1. INTRODUCTION

1.1 PROJECT INTRODUCTION

The "Movie Recommendation Engine" is a recommendation system. It is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences. There are a wide variety of applications for recommendation systems. These have become increasingly popular over the last few years and are now utilized in most online platforms that we use. The content of such platforms varies from movies, music, books and videos, to friends and stories on social media platforms, to products on e-commerce websites, to people on professional and dating websites, to search results returned on Google. Often, these systems are able to collect information about a user's choices, and can use this information to improve their suggestions in the future.

1.2 PURPOSE OF THE PROJECT

The purpose of this project is to create an application by the help of computerized equipment and full-fledged computer software, fulfilling their requirements so that they get valuable data/information about the movies and series which a they users wishes to watch. The required software and hardware are easily available and easy to work with.

The project, as described above, can lead to error free, secure, reliable and fast management system. It can assist the user to concentrate on their other activities rather to concentrate on the record keeping. That means that the user need not be distracted by the information that is not relevant, while being able to reach the information.

2 LITERATURE SURVEY

2.1 EXISTING SYSTEM

The existing system for watching movies online includes a single website where the user can watch movies or series which are present on that particular website or application. With the advancement of the internet, watching movies on the internet is very convenient, easy and fun thing to do in free time, which makes watching movies online a common thing now a days.

2.1.1 PROPOSED SYSTEM

Movie Recommendation Engine is developed to provide an effective means for the users to get recommendations for the movies which he/she wishes to watch. In addition, Users can get recommendations based on the genres selected by the user. Movie Recommendation Engine is a web-based application providing the flexibility for the users.

2.2 FEASIBILITY STUDY

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential. Three key considerations involved in the feasibility analysis are:

2.2.1 Economic Feasibility

This study is carried out to check the economic impact will have on the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system was within the budget and this was achieved because most of the technologies used are freely available. Only the customized products have to be purchased.

2.2.2 Technical Feasibility

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes for the implementing this system.

2.2.3 Operational Feasibility

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

2.3 TOOLS AND TECHNOLOGIES

2.3.1 React: Front-End Library

React is a JavaScript library that is used for building user interfaces. React is used for the development of single-page applications and mobile applications because of its ability to handle rapidly changing data. React allows users to code in JavaScript and create UI components. Why use React?

- Virtual DOM – A virtual DOM object is a representation of a DOM object. Virtual DOM is actually a copy of the original DOM. Any modification in the web application causes the entire UI to re-render the virtual DOM. Then the difference between the original DOM and this virtual DOM is compared and the changes are made accordingly to the original DOM.
- JSX – Stands for JavaScript XML. It is an HTML/XML JavaScript Extension which is used in React. Makes it easier and simpler to write React components.
- Components – ReactJS supports Components. Components are the building blocks of UI wherein each component has a logic and contributes to the overall UI. These components also promote code reusability and make the overall web application easier to understand.
- High Performance – Features like Virtual DOM, JSX and Components makes it much faster than the rest of the frameworks out there.
- Developing Android/iOS Apps – With React Native you can easily code Android-based or IOS-Based apps with just the knowledge of JavaScript and ReactJS.

2.3.2 API: Application Programming Interface

APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols. For example, the weather bureau's software system contains daily weather data. The weather app on your phone "talks" to this system via APIs and shows you daily weather updates on your phone.

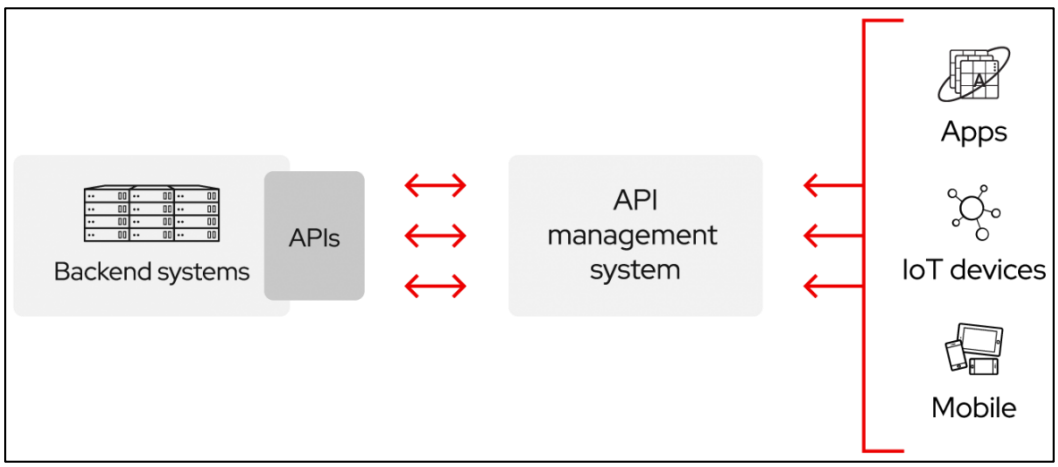
How do APIs work?

APIs let your product or service communicate with other products and services without having to know how they're implemented. This can simplify app development, saving time and money. When you're designing new tools and products—or managing existing ones—APIs give you flexibility; simplify design, administration, and use; and provide opportunities for innovation.

APIs are sometimes thought of as contracts, with documentation that represents an agreement between parties: If party 1 sends a remote request structured a particular way, this is how party 2's software will respond.

Because APIs simplify how developers integrate new application components into an existing architecture, they help business and IT teams collaborate. Business needs often change quickly in response to ever shifting digital markets, where new competitors can change a whole industry with a new app. In order to stay competitive, it's important to support the rapid development and deployment of innovative services. Cloud-native application development is an identifiable way to increase development speed, and it relies on connecting a microservices application architecture through APIs.

APIs are a simplified way to connect your own infrastructure through cloud-native app development, but they also allow you to share your data with customers and other external users. Public APIs represent unique business value because they can simplify and expand how you connect with your partners, as well as potentially monetize your data (the Google Maps API is a popular example).



3 SOFTWARE AND HARDWARE REQUIREMENTS

3.1 INTRODUCTION

To be used efficiently, all computer software needs certain hardware components or the other software resources to be present on a computer. These pre-requisites are known as(computer) system requirements and are often used as a guideline as opposed to an absolute rule. Most software defines two sets of system requirements: minimum and recommended. With increasing demand for higher processing power and resources in newer versions of software, system requirements tend to increase over time. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

3.2 SYSTEM REQUIREMENTS

3.2.1 HARDWARE REQUIREMENTS

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware. A hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatibility and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Hardware Requirements for Present Project:

PROCESSOR: Intel i3 or higher

RAM: 4 GB

HARD DISK: 8 GB

3.2.2 SOFTWARE REQUIREMENTS

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

SOFTWARE REQUIREMENTS FOR PRESENT PROJECT:

OPERATING SYSTEM: Windows 8 or higher / Linux (Ubuntu 20.04 LTS)

FRONT-END: ReactJS, Material-UI (Framework)

3.3 MODULES

This project has the following modules: -

3.3.1 TRENDING MANAGEMENT MODULE

- The user can see the latest movies which are trending recently
- The user can see the details of the content selected like the title, description, cast etc.

3.3.2 GENRE MANAGEMENT MODULE

- There are multiple genres from which the user can select the genre of his/her choice.
- The categories are displayed in a lexicographical order.
- The movies/series can be filtered out according to the categories selected on the page.

3.3.3 MOVIE MANAGEMENT MODULE

- This module handles the movies displayed on the website.
- In this module multiple movies are displayed.
- The user can select and filter the movies according to the genre of their choice.

3.3.4 SERIES MANAGEMENT MODULE

- This module handles the series displayed on the website.
- In this module multiple series are displayed.
- The user can select and filter the series according to the genre of their choice.

3.3.5 SEARCH MODULE

- This module handles the searches of the user.
- In this module, the content is displayed on the page based on the keyword given by the user.
- There are two sections in this module, one for the movies and the other for the series.

4 SYSTEM DESIGN

4.1 INTRODUCTION TO UML

UML DESIGN

The Unified Modeling Language (UML) is a standard language for specifying, visualizing, constructing, and documenting the software system and its components. It is a graphical language, which provides a vocabulary and set of semantics and rules. The UML focuses on the conceptual and physical representation of the system. It captures the decisions and understandings about systems that must be constructed. It is used to understand, design, configure, maintain, and control information about the systems.

The UML is a language for:

• Visualizing

• Specifying

• Constructing

• Documenting

Visualizing

Through UML we see or visualize an existing system and ultimately, we visualize how the system is going to be after implementation. Unless we think, we cannot implement. UML helps to visualize, how the components of the system communicate and interact with each other.

Specifying

Specifying means building, models that are precise, unambiguous and complete UML addresses the specification of all the important analysis design, implementation decisions that must be made in developing and deploying a software system.

Constructing

UML models can be directly connected to a variety of programming language through mapping a model from UML to a programming language like JAVA or C++ or VB. Forward Engineering and Reverse Engineering is possible through UML.

Documenting

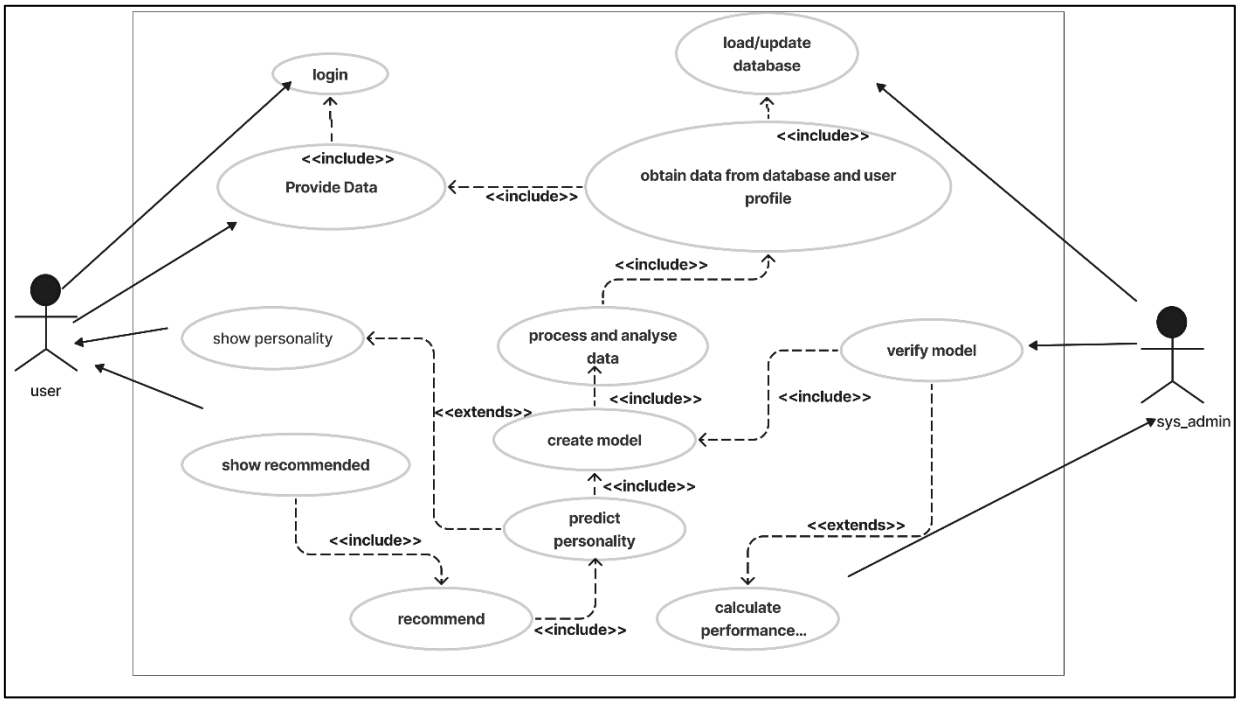
The Deliverables of a project apart from coding are some Artifacts, which are critical in controlling, measuring and communicating about a system during its developing requirements, architecture, desire, source code, project plans, tests, prototypes releasers, etc.

4.1.1 USE CASE DIAGRAM

A use case diagram in the Unified Modeling Language (UML) is type of behavioral diagram defined by and created from a use-case analysis. its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases.

