that users do not directly interact with. It is responsible for manipulating and storing data of an application. This section includes the explanation of the technologies that were used in the back-end.

4.1 Node.js

Node.js is a runtime environment for executing JavaScript code outside of a browser. It is used to build back-end service, also known as API, which powers a front-end application. Node.js is ideal for building highly-scalable, faster, data-intensive, and real-time applications. It uses a non-blocking and event-driven Input/Output model, which makes it efficient and lightweight. It is used by many large companies, such as Netflix, LinkedIn, Trello, and PayPal [19]. [20.]

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello Node.js users');
});

server.listen(port, hostname, () => {
  console.log(`Server is running at http://${hostname}:${port}/`);
});
```

FIGURE 13. An example of HTTP server rendering using Node.js. [20]

FIGURE 13 shown above is an example HTTP server built using Node.js. The code in the above figure returns the text “Hello Node.js users” on the server with a hostname “127.0.0.1” and a port “3000”. The server also logs a message “Server is running at http://127.0.0.1:3000/” in command.

4.2 Express.js

Express, also known as Express.js or ExpressJS, is a flexible and open-source web application framework that was developed by TJ Holowaychuk and is maintained by the Node.js Foundation. Express provides a minimal interface to build applications. It also provides the tools to make the development process easy and fast. Using HTTP utility methods and the middleware of Express makes it quick and easy to create robust APIs. Feathers, Kraken, NestJs are some of the most popular frameworks that are built on Express. [21.]

```
const express = require('express');
const app = express();

const port = 3000;

app.get('/', (req, res) => res.send('Hello Express.js users'));

app.listen(port, () => console.log(`App is running at port: ${port}`));
```

FIGURE 14. An example of a server built using Express.

FIGURE 14 shown above is an example of a server built using Express. In the above figure, the server will run on the port “3000” with the hostname “localhost” and when the URL (Uniform Resource Locator) is http://localhost:3000/, the app will send a response “Hello Express.js users”. The app also logs a message “App is running at port: 3000” in command.

4.3 MLab

MLab is a cloud-based Database-as-a-Service for MongoDB. MLab provides features, such as Cloud Automation, Backup and Recovery, Monitoring and Analytics Tools, Advanced Security, Easy-to-use Data Browser, and Database Support. Facebook, Toyota, Verizon are some of the companies that use mLab. [22.]

mongodb://<username>:<password>@ds045618.mlab.com:45618/<dbname> is an example of the MongoDB URL to connect to the database in mLab.

4.4 Other Libraries (back-end)

To make the back-end development of the website pleasant, fast, and seamless, additional libraries were used by the author. Those libraries helped the author to make the website secure and easy to develop.

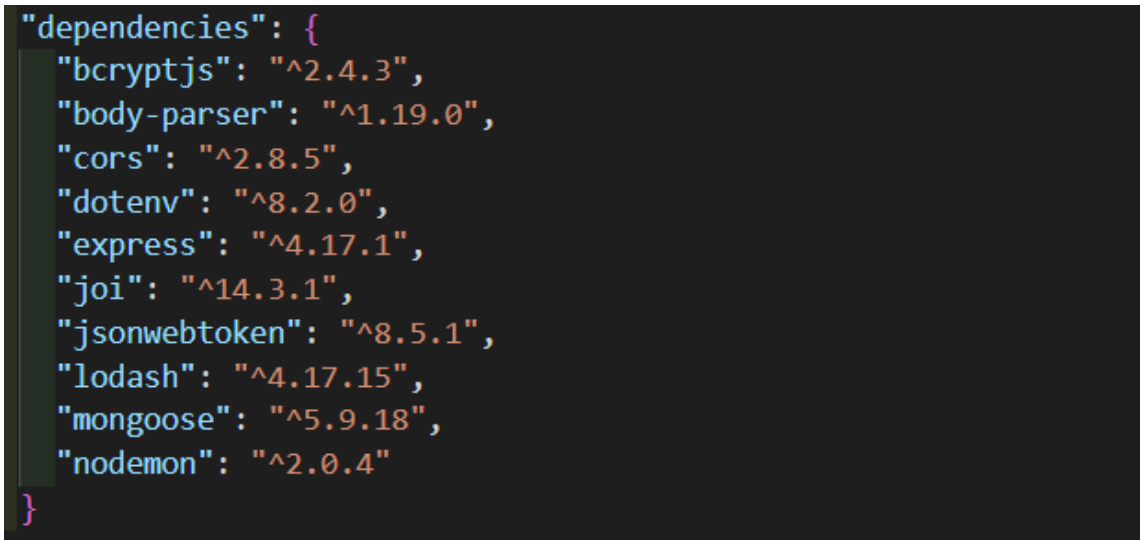

FIGURE 15. The version of the libraries installed in the back-end.

FIGURE 15 shown above is the list of libraries with their respective versions that were installed in the back-end during the development of the website. The use of the libraries is described below in TABLE 1.

TABLE 1. The use of libraries installed in the dependency of the project.

Libraries	**Use of the library**
bcryptjs	Hashing and Comparing the password.
body-parser	Validating and Parsing the incoming requests input.
cors	Enabling CORS (Cross-Origin Resource Sharing).
jsonwebtoken	Authenticating to the APIs.
lodash	Providing utility functions to simplify the tasks.
joi	Validating the data and describing the schema.

5 IMPLEMENTATION OF THE WEBSITE

5.1 Development Tools

Tools that are used to create, edit, maintain, support, and debug are known to be development tools. A Code editor, Terminal, Web Browser, a Code hosting platform, a Testing Tool, and a UI designer app are some of the most important development tools that are used to develop a website. These development tools were used by the author for maintaining the workflow of the project.

5.1.1 Code Editor

Prior to developing any application, selecting a good code editor can have a huge impact on the productivity of an application. Choosing the code editor that maintains a workflow, provides features that help save time while coding. Maintaining the quality of the code is a necessary factor in development.

Among many code editors, VS Code (Visual Studio Code) was selected for the development of the website. VS Code is very robust and quick. It has a built-in terminal and it supports Git Commit. Because of its Intellisense feature, it provides auto-completion of the code and suggestions throughout the known variable names and functional parameters. Prettier Code Formatter, Colour Info, Bracket Pair Colorizer, Auto Rename Tag are some of the add-ons or extensions that are provided by the VS Code and were used by the author during the development of the website.

5.1.2 Operating System, Browser and Terminal

The Microsoft Windows operating system, also known as Windows, was used to develop the website. Windows 10 and Nokia 8 Sirocco were used to preview and test the responsiveness of the website. Google Chrome, as well as Mozilla Firefox web browsers, were used to preview the website in both desktop and mobile devices. Additionally, the responsiveness of the website was inspected using DevTools of the browser. In Windows, the command prompt is same as the terminal. Command Prompt was used to install the required package man-

agers using NPM. The React project was initialized using a “create-react-app frontend” command, where the frontend is inside the root folder, named “tmdb-cine”, of the project. Similarly, the backend project was initialized using an “npm init” command inside the folder named “backend”.

5.1.3 Code hosting platform

GitHub was used as a source code hosting platform to store the codes of the project. GitHub keeps track of all the modifications made in the project. When we put our source codes in GitHub, it is publicaly available and can be used and evaluated by other programmers. It also has the option to keep our source codes private, in which case others will not be able to access those codes.

