A MINI PROJECT REPORT

ON

MOVIES RECOMMENDATION SYSTEM

Submitted to Mumbai University

In the partial fulfillment of the requirement for the award of the degree of

Bachelor of Engineering

In

COMPUTER ENGINEERING

By

Mr. Shaikh Manzar Jamal(16CO50)
Mr. Poonawala Ayan Ishtiaque(16CO44)
Mr. Mohd Zeeshan Abbas (16CO41)

Under the guidance of

Mrs. Kumari Rajesh

Professor

Department of Computer Engineering

Anjuman-I-Islam's Kalsekar Technical Campus

Affiliated to Mumbai University

KHANDA GOAN, NEW PANVEL, NAVI MUMBAI, MAHARASHTRA

2018-2019

Department of Computer Engineering

Anjuman-I-Islam's Kalsekar Technical Campus

Affiliated to Mumbai University

KHANDA GOAN, NEW PANVEL, NAVI MUMBAI, MAHARASHTRA

2018-2019

DECLARATION BY THE CANDIDATE

Shaikh Manzar, poonawala Ayan, Mohd Zeeshan bearing **Roll number: 16CO50, 16CO4416CO41**, hereby declare that the mini project report entitled **"Movies Recommender System"**, is a record of bonafide work carried out by me and the results embodied in this project have not been reproduced or copied from any source. The results of this project report have not been submitted to any other University or Institute for the award of any other Degree or Diploma.

Shaikh Manzar(16CO50)

Poonawala Ayan(16CO44)

Mohd Zeeshan(16Co41)

Department of Computer Engineering

Anjuman-I-Islam's Kalsekar Technical Campus

Affiliated to Mumbai University

KHANDA GOAN, NEW PANVEL, NAVI MUMBAI, MAHARASHTRA

2018-2019

CERTIFICATE

This is to certify that the project report entitled **"Movies Recommender System"**, submitted by **Mr. Mohd Zeeshan** , bearing **Roll. No.: 16CO41** in the partial fulfillment of the requirements for the award of the degree of **Bachelor of Computer Engineering** is a record of bonafide work carried out by him/her for the course **Mini Project CSP605**.

Mini project Guide

(Prof. Kumari Rajesh)

Mini Project Coordinator

(Prof. Muhammed Salman Shamsi)

Program Owner

(Prof. Tabrez Khan)

INDEX

CONTENTS

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

A recommendation system is a type of information filtering system which attempts to predict the preferences of a user, and make suggests based on these preferences.Recommender System is a system that seeks to predict or filter preferences according to the user's choices. Recommender systems are utilized in a variety of areas including movies, music, news, books, research articles, search queries, social tags, and products in general. Movies Recommendation systems imitate this social process to enable quick filtering of the information on the web about movies.

The purpose of a recommendation system is to predict a rating that a user will give to an item that they have not yet rated.

This rating is produced by analyzing either item characteristics or other user/item ratings (or both) to provide personalized recommendations to users.

There are 2 main approaches to recommendation systems:

- Content Filtering. Recommendations depend on item characteristics.
- Collaborative Filtering. Recommendations depend on user-item ratings.

1.1.1 Collaborative filtering

Collaborative filtering system recommends items based on similarity measures between users and/or items. The system recommends those items that are preferred by similar kind of users.

1.1.1.1 Collaborative filtering has many advantages :

- It is content-independent i.e. it relies on connections only
- Since in CF people makes explicit ratings so real quality assessment of items are done.
- It provides serendipitous recommendations because recommendations are base on user's similarity rather than item's similarity.

1.1.2 Content-based filtering

Content-based filtering is based on the profile of the user's preference and the item's description. In CBF to describe items we use keywords apart from user's profile to indicate user's preferred liked or dislikes. In other words CBF algorithms recommend those items or similar to those items that were liked in the past. It examines previously rated items and recommends best matching item. There are various approaches proposed in various research papers listed below. These approaches are often combined in Hybrid Recommender Systems.Al for the recommendation of movies through MOVIEGEN had certain drawbacks such as , it asks a series of questions to users which was time taking . On the other hand it was not user friendly for the fact that it proved to be stressful to a certain extent. Keeping in mind these shortcomings, we have developed Movie REC, a movie recommendation system that recommends movies to users based on the information provided by the users themselves. In the present study, a user is given the option to select his choices from a set of attributes which include actor, director, genre, year and rating etc. We predict the users choices based on the choices of the previous visited history of users. The system has been developed in PHP and currently uses a simple console based interface.

1.2 SCOPE

The targeted software product is a Movies Recommendation System. The system makes an extensive use of user data to come up with reasonable track predictions about user preferences and there are two main paradigms in this sense: Content-based and collaborative filtering. As a beginning, the scope of the system is scaled down to collaborative filtering. This type of filtering is based on collecting and analyzing huge amount of user information; users' behaviors, activities, preferences and similarities to other users. The scope consists of determining similar users according to the similarities between their evaluations of certain track attributes. The system is aimed to have additional solutions for big data and sparse information. The test data will be provided from Argedor. The test data for the system are planned to contain millions of users and irregular data such that some user-track relation remains unknown. The project limits the improvement of the recommendation system around these challenges and favors accuracy over performance while giving the final suggestions. As a milestone, the system should follow the methods to enhance accuracy, the performance enhancements are not included in the scope. The Movies Recommender System will be a Web Service and the end users will be directly exposed to the suggestions made for them.

