

MOVIE RECOMMENDATION SYSTEM

A MINI PROJECT REPORT

18CSC305J - ARTIFICIAL INTELLIGENCE

Submitted by

N MD ATAHAR HAJEE[RA2011003010892]

T CHARANDEEP REDDY[RA2011003010881]

Under the guidance of

DR.RAJALAKSHMI M

Assistant Professor, Department of Computer Science and Engineering

in partial fulfillment for the award of the degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE & ENGINEERING

of

FACULTY OF ENGINEERING AND TECHNOLOGY



S.R.M. Nagar, Kattankulathur, Chengalpattu District

MAY 2023

SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

(Under Section 3 of UGC Act, 1956)

BONAFIDE CERTIFICATE

Certified that Mini project report titled “**MOVIE RECOMMENDATIONS SYSTEM**” is the bonafide work of **N MD ATAHAR HAJEE[RA2011003010892] AND T CHARANDEEP REDDY[RA2011003010881]** who carried out the minor project under my supervision. Certified further, that to the best of my knowledge, the work reported herein does not form any other project report or dissertation on the basis of which a degree or award was conferred on an earlier occasion on this or any other candidate.

SIGNATURE

Dr. Rajalakshmi M
Assistant Professor
Department of Computing
Technologies

SIGNATURE

Dr.M.Pushpalatha
Head of the department
Professor and Head
Department of Computing Technologies

ABSTRACT

A movie recommendation is important in our social life due to its strength in providing enhanced entertainment. Such a system can suggest a set of movies to users based on their interest, or the popularities of the movies. Although, a set of movie recommendation systems have been proposed, most of these either cannot recommend a movie to the existing users efficiently or to a new user by any means. In this paper we propose a movie recommendation system that has the ability to recommend movies to a new user as well as the others. It mines movie databases to collect all the important information, such as, popularity and attractiveness, required for recommendation. It generates movie swarms not only convenient for movie producer to plan a new movie but also useful for movie recommendation. Experimental studies on the real data reveal the efficiency and effectiveness of the proposed system.

TABLE OF CONTENTS

ABSTRACT	iii
TABLE OF CONTENTS	iv
LIST OF FIGURES	v
ABBREVIATIONS	vi
1 INTRODUCTION	7
2 LITERATURE SURVEY	8
3 SYSTEM ARCHITECTURE AND DESIGN	9
Architecture diagram	9
technique used	9
4 METHODOLOGY	10
cosine similarity	10
5 DATASET	11
6 CODING AND TESTING	12
7 SREENSHOTS AND RESULTS	24
webpage	24
movie recommendations	25
8 CONCLUSION AND FUTURE ENHANCEMENT	26
Conclusion	26
Future Enhancement	26
REFERENCES	27

LIST OF FIGURES

3.1.1 Architecture block	9
4.1.1 cosine similarity	10
7.1.1 web page	
movie recommendations	25

ABBREVIATIONS

API	application program interface
IMDB	internet movie database
CF	Collaborative filtering
SLR	systematic literature review

CHAPTER 1

INTRODUCTION

Given the huge amount of movies are available all over the world, it is challenging for a user to find the appropriate movies suitable for his/her tastes. Different users like different movies or actors. It is important to find a method of filtering irrelevant movies and/or find a set of relevant movies.

Movie recommendation system is a process of exactly doing above tasks. Such a system has lot of implications and is inspired by the success of recommendation systems in different domains such as books, TV program, jokes, news articles . It is one of the most important research in the digital television domain.

The most well-known recommendation systems are mainly based on Collaborative Filtering (CF) and Content-based Filtering. CF first tries to find out the groups of similar users automatically from a set of active users. The similarities between users are computed using correlation measure. It then recommends items to a user based on the opinions of the users groups. Although CF is successful in many domains, however, it has shortcomings such as, sparsity and scalability . CF uses user ratings to find similar users. However, it is very difficult to find such since very few movies have ratings.

CHAPTER 2

LITERATURE SURVEY

The era of information and communication technology makes the information available on the internet growing rapidly. Recommender Systems are one of the technologies that are widely used to filter information to handle the huge of information. One of the developing information is film. The increasing number of films released every year has led to the development of applications that offer movie streaming services such as Netflix, Yiu, Disney Hotstar, etc. Therefore, movie recommender systems technology is needed to facilitate and provide a good experience when users use these services. The purpose of this study is to conduct a Systematic Literature Review (SLR) to analyze methods against the algorithm developed in building a movie recommender system. SLR method consists of three stages, namely, planning, conducting, and reporting processes. Studies published from 2010 to 2020 were considered. There were 21 main studies in which the collaborative filtering method was used in 16 studies, knowledge-based filtering was used in 2 studies, and hybrid filtering method was used in 3 studies. The results of the SLR process can be concluded that there are advantages and disadvantages to each method developed in building the movie recommender system. However, the model-based collaborative filtering method is one method that can minimize cold start, data sparsity, and scalability problems.