5.2 Project File Structure

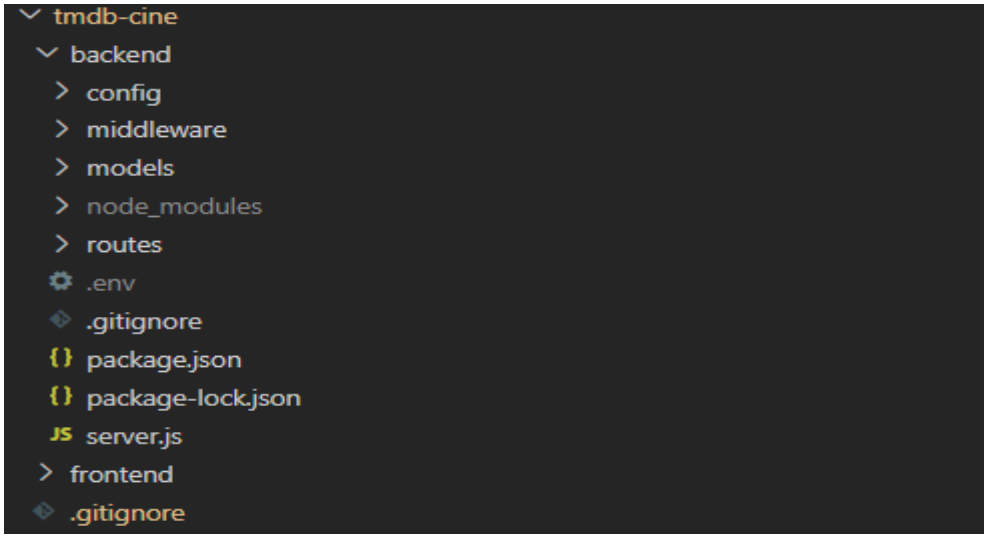

FIGURE 16. The project file structure for back-end.

FIGURE 16 shown above is the project structure created for the back-end development of the website.

In the “backend” project structure, the “routes” folder includes all the routing for login, register and media. The “models” folder includes the schema for the MongoDB. Similarly, the “middleware” folder includes the authentication file for authenticating the user by checking the unique token set in the header of an

API request. The “config” folder includes the database configuration file. The server.js file was used to start the server and connect with the database.

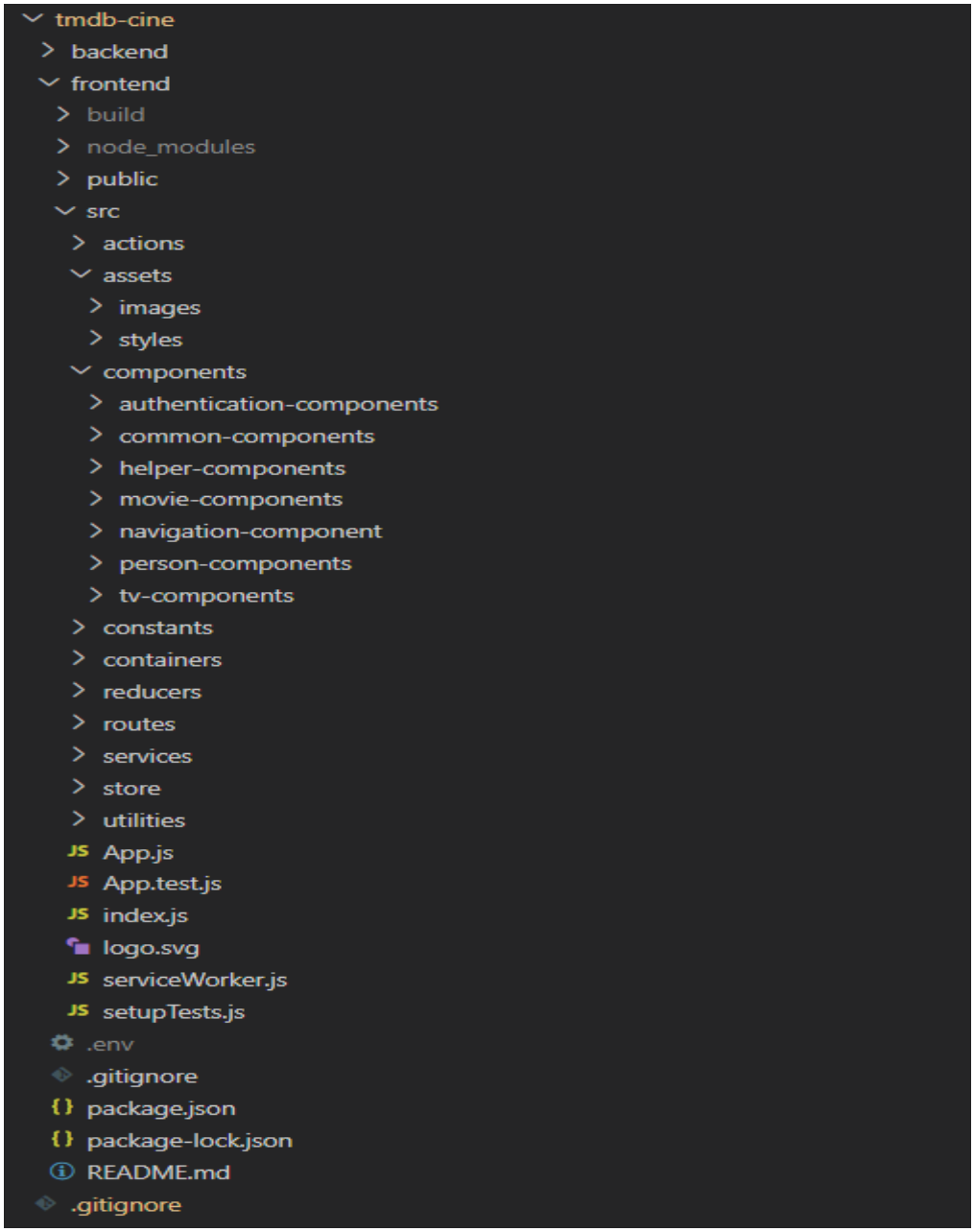

FIGURE 17. The project file structure for front-end.

FIGURE 17 shown above is the project structure created for the front-end development of the project.

In the “frontend” project structure, the entry point of an application is an index.html file, which is located inside the “public” folder. An index.js file tells the

application which component to render and where. Generally, the App.js file is rendered because it contains the other components. App.js is at the top of the component hierarchy. However, the App.js file can be renamed. Images and SCSS files are inside the “images” and “styles” folder of the “assets” folder. UI components are inside the “components” folder, which contains multiple sub-folders such as movies, TV, person, navigation, authentication, common, and helper components. The files that are used to retrieve data from the API, and to import the UI component to display that data, are kept inside the “container” folder. All the constant values, such as configuration, action types, navigation items, and other options are kept inside the “constants” folder. The reducer methods, action functions, and a store for Redux are inside the “reducers”, “actions”, and “store” folder simultaneously. The “routes” folder contains the routing of the website. The “services” folder contains the additional services that were required to make the development process fast and easy. Similarly, the “utilities” folder contains the functions that were used to make API calls.

Furthermore, both “frontend” and “backend” folders are contained inside the “tmdb-cine” folder.

5.3 UI of the Website

5.3.1 Navigation Bar, Side-Navigation Bar and Search Bar

The navigation bar and side-navigation bar both allow the user to navigate through the different pages of the website without having to go back to the previous page or home page. Both the navigation bar and side-navigation bar provide direct access to Home, Movies, TV Shows, Browse by genre, and Log-in/Register or Dashboard page of the website. However, the side-navigation bar has the option that turns on and turns off the dark-theme of the website. And, the navigation bar has the search button that opens up the search bar in place of the navigation bar. Additionally, the navigation bar has a menu button that opens the side-navigation bar.

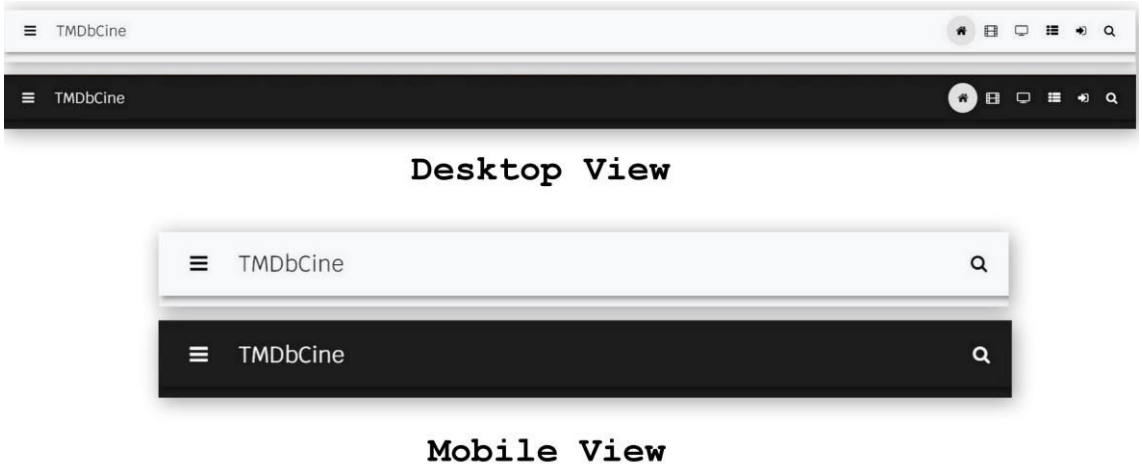

FIGURE 18. The navigation bar for desktop view and mobile view.