1.3 PROBLEM STATEMENT

Providing related content out of relevant and irrelevant collection of items to users of online service providers OMRES (Online Movie Recommendation System) aims to recommend movies to users based on user-movie (item) ratings.

Given a set of users with their previous ratings for a set of movies, can we predict the rating they will assign to a movie they have not previously rated?

• Ex.“Which movie will you like” given that you have seen X-Men, X-Men II, X-Men : The Last Stand and users who saw these movies also liked “X-Men Origins:Wolverine”?

CHAPTER 2

SYSTEM SPECIFICATION

2.1 SYSTEM REQUIREMENT

HARDWARE REQUIREMENT

Name of Component	Specification
PROCESSOR:	INTEL core i3,i5,i7
RAM:	1GB
HDD:	20GB

SOFTWARE REQUIREMENT

Name of Component	Specification
OPERATING SYSTEM	LINUX**,**WINDOWS(Version 7or Up to)
Language:	Python3
Software:	Anaconda – Jupyter
Database:	MySQL server
Browser:	Mozilla**,** chrome etc.
Web server:	XAMP**,** LAMP
Libaries	Scikit-learn, Scikit-Surprise

2.2 SYSTEM FEATURES

- Responsive and Intuitive User Interface.
- Comprehensive user experience.
- More accurate model.
- User Friendly-Easy to use.
- Consumes less time for classifications.

CHAPTER 3

SYSTEM DESIGN

3.1 SYSTEM ARCHITECTURE

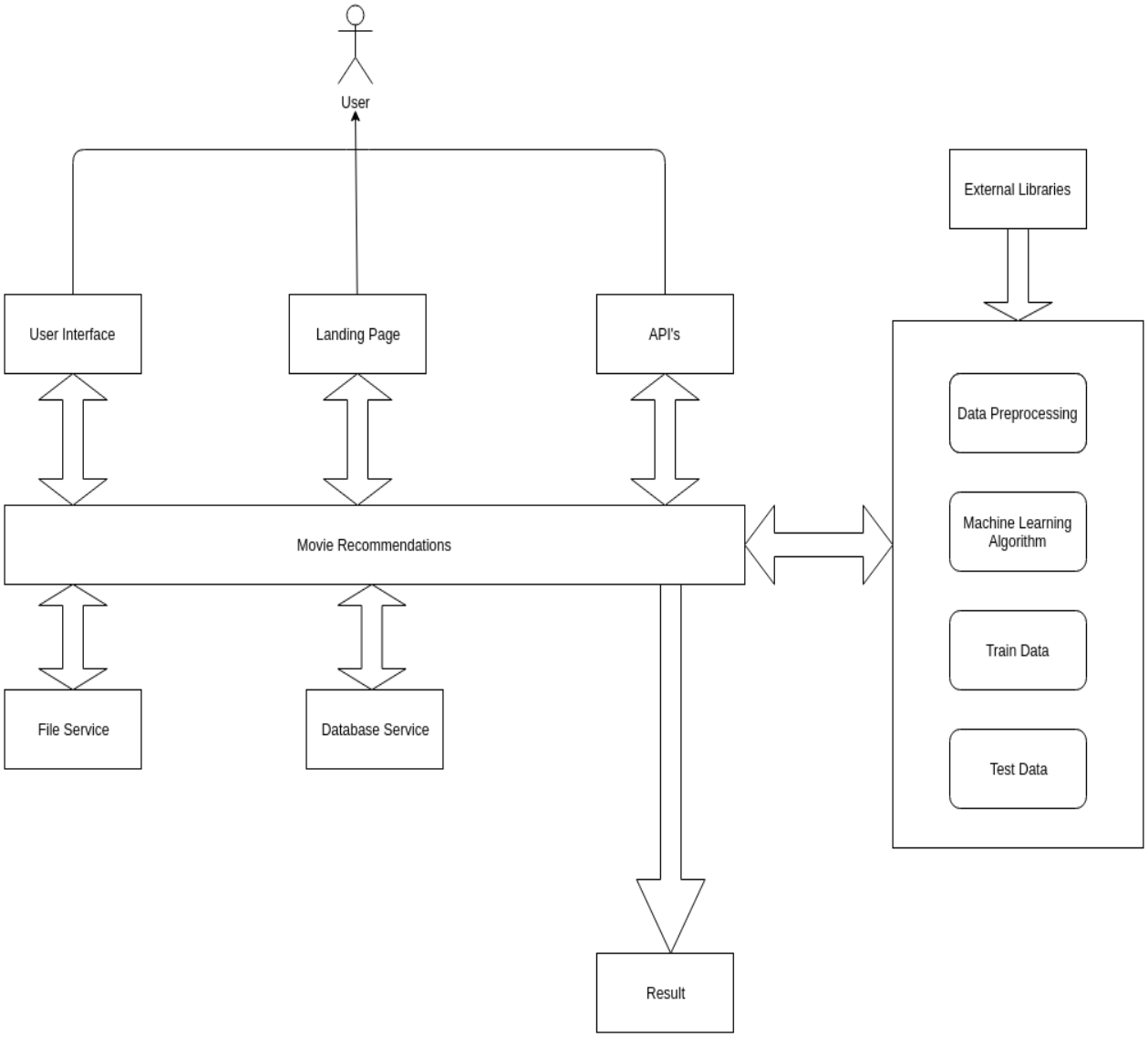


fig 1: System architecture for Movies Recommender System.