CHAPTER 3

SYSTEM ARCHITECTURE AND DESIGN

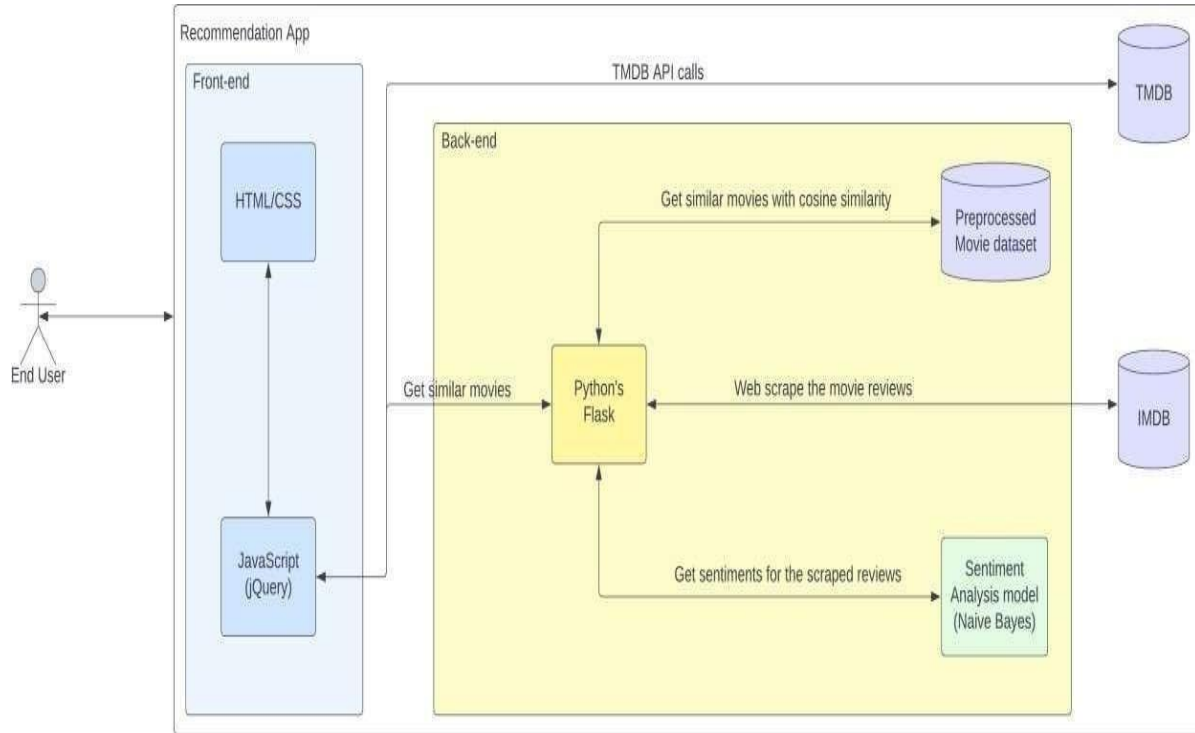


Fig: 3.1

TECHNIQUE USED

In data analysis, cosine similarity is a measure of similarity between two sequences of numbers. Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The cosine similarity is advantageous because even if the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

For example, in information retrieval and text mining, each word is assigned a different coordinate and a document is represented by the vector of the numbers of occurrences of each word in the document. Cosine similarity then gives a useful measure of how similar two documents are likely to be, in terms of their subject matter, and independently of the length of the documents.

CHAPTER 4

METHODOLOGY

SIMILARITY SCORE

It is a numerical value ranges between zero to one which helps to determine how much two items are similar to each other on a scale of zero to one. This similarity score is obtained measuring the similarity between the text details of both of the items. So, similarity score is the measure of similarity between given text details of two items. This can be done by cosine-similarity.

Cosine similarity is a metric used to measure how similar the documents are irrespective of their size. Mathematically, it measures the cosine of the angle between two vectors projected in a multi- dimensional space. The cosine similarity is advantageous because evenif the two similar documents are far apart by the Euclidean distance (due to the size of the document), chances are they may still be oriented closer together. The smaller the angle, higher the cosine similarity.