FIGURE 18 shown above is an image of the navigation bar in both light-theme and dark-theme that is displayed when the user is on a desktop or mobile device, respectively. When a user is logged-in, the login icon will change into a dashboard icon, which makes it possible for the user to navigate to the dashboard page of the website.

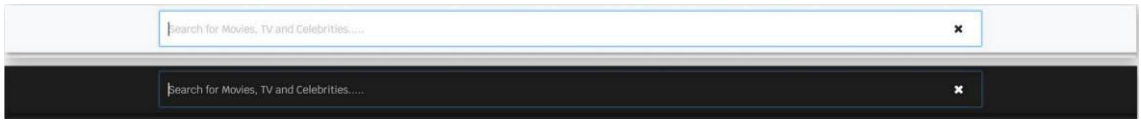

FIGURE 19. The search bar.

FIGURE 19 shown above is an image of the search bar in both light-theme and dark-theme for all the devices.

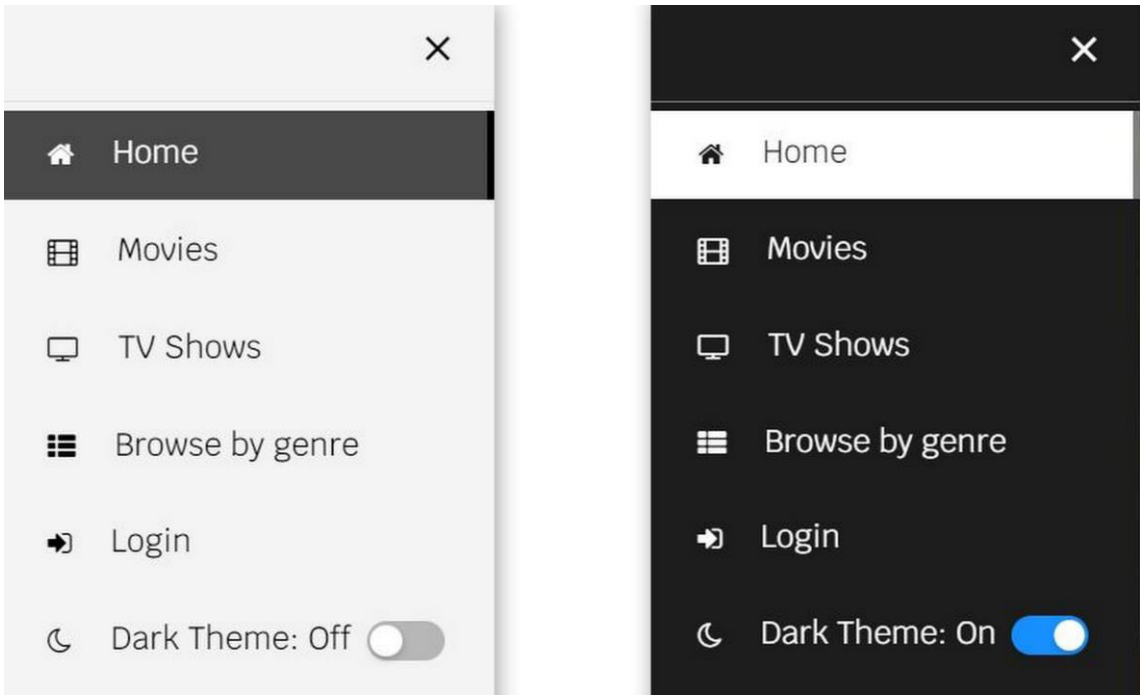

FIGURE 20. The side-navigation bar.

FIGURE 20 shown above is an image of the side-navigation bar in both light-theme and dark-theme, respectively. When a user is logged-in, the Login button will change into the Dashboard button.

5.3.2 Search Suggestion, Search Results Page and Footer

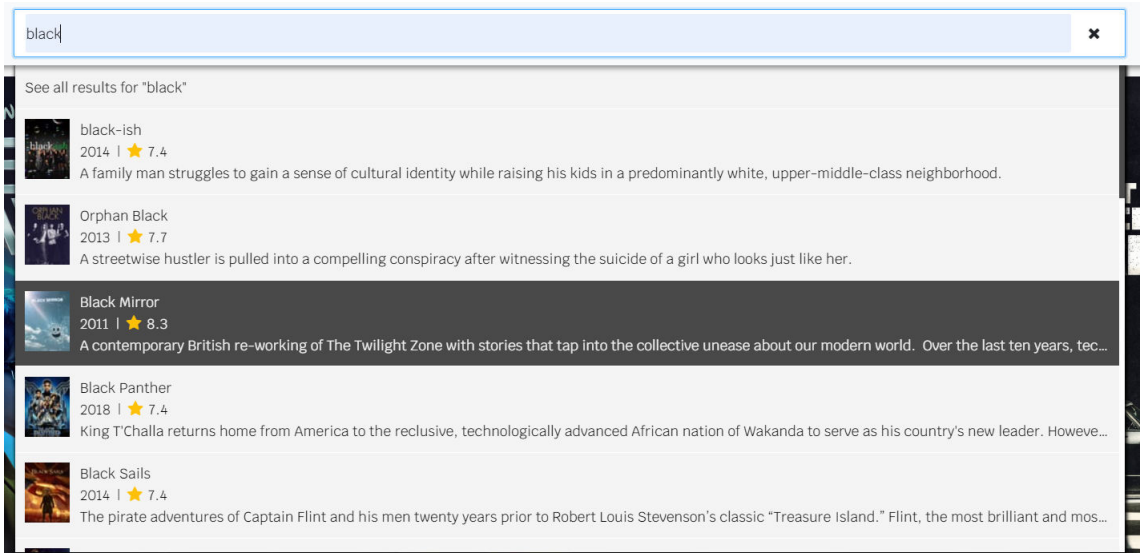

FIGURE 21. Search suggestion box.

FIGURE 21 shown above shows the suggestions that are generated when the user starts typing the text on the search bar. For instance, the text “black” was typed and, movies, TV shows, and celebrities that has “black” text on their name are shown in the suggestion box with the limit result of twenty suggestions.