3.2 SYSTEM MODULE

- **User Interface:** The User Interface is in the form of a Web Application.The user interface is design using HTML5, CSS3, Javascript and Bootstrap for Responsiveness.The Interface shows the movies images and details which are fetched from Movie Recommender System. System also display 8 Movies based on User History.
- **Data Preprocessing:** In this module,we have large amount of data this data, Contain many null value such null value will be removed.
- **Login Module :** Used for managing the login details.It will take Username and password for Login.
- **Register Module :** For a new User, This module will take username,password ,Email Id and create a New DataBase Entry.
- **DataBase Module :** This module is used to store Data regarding User Credentials **.**
- **Movies Recommendation :** Here main task of Recommendation Happen,Based on the previous history of user the module extract required Data Such as Genre of Movies He has rated and rating.Based of this the algorithm Recommend similar movies with matching genres to The login User.
- **User Module :**Used for managing the users of the system.
- **The Main Algorithm we used here is KNN (K- nearest Neighbour)**

K-Nearest Algorithm:

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique

Algorithm:

A case is classified by a majority vote of its neighbors, with the case being assigned to the class most common amongst its K nearest neighbors measured by a distance function. If K = 1, then the case is simply assigned to the class of its nearest neighbour.

Distance functions

Euclidean $\sqrt{\sum_{i=1}^{k}(x_i - y_i)^2}$

It should also be noted that the distance measures are only valid for continuous variables. In the instance of categorical variables the Hamming distance must be used. It also brings up the issue of standardization of the numerical variables between 0 and 1 when there is a mixture of numerical and categorical variables in the dataset.

Hamming Distance

$$D_H = \sum_{i=1}^{k} |x_i - y_i|$$

$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Choosing the optimal value for K is best done by first inspecting the data. In general, a large K value is more precise as it reduces the overall noise but there is no guarantee.