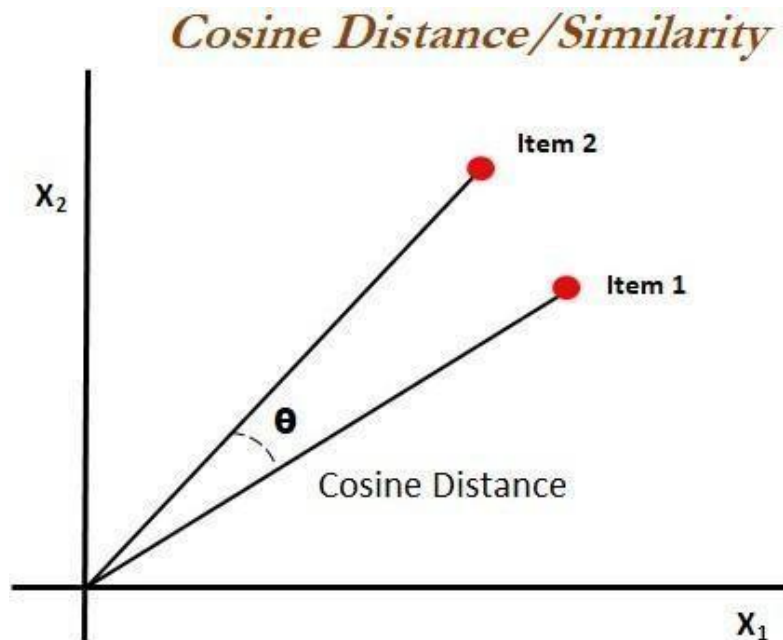
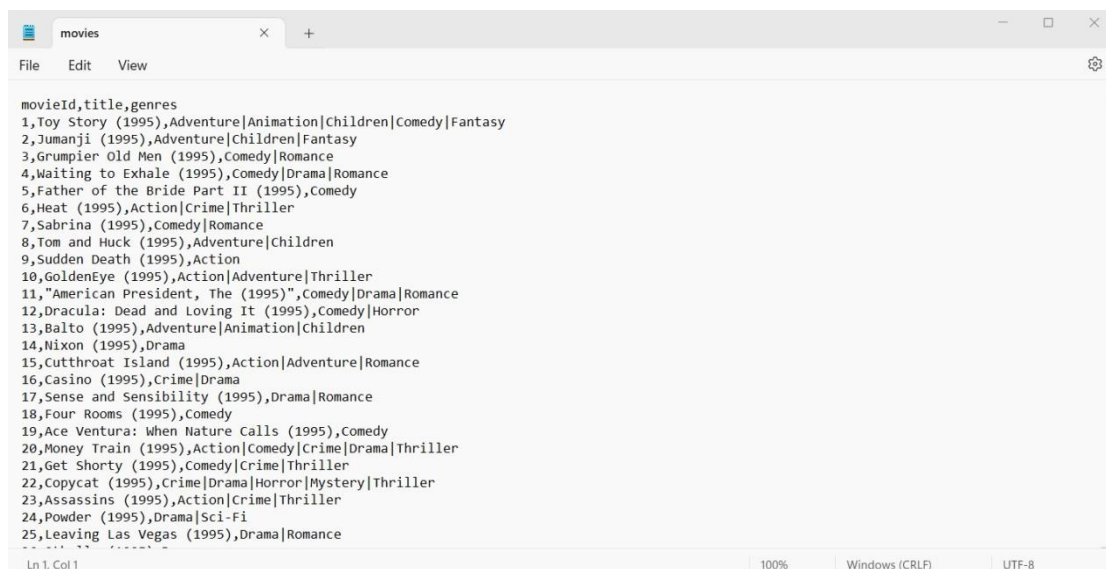


Fig: 4.1

CHAPTER 5

DATASET

- **MovieLens Dataset:** This is one of the most widely used datasets for movie recommendation systems. It contains movie ratings from users on a scale of 1 to 5, along with movie metadata such as title, genre, and year of release. It is available in different sizes ranging from 100,000 ratings to 25 million ratings.
- **IMDB Dataset:** The Internet Movie Database (IMDB) is a popular website that provides information on movies, TV shows, and celebrities. The IMDB dataset contains information on movies such as title, genre, rating, and cast.
- **Kaggle Dataset:** This dataset contains 100K data points of various movies and users.
- **Flixster Dataset:** Flixster is a movie recommendation website that provides information on movies, ratings, and reviews. The Flixster dataset contains information on movies such as title, genre, rating, and cast, along with user ratings and reviews.
- **Yahoo! Movies Dataset:** This dataset contains information on movies such as title, genre, rating, and cast, along with user ratings and reviews.
- **The Movies Dataset:** This dataset contains information on movies such as title, genre, rating, and cast, along with additional metadata such as budget, revenue, and runtime. It also includes links to movie trailers and posters.



```
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
6,Heat (1995),Action|Crime|Thriller
7,Sabrina (1995),Comedy|Romance
8,Tom and Huck (1995),Adventure|Children
9,Sudden Death (1995),Action
10,GoldenEye (1995),Action|Adventure|Thriller
11,"American President, The (1995)",Comedy|Drama|Romance
12,Dracula: Dead and Loving It (1995),Comedy|Horror
13,Balto (1995),Adventure|Animation|Children
14,Nixon (1995),Drama
15,Cutthroat Island (1995),Action|Adventure|Romance
16,Casino (1995),Crime|Drama
17,Sense and Sensibility (1995),Drama|Romance
18,Four Rooms (1995),Comedy
19,Ace Ventura: When Nature Calls (1995),Comedy
20,Money Train (1995),Action|Comedy|Crime|Drama|Thriller
21,Get Shorty (1995),Comedy|Crime|Thriller
22,Copcat (1995),Crime|Drama|Horror|Mystery|Thriller
23,Assassins (1995),Action|Crime|Thriller
24,Powder (1995),Drama|Sci-Fi
25,Leaving Las Vegas (1995),Drama|Romance
```