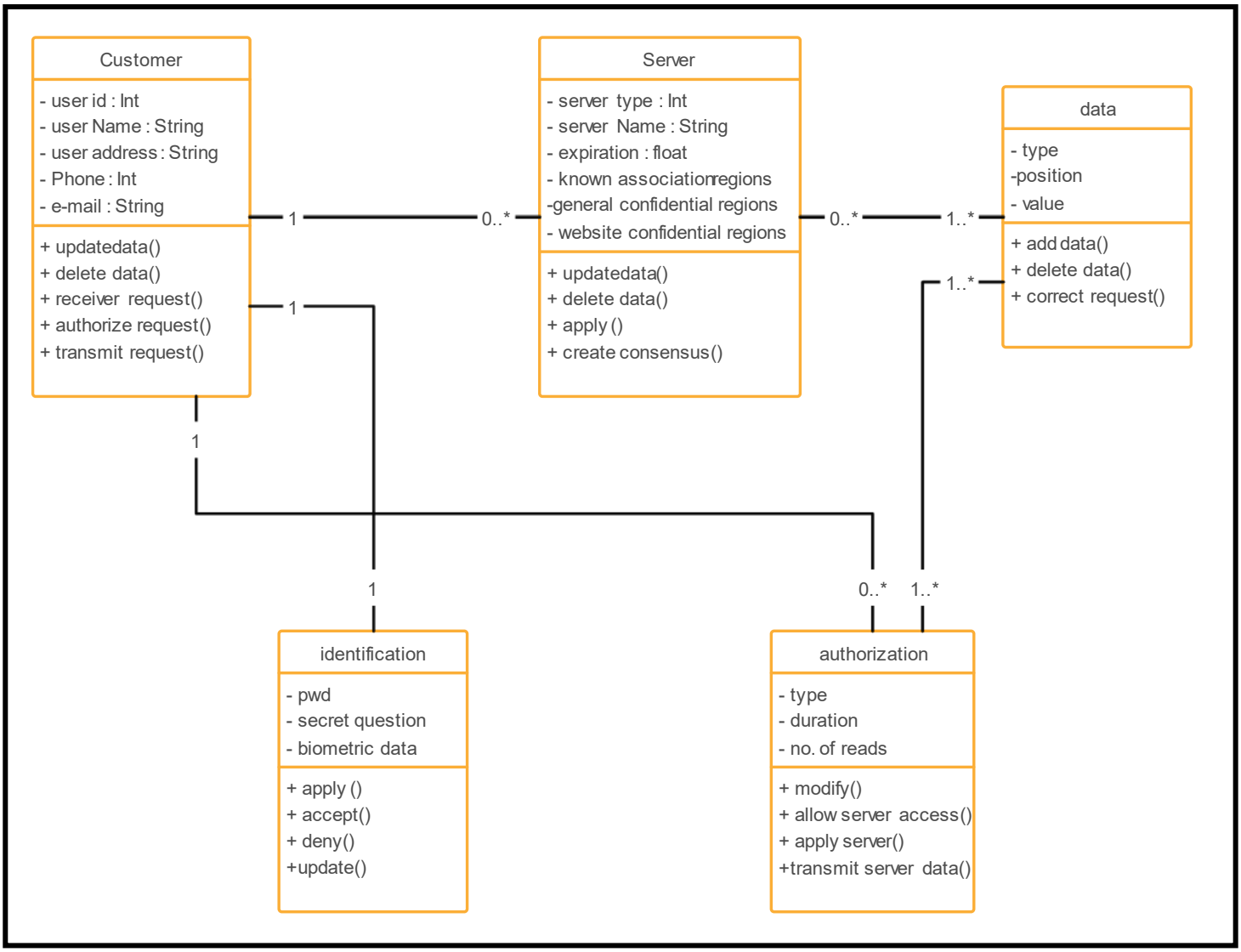
Use case diagrams are formally included in two modeling languages defined by the OMG: the unified modeling language (UML) and the systems modeling language (sys ML)

Use case diagram of our project: Use Case Diagram: User

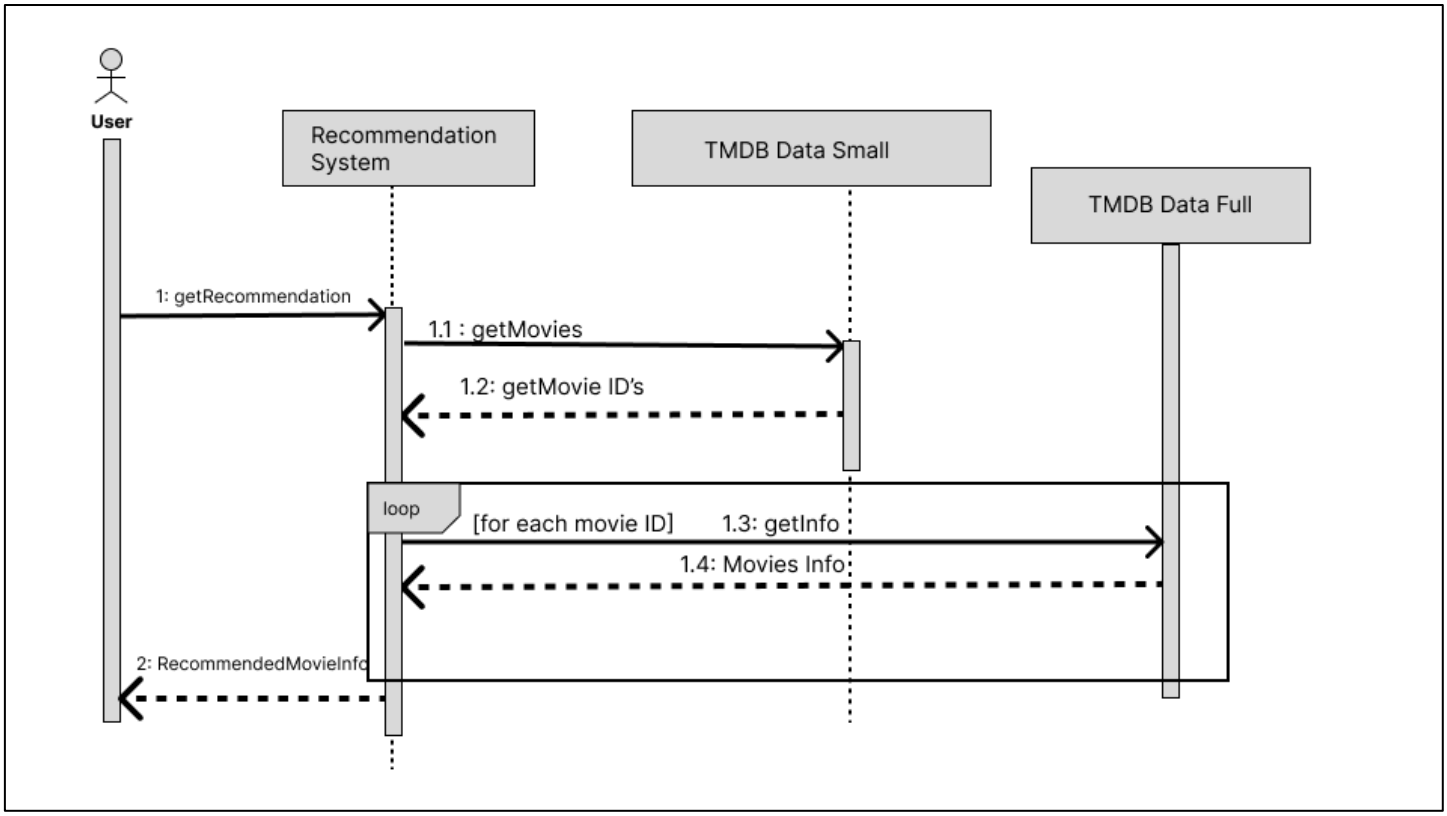


4.1.2 CLASS DIAGRAM

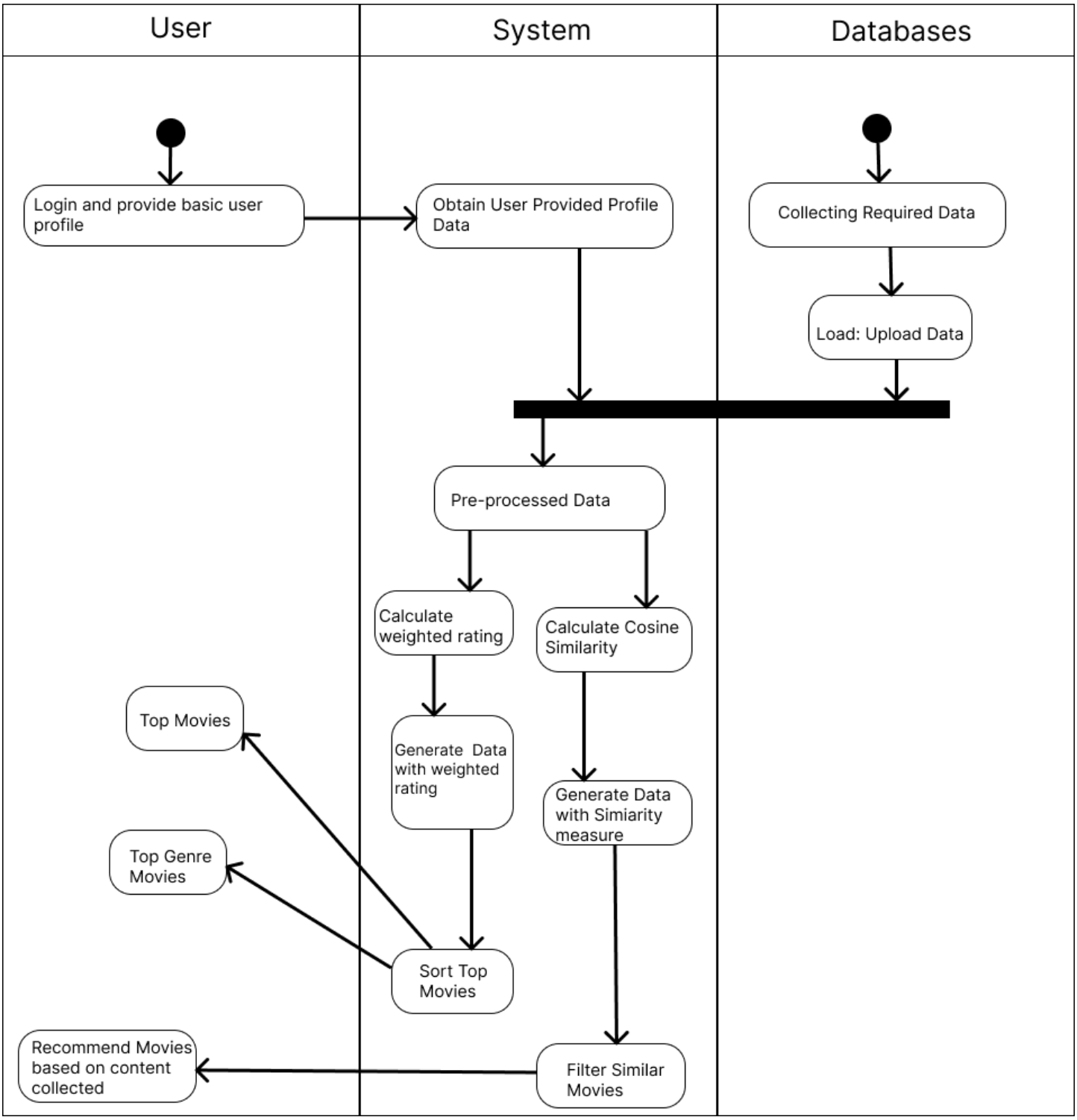
A Class is a category or group of things that has similar attributes and common behavior. A Rectangle is the icon that represents the class it is divided into three areas. The upper most area contains the name, the middle; area contains the attributes and the lowest areas show the operations. Class diagrams provides the representation that developers work from. Class diagrams help on the analysis side, too.