3.3 USE CASE DIAGRAM

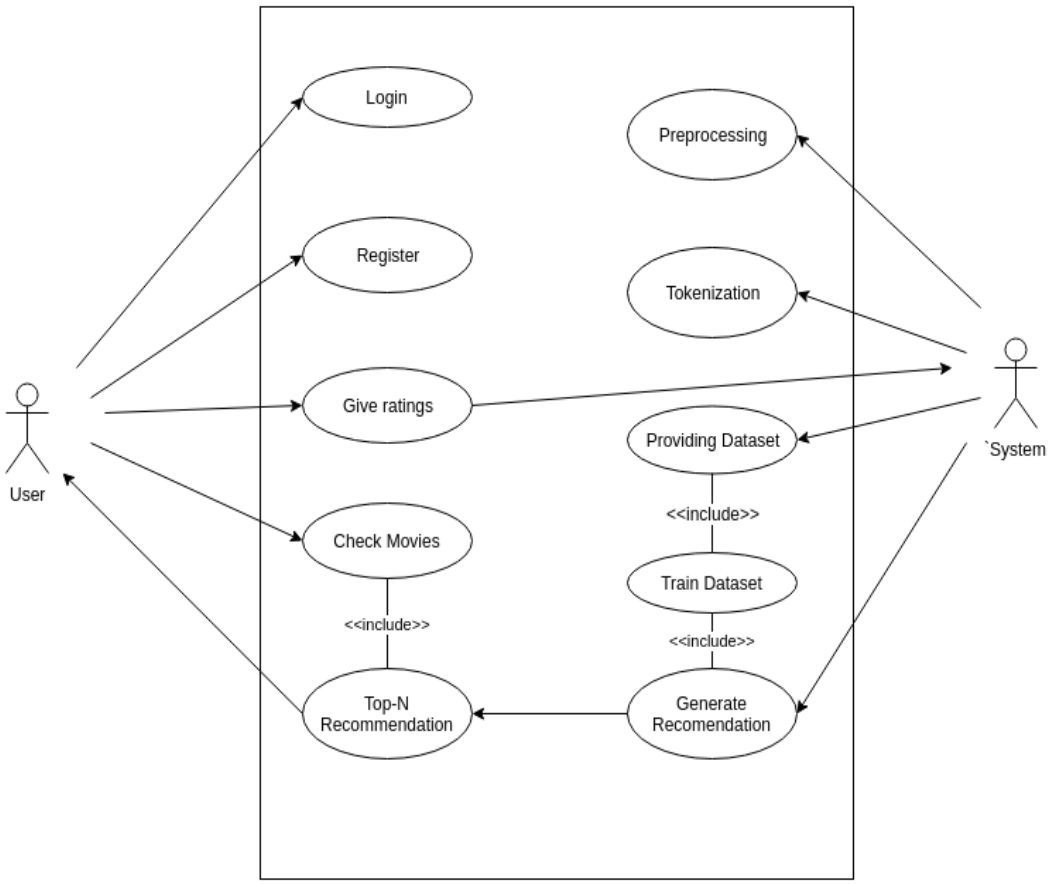


Fig 2: use case diagram for Movies Recommender System

3.4 Activity Sequence Diagram

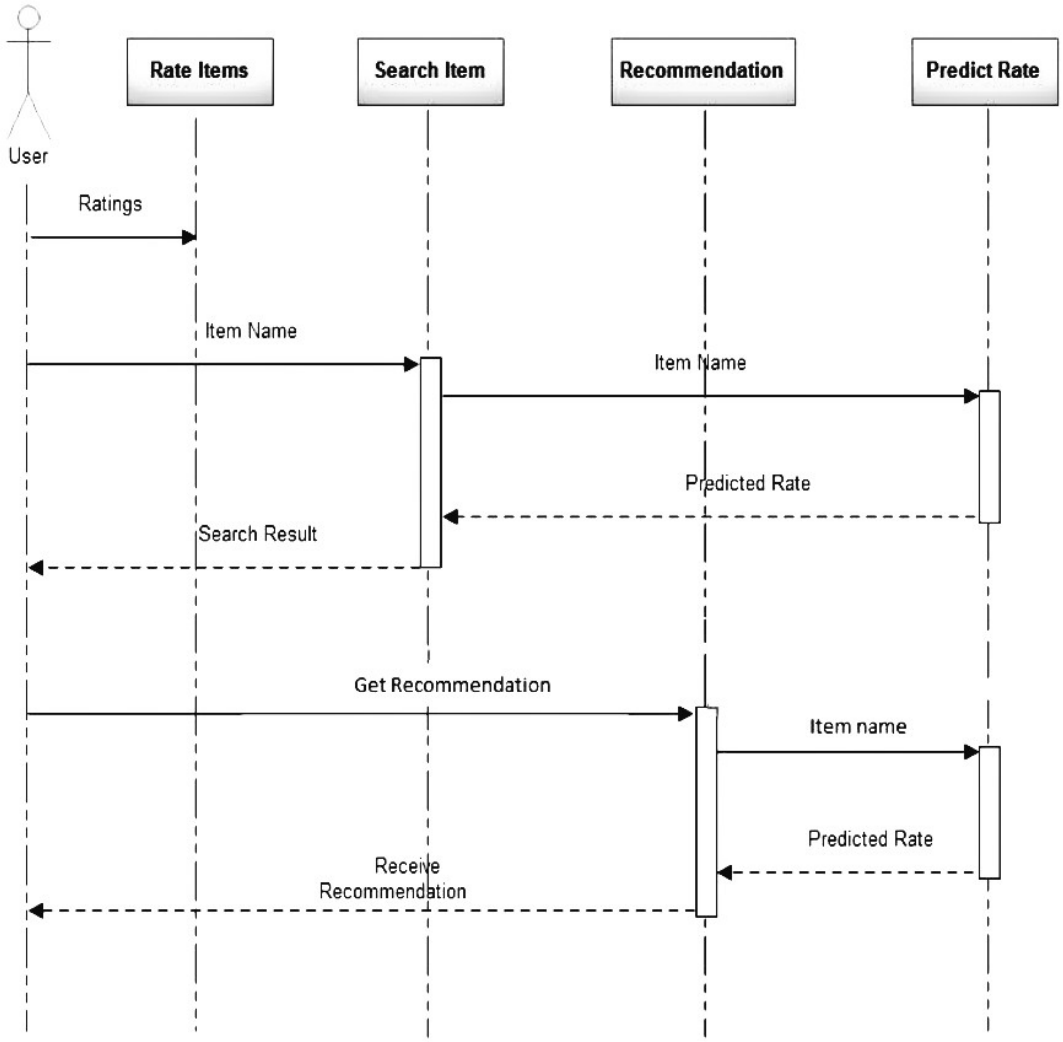


Fig 3: use case diagram for Movies Recommender System.