Fig: 5.1

CHAPTER 6

CODING AND TESTING

Index.html :

```
<!doctype html>
<html lang="en">
<head>

  <meta charset="utf-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <meta name="description" content="Movie Recommender System">
  <meta name="author" content="Martin Kondor">

  <title>Movie Recommender</title>

  <!-- Icons -->
  <link rel="icon" sizes="115x93" href="https://martinkondor.github.io/img/icon.bmp">
  <link rel="shortcut icon" sizes="115x93" href="https://martinkondor.github.io/img/icon.bmp">
  <link rel="apple-touch-icon" sizes="115x93" href="https://martinkondor.github.io/img/icon.bmp">
  <!-- / Icons -->

  <!-- Styles -->
  <link rel="stylesheet" href="public/css/libs/bootstrap-4.3.1.min.css">
  <link href="public/css/main.css?v=2022062" rel="stylesheet">
  <!-- / Styles -->

</head>
<body class="bg">

  <div id="loaderSign" class="text-center">
    <div class="spinner-border text-white" role="status" style="width: 5rem; height: 5rem;"></div>
    <br>
    <p class="font-weight-bold text-uppercase" style="font-size: 20px;">
      Loading...
    </p>
  </div>
  <div class="container-fluid text-center mt-1">
```

```

<h1 class="font-weight-bold h2 mt-4 mb-4">
  <a href="/" style="text-decoration: none; color: white;">🎬 Movie Recommender</a>
</h1>

<div id="box">
  <input id="title" type="text" class="form-control" placeholder="Type in a movie's title you liked
watching (e. g. cars, inception, titanic)" required="">
  <input id="year" type="text" class="form-control mt-2" placeholder="In what year it was released?
(Optional)" minlength="4" style="display: none;">
  <button id="submit" type="submit" class="btn btn-primary mt-2 btn-block text-
uppercase">Recommend Movies!</button>
  <hr>

<div id="box-list" class="mt-4">

  <p id="emptyText" class="text-muted mt-5">
    <i>
      The movies recommended for you will appear here...
    </i>
  </p>

  <h3 id="didYouMeanTitle" class="font-weight-bold mb-4">
    <div class="text-uppercase">
      Did you mean ...?
    </div>
    <p class="h5 text-muted">
      Choose the movie you've watched below.
    </p>
  </h3>

  <h3 id="weFoundNothing" class="text-uppercase font-weight-bold mb-4">
    We Found No Movies With This Title
  </h3>

  <h4 id="chosenMovie" class="mb-4">

  <div id="movie-list">
    <!--
    <div class="movie-li">
      1. Movie title
    </div>
    <div class="movie-li">
      2. Movie title
    </div>

```

```

        <div class="movie-li">
            3. Movie title
        </div>
    -->
</div>

</div>

<footer class="footer mt-5">
    <p class="text-muted" style="font-size: 12px;">
        This project is made by <a href="https://MartinKondor.github.io/">Martin Kondor</a>,
        see It's <a href="https://github.com/MartinKondor/MovieRecommender">GitHub</a> page
for more information.
    <br>
    Copyright &copy; Martin Kondor 2022
    </p>
</footer>
</div>
</div>
<!-- Scripts -->
<script src="https://code.jquery.com/jquery-3.6.0.min.js" integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script>
<script src="https://cdn.jsdelivr.net/npm/danfojs@1.1.0/lib/bundle.min.js"></script>
<script src="public/js/main.js?v=2022062"></script>
<!-- / Scripts -->

</body>
</html>

```

Main.css :

```

@charset "UTF-8";

html,
body {
    margin: 0;
    height: 100%;
    width: 100%;
    overflow-x: hidden;
    color: #fff;
    background: url('../img/andreas-gabler-XEW_Wd4240c-unsplash.jpg');
    background-size: cover;
    background-position: center;
}