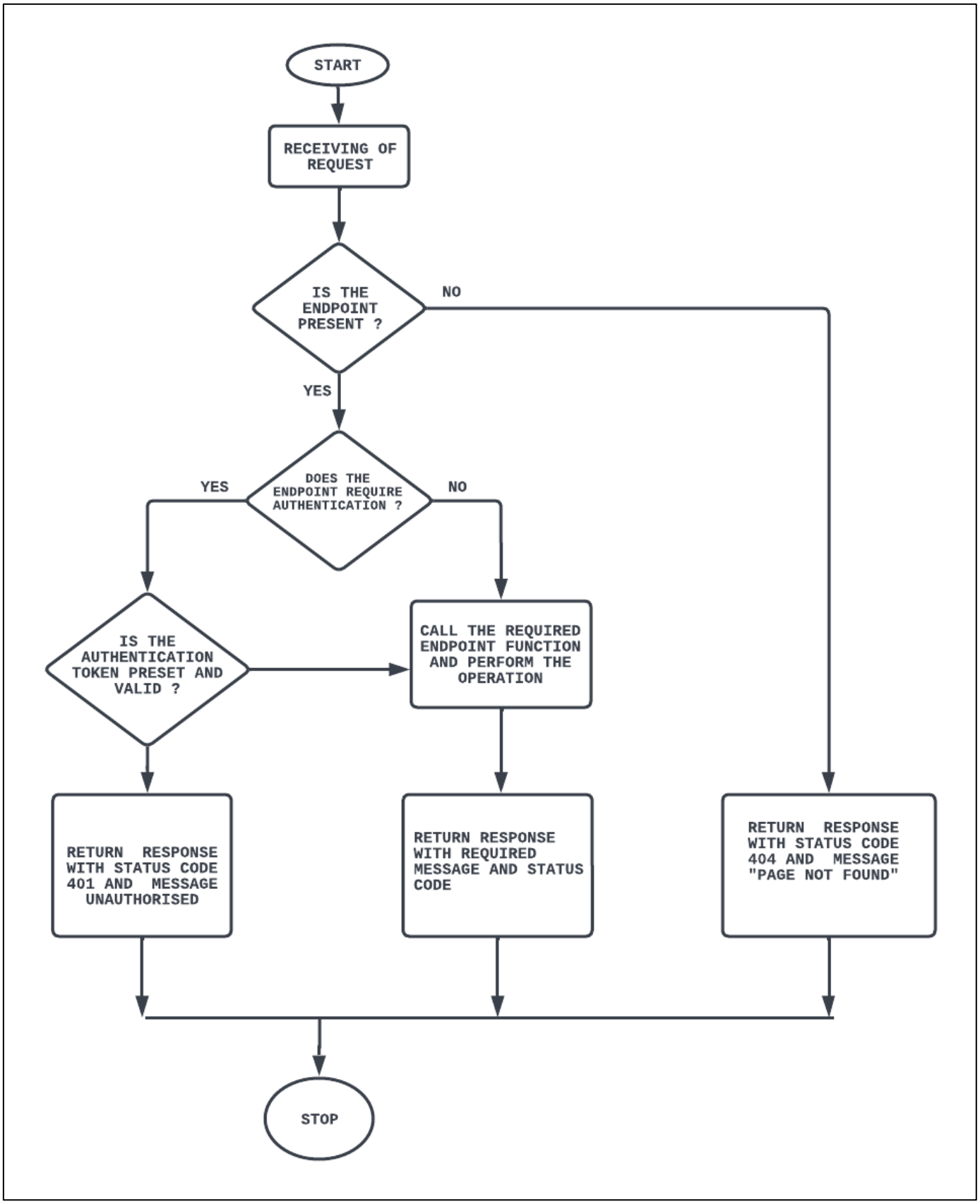
4.1.3 SEQUENCE DIAGRAM



4.1.4 ACTIVITY DIAGRAM



4.1.5 FLOW CHART

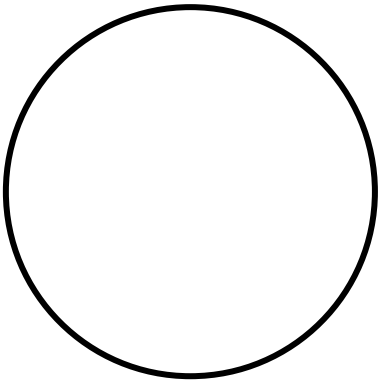


4.1.6 DATA FLOW DIAGRAM

A DFD does not show a sequence of steps. A DFD only shows what the different process in a system is and what data flows between them.

The following are some DFD symbols used in the project.

EXTERNAL ENTITIES



Process: A transaction of information that resides within the bounds of the system to be module.

DATAFLOWS

DATASTORE: A repository of data that is to be stored for use by one or more processes, may be as simple as buffer of queue or as a relational database.

RULES FOR DFD:

• Fix the scope of the system by means of context diagrams.

• Organize the DFD so that the main sequence of the actions reads left to right and top to bottom.

• Identify all inputs and outputs.

• Identify and label each process internal to the system with rounded circles.

• A process is required for all the data transformation and transfers. Therefore, never connect a data store to a data source or the destinations or another data store with just a data flow arrow.

• Do not indicate hardware and ignore control information.

• Make sure the names of the processes accurately convey everything the process is done.

• There must not be unnamed process.

• Indicate external sources and destinations of the data, with squares.

• Number each occurrence of repeated external entities.

• Identify all data flows for each process step, except simple Record retrievals.

• Label data flow on each arrow.

• Use details flow on each arrow.

• Use the details flow arrow to indicate data movements.

• There can’t be unnamed data flow.

• A data flow can’t connect two external entities.

LEVELS OF DFD:

The complexity of the business system means that it is a responsible to represent the operations of any system of single data flow diagram. At the top level, an Overview of the different systems in an organization is shown by the way of context analysis diagram. When exploded into DFD

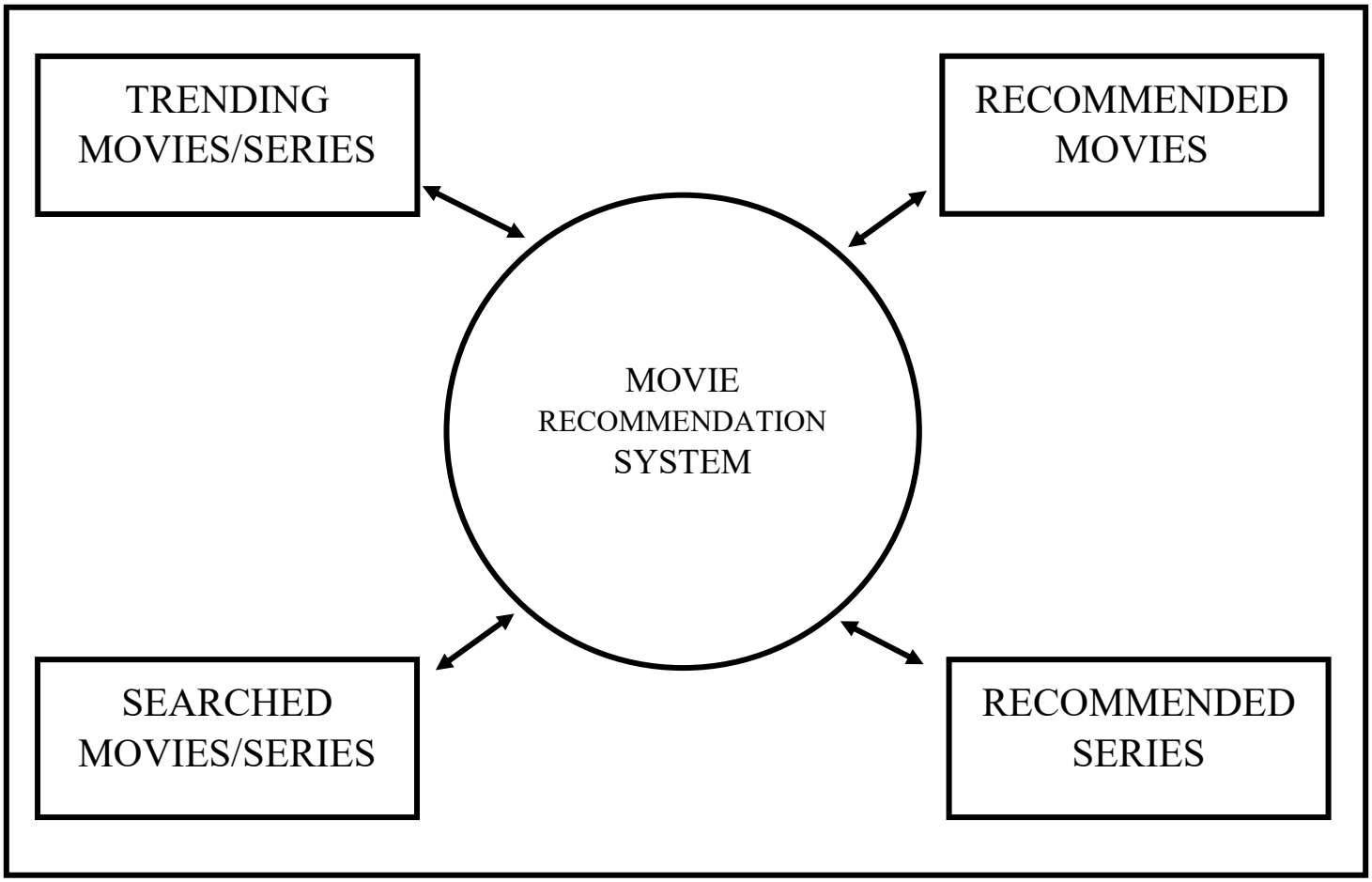
They are represented by:

- LEVEL-0: SYSTEM INPUT/OUTPUT
- LEVEL-1: SUBSYSTEM LEVEL DATAFLOW FUNCTIONAL
- LEVEL-2: FILE LEVEL DETAIL DATA FLOW.

The input and output data shown should be consistent from one level to the next.

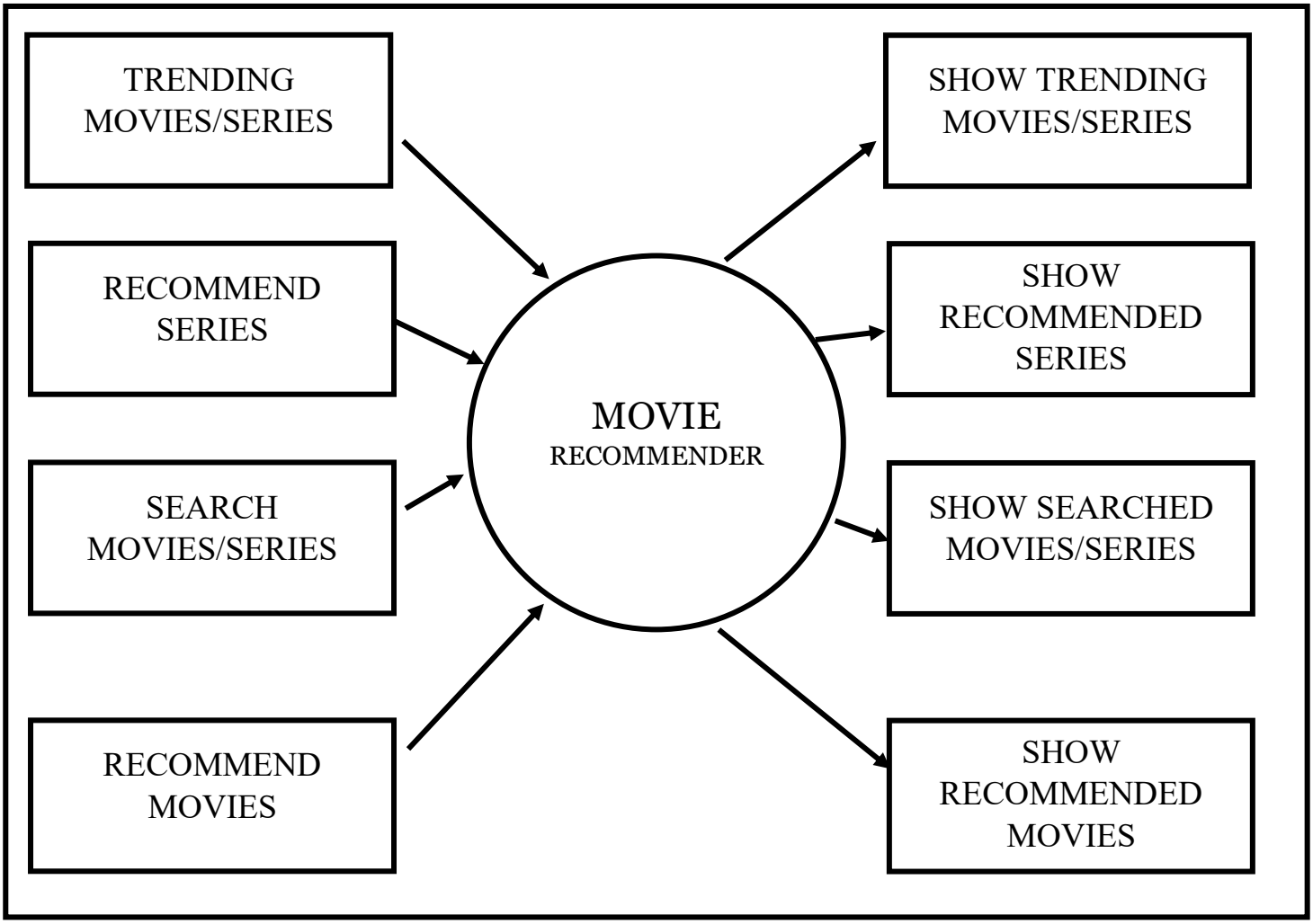
LEVEL-0: SYSTEM INPUT/OUTPUT LEVEL

A level-0 DFD describes the system-wide boundaries, dealing inputs to and outputs from the system and major processes. This diagram is similar to the combined user-level context diagram.



LEVEL-1: SUBSYSTEM LEVEL DATA FLOW

A level-1 DFD describes the next level of details within the system, detailing the data flows between subsystems, which makeup the whole.