CHAPTER 4

IMPLEMENTATION

4.1 CODE SNIPPETS

4.1.1 TEMPLATES

4.1.1.1 Index.html

```
{% load staticfiles %}
<!DOCTYPE HTML>
<html>
  <head>
    <title>Movies Now: Home</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
     <meta name="keywords" content="Movie_store Responsive web template, Bootstrap Web Templates, Flat Web Templates, Andriod Compatible web template,
        Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyErricsson, Motorola web design" />
    <script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>
    <link href='{% static "css/bootstrap.css" %}' rel='stylesheet' type='text/css' />
    <link href='{% static "css/style.css" %}' rel="stylesheet" type="text/css" media="all" />
    <!-- start plugins -->
    <script type="text/javascript" src='{% static "js/jquery-1.11.1.min.js" %}'></script>
                                                    <link href='http://fonts.googleapis.com/css?family=Roboto+Condensed:100,200,300,400,500,600,700,800,900' rel='stylesheet' type='text/css'>
    <script src='{% static "js/responsiveslides.min.js" %}'></script>
    <script>
      $(function () {
       $("#slider").responsiveSlides({
        auto: true,
        nav: true,
        speed: 500,
         namespace: "callbacks",
         pager: true,
       });
      });
    </script>
    <style type="text/css">
      .m_3{
      background-color: #ED2B2D;
      text-align: center;
      color: green;
      font-family: "HelveticaNeue", "Helvetica Neue", Helvetica, Arial, "Lucida Grande", sans-serif;
      }
      .message_ticker:hover {
      -webkit-transition: 4s ease-in;
      -moz-transition: 4s ease-in;
      -o-transition: 4s ease-in;
      transition: 4s ease-in;
      text-indent: -300px;
      }
      .header {
```

```
        width:100%
        margin:0 auto;
        background: white;
        }
        .message_ticker {
        width: 100%;
        height:50px;
        overflow:hidden;
        float:left;
        border-right:1px solid #ED2B2D;
        -webkit-transition: 0.5s ease-in;
        -moz-transition: 0.5s ease-in;
        -o-transition: 0.5s ease-in;
        transition: 0.5 s ease-in;
        }
        .img2{
        width: 100%;
        }
    </style>
  </head>
  <body>
    <div class="container">
      <div class="container_wrap">
        <div class="header_top">
             <div class="col-sm-3 logo"><a href="index.html"><img src="{% static 'images/logos.png' %}"
alt=""/></a></div>
          <div class="col-sm-3 header_right navbar-right">
            <ul class="header_right_box">
              <li><img src="{% static 'images/p1.png' %}" alt=""/></li>
              <li>
                <p><a href="login.html">Manzar</a></p>
              </li>
              <div class="clearfix"> </div>
            </ul>
          </div>
          <div class="clearfix"> </div>
        </div>
        <div class="slider">
          <div class="callbacks_container">
            <ul class="rslides" id="slider">
              <li>
                <img src='{% static "images/banner.jpg" %}' class="img-responsive" alt=""/>
                <div class="button">
                    <a href="https://www.imdb.com/title/tt2015381" class="hvr-shutter-out-horizontal">Click
Here</a>
                </div>
              </li>
              <li>
                <img src= '{% static "images/banner1.jpg" %}'class="img-responsive" alt=""/>
                <div class="button">
```

```
                    <a href="https://www.imdb.com/title/tt3896198" class="hvr-shutter-out-horizontal">Click Here</a>
                  </div>
                </li>
                <li>
                  <img src='{% static "images/banner2.jpg" %}'class="img-responsive" alt=""/>
                  <div class="button">
                     <a href="https://www.imdb.com/title/tt0944947" class="hvr-shutter-out-horizontal">Click Here</a>
                  </div>
                </li>
              </ul>
            </div>
          </div>
          <div class="content">
            <div class="box_1">
              <h1 class="m_2">Featurd Movies</h1>
              <div class="clearfix"> </div>
            </div>
            <div class="box_2">
              <div class="row_4">
                <div class="col-md-3 grid_2">
                  <div class="col_2">
                    <ul class="list_4">
                      <li>Rating : {{ rating.0 }}/5 </li>
                      <li>Released Year : {{ year.0 }} </li>
                      <div class="clearfix"> </div>
                    </ul>
                    <form id = 'form1' method = "get" action = "{% url 'single'%}" >
                    <div class="m_5">
                      <input type = "hidden" name = "id" value = "{{ids.0}}" >
                                  <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                         <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>    -->
                         <div class="caption1 header">
                           <p class="m_3 message_ticker"> {{ name.0}} </p>
                         </div>
                    </div>
                  </form>
                  </div>
                  <div class="clearfix"> </div>
                </div>
                <div class="col-md-3 grid_2">
                  <div class="col_2">
                    <ul class="list_4">
                      <li>Rating : {{ rating.1 }}/5 </li>
                      <li>Released Year : {{ year.1 }} </li>
                      <div class="clearfix"> </div>
                    </ul>
                    <form id = 'form1' method = "get" action = "{% url 'single'%}" >
                    <div class="m_5">
```

```
                    <input type = "hidden" name = "id" value = "{{ids.1}}" >
                                <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                      <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>   -->
                      <div class="caption1 header">
                        <p class="m_3 message_ticker"> {{ name.1}} </p>
                      </div>
                    </div>
                  </form>
                  </div>
                  <div class="clearfix"> </div>
                </div>
                <div class="col-md-3 grid_2">
                  <div class="col_2">
                    <ul class="list_4">
                      <li>Rating : {{ rating.2 }}/5 </li>
                      <li>Released Year : {{ year.2 }} </li>
                      <div class="clearfix"> </div>
                    </ul>
                    <form id = 'form1' method = "get" action = "{% url 'single'%}" >
                    <div class="m_5">
                      <input type = "hidden" name = "id" value = "{{ids.2}}" >
                                <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                      <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>   -->