```

```

label,
button {
    border-radius: 0px !important;
}

#box {
    width: 40%;
    min-height: 30%;
    padding: 40px 40px 5px 40px;
    box-shadow: 0px 5px 20px rgba(0, 0, 0, 0.222);
    border-radius: 10px;
    background-color: #fffffffb;
    margin: 0 auto;
    z-index: 1;
    color: #222222;
}
#weFoundNothing {
    display: none;
}
#didYouMeanTitle {
    display: none;
}

#choosenMovie {
    display: none;
}

#box-list {
    color: #000;
    text-shadow: none;
}

#loaderSign {
    position: absolute;
    margin: 0 auto;
    justify-content: center;
    justify-items: center;
    justify-self: center;
    background-color: rgba(0, 0, 0, 0.5);
    width: 100%;
    height: 100%;
    z-index: 2;
    padding-top: 150px;
}

```

```

.movie-li {
    margin: 20px 20px20px20px;
    border-style: dotted;
    padding: 20px;
    text-transform: uppercase;
}

.movie-li:hover {
    border-style: solid;
    cursor: pointer;
    transition: all 0.3s;
}

@media screen and (max-width: 1500px) {
    html,body {
        padding: 1px !important;
    }

    #box {
        width: 60% !important;
        padding: 20px 20px 5px 20px;
    }
}

@media screen and (max-width: 1200px) {
    html,body {
        padding: 0px !important;
    }

    #box {
        width: 95% !important;
    }
}

.bg {
    width: 100%;
    height: 100vh;
    display: flex;
    background-size: 300% 300%;
    background-image: linear-gradient(-45deg, #3493e6b2 0%, #EC6EADb2 100%);
    -webkit-animation: AnimateBG 10s ease infinite;
    animation: AnimateBG 10s ease infinite;
}

```



```
@-webkit-keyframes AnimateBG {
  0% {
    background-position: 0% 50%;
  }

  50% {
    background-position: 100% 50%;
  }
  100% {
    background-position: 0% 50%;
  }
}
```

```
@keyframes AnimateBG {
  0% {
    background-position: 0% 50%;
  }
  50% {
    background-position: 100% 50%;
  }
  100% {
    background-position: 0% 50%;
  }
}
```

Main.js :

```
'use strict';
(function ($) {
  // Prepare data
  function resetUI() {
    $("#emptyText").css("display", "none");
    $("#weFoundNothing").css("display", "none");
    $("#didYouMeanTitle").css("display", "none");
    $("#chosenMovie").css("display", "none");
    $("#loaderSign").hide();
    $("#movie-list").html("");
  }

  // Download data
  function load(callback) {
    resetUI();
    $("#loaderSign").show();
  }
}
```

```

$.getJSON("https://martinkondor.github.io/MovieRecommender/data/json/movies.json", function
(moviesData) {
  $.getJSON("https://martinkondor.github.io/MovieRecommender/data/json/ratings.json", function
(ratingsData) {
    let movies = new dfd.DataFrame(moviesData);
    let ratings = new dfd.DataFrame(ratingsData);
    let df = dfd.merge({ left: movies, right: ratings, on: ["movieId"], how: "inner" })
    callback(df);
  });
});
}

```

```

function getMovieNameWords(title, callback) {
  let movieNameWords = [];
  let words = title.toLowerCase().trim().split(" ");
  for (let word of words) {
    if (word == "the") continue;
    movieNameWords.push(word);
  }
  callback(movieNameWords);
}

```

```

function findSimilarMovies(df, title) {
  let recommendedMovies = [];
  df.sortValues("rating", { inplace: true, ascending: false });

  // let movies = df.iloc({ rows: ["0:5"] });
  // console.log(movies.values);

  // Get all ratings of the movie
  let ratings = [];
  for (let m of df.values) {
    if (m[1] != title) continue;
    ratings.push(m);
  }
  ratings = ratings.sort(function (a, b) {
    return a[4] < b[4];
  });

  // Get the top 5 ratings
  ratings = ratings.slice(0, 5);

  for (let rating of ratings) {
    let userid = rating[3];

```

```

// Find movies rated by this user
let userMovies = [];
for (let m of df.values) {
  if (m[3] !== userid || m[1] !== title) continue;
  userMovies.push(m);
}

// Weight movies about the similarity of genres
let scores = [];
let genres = ratings[0][2].split("|");

for (let m of userMovies) {
  let umGenres = m[2].split("|");
  let score = 0;

  for (let umg of umGenres) {
    if (genres.includes(umg)) {
      score += 1;
    }
  }

  scores.push([m, score]);
}

// Sort by genre similarity
scores = scores.sort(function (a, b) {
  return a[1] < b[1];
}).slice(0, 5);

userMovies = [];
for (let m of scores) {
  userMovies.push(m[0]);
}

userMovies = userMovies.sort(function (a, b) {
  return a[4] < b[4];
}).slice(0, 5);

for (let m of userMovies) {
  if (recommendedMovies.includes(m)) continue;
  recommendedMovies.push(m);
}
}