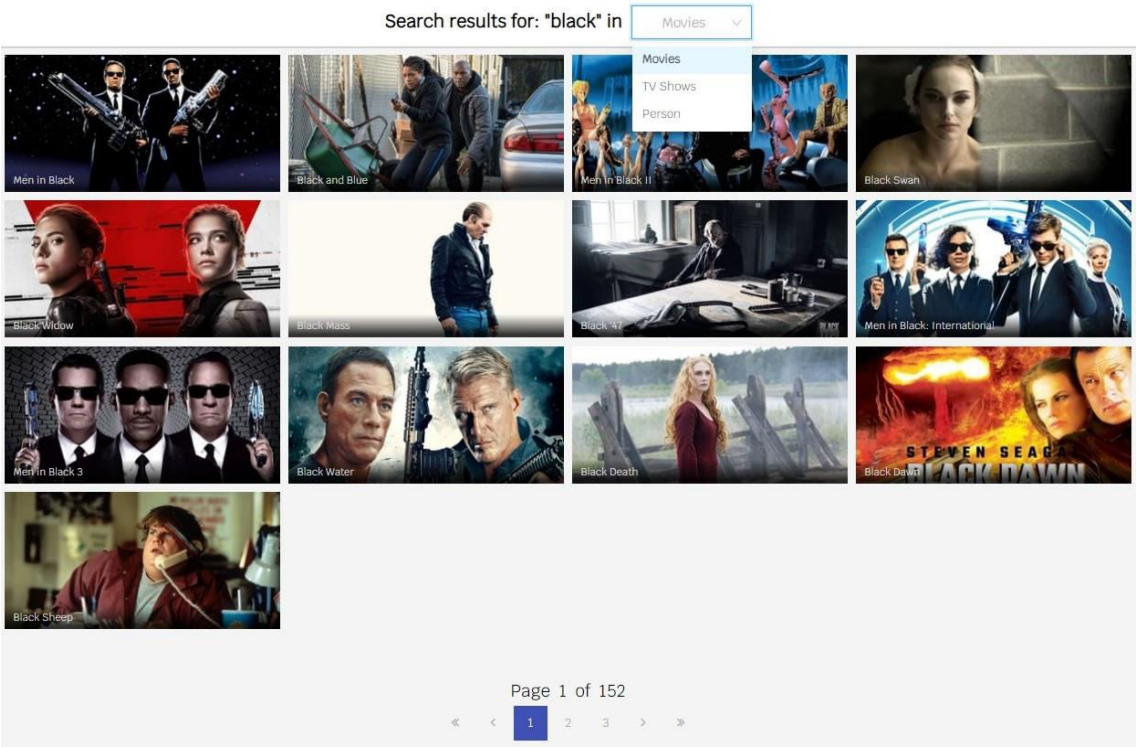

FIGURE 22. Search results page.

FIGURE 22 shown above is the search results page of the website. Users are directed to that page when they type and hit enter on the search bar. In the above figure, the search results related to movies for the text “black” are shown. Users can select between movies, TV shows and celebrities to show the related results. It has the pagination at the bottom of the page, before the footer.

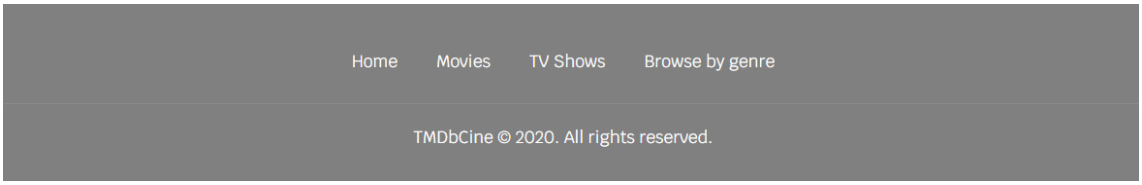

FIGURE 23. The footer of the website.

FIGURE 23 shown above is an image of the footer of the website that includes the links to go to the other pages of the website. It also includes the website name with the current year and copyright logo.

5.3.3 Home Page

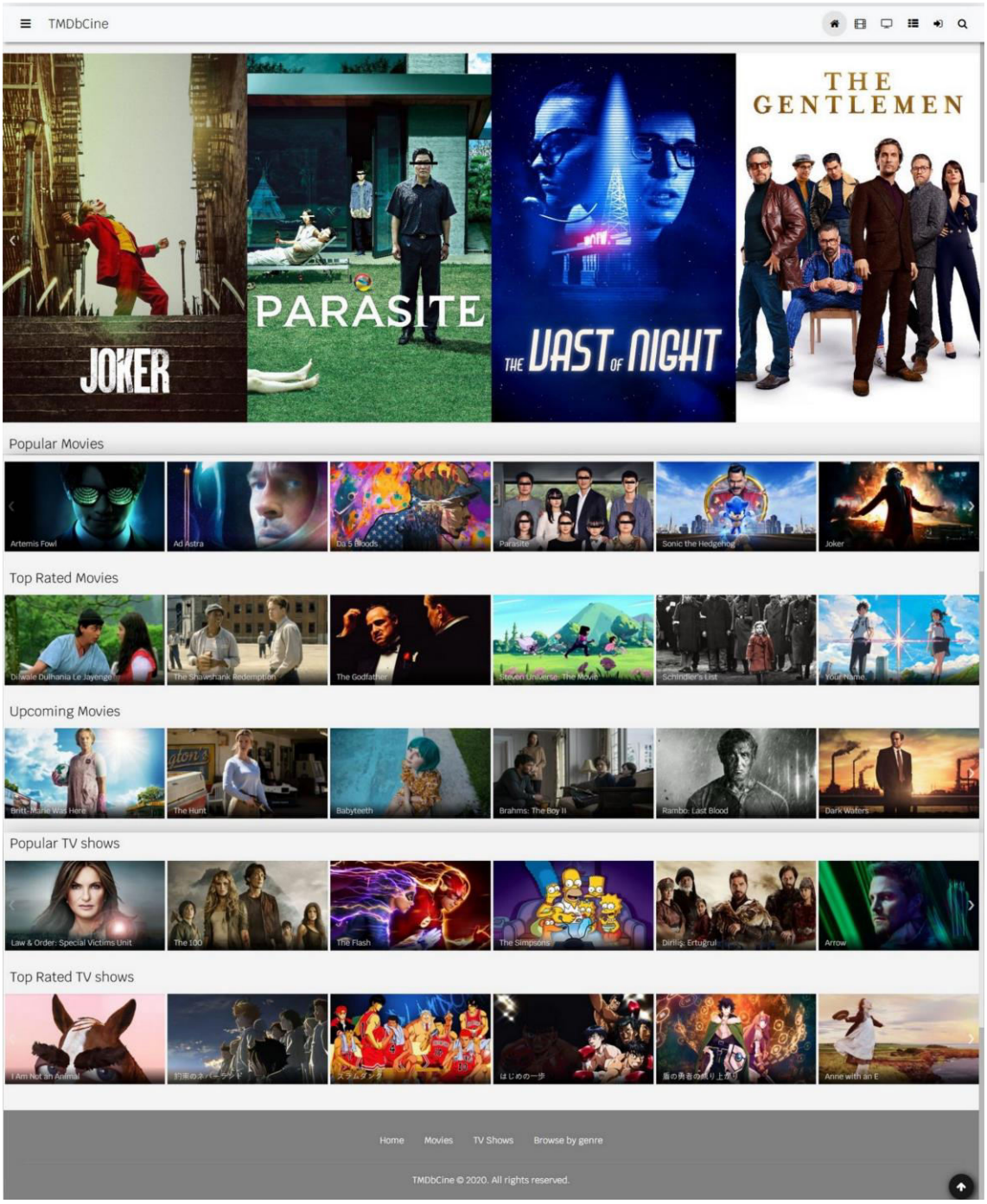

FIGURE 24. Home page of the website.

FIGURE 24 shown above is an image of the home page of the website. The home page is also known as the main page, front page, or landing page. It is the first page of the website that the users get directed to when they enter the URL of the website. The home page renders the trending, popular, top-rated, and upcoming movies along with popular and top-rated TV shows.