                      <div class="caption1 header">
                        <p class="m_3 message_ticker"> {{ name.2}} </p>
                      </div>
                    </div>
                  </form>
                  </div>
                  <div class="clearfix"> </div>
                </div>
                <div class="col-md-3 grid_2">
                  <div class="col_2">
                    <ul class="list_4">
                      <li>Rating : {{ rating.3 }}/5 </li>
                      <li>Released Year : {{ year.3 }} </li>
                      <div class="clearfix"> </div>
                    </ul>
                    <form id = 'form1' method = "get" action = "{% url 'single'%}" >
                    <div class="m_5">
                      <input type = "hidden" name = "id" value = "{{ids.3}}" >
                                <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                      <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>   -->
                      <div class="caption1 header">
                        <p class="m_3 message_ticker"> {{ name.3}} </p>
                      </div>
                    </div>
                  </form>
```

```
            </div>
            <div class="clearfix"> </div>
          </div>
        </div>
        <div class="clearfix"> </div>
        <div class="box_1">
          <div class="row_2"></div>
          <div class="clearfix"> </div>
          <div class="col-md-3 grid_2">
            <div class="col_2">
              <ul class="list_4">
                <li>Rating : {{ rating.4 }}/5 </li>
                <li>Released Year : {{ year.4 }} </li>
                <div class="clearfix"> </div>
              </ul>
              <form id = 'form1' method = "get" action = "{% url 'single'%}" >
              <div class="m_5">
                <input type = "hidden" name = "id" value = "{{ids.4}}" >
                                <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                  <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>    -->
                  <div class="caption1 header">
                    <p class="m_3 message_ticker"> {{ name.4}} </p>
                  </div>
              </div>
            </form>
            </div>
            <div class="clearfix"> </div>
          </div>
          <div class="col-md-3 grid_2">
            <div class="col_2">
              <ul class="list_4">
                <li>Rating : {{ rating.5 }}/5 </li>
                <li>Released Year : {{ year.5 }} </li>
                <div class="clearfix"> </div>
              </ul>
              <form id = 'form1' method = "get" action = "{% url 'single'%}" >
              <div class="m_5">
                <input type = "hidden" name = "id" value = "{{ids.5}}" >
                                <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                  <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>    -->
                  <div class="caption1 header">
                    <p class="m_3 message_ticker"> {{ name.5}} </p>
                  </div>
              </div>
            </form>
            </div>
            <div class="clearfix"> </div>
          </div>
          <div class="col-md-3 grid_2">
```

```
                <div class="col_2">
                  <ul class="list_4">
                    <li>Rating : {{ rating.6 }}/5 </li>
                    <li>Released Year : {{ year.6 }} </li>
                    <div class="clearfix"> </div>
                  </ul>
                  <form id = 'form1' method = "get" action = "{% url 'single'%}" >
                  <div class="m_5">
                    <input type = "hidden" name = "id" value = "{{ids.6}}" >
                                    <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                      <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>    -->
                      <div class="caption1 header">
                        <p class="m_3 message_ticker"> {{ name.6}} </p>
                      </div>
                  </div>
                </form>
                </div>
                <div class="clearfix"> </div>
              </div>
              <div class="col-md-3 grid_2">
                <div class="col_2">
                  <ul class="list_4">
                    <li>Rating : {{ rating.7 }}/5 </li>
                    <li>Released Year : {{ year.7 }} </li>
                    <div class="clearfix"> </div>
                  </ul>
                  <form id = 'form1' method = "get" action = "{% url 'single'%}" >
                  <div class="m_5">
                    <input type = "hidden" name = "id" value = "{{ids.7}}" >
                                    <input type = "image" class = "img-responsive img2" src ='{% static "images/banner2.jpg" %}' >
                      <!-- <img src='{{ link.0 }}' class="img-responsive img2" alt=""/>    -->
                      <div class="caption1 header">
                        <p class="m_3 message_ticker"> {{ name.7}} </p>
                      </div>
                  </div>
                </form>
                </div>
              </div>
            </div>
          </div>
          <div class="clearfix"> </div>
        </div>
      </div>
    </div>
    </div>
    <div class="container">
      <footer id="footer">
        <div id="footer-3d">
          <div class="gp-container">
```

```
            <span class="first-widget-bend"> </span>
          </div>
        </div>
        <div id="footer-widgets" class="gp-footer-larger-first-col">
          <div class="gp-container">
            <div class="footer-widget footer-1">
              <div class="wpb_wrapper">
                <img src="{% static 'images/logos.png' %}" alt=""/>
              </div>
              <br>
                  <p>It is a long established fact that a reader will be distracted by the readable content of a page.</p>
                   <p class="text">There are many variations of passages of Lorem Ipsum available, but the majority have suffered.</p>
            </div>
            <div class="footer_box">
              <div class="col_1_of_3 span_1_of_3">
                <h3>Follow Us</h3>
                <ul class="first">
                  <li><a href="https://www.facebook.com">Facebook</a></li>
                  <li><a href="https://www.twitter.com">Twitter</a></li>
                  <li><a href="htts://www.instagram.com/manz.ar">Instagram</a></li>
                </ul>
                <div class="copy">
                                <p>&copy; Made by <a href="manzars.github.io" target="_blank"> Team ERRORist</a></p>
                </div>
              </div>
              <div class="clearfix"> </div>
            </div>
            <div class="clearfix"> </div>
          </div>
        </div>
      </footer>
    </div>
  </body>
</html>
```