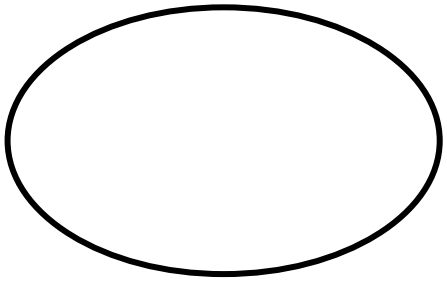
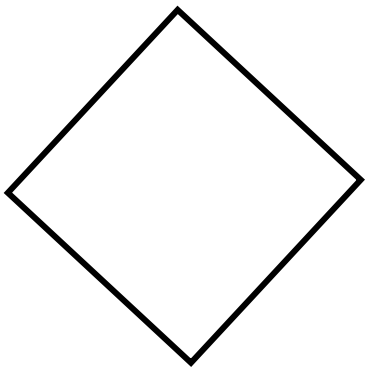


4.1.7 ER DIAGRAM

ENTITY-RELATIONSHIP Diagrams

E-R (Entity-Relationship) Diagram is used to represents the relationship between entities in the table.

The symbols used in E-R diagrams are:

SYMBOLS	PURPOSE
	Represents Entity Set
	Represent Attributes
	Represents Relationship Set
	Represents Line flow

Structured analysis is a set of tools and techniques that the analyst.

To develop a new kind of a system:

The traditional approach focuses on the cost benefit and feasibility analysis, Project management, and hardware and software selection a personal consideration.

5 IMPLEMENTATION

5.1 INTRODUCTION

Implementation is the stage of the project when the theoretical design is turned out into a working system. Thus, it can be considered to be the most critical stage in achieving a successful new system and in giving the user, confidence that the new system will work and be effective. The implementation stage involves careful planning, investigation of the existing system and its constraints on implementation, designing of methods to achieve changeover and evaluation of changeover methods.

5.2 CODES

5.2.1 Index.html [public / index.html]

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <link rel="icon" href="favicon.ico" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <meta name="theme-color" content="#000000" />
    <meta
      name="description"
      content="Get info for all your favorite movies and series"
    />
    <link rel="shortcut icon" href="favicon.ico" />
    <link rel="apple-touch-icon" href="apple-touch-icon.png" />
    <link
      rel="stylesheet"

href="https://fonts.googleapis.com/css?family=Roboto:100,300,400,500,700&display=swap"
    />
    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
    <title>Movie Recommender</title>
```

```
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>
```

5.2.2 Carousel

5.2.2.1 Carousel.js [src/component/carousel/ Carousel.js]

```
import axios from "axios";
import React, { useEffect, useState } from "react";
import AliceCarousel from "react-alice-carousel";
import "react-alice-carousel/lib/alice-carousel.css";
import { img_300, noPicture } from "../../config/config";
import "./Carousel.css";
const apiKey = '0ce524ee34cb6a3d23c4c6c1200883a0';
const handleDragStart = (e) => e.preventDefault();
const Gallery = ({ id, media_type }) => {
  const [credits, setCredits] = useState([]);
  const items = credits.map((c) => (
    <div className="carouselItem">
      <img
        src={c.profile_path ? `${img_300}/${c.profile_path}` : noPicture}
        alt={c?.name}
        onDragStart={handleDragStart}
        className="carouselItem__img"
      />
      <b className="carouselItem__txt">{c?.name}</b>
    </div>
  ));
  const responsive = {
```

```
    0: {
      items: 3,
    },
    512: {
      items: 5,
    },
    1024: {
      items: 7,
    },
  };
  const fetchCredits = async () => {
    const { data } = await axios.get(

`https://api.themoviedb.org/3/${media_type}/${id}/credits?api_key=${apiKey}&langua
ge=en-US`
    );
    setCredits(data.cast);
  };
  useEffect(() => {
    fetchCredits();
  }, []);
  return (
    <AliceCarousel
      mouseTracking
      infinite
      disableDotsControls
      disableButtonsControls
      responsive={responsive}
      items={items}
      autoPlay
    />
```

```
  );
};
export default Gallery;
```

5.2.2.2 Carousel.css [src/component/carousel/ Carousel.css]

```
.carouselItem {
  display: flex;
  flex-direction: column;
  object-fit: contain;
  padding: 10px;
}
.carouselItem__img {
  width: 75px;
  height: auto;
  border-radius: 10px;
  margin-bottom: 10px;
  box-shadow: 0px 0px 5px black;
}
```

5.2.3 ContentModal

5.2.3.1 ContentModal.js [src/component/contentModal/contentModal.js]

```
import React, { useEffect, useState } from "react";
import { makeStyles } from "@material-ui/core/styles";
import Modal from "@material-ui/core/Modal";
import Backdrop from "@material-ui/core/Backdrop";
import Fade from "@material-ui/core/Fade";
import axios from "axios";
import {
  img_500,
```

```
  unavailable,
  unavailableLandscape,
} from "../../config/config";
import "./ContentModal.css";
import { Button } from "@material-ui/core";
import YouTubeIcon from "@material-ui/icons/YouTube";
import Carousel from "../Carousel/Carousel";
const apiKey = "0ce524ee34cb6a3d23c4c6c1200883a0";

const useStyles = makeStyles((theme) => ({
  modal: {
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
  },
  paper: {
    width: "85%",
    height: "85%",
    backgroundColor: "#252424",
    border: "3.5px solid #0590c1",
    borderRadius: 10,
    color: "white",
    boxShadow: theme.shadows[5],
    padding: theme.spacing(1, 1, 3),
  },
}));

export default function TransitionsModal({ children, media_type, id }) {
  const classes = useStyles();
  const [open, setOpen] = useState(false);
  const [content, setContent] = useState();
```

```
  const [video, setVideo] = useState();

  const handleOpen = () => {
    setOpen(true);
  };

  const handleClose = () => {
    setOpen(false);
  };

  const fetchData = async () => {
    const { data } = await axios.get(

`https://api.themoviedb.org/3/${media_type}/${id}?api_key=${apiKey}&languag
e=en-US`
    );

    setContent(data);
    console.log(data);
  };

  const fetchVideo = async () => {
    const { data } = await axios.get(

`https://api.themoviedb.org/3/${media_type}/${id}/videos?api_key=${apiKey}&l
anguage=en-US`
    );

    setVideo(data.results[0]?.key);
  };
```

```
useEffect(() => {
  fetchData();
  fetchVideo();
}, []);

return (
  <>
    <div
      className="media"
      style={{ cursor: "pointer" }}
      color="inherit"
      onClick={handleOpen}
    >
      {children}
    </div>
    <Modal
      aria-labelledby="transition-modal-title"
      aria-describedby="transition-modal-description"
      className={classes.modal}
      open={open}
      onClose={handleClose}
      closeAfterTransition
      BackdropComponent={Backdrop}
      BackdropProps={{
        timeout: 500,
      }}
    >
      <Fade in={open}>
        {content && (
          <div className={classes.paper}>
            <div className="ContentModal">
```

```
<img
  src={
    content.poster_path
      ? `${img_500}/${content.poster_path}`
      : unavailable
  }
  alt={content.name || content.title}
  className="ContentModal__portrait"
/>
<img
  src={
    content.backdrop_path
      ? `${img_500}/${content.backdrop_path}`
      : unavailableLandscape
  }
  alt={content.name || content.title}
  className="ContentModal__landscape"
/>
<br />
<br />
<div className="ContentModal__about">
  <span className="ContentModal__title">
    {content.name || content.title} (
    {(
      content.first_air_date ||
      content.release_date ||
      "-----"
    ).substring(0, 4)}
    )
  </span>
  {content.tagline && (
```

```
            <i className="tagline">{content.tagline}</i>
          )}
          <br />
          <span className="ContentModal__description">
            {content.overview}
          </span>
          <br />
          <div>
            <Carousel id={id} media_type={media_type} />
          </div>

          <Button
            variant="contained"
            startIcon={<YouTubeIcon />}
            color="secondary"
            target="__blank"
            href={`https://www.youtube.com/watch?v=${video}`}
          >
            Trailer
          </Button>
        </div>
      </div>
    </div>
  )}
 </Fade>
</Modal>
</>
);
}
```

5.2.3.2 ContentModal.js [src/component/contentModal/contentModal.css]

```
@import
url("https://fonts.googleapis.com/css2?family=Roboto:wght@100&display=swap
");

.ContentModal__landscape {
  object-fit: contain;
  border-radius: 10px;
}

.ContentModal__portrait {
  display: none;
  object-fit: contain;
  border-radius: 10px;
}
.tagline {
  padding-bottom: 10px;
  align-self: center;
}

.ContentModal {
  display: flex;
  flex-direction: column;
  justify-content: space-between;
  height: 100%;
  width: 100%;
  overflow-y: scroll;
  scrollbar-width: none;
}

.ContentModal::-webkit-scrollbar {
```

```
  display: none;
}

.ContentModal__about {
  padding: 10px;
  width: 95%;
  height: 90%;
  display: flex;
  flex-direction: column;
  font-family: "Roboto";
  justify-content: space-evenly;
  font-weight: 300;
}

.ContentModal__title {
  height: 18%;
  font-size: 5vw;
  display: flex;
  align-items: center;
  justify-content: center;
}
.ContentModal__description {
  display: flex;
  height: auto;
  /* overflow-y: scroll; */
  padding: 15px 10px;
  border-radius: 15px;
  box-shadow: inset 0 0 5px #000000;
  text-align: justify;
}
```

```
.ContentModal__description::-webkit-scrollbar {
  display: none;
}

@media (min-width: 835px) {
  .ContentModal__landscape {
    display: none;
  }
  .ContentModal__portrait {
    display: flex;
    width: 38%;
  }
  .ContentModal {
    flex-direction: row;
    justify-content: space-around;
    padding: 10px 0;
  }
  .ContentModal__about {
    width: 58%;
    padding: 0;
    height: 100%;
  }
  .ContentModal__title {
    font-size: 3.5vw;
  }
  .ContentModal__description {
    font-size: 20px;
  }
}
```

5.2.4 Genre [src/component/genre/genre.js]

```
import { Chip } from "@material-ui/core";
import axios from "axios";
import { useEffect } from "react";
const apiKey = '0ce524ee34cb6a3d23c4c6c1200883a0';
const Genres = ({
  selectedGenres,
  setSelectedGenres,
  genres,
  setGenres,
  type,
  setPage,
}) => {
  const handleAdd = (genre) => {
    setSelectedGenres([...selectedGenres, genre]);
    setGenres(genres.filter((g) => g.id !== genre.id));
    setPage(1);
  };

  const handleRemove = (genre) => {
    setSelectedGenres(
      selectedGenres.filter((selected) => selected.id !== genre.id)
```

```
    );
    setGenres([...genres, genre]);
    setPage(1);
  };

  const fetchGenres = async () => {
    const { data } = await axios.get(
      `https://api.themoviedb.org/3/genre/${type}/list?api_key=${apiKey}&language=en-US`
    );
    setGenres(data.genres);
  };

  useEffect(() => {
    fetchGenres();
    return () => {
      setGenres({}); // unmounting
    };
  }, []);

  return (
    <div style={{ padding: "6px 0" }}>
      {selectedGenres.map((genre) => (
```

```
        <Chip
          style={{ margin: 2 }}
          label={genre.name}
          key={genre.id}
          color="primary"
          clickable
          size="small"
          onDelete={() => handleRemove(genre)}
        />
      ))}
      {genres.map((genre) => (
        <Chip
          style={{ margin: 2 }}
          label={genre.name}
          key={genre.id}
          clickable
          size="small"
          onClick={() => handleAdd(genre)}
        />
      ))}
    </div>
  );
};
```

```
export default Genres;
```

5.2.5 HEADER

5.2.5.1 Header.js [src/component/header/header.js]

```
import "./Header.css";

const Header = () => {
  return (
    <span onClick={() => window.scroll(0, 0)} className="header">
      Movie Recommender
    </span>
  );
};

export default Header;
```

5.2.5.2 Header.css [src/component/header/header.css]

```
@import
url('https://fonts.googleapis.com/css2?family=Cinzel+Decorative:wght@900&dis
play=swap');

.header {
  width: 100%;
  cursor: pointer;
  position: fixed;
  display: flex;
  justify-content: center;
  text-transform: uppercase;
  background-color: #000000;
```

```
  font-family: 'Cinzel Decorative', 'Segoe UI', Tahoma, Verdana,'Times New
Roman';
  font-size: 5vw;
  padding-bottom: 15px;
  color: rgba(168, 168, 168, 0.75);
  z-index: 100;
}

@media (max-width: 1000px) {
  .header {
    padding-top: 15px;
    font-size: 6.4vw;
  }
}
```

5.2.6 customPagination

5.2.6.1 customPagination.js [src/component/Pagination/customPagination.js]

```
import React from "react";
import Pagination from "@material-ui/lab/Pagination";
import { createMuiTheme, ThemeProvider } from "@material-ui/core";

const darkTheme = createMuiTheme({
  palette: {
    type: "dark",
  },
});

export default function CustomPagination({ setPage, numOfPages = 15 }) {
  // Scroll to top when page changes
  const handlePageChange = (page) => {
```