5.3.4 Movies and TV Shows Page

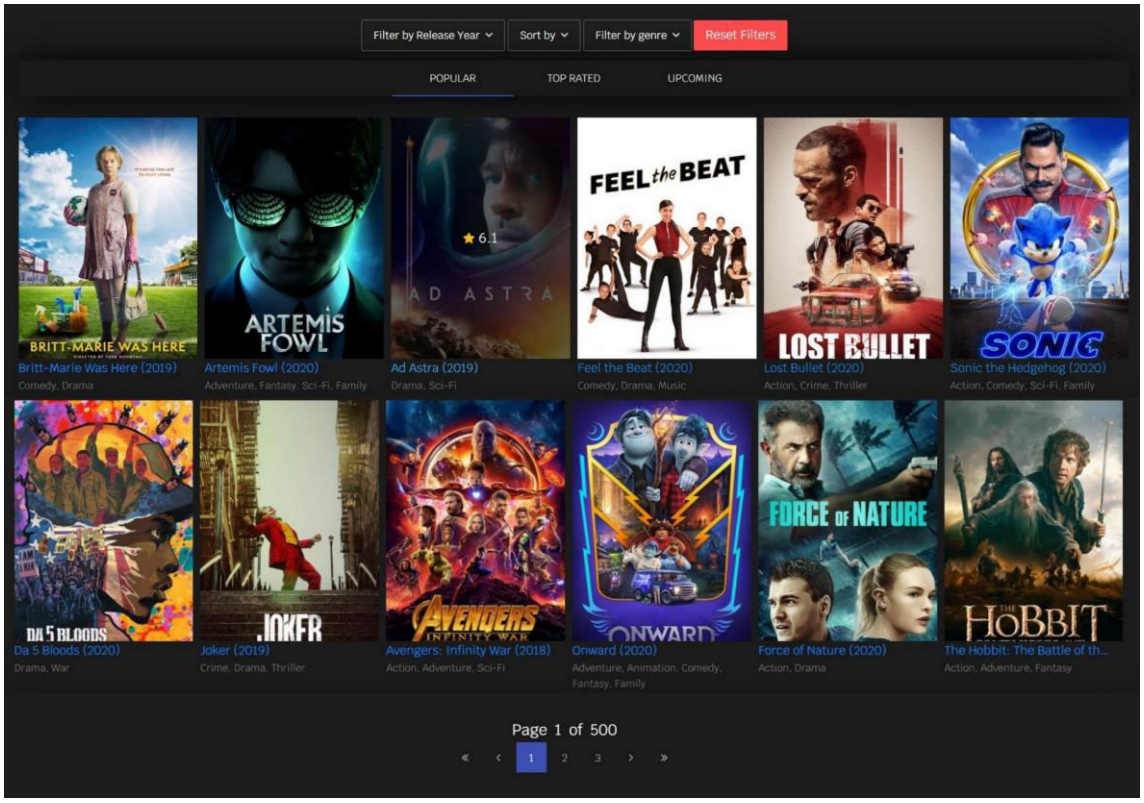

FIGURE 25. Movies page of the website.

FIGURE 25 shown above is an image of the movies page of the website. The page has the filter options that consist of “filter by release year”, “sort by”, and “filter by genre”. In the context of filtering, it has the reset filters button that will clear all the filter options that were applied. The filter by release year option has the list of years that can be pressed or clicked to get the movies that were released in the year that the user has selected in the filter. Similarly, the filter by genre option consists of the genres of the movies. The sort by option can sort the movies accordingly based on the popularity, release dates, and ratings of the movie. Just under the filter options, there is a tab box that consists of cate-

gories of the movies, such as Popular, Top Rated, and Upcoming. All the options that are on the movies page are also on the TV shows page. Apart from the categories tab, the movies page and the TV shows page are identical. The TV shows page does not have an upcoming category. Furthermore, both the movies page as well as the TV shows page have a pagination system that helps to jump from one-page number to another page number on the same page of the website. Additionally, twenty movies and TV shows are displayed per a page number.

When the user clicks or presses one of the movies, it redirects them to the page where all the necessary details about the movie are displayed. The link to this page is not listed in the navigation bar or the side-navigation bar. As shown in FIGURE 26 below, the page contains the poster on the left and the title, tagline, genre lists, average rating, released date, the runtime of the movie on the right side of the page. Additionally, the right side also contains the release status of the movie. For instance, it shows if the movie is released or is under production. It also contains the storyline, the languages that the movie was released on, their budget and revenue, the homepage of the movie, the add to a favorite option, and the keywords of the movie. After that, scrolling down the page shows the cast and crew members of the movie with their original name and character name in the movie. This page also shows the posters and background posters along with videos and trailers of the movie. Lastly, the page shows similar movies and also recommends other movies based on the movie that the user is visiting.

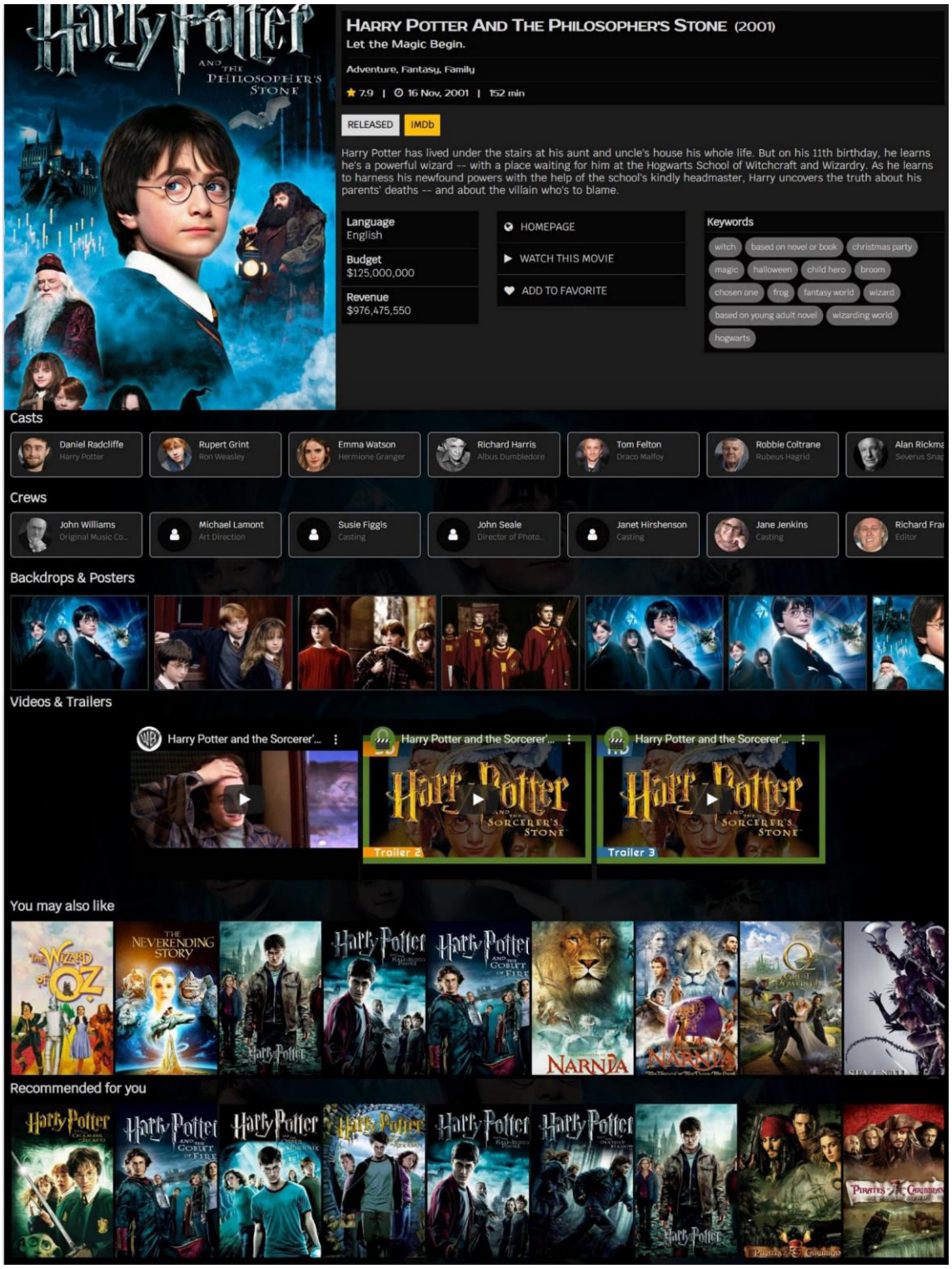

FIGURE 26. Single movie page of the website.

FIGURE 26 shown above is an image of the page that opens when the user clicks or presses on one of the listed movies. In this case, the movie “Harry Potter and the Philosopher’s stone” was clicked.

FIGURE 27. Single TV show page of the website.

FIGURE 27 shown above is an image of the page that opens when the user clicks or presses on one of the listed TV shows. In this case, the TV show “The Flash” was clicked. The UI of the single TV show page is identical to a single movie page except some details change. For instance, instead of the tagline, it shows the total number of episodes and seasons of the TV show. And, the lists of TV show seasons are displayed as shown in the above FIGURE 27.