4.1.1.2 Single.html

```
{% load staticfiles %}
<!DOCTYPE HTML>
<html>
  <head>
      <title>Movie_store A Entertainment Category Flat Bootstarp Resposive Website Template | Single :: w3layouts</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
     <meta name="keywords" content="Movie_store Responsive web template, Bootstrap Web Templates, Flat Web Templates, Andriod Compatible web template,
        Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyErricsson, Motorola web design" />
    <script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>
    <link href='{% static "css/bootstrap.css" %}' rel='stylesheet' type='text/css' />
    <link href='{% static "css/style.css" %}' rel="stylesheet" type="text/css" media="all" />
    <!-- start plugins -->
    <script type="text/javascript" src='{% static "js/jquery-1.11.1.min.js" %}'></script>
                                                    <link href='http://fonts.googleapis.com/css?family=Roboto+Condensed:100,200,300,400,500,600,700,800,900' rel='stylesheet' type='text/css'>
    <script src='{% static "js/responsiveslides.min.js" %}'></script>
  </head>
  <body>
    <div class="container">
      <div class="container_wrap">
        <div class="header_top">
          <div class="col-sm-3 logo"><a href="{% url 'index' %}"><img src="{% static 'images/logos.png' %}"alt=""/></a></div>
          <div class="col-sm-3 header_right navbar-right">
            <ul class="header_right_box">
              <li><img src="{% static 'images/p1.png' %}" alt=""/></li>
              <li>
                <p><a href="{% url 'login' %}">Manzar</a></p>
              </li>
              <div class="clearfix"> </div>
            </ul>
          </div>
          <div class="clearfix"> </div>
        </div>
        <div class="content">
          <div class="movie_top">
            <div class="col-md-9 movie_box">
              <div class="grid images_3_of_2">
                <div class="movie_image">
                  <span class="movie_rating">5.0</span>
                  <img src="{% static 'images/single.jpg' %}" class="img-responsive" alt=""/>
                </div>
              </div>
```

```
            <div class="desc1 span_3_of_2">
                  <p class="movie_option"><strong>Country: </strong><a href="#">established</a>, <a href="#">USA</a></p>
                <p class="movie_option"><strong>Year: </strong>2014</p>
                    <p class="movie_option"><strong>Category: </strong><a href="#">Adventure</a>, <a href="#">Fantazy</a></p>
                <p class="movie_option"><strong>Release date: </strong>December 12, 2014</p>
                <p class="movie_option"><strong>Director: </strong><a href="#">suffered </a></p>
                        <p class="movie_option"><strong>Actors: </strong><a href="#">anything</a>, <a href="#">Lorem Ipsum</a>, <a href="#" discovered</a>, <a href="#"> Virginia</a>, <a href="#"> Virginia</a>, <a href="#">variations</a>, <a href="#">variations</a>, <a href="#">variations</a>, <a href="#"> Virginia</a> <a href="#">...</a></p>
                <p class="movie_option"><strong>Age restriction: </strong>13</p>
                <div class="down_btn"><a class="btn1" href="#"><span> </span>Watch Now</a></div>
              </div>
              <div class="clearfix"> </div>
                 <p class="m_4">There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet.</p>
              <div class="clearfix"></div>
              </li>
              </ul>
            </div>
          </div>
          <div class="col-md-3">
            <div class="movie_img">
              <div class="grid_2">
                <img src="{% static 'images/pic6.jpg' %}" class="img-responsive" alt="">
                <div class="caption1">
                  <ul class="list_5 list_7">
                    <li>
                      <i class="icon5"> </i>
                      <p>3,548</p>
                    </li>
                  </ul>
                  <i class="icon4 icon6 icon7"> </i>
                  <p class="m_3">{{ message }}</p>
                </div>
              </div>
            </div>
            <div class="grid_2 col_1">
              <img src="{% static 'images/pic2.jpg' %}" class="img-responsive" alt="">
              <div class="caption1">
                <ul class="list_3 list_7">
                  <li>
                    <i class="icon5"> </i>
                    <p>3,548</p>
                  </li>
                </ul>
```

```
                <i class="icon4 icon7"> </i>
                <p class="m_3">Guardians of the Galaxy</p>
              </div>
            </div>
            <div class="grid_2 col_1">
              <img src="{% static 'images/pic9.jpg' %}" class="img-responsive" alt="">
              <div class="caption1">
                <ul class="list_3 list_7">
                  <li>
                    <i class="icon5"> </i>
                    <p>3,548</p>
                  </li>
                </ul>
                <i class="icon4 icon7"> </i>
                <p class="m_3">Guardians of the Galaxy</p>
              </div>
            </div>
          </div>
          <div class="clearfix"> </div>
        </div>
      </div>
    </div>
    </div>
    <div class="container">
      <footer id="footer">
        <div id="footer-3d">
          <div class="gp-container">
            <span class="first-widget-bend"> </span>
          </div>
        </div>
        <div id="footer-widgets" class="gp-footer-larger-first-col">
          <div class="gp-container">
            <div class="footer-widget footer-1">
              <div class="wpb_wrapper">
                <img src="{% static 'images/logos.png' %}" alt=""/>
              </div>
              <br>
                 <p>It is a long established fact that a reader will be distracted by the readable content of a page.</p>
                  <p class="text">There are many variations of passages of Lorem Ipsum available, but the majority have suffered.</p>
            </div>
            <div class="footer_box">
              <div class="col_1_of_3 span_1_of_3 row align-items-center">
                <h3>Follow Us</h3>
                <ul class="first">
                  <li><a href="https://www.facebook.com">Facebook</a></li>
                  <li><a href="https://www.twitter.com"">Twitter</a></li>
                  <li><a href="htts://www.instagram.com/manz.ar">Instagram</a></li>
                </ul>
                <div class="copy">
```

```
                        <p>&copy; Made by <a href="manzars.github.io" target="_blank"> Team ERRORist</a></p>
                    </div>
                  </div>
                  <div class="clearfix"> </div>
                </div>
                <div class="clearfix"> </div>
              </div>
            </div>
          </footer>
        </div>
    </body>
</html>
```