```

```

// Adding recommendedMovies to html
let alreadyUsedMovieTitles = [];
let index = 0;

for (let m of recommendedMovies) {
  if (alreadyUsedMovieTitles.includes(m[1])) continue;
  alreadyUsedMovieTitles.push(m[1]);

  $("#movie-list").html($("#movie-list").html() + `
    <div id="movie-${index}" class="movie-li" title="${m[1]}">
      ${m[1]}
    </div>
  `);

  // Recommend similar to the chosen movie
  // $("#movie-list").on("click", `#movie-${index}`, createMovieCallback(df, index, m[1]));
  $("#movie-list").on("click", `#movie-${index}`, function () {}); // Remove old listeners
  $(`#movie-${index}`).css("pointer-events", "none");
  $(`#movie-${index}`).css("cursor", "pointer");

  index++;
}

$("#title").val(title);
$("#year").val("");
$("#loaderSign").hide();
}

function createMovieCallback(df, index, title) {
  return function() {
    resetUI();
    $("#loaderSign").show();

    $("#chosenMovie").html("If you like <strong>" + title + "</strong> you might like these");
    $("#chosenMovie").css("display", "block");

    findSimilarMovies(df, title);
  }
}

function findMovie(df, title, callback) {
  getMovieNameWords(title, function (movieNameWords) {
    let similarTitles = [];
    let similarTitlesWithPoints = [];

```

```

for (let dfTitle of df.iloc({columns: [1]}).values) {
  dfTitle = dfTitle[0];
  let wordsOfDfTitle = []
  let wordsOfDfTitleTemp = dfTitle.toLowerCase().trim().split(" ");

  for (let w of wordsOfDfTitleTemp) {
    if (w.trim() == "the") continue;
    if (w.trim()[0] == "(") continue;
    wordsOfDfTitle.push(w.trim());
  }

  let points = 0;

  for (let word of wordsOfDfTitle) {
    if (movieNameWords.includes(word)) {
      points += 1;
    }
  }

  if (points != 0 && !similarTitles.includes(dfTitle)) {
    similarTitles.push(dfTitle);
    similarTitlesWithPoints.push([dfTitle, points]);
  }
}

similarTitlesWithPoints = similarTitlesWithPoints.sort(function (a, b) {
  return a[1] < b[1];
});

if (similarTitlesWithPoints.length == 0) {
  $("#weFoundNothing").css("display", "block");
  return;
}

$("#didYouMeanTitle").css("display", "block");
let index = 0;

for (let m of similarTitlesWithPoints) {
  $("#movie-list").html($("#movie-list").html() + `
    <div id="movie-${index}" class="movie-li">
      ${m[0]}
    </div>
  `);
}

```

```

// On Click for each movie
    $("#movie-list").on("click", `#movie-${index}`, createMovieCallback(df, index, m[0]));
    index++;
}

    callback();
});
}
// / Prepare data

function search(title) {
    load(function (df) {
        findMovie(df, title, function () {
            $("#loaderSign").hide();
        });
    });
}

function searchWithYear(title, year) {
    load(function (df) {

    });
}

function isYear(val){
    return !isNaN(val) &&val.length === 4;
}

$("#submit").on("click", function () {
    let title = $("#title").val();
    let year = $("#year").val() || "";

    // Running criterias
    if (title.length < 1) {
        /// TODO: Error messages
        return;
    }
    if (year.length !== 0 && !isYear(year)) {
        /// TODO: Error messages
        return;
    }

    // Convert year if given to int
    let results = [];
    if (year.length !== 0) {
        year = parseInt(year);
    }

```

```
        results = searchWithYear(title, year);
    }
    else {
        results = search(title);
    }

    console.log(results);

});

resetUI();
$("#emptyText").css("display", "block");

})(jQuery);
```