```
  setPage(page);
  window.scroll(0, 0);
 };

 return (
  <div
   style={{
    width: "100%",
    display: "flex",
    justifyContent: "center",
    marginTop: 10,
   }}
  >
   <ThemeProvider theme={darkTheme}>
    <Pagination
     onChange={(e) => handlePageChange(e.target.textContent)}
     count={numOfPages}
     color="primary"
     hideNextButton
     hidePrevButton
    />
   </ThemeProvider>
  </div>
 );
}
```

5.2.6.2 customPagination.js [src/component/Pagination/customPagination.js]

```
.media {
 display: flex;
 flex-direction: column;
 width: 23%;
```

```
  padding: 5px;
  margin: 5px 0;
  background-color: #252424;
  border-radius: 10px;
  position: relative;
  font-family: Georgia, "Times New Roman", serif, "Courier New", Courier;
}

.media:hover {
  background-color: #0590c1;
  color: black;
  transform: scale(1.02);
  transition: 0.5s ease-in-out;
}
@media (max-width: 550px) {
  .media {
    width: 80%;
  }
  .media:hover {
    transform: none;
  }
}

.poster {
  border-radius: 2.5px;
  border-bottom-left-radius: 5px;
  border-bottom-right-radius: 5px;
}

.title {
  width: 100%;
```

```
  text-align: center;
  font-size: 17px;
  padding: 8px 0;
}

.subTitle {
  display: flex;
  justify-content: space-between;
  padding: 0 2px;
  padding-bottom: 3px;
}
```

5.2.7 singleContent

5.2.7.1 singleContent.js [src/component/ singleContent / singleContent.js]

```
import { Badge } from "@material-ui/core";
import { img_300, unavailable } from "../../config/config";
import "./SingleContent.css";
import ContentModal from "../ContentModal/ContentModal";

const SingleContent = ({
  id,
  poster,
  title,
  date,
  media_type,
  vote_average,
}) => {
  return (
    <ContentModal media_type={media_type} id={id}>
```

```
      <Badge
        badgeContent={vote_average}
        color={vote_average > 6 ? "primary" : "secondary"}
      />
      <img
        className="poster"
        src={poster ? `${img_300}${poster}` : unavailable}
        alt={title}
      />
      <b className="title">{title}</b>
      <span className="subTitle">
        {media_type === "tv" ? "TV Series" : "Movie"}
        <span className="subTitle">{date}</span>
      </span>
    </ContentModal>
  );
};
export default SingleContent;
```

5.2.7.2 singleContent.css [src/component/ singleContent / singleContent.css]

```
.media {
  display: flex;
  flex-direction: column;
  width: 23%;
  padding: 5px;
  margin: 5px 0;
  background-color: #252424;
  border-radius: 10px;
  position: relative;
  font-family: Georgia, "Times New Roman", serif, "Courier New", Courier;
}
```

```
.media:hover {
  background-color: #0590c1;
  color: black;
  transform: scale(1.02);
  transition: 0.5s ease-in-out;
}
@media (max-width: 550px) {
  .media {
    width: 80%;
  }
  .media:hover {
    transform: none;
  }
}

.poster {
  border-radius: 2.5px;
  border-bottom-left-radius: 5px;
  border-bottom-right-radius: 5px;
}

.title {
  width: 100%;
  text-align: center;
  font-size: 17px;
  padding: 8px 0;
}

.subTitle {
  display: flex;
```

```
  justify-content: space-between;
  padding: 0 2px;
  padding-bottom: 3px;
}
```

5.2.8 MainNav.js [src/component/MainNav.js]

```
import React, { useEffect } from "react";
import { makeStyles } from "@material-ui/core/styles";
import BottomNavigation from "@material-ui/core/BottomNavigation";
import BottomNavigationAction from "@material-
ui/core/BottomNavigationAction";
import TvIcon from "@material-ui/icons/Tv";
import MovieIcon from "@material-ui/icons/Movie";
import SearchIcon from "@material-ui/icons/Search";
import WhatshotIcon from "@material-ui/icons/Whatshot";
import { useHistory } from "react-router-dom";

const useStyles = makeStyles({
  root: {
    width: "100%",
    position: "fixed",
    bottom: 0,
    backgroundColor: "#000000",
    zIndex: 100,
  },
});

export default function SimpleBottomNavigation() {
  const classes = useStyles();
  const [value, setValue] = React.useState(0);
```

```
const history = useHistory();

useEffect(() => {
  if (value === 0) {
    history.push("/");
  } else if (value === 1) {
    history.push("/movies");
  } else if (value === 2) {
    history.push("/series");
  } else if (value === 3) {
    history.push("/search");
  }
}, [value, history]);

return (
  <BottomNavigation
    value={value}
    onChange={(event, newValue) => {
      setValue(newValue);
    }}
    showLabels
    className={classes.root}
  >
    <BottomNavigationAction
      style={{ color: "white" }}
      label="Trending"
      icon={<WhatshotIcon />}
    />
    <BottomNavigationAction
      style={{ color: "white" }}
      label="Movies"
```

```
        icon={<MovieIcon />}
      />
      <BottomNavigationAction
        style={{ color: "white" }}
        label="TV Series"
        icon={<TvIcon />}
      />
      <BottomNavigationAction
        style={{ color: "white" }}
        label="Search"
        icon={<SearchIcon />}
      />
    </BottomNavigation>
  );
}
```

5.2.9 Config.js [src/component/config/config.js]

```
//image sizes for tmdb
export const img_300 = "https://image.tmdb.org/t/p/w300";
export const img_500 = "https://image.tmdb.org/t/p/w500";

// contentModal and singleContent
export const unavailable =
  "https://www.movienewz.com/img/films/poster-holder.jpg";

// contentModal
export const unavailableLandscape =
  "https://user-images.githubusercontent.com/10515204/56117400-9a911800-
5f85-11e9-878b-3f998609a6c8.jpg";
```

```
// For Carousel
export const noPicture =
  "https://upload.wikimedia.org/wikipedia/en/6/60/No_Picture.jpg";
```

5.2.10 Movies.js [src/Pages/Movies/movies.js]

```
import axios from "axios";
import { useEffect, useState } from "react";
import Genres from "../../components/Genres/Genres";
import SingleContent from "../../components/SingleContent/SingleContent";
import useGenre from "../../hooks/useGenre";
import CustomPagination from "../../components/Pagination/CustomPagination";

const Movies = () => {
  const [genres, setGenres] = useState([]);
  const [selectedGenres, setSelectedGenres] = useState([]);
  const [page, setPage] = useState(1);
  const [content, setContent] = useState([]);
  const [numOfPages, setNumOfPages] = useState();
  const genreforURL = useGenre(selectedGenres);
  // console.log(selectedGenres);
const apiKey = '0ce524ee34cb6a3d23c4c6c1200883a0'

  const fetchMovies = async () => {
    const { data } = await axios.get(

`https://api.themoviedb.org/3/discover/movie?api_key=${apiKey}&language=en-
US&sort_by=popularity.desc&include_adult=false&include_video=false&page=$
{page}&with_genres=${genreforURL}`
    );
    setContent(data.results);
```

```
  // setNumOfPages(data.total_pages);
  (data.total_pages>15)?setNumOfPages(15):setNumOfPages(data.total_pages);
  // console.log(data);
};

useEffect(() => {
  window.scroll(0, 0);
  fetchMovies();
}, [genreforURL, page]);

return (
  <div>
    <span className="pageTitle">Explore Movies</span>
    <Genres
      type="movie"
      selectedGenres={selectedGenres}
      setSelectedGenres={setSelectedGenres}
      genres={genres}
      setGenres={setGenres}
      setPage={setPage}
    />
    <div className="trending">
      {content &&
        content.map((c) => (
          <SingleContent
            key={c.id}
            id={c.id}
            poster={c.poster_path}
            title={c.title || c.name}
            date={c.first_air_date || c.release_date}
            media_type="movie"
```

```
              vote_average={c.vote_average}
            />
          ))}
        </div>
        {numOfPages > 1 && (
          <CustomPagination setPage={setPage} numOfPages={numOfPages} />
        )}
      </div>
    );
};

export default Movies;
```

5.2.11 search.js [src/Pages/Search/search.js]

```
import {
  Button,
  createMuiTheme,
  Tab,
  Tabs,
  TextField,
  ThemeProvider,
} from "@material-ui/core";
import "./Search.css";
import SearchIcon from "@material-ui/icons/Search";
import { useEffect, useState } from "react";
import axios from "axios";
import CustomPagination from "../../components/Pagination/CustomPagination";
import SingleContent from "../../components/SingleContent/SingleContent";

const apiKey = '0ce524ee34cb6a3d23c4c6c1200883a0';
```

```
const Search = () => {
  const [type, setType] = useState(0);
  const [searchText, setSearchText] = useState("");
  const [page, setPage] = useState(1);
  const [content, setContent] = useState([]);
  const [numOfPages, setNumOfPages] = useState();
  const darkTheme = createMuiTheme({
    palette: {
      type: "dark",
      primary: {
        main: "#fff",
      },
    },
  });

  const fetchSearch = async () => {
    try {
      const { data } = await axios.get(
        `https://api.themoviedb.org/3/search/${type ? "tv" :
"movie"}?api_key=${apiKey}&language=en-
US&query=${searchText}&page=${page}&include_adult=false`
      );
      setContent(data.results);
      setNumOfPages(data.total_pages);
      // console.log(data);
    } catch (error) {
      console.error(error);
    }
  };

  useEffect(() => {
```

```
  window.scroll(0, 0);
  fetchSearch();
}, [type, page]);

return (
  <div>
    <ThemeProvider theme={darkTheme}>
      <div className="search">
        <TextField
          style={{ flex: 1 }}
          className="searchBox"
          label="Search"
          variant="filled"
          onChange={(e) => setSearchText(e.target.value)}
        />
        <Button
          onClick={fetchSearch}
          variant="contained"
          style={{ marginLeft: 10 }}
        >
          <SearchIcon fontSize="large" />
        </Button>
      </div>
      <Tabs
        value={type}
        indicatorColor="primary"
        textColor="primary"
        onChange={(event, newValue) => {
          setType(newValue);
          setPage(1);
        }}
```

```
          style={{ paddingBottom: 5 }}
          aria-label="disabled tabs example"
        >
          <Tab style={{ width: "50%" }} label="Search Movies" />
          <Tab style={{ width: "50%" }} label="Search TV Series" />
        </Tabs>
      </ThemeProvider>
      <div className="trending">
        {content &&
          content.map((c) => (
            <SingleContent
              key={c.id}
              id={c.id}
              poster={c.poster_path}
              title={c.title || c.name}
              date={c.first_air_date || c.release_date}
              media_type={type ? "tv" : "movie"}
              vote_average={c.vote_average}
            />
          ))}
        {searchText &&
          !content &&
          (type ? <h2>No Series Found</h2> : <h2>No Movies Found</h2>)}
      </div>
      {numOfPages > 1 && (
        <CustomPagination setPage={setPage} numOfPages={numOfPages} />
      )}
    </div>
  );
};
```

```
export default Search;
```

5.2.12 search.js [sr/Pages/Search/search.js]

```
import axios from "axios";
import { useEffect, useState } from "react";
import Genres from "../../components/Genres/Genres";
import CustomPagination from "../../components/Pagination/CustomPagination";
import SingleContent from "../../components/SingleContent/SingleContent";
import useGenre from "../../hooks/useGenre";

const apiKey = '0ce524ee34cb6a3d23c4c6c1200883a0';
const Series = () => {
  const [genres, setGenres] = useState([]);
  const [selectedGenres, setSelectedGenres] = useState([]);
  const [page, setPage] = useState(1);
  const [content, setContent] = useState([]);
  const [numOfPages, setNumOfPages] = useState();
  const genreforURL = useGenre(selectedGenres);

  const fetchSeries = async () => {
    const { data } = await axios.get(
      `https://api.themoviedb.org/3/discover/tv?api_key=${apiKey}&language=en-
US&sort_by=popularity.desc&include_adult=false&include_video=false&page=$
{page}&with_genres=${genreforURL}`
    );
    setContent(data.results);
    // setNumOfPages(data.total_pages);
    (data.total_pages>15)?setNumOfPages(15):setNumOfPages(data.total_pages);
    // console.log(data);
  };
```

```
useEffect(() => {
  window.scroll(0, 0);
  fetchSeries();
}, [genreforURL, page]);

return (
  <div>
    <span className="pageTitle">Explore Series</span>
    <Genres
      type="tv"
      selectedGenres={selectedGenres}
      setSelectedGenres={setSelectedGenres}
      genres={genres}
      setGenres={setGenres}
      setPage={setPage}
    />
    <div className="trending">
      {content &&
        content.map((c) => (
          <SingleContent
            key={c.id}
            id={c.id}
            poster={c.poster_path}
            title={c.title || c.name}
            date={c.first_air_date || c.release_date}
            media_type="tv"
            vote_average={c.vote_average}
          />
        ))}
    </div>
```