FIGURE 28 shown below is an image of the single season page of a TV show of the website. It contains the details about the season of a TV show. In the details, it shows the number of episodes that the season has and the title of each episode. When scrolled down the page, it shows the poster, storyline, average ratings, released date, crew members, and the guest stars of the episode. Additionally, the page also has the “Next Season” button to go to the next season of the TV show and the “Previous Season” button to go to the previous season of the TV show.

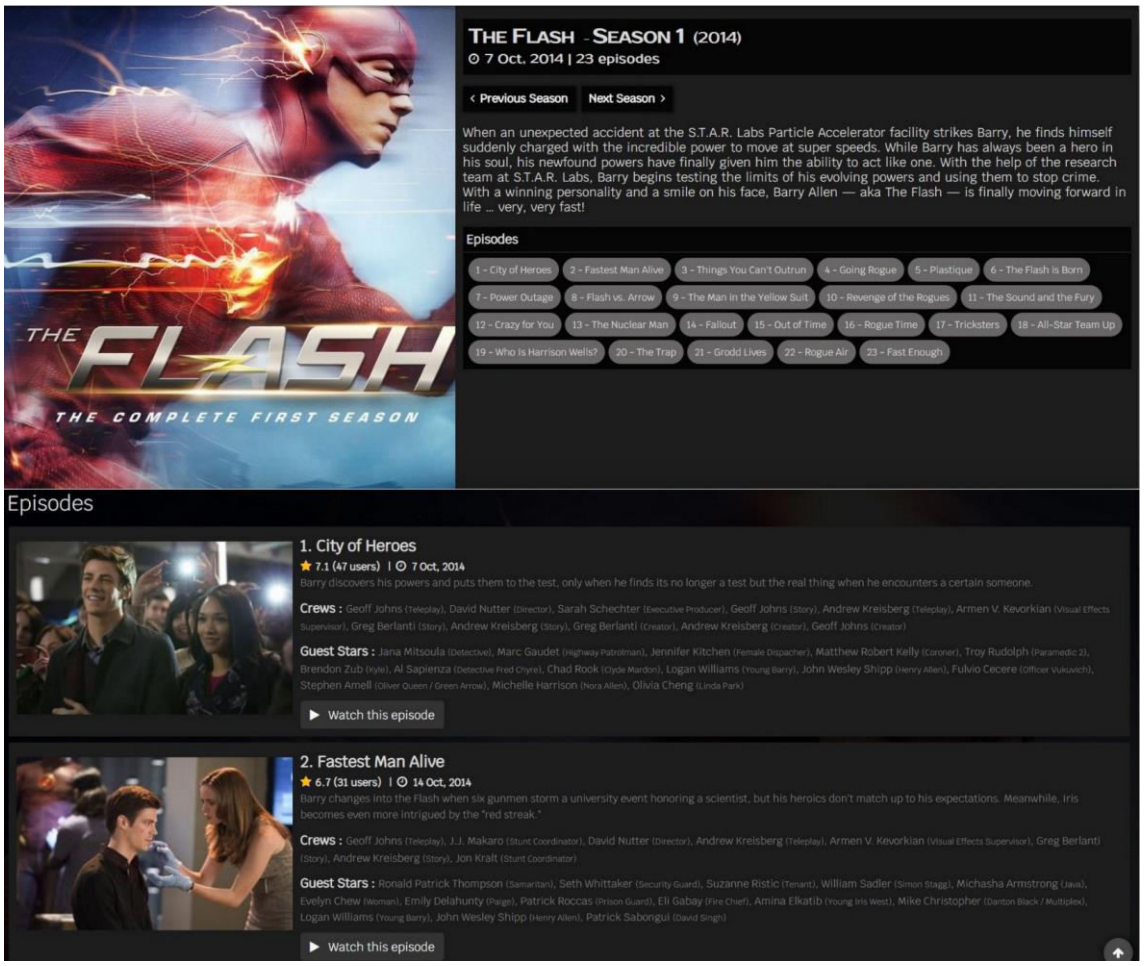

FIGURE 28. Single season page of a TV show of the website.

5.3.5 Celebrity Page

FIGURE 29 shown below is an image of the UI of the celebrity page of the website. The website shows the details of a celebrity that was opened when the user clicked or pressed the cast or the crew member list from the movie or the TV show page. The link to this page is not provided in any of the navigational bar. On this page, the left side of the page shows the profile picture of the celebrity. And, the right side contains all the other details, such as the name at the top. After that, it shows the main profession, their nickname or another name, birthdate, birthplace, link to their social media accounts, their biography, list of their known movies or TV shows, additional photos of the celebrity. Additionally, it shows the list of all the movies and TV shows they have worked in. The lists show the name of the movies and TV shows with the year of their release date

and the character name of that particular celebrity (“Danielle Panabaker” in this case). It has filter options from which the user can filter with a cast or crew option as well as a movie or TV show option.

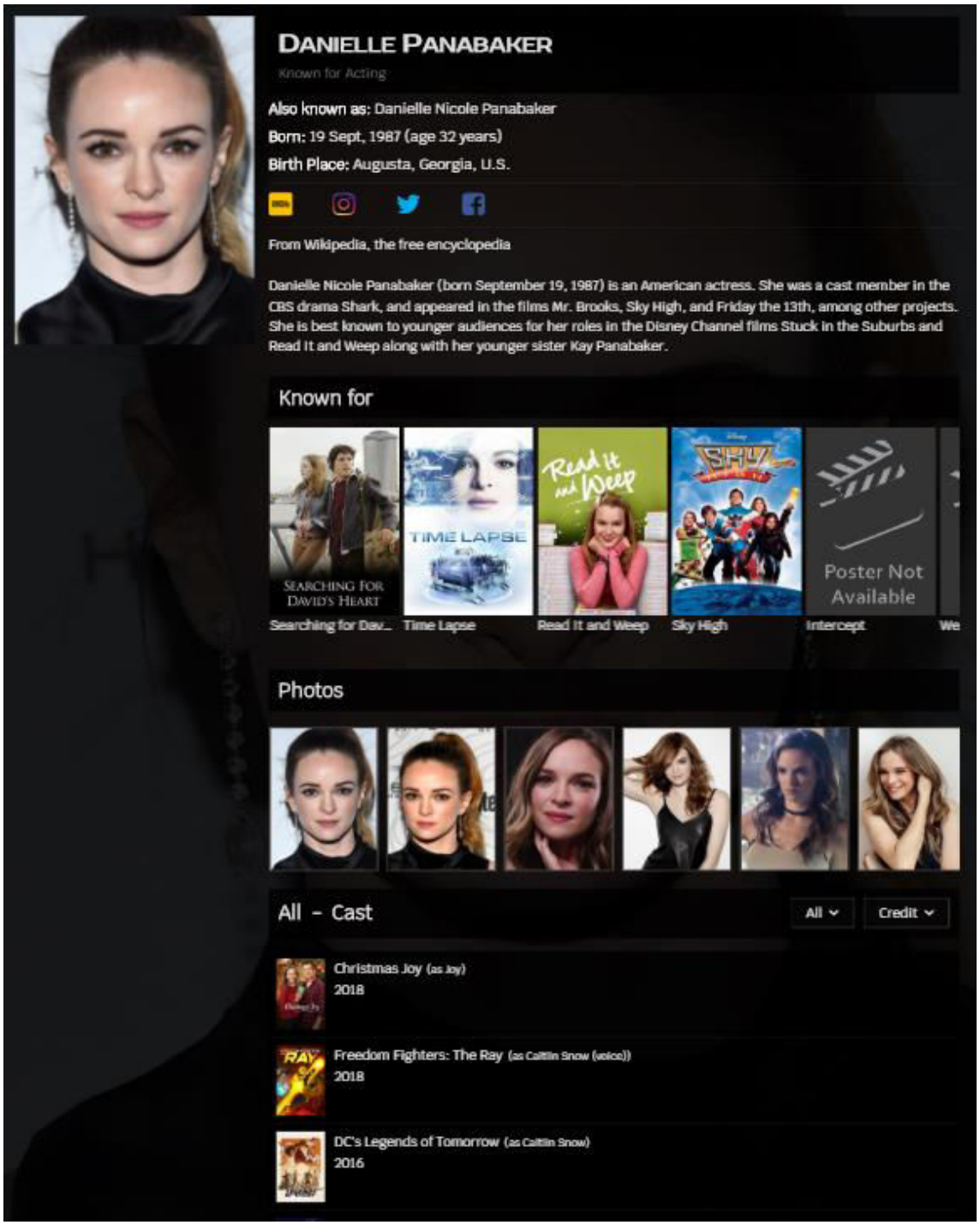

FIGURE 29. Celebrity page of the website.