CHAPTER 7

SCREENSHOTS AND RESULTS

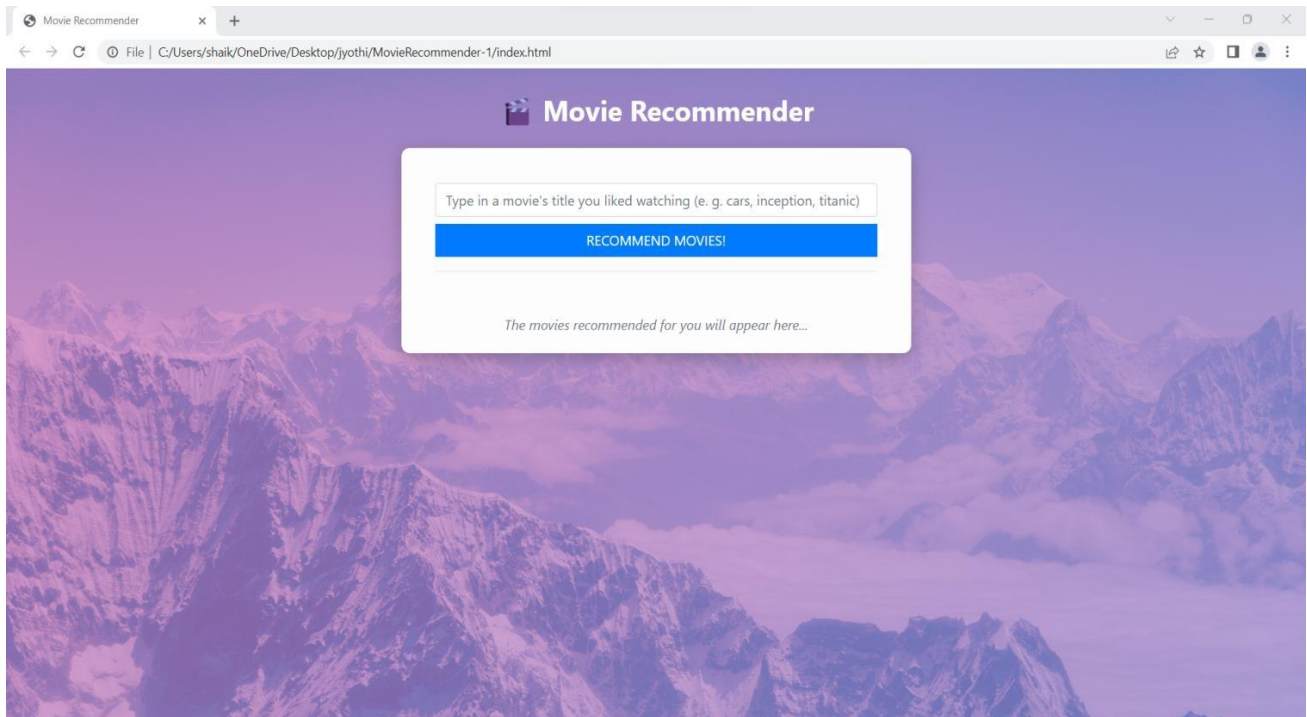


Fig: 7.1

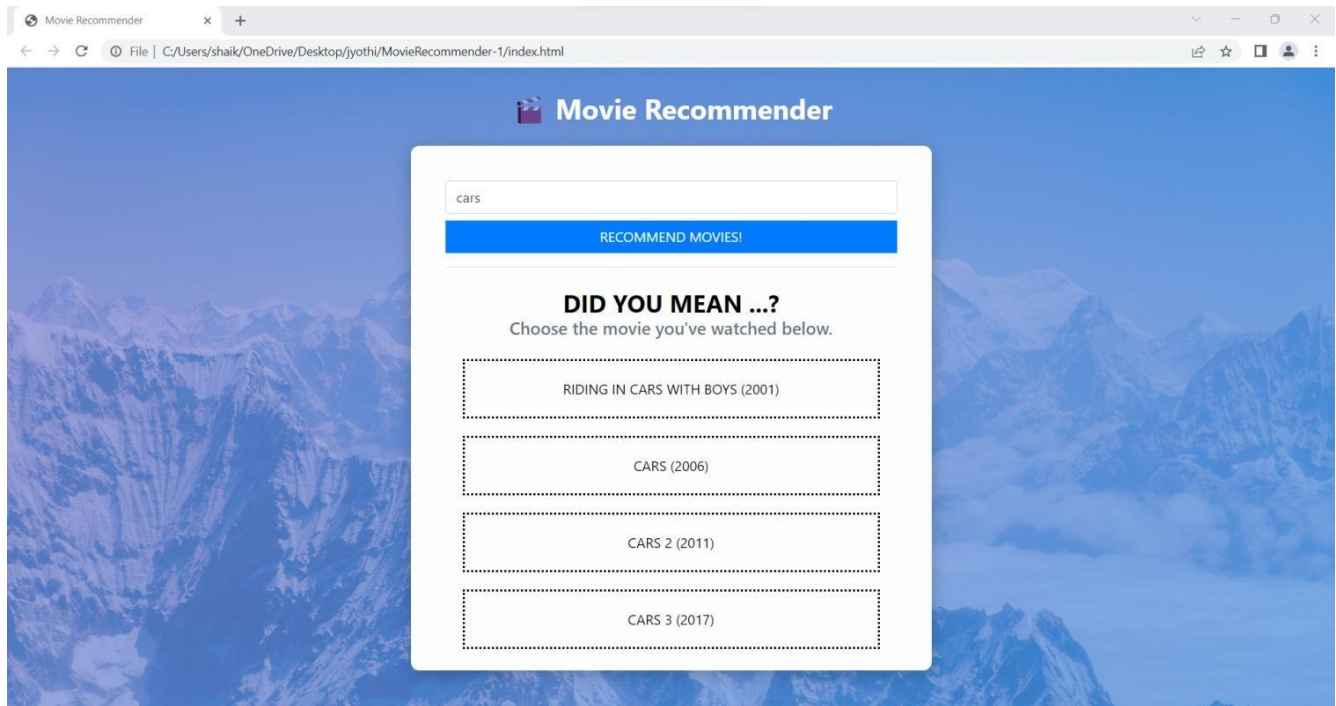


Fig: 7.2

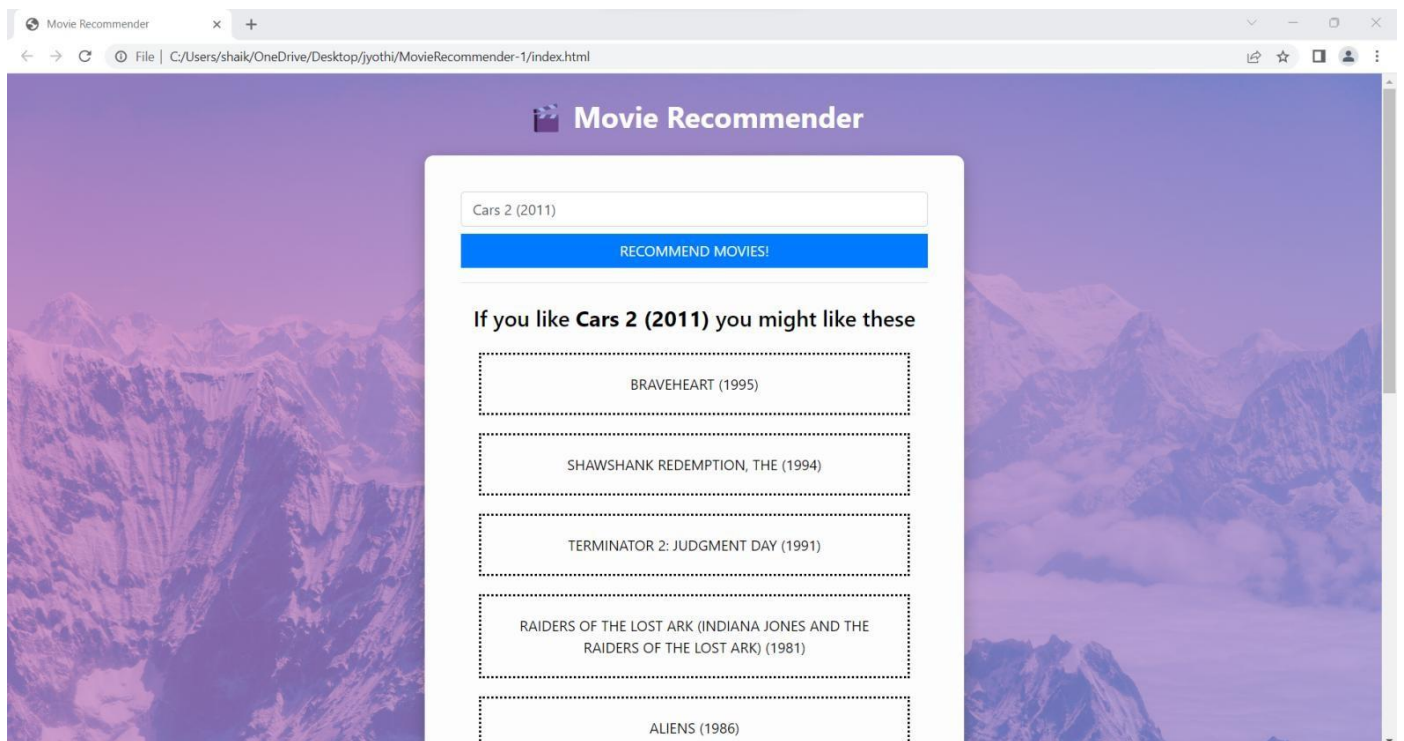


Fig: 7.2

CHAPTER 8

CONCLUSION AND FUTURE ENHANCEMENTS

A movie recommendation system is built using cosine similarity in artificial intelligence. Cosine similarity is a metric used to measure how similar the documents are irrespective of their size, here data of several movies and reviews are collected with the help of API's and are compared. The recommendations are given based on similarity and cast. We can use the movie recommendation system for recommendations of shows worth watching based on past interests.

REFERENCES

1. <https://www.kaggle.com/>
2. **Pratap Dangeti** “Statistics for Machine Learning [Book]”
3. <https://www.irjet.net/archives/V7/i9/IRJET-V7I9633.pdf>
4. <https://www.ijraset.com/research-paper/paper-on-movie-recommendation-system>
5. M. Kumar, D. K. Yadav, A. Singh, and Y. Kr., International Journal Computer Applications, 7–11 (2015).