```
      {numOfPages > 1 && (
        <CustomPagination setPage={setPage} numOfPages={numOfPages} />
      )}
    </div>
  );
};
export default Series;
```

5.2.13 trending.js [src/Pages/Trending/trending.js]

```
import axios from "axios";
import "./Trending.css";
import { useEffect, useState } from "react";
import SingleContent from "../../components/SingleContent/SingleContent";
import CustomPagination from "../../components/Pagination/CustomPagination";

const apiKey = '0ce524ee34cb6a3d23c4c6c1200883a0';
const Trending = () => {
  const [page, setPage] = useState(1);
  const [content, setContent] = useState([]);

  const fetchTrending = async () => {
    const { data } = await axios.get(

`https://api.themoviedb.org/3/trending/all/day?api_key=${apiKey}&page=${page
}`
    );
    setContent(data.results);
  };

  useEffect(() => {
    window.scroll(0, 0);
```

```
  fetchTrending();
 }, [page]);

 return (
  <div>
   <span className="pageTitle">Trending Today</span>
   <div className="trending">
    {content &&
     content.map((c) => (
      <SingleContent
       key={c.id}
       id={c.id}
       poster={c.poster_path}
       title={c.title || c.name}
       date={c.first_air_date || c.release_date}
       media_type={c.media_type}
       vote_average={c.vote_average}
      />
     ))}
   </div>
   <CustomPagination setPage={setPage} />
  </div>
 );
};

export default Trending;
```

5.2.14 App.js [src/app.js]

```
import { BrowserRouter, Route, Switch } from "react-router-dom";
import "./App.css";
import Header from "./components/Header/Header";
import SimpleBottomNavigation from "./components/MainNav";
import Movies from "./Pages/Movies/Movies";
import Series from "./Pages/Series/Series";
import Trending from "./Pages/Trending/Trending";
import Search from "./Pages/Search/Search";
import { Container } from "@material-ui/core";

function App() {
  return (
    <BrowserRouter>
      <Header />
      <div className="app">
        <Container>
          <Switch>
            <Route path="/" component={Trending} exact />
            <Route path="/movies" component={Movies} />
            <Route path="/series" component={Series} />
            <Route path="/search" component={Search} />
          </Switch>
        </Container>
      </div>
      <SimpleBottomNavigation />
    </BrowserRouter>
  );
}
export default App;
```

5.2.15 INDEX

5.2.15.1 index.js [src/index.js]

```
import React from "react";
import ReactDOM from "react-dom";
import "./index.css";
import App from "./App";

ReactDOM.render(<App />, document.getElementById("root"));
// Diable inspect element from user
(document.onkeydown = function (event) {
  if (event.keyCode == 123) {
    return false;
  } else if (event.ctrlKey && event.shiftKey && event.keyCode == 73) {
    return false;
  } else if (event.ctrlKey && event.shiftKey && event.keyCode == 67) {
    return false;
  } else if (event.ctrlKey && event.shiftKey && event.keyCode == 86) {
    return false;
  } else if (event.ctrlKey && event.shiftKey && event.keyCode == 117) {
    return false;
  } else if (event.ctrlKey && event.keyCode == 85) {
    return false;
  }
}),
  false;
if (document.addEventListener) {
  document.addEventListener(
    "contextmenu",
    function (e) {
      e.preventDefault();
    },
```

```
    false
  );
} else {
  document.attachEvent("oncontextmenu", function () {
    window.event.returnValue = false;
  });
}
```

5.2.15.2 index.css[src/index.css]