5.3.6 Genre Lists Page

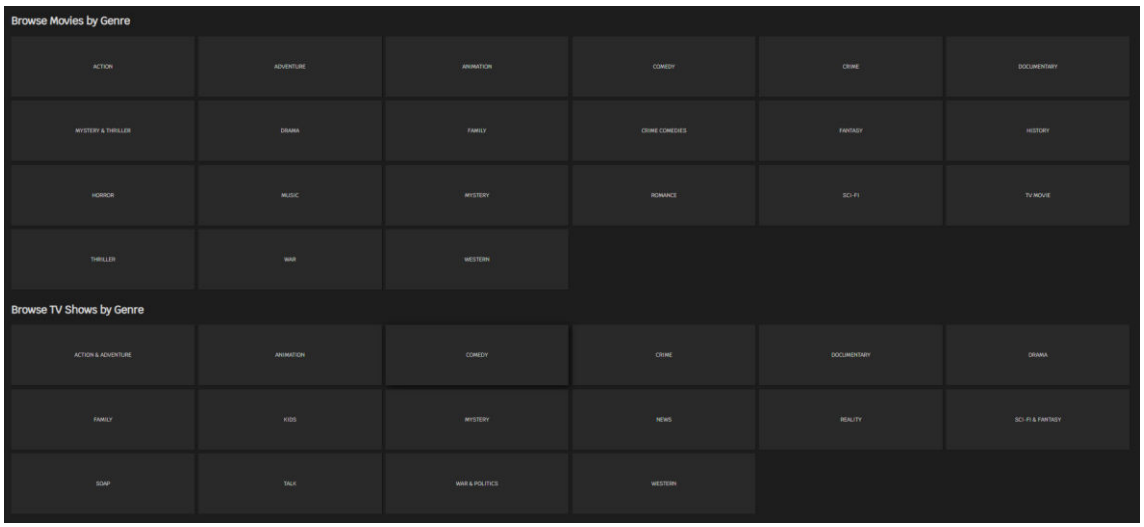

FIGURE 30. Genre Lists page of the website.

FIGURE 30 shown above is an image of an UI of a page where genres for both movies and TV shows are listed. The link to redirect to this page is provided in the navigation bar, side-navigation bar, and footer of the website. When clicked or pressed on a genre, it will redirect the page to the respective movies or TV shows page with the ID of the genre in the URL.

https://tmdbcine.netlify.app/movies?with_genre_id=16 is an example of the redirected URL, when the user clicked the animation genre of the movie on the genre lists page.

5.3.7 Login, Register and Dashboard Page

The user needs to register and then log in to the website to mark their favorite movies and TV shows. So, to do that register and login pages are used. The link to this page is provided in all navigations. The UI of the login and the register page is built in a way that it can detect if the input box is filled or not. In the beginning, both the login and the register button in their respective pages is disabled and is not clickable and once the user fills the input boxes, the button is enabled and will be clickable. Both the login and the register page have “username” and “password” input boxes. However, the register page contains two additional input boxes. They are “firstname” and “lastname”. Users can go

to the register page from the login page and vice versa with the link “Register here” and “Login here”, respectively.

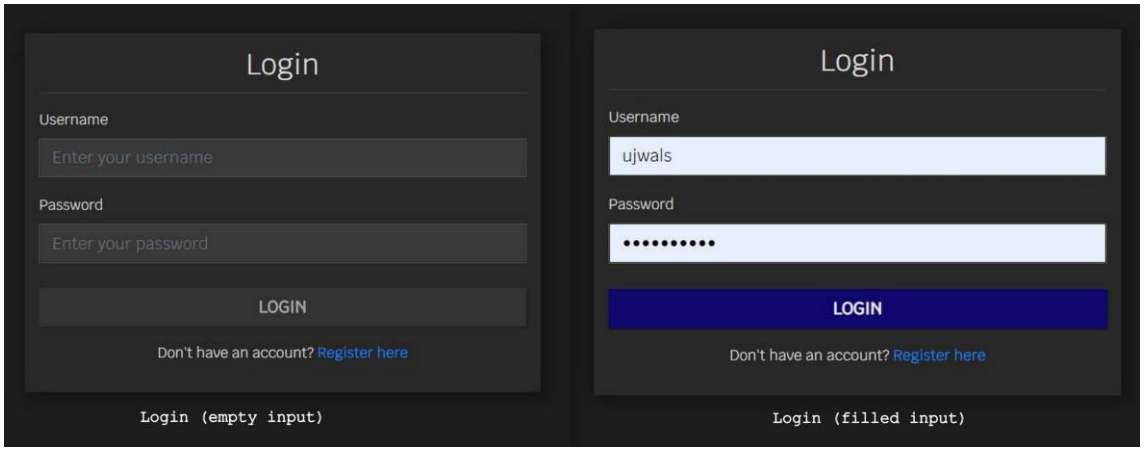

FIGURE 31. Login page.

FIGURE 31 is an image of login page, with empty input boxes on the left and filled input boxes on the right of the image, of the website.

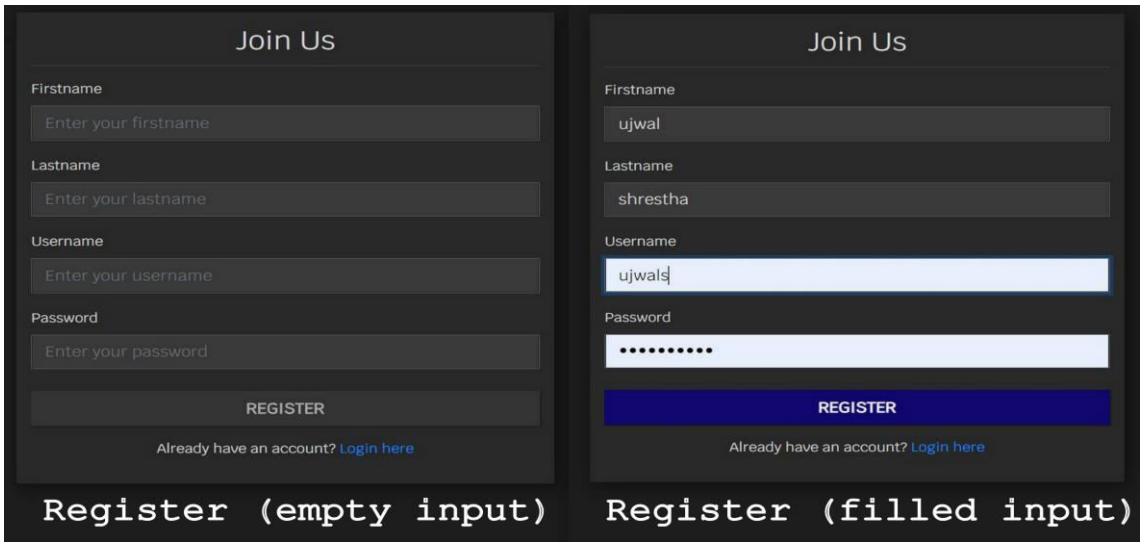

FIGURE 32. Register page.

FIGURE 32 is an image of register page, with empty input boxes on the left and filled input boxes on the right of the image, of the website.

Once the user is logged in, they can visit the dashboard page. The page contains the lists of their favorite movies as well as TV shows. If there is no favorite

movie or TV shows listed, then, it shows a message to inform the user. This page cannot be accessed if the user is not logged in. At the top of the page, it shows welcome message and logout button at the bottom of the message. On the bottom-left corner of an image of movie or TV show, there is a remove button with trash bin icon. When the user clicks or presses that button, the respective movie or TV show is removed from the favorite list of the user.