```
@import
url("https://fonts.googleapis.com/css2?family=Montserrat:wght@100&display=s
wap");
@import "react-alice-carousel/lib/alice-carousel.css";
a {
  text-decoration: none;
  color: inherit;
}
body {
  margin: 0;
  font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto",
"Oxygen",
    "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
    sans-serif;
  -webkit-font-smoothing: antialiased;
  -moz-osx-font-smoothing: grayscale;
}
code {
  font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
    monospace;
}
```

6 TESTING AND REPORT

6.1 INTRODUCTION TO SYSTEM 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.

6.2 TYPE OF TESTING

6.2.1 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.

6.2.2 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.

6.2.3 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.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

6.2.4 SYSTEM TESTING

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

6.2.5 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.

6.2.6 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.

6.2.7 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:

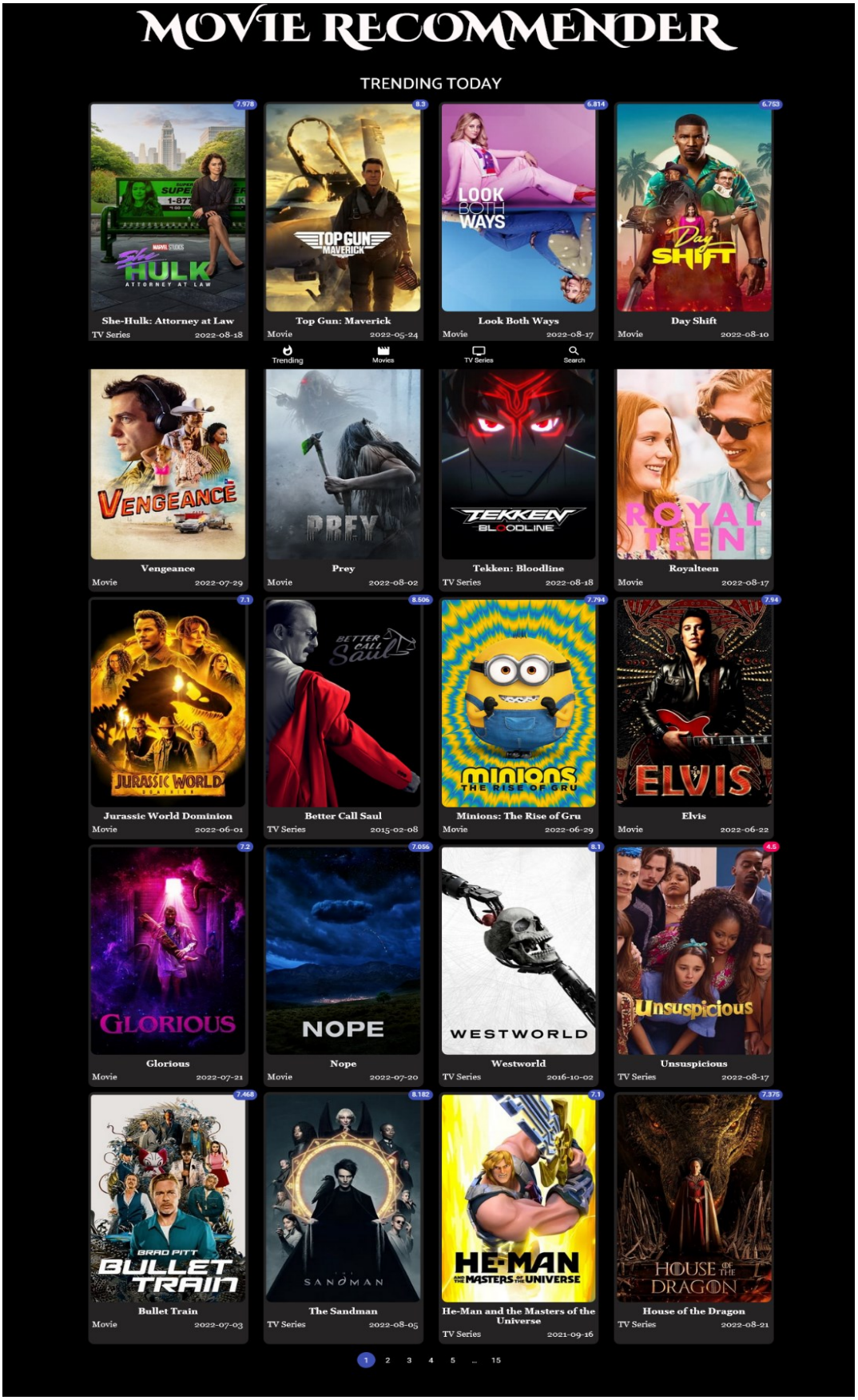
All the test cases mentioned above passed successfully. No defects encountered.

*Below are the test cases for the **Movie Recommendation Engine** web application.*

TEST MODULE	TEXT CASE	EXPECTED RESULT	TEST RESULT
USER	Selects a specific content displayed on the home page of the application.	A detailed description about the selected content is displayed.	**PASS**
USER	Selects a single or multiple genres according to their choice of movie which they want to watch.	Movies will be filtered out according to the genres selected by the user.	**PASS**
USER	Selects a single or multiple genres according to their choice of series which they want to watch.	Series will be filtered out according to the genres selected by the user.	**PASS**
USER	Searches a specific movie or series according to their choice.	If the specific movie or series is available, then it will be displayed otherwise it displays “Movie/Series not available”.	**PASS**

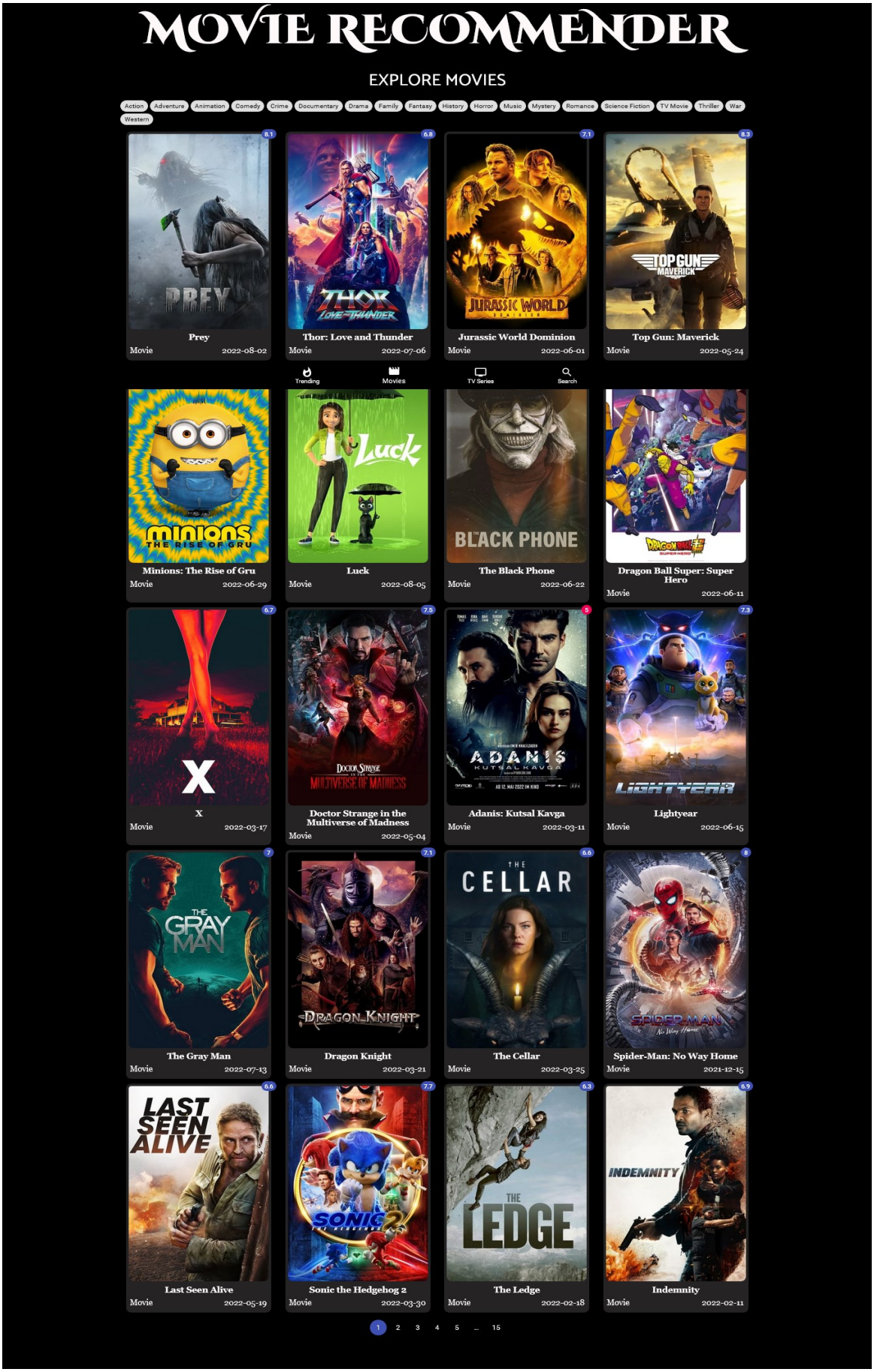
7 SAMPLE SCREENSHOTS

HOME PAGE



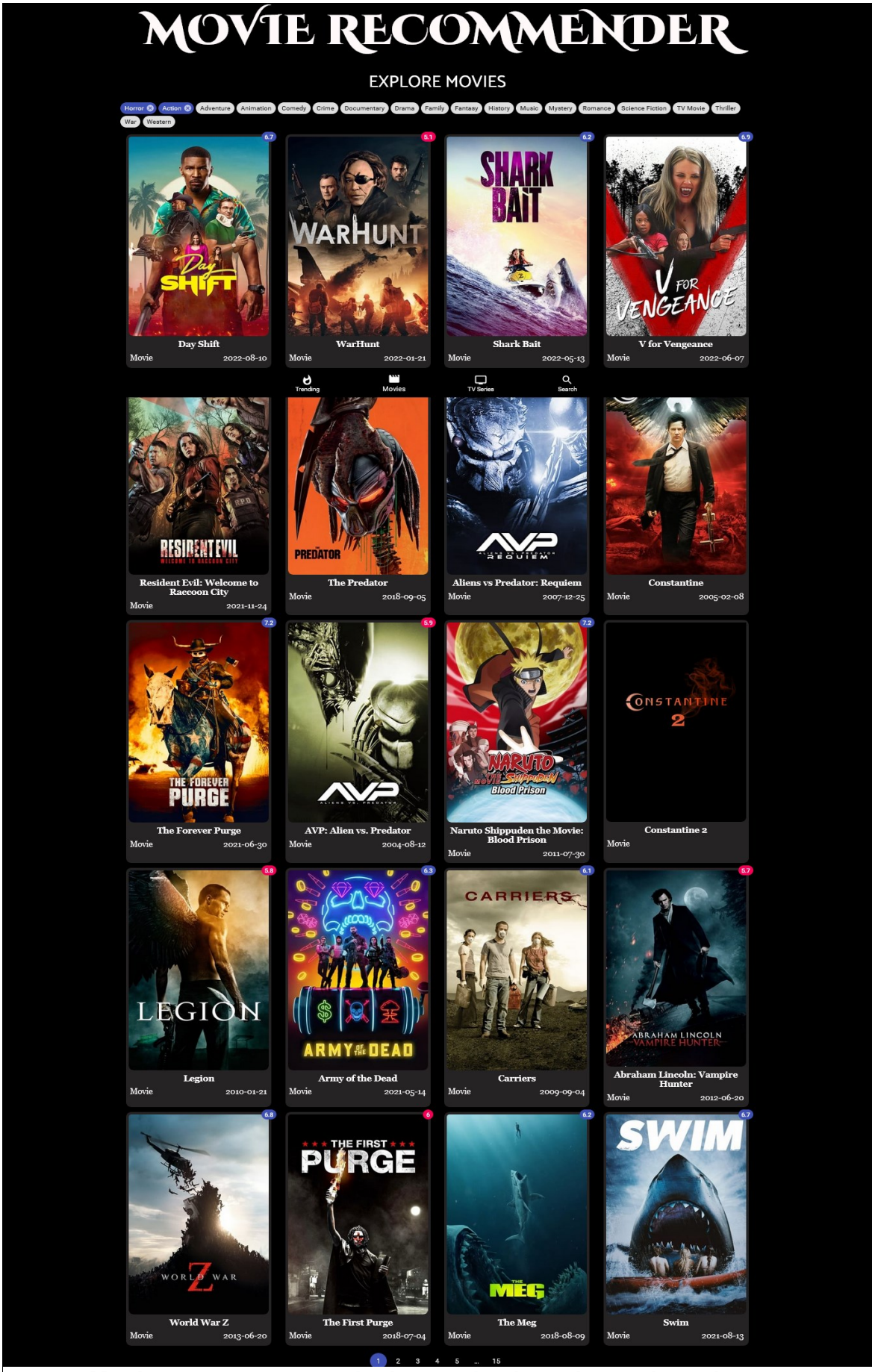
Home page displays the trending movies and series

MOVIES PAGE



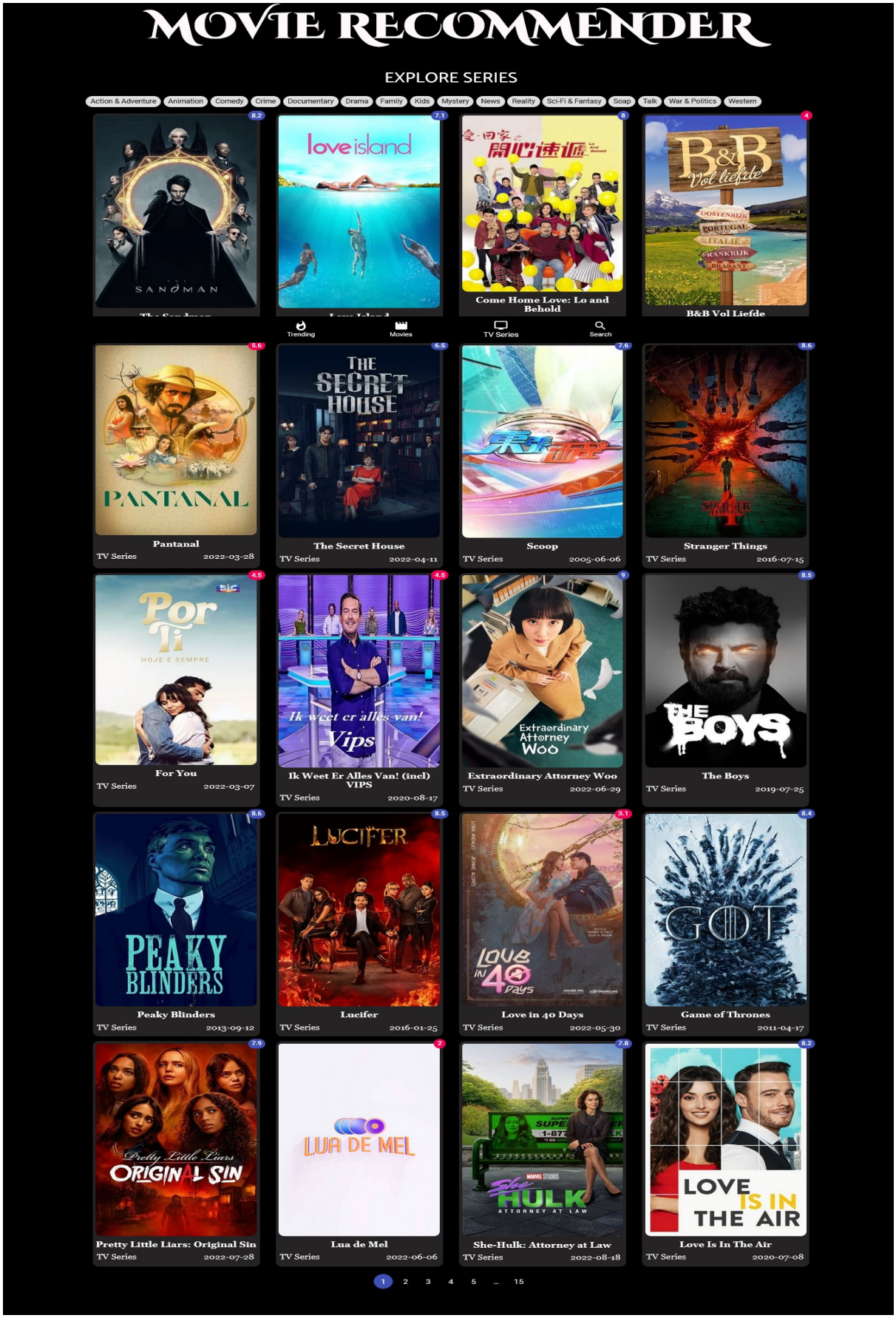
Movie page displays the trending movies

MOVIES PAGE (with recommendation)



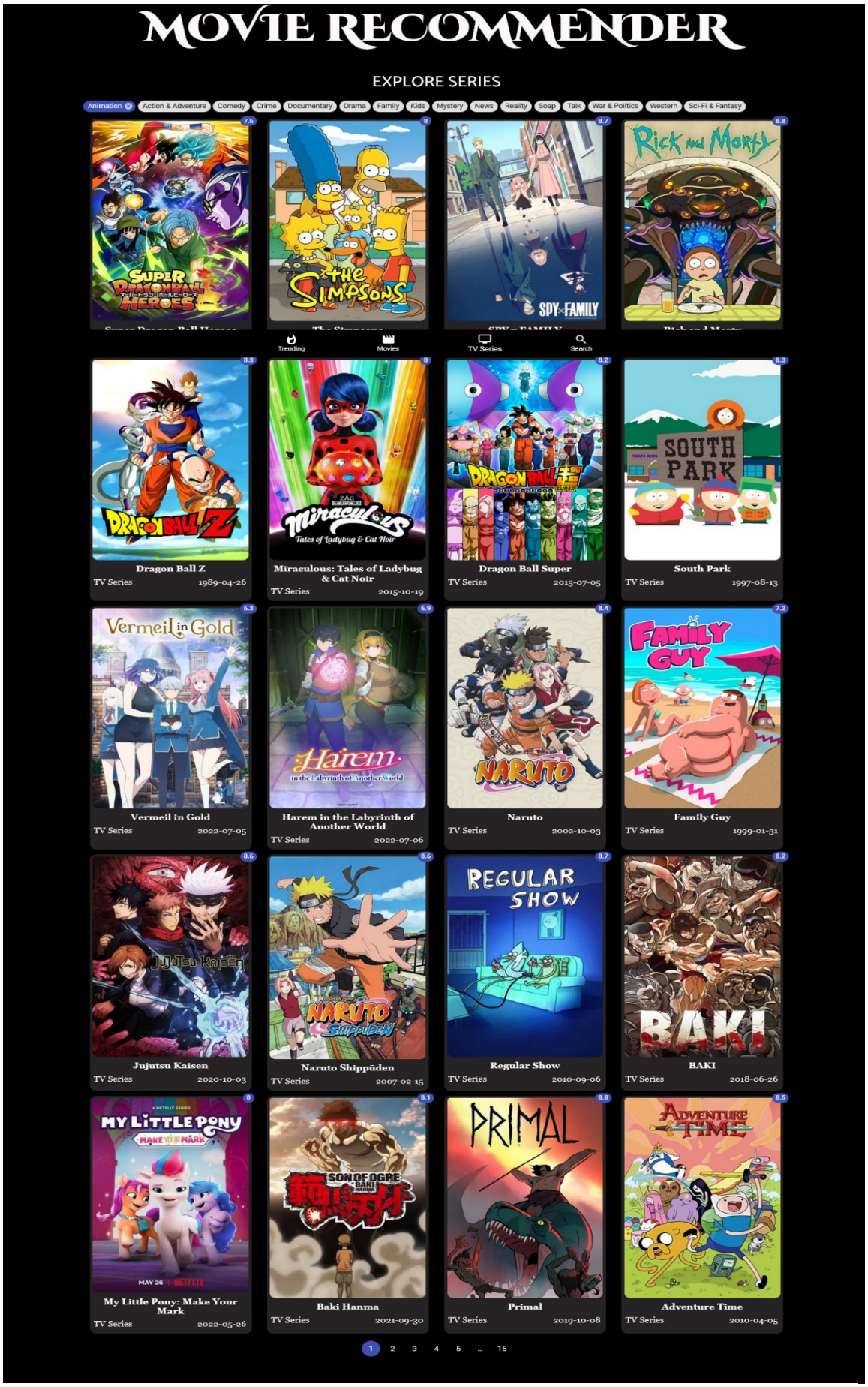
Movie page recommends the movies based on selected genre

SERIES PAGE



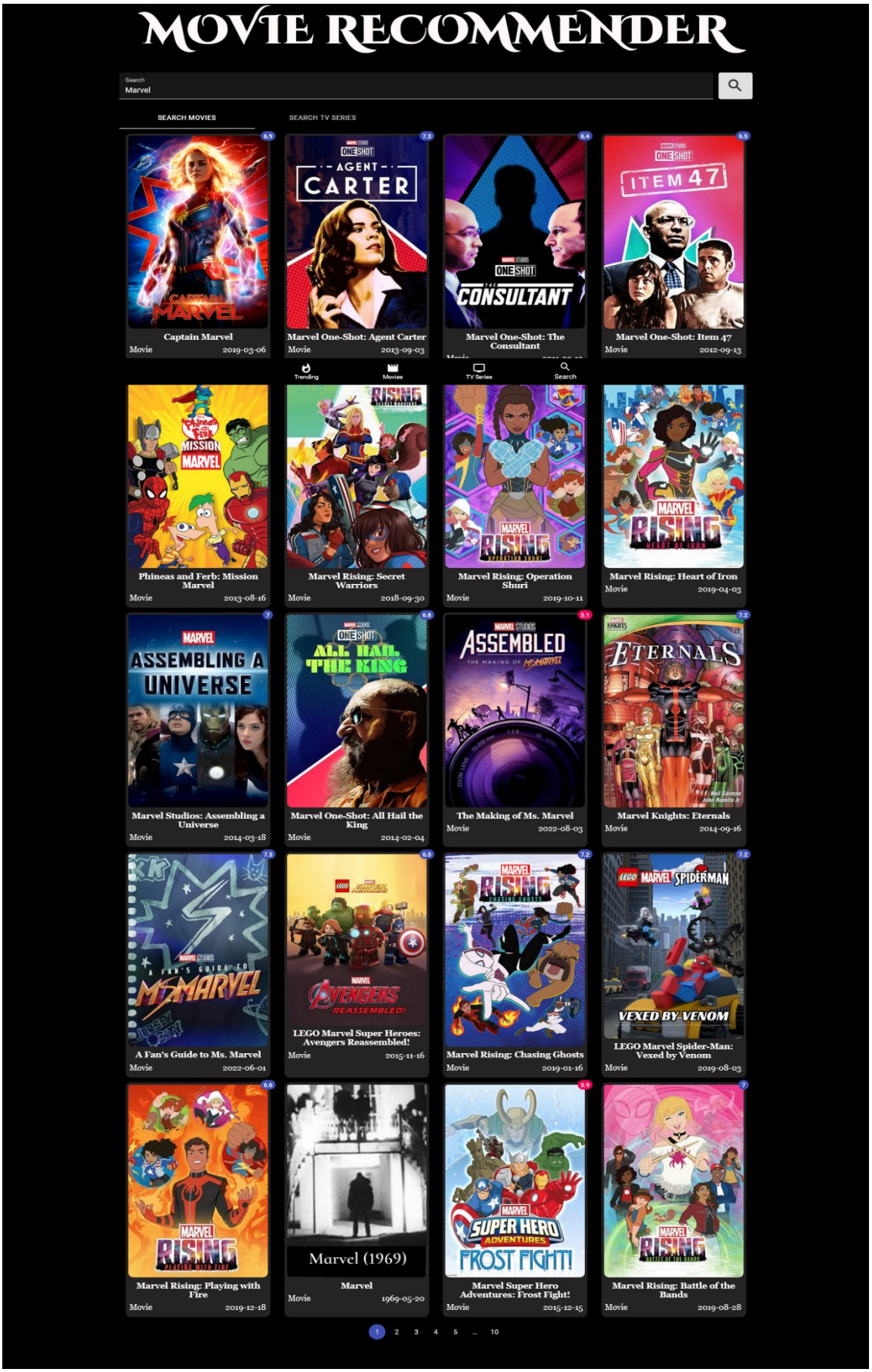
Series page displays the trending series

SERIES PAGE (with recommendation)



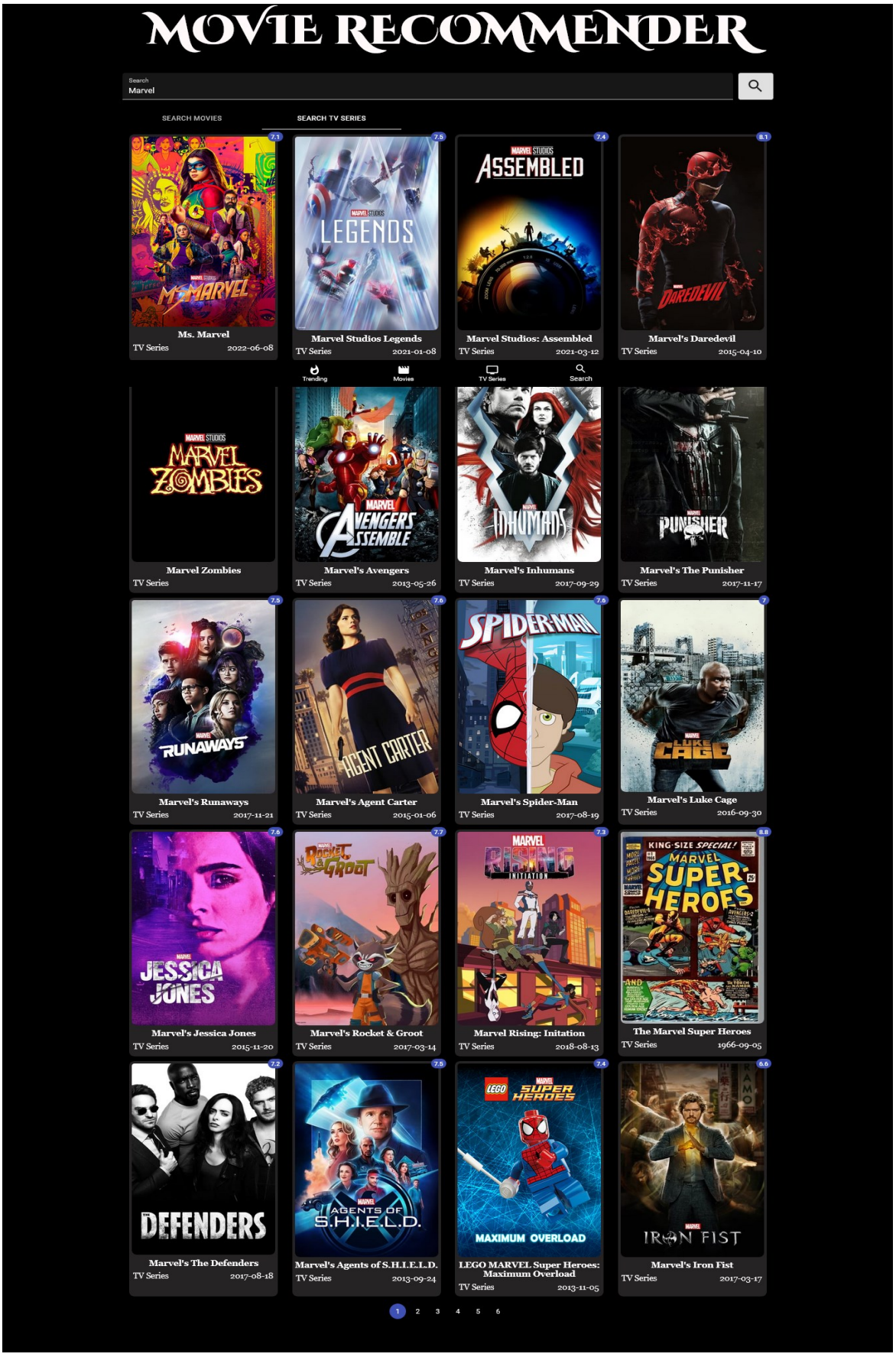
Movie page recommends the series based on selected genre

SEARCH PAGE [MOVIE]



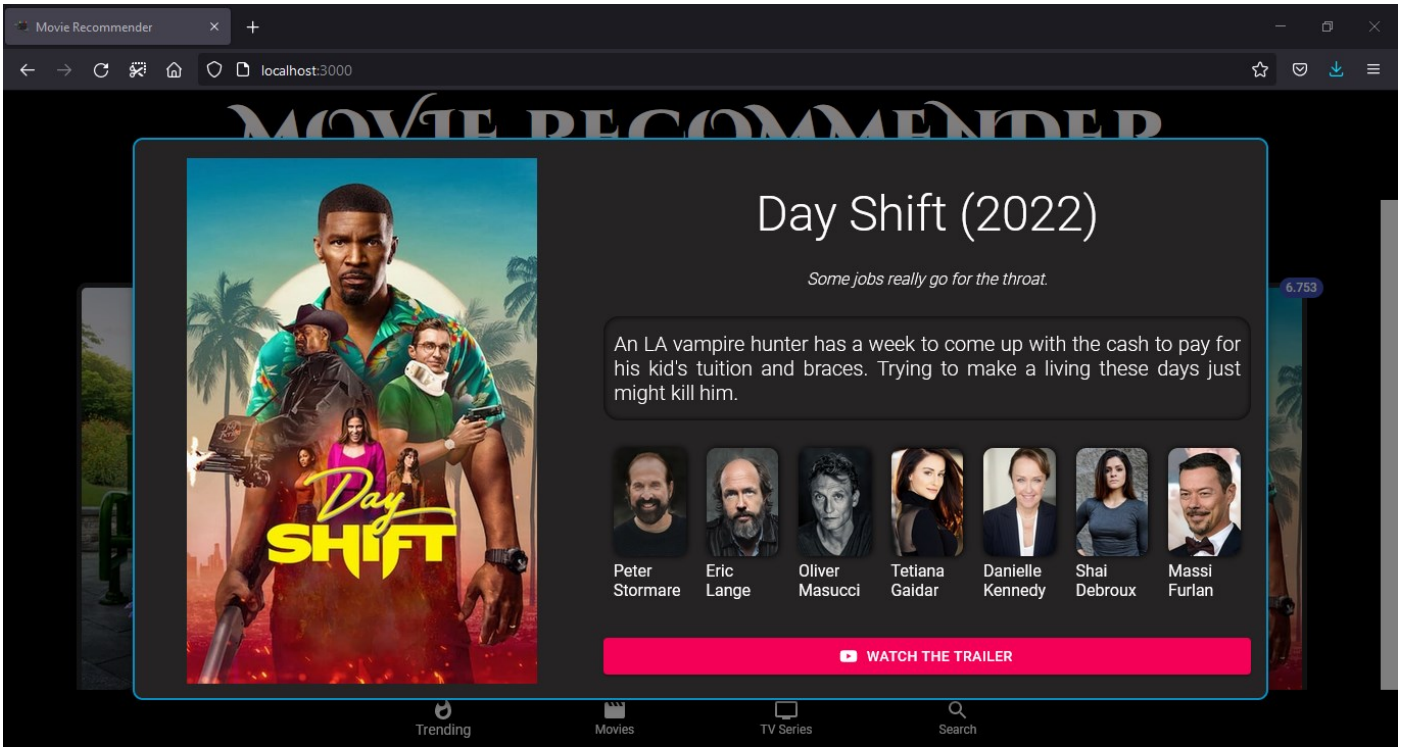
Search page displays the searched movies

SEARCH PAGE [SERIES]



Search page displays the searched series

DETAILED VIEW OF SELECTED CONTENT



Detail view displays the details of the selected movie/series

8 CONCLUSION

Our project is only a humble venture to satisfy the needs to manage their project work. Several user-friendly coding has also adopted. This package shall prove to be a powerful package in satisfying all the requirements of the college. The objective of software planning is to provide a framework that enables the user to create a blog with minimal efforts.