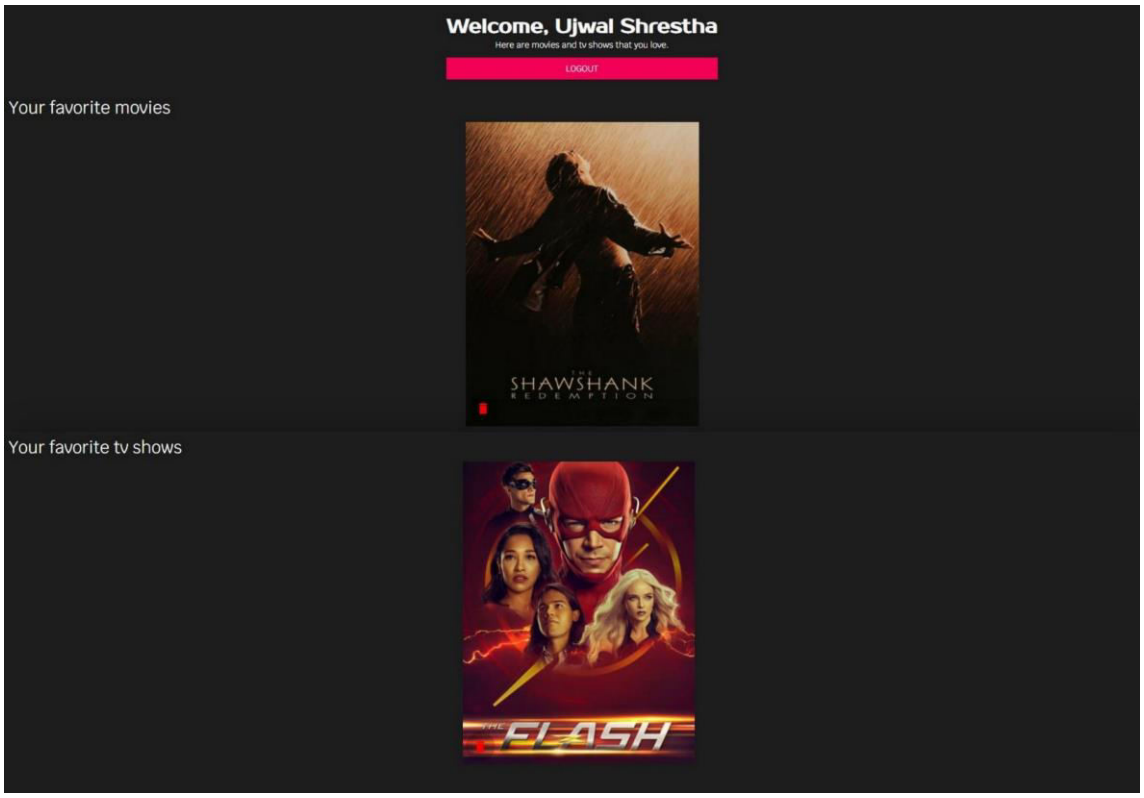

FIGURE 33. Dashboard page of a user of the website.

FIGURE 33 shown above is an image of a dashboard page of a user (“Ujwal Shrestha” in this case) of the website.

6 CONCLUSION

The main aim of the thesis was to learn about API and implementing it with ReactJS along with Redux state management and developing API using back-end services, such as Node.js, Express, and mLab. Apart from the difficulties faced during the learning and developing phase, the main aim of this thesis project was achieved. The implementation process, development of UI, and project structures are described in Chapter 5.

During the development, several other small React components and libraries were used which were new to the author. The author learned to develop and implement APIs in ReactJS and to manage states with the Redux state management tool. Additionally, the author learned to use back-end services and Axios to make a call to back-end and get a response. Reducer, Action, and Store are three principles that were followed while using Redux. Consequently, managing and interacting with the state tree were made easy and efficient. Furthermore, functional components of ReactJS were used for the UI part, and class components of ReactJS were used for calling the action and retrieving data from the Redux state into the application. After the development, the author concluded that the use of the state management system, such as Redux in small scale projects is not very useful.

While developing, the author also gained knowledge about other libraries, such as react-slick, dotenv, node-sass, react-color-extractor, react-js-pagination, mongoose, nodemon, which were used in the development of the website. Implementation of search suggestions, dark-theme, and user authentication system was new and kind of an obstacle to the author. However, he overcame those obstacles and implemented the features on the website.

The website was tested for its smooth and efficient run. According to the users, the UI of the website was responsive, user-friendly, interactive, and informative. The users also said that the website was attractive especially when the dark-theme of the website was turned on. The technologies, the concepts and the

implementation processes that were explained in this thesis report can be valuable for others to develop similar projects.

The implementation of the project was successful. The results of the project are as same as the objectives. The author did what he was set out to do. However, there is still space for some improvements, and new features, such as multiple language systems, can be added in the future.

Overall, the development of the website and the writing of this thesis was a great learning experience for the author.

REFERENCES

1 Faraz, K. 2019. Axios or fetch (): Which should you use? Date of retrieval 15 June 2020
https://blog.logrocket.com/axios-or-fetch-api/

2 SASS. Syntax. Date of retrieval 15 June 2020
https://sass-lang.com/documentation/syntax

3 React 2019. Getting Started. Date of retrieval 15 June 2020
https://reactjs.org/docs/getting-started.html

4 React 2019. React: A JavaScript Library for building user interfaces. Date of retrieval 15 June 2020
https://reactjs.org/

5 Laurie, V. 2018. The State of JavaScript Frameworks, 2017. Date of retrieval
https://www.npmjs.com/npm/state-of-javascript-frameworks-2017-part-1

6 Alex 2016. Redux - An Introduction. Date of retrieval 15 June 2020
https://www.smashingmagazine.com/2016/06/an-introduction-to-redux/

7 Redux. Three Principles. Date of retrieval 15 June 2020
https://redux.js.org/introduction/three-principles

8 Rajaraodv 2016. Middlewares and React Redux Life Cycle. Date of retrieval 15 June 2020
https://medium.com/@rajaraodv/using-middlewares-in-react-redux-apps-f7c9652610c6

9 React Redux. Quick Start. Date of retrieval 15 June 2020
https://react-redux.js.org/introduction/quick-start

10 NPM. Redux-Thunk. Date of retrieval 18 June 2020
https://www.npmjs.com/package/redux-thunk

11 Sebastian Eschweiler 2017. Getting Started with Axios. Date of retrieval 15 June 2020
https://medium.com/codingthesmartway-com-blog/getting-started-with-axios-166cb0035237

12 Code Realm 2018. Meet Material-UI – your new favorite user interface library. Date of retrieval 15 June 2020
https://www.freecodecamp.org/news/meet-your-material-ui-your-new-favorite-user-interface-library-6349a1c88a8c/

13 Alex Taylor. Crafting Beautiful UIs in React using Ant Design. Date of retrieval 15 June 2020
https://alligator.io/react/beautiful-uis-ant-design/

14 Eising Perry 2017. What exactly is an API? Date of retrieval 15 June 2020
https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f

15 The Movie Database API. Getting Started. Date of retrieval 1 July 2020
https://developers.themoviedb.org/3/getting-started/introduction

16 The Movie Database API. Search. Date of retrieval 1 July 2020
https://developers.themoviedb.org/3/search/multi-search

17 The Movie Database API. Discover. Date of retrieval 2 July 2020
https://developers.themoviedb.org/3/discover/movie-discover

18 The Movie Database API. Movies. Date of retrieval 2 July 2020
https://developers.themoviedb.org/3/movies/get-movie-details

19 YouTeam Editorial Team 2019. Top Companies that Used Node.js in Production. Date of retrieval 15 June 2020
https://youteam.io/blog/top-companies-that-used-node-js-in-production/

20 Node.js. About Node.js. Date of retrieval 15 June 2020
https://nodejs.org/en/about/

21 Tutorialspoint. ExpressJS – Overview. Date of retrieval 16 June 2020
https://www.tutorialspoint.com/expressjs/expressjs_overview.htm

22 mLab. mLab - home. Date of retrieval 16 June 2020
https://mlab.com/