Recommender systems open new opportunities of retrieving personalized information on the Internet. It also helps to alleviate the problem of information overload which is a very common phenomenon with information retrieval systems and enables users to have access to products and services which are not readily available to users on the system. We come up with a strategy that focuses on dealing with user's personal interests and based on genres selected by the user. This strategy helps in improving accuracy of the recommendations.

At the end it is concluded that we have made effort on the following points: -

- A description of the background and context of the project and its relation to work already done in the area.
- We mentioned the technologies used in order to make this project.
- We understood the problems faced the by the user and tried to developed an application to eradicate the problem.
- We included features and operations in detail, including screen layouts.

Starting from requirements elicitation to design, construction, implementation and testing, I have gained a very good experience working with various technologies at every phase. Development of this project boosted my confidence in web development.

9 FUTURE ENHANCEMENTS

- ✓ We can give more advance software for recommender application including more facilities.
- ✓ We will host platform on online servers to make it accessible worldwide.
- ✓ Integrated multiple load balancer to distribute the loads of the system.
- ✓ Create the master and slave database structure to reduce the overload of the database queries.
- ✓ A section to provide a set of related movies and web series based on the movie or series selected by the user.
- ✓ Links to stream the selected movie or series via online steaming platforms like Netflix, Amazon Prime, Disney etc.
- ✓ Description of the cast starring on the selected content along with other movies and series in which they have starred in previously.
- ✓ A module to create accounts for the user in which a personal profile is created for each user, where each user has access to his own history, his likes, ratings, comments, password modification processes. It also helps in collecting authentic data with improved accuracy and makes the system more responsive.

The above-mentioned points are the enhancement which can be done to increase the applicable and usage of this project. Here we can maintain the records of blogs and comment. Also, as it can be seen now-a-days the players are versatile, i.e., so there is a scope for introducing a method to maintain all the blogs, comment and image.

10 BIBLOGRAPHY

https://reactjs.org/docs/getting-started.html

https://reactjs.org/docs/faq-internals.html

https://reactjs.org/docs/components-and-props.html

https://reactjs.org/docs/state-and-lifecycle.html

https://docs.aws.amazon.com/apigateway/latest/developerguide/http-api-vs-rest.html?pg=wianapi&cta=restapi

https://aws.amazon.com/what-is/api/#:~:text=API%20stands%20for%20Application%20Programming,other%20using%20requests%20and%20responses.

https://www.howtogeek.com/343877/what-is-an-api/

https://developers.themoviedb.org/3/getting-started/introduction

https://github.com/anubhavlal07/Entertainment-Hub (this entire project with documentation)