4.1.1.3 Register.html

```
{% load staticfiles %}
<!DOCTYPE HTML>
<html>
  <head>
     <title>Movie_store A Entertainment Category Flat Bootstarp Resposive Website Template | Register :: w3layouts</title>
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
     <meta name="keywords" content="Movie_store Responsive web template, Bootstrap Web Templates, Flat Web Templates, Andriod Compatible web template,
        Smartphone Compatible web template, free webdesigns for Nokia, Samsung, LG, SonyErricsson, Motorola web design" />
    <script type="application/x-javascript"> addEventListener("load", function() { setTimeout(hideURLbar, 0); }, false); function hideURLbar(){ window.scrollTo(0,1); } </script>
    <link href='{% static "css/bootstrap.css" %}' rel='stylesheet' type='text/css' />
    <link href='{% static "css/style.css" %}' rel="stylesheet" type="text/css" media="all" />
    <!-- start plugins -->
    <script type="text/javascript" src='{% static "js/jquery-1.11.1.min.js" %}'></script>
                                                    <link href='http://fonts.googleapis.com/css?family=Roboto+Condensed:100,200,300,400,500,600,700,800,900' rel='stylesheet' type='text/css'>
    <script src='{% static "js/responsiveslides.min.js" %}'></script>
  </head>
  <body>
    <div class="container">
      <div class="container_wrap">
        <div class="header_top">
          <div class="col-sm-3 logo"><a href="{% url 'index' %}"><img src="{% static 'images/logos.png' %}" alt=""/></a></div>
          <div class="col-sm-3 header_right navbar-right">
            <ul class="header_right_box">
              <li><img src="{% static 'images/p1.png' %}" alt=""/></li>
              <li>
                <p><a href="{% url 'login' %}">Manzar</a></p>
              </li>
```

```
            <div class="clearfix"> </div>
          </ul>
        </div>
        <div class="clearfix"> </div>
      </div>
      <div class="content">
        <div class="register">
          <form>
            <div class="register-top-grid">
              <h3>Personal Information</h3>
              <div>
                <span>First Name<label>*</label></span>
                <input type="text">
              </div>
              <div>
                <span>Last Name<label>*</label></span>
                <input type="text">
              </div>
              <div>
                <span>Email Address<label>*</label></span>
                <input type="text">
              </div>
              <div class="clearfix"> </div>
              <a class="news-letter" href="#">
                    <label class="checkbox"><input type="checkbox" name="checkbox" checked=""><i>
</i>Sign Up for Newsletter</label>
              </a>
            </div>
            <div class="register-bottom-grid">
              <h3>Login Information</h3>
              <div>
                <span>Password<label>*</label></span>
                <input type="text">
              </div>
              <div>
                <span>Confirm Password<label>*</label></span>
                <input type="text">
              </div>
              <div class="clearfix"> </div>
            </div>
          </form>
          <div class="clearfix"> </div>
          <div class="register-but">
            <form>
              <input type="submit" value="submit">
              <div class="clearfix"> </div>
            </form>
          </div>
        </div>
      </div>
```

```
    </div>
    <div class="container">
      <footer id="footer">
        <div id="footer-3d">
          <div class="gp-container">
            <span class="first-widget-bend"> </span>
          </div>
        </div>
        <div id="footer-widgets" class="gp-footer-larger-first-col">
          <div class="gp-container">
            <div class="footer-widget footer-1">
              <div class="wpb_wrapper">
                <img src="{% static 'images/logos.png' %}" alt=""/>
              </div>
              <br>
                  <p>It is a long established fact that a reader will be distracted by the readable content of a page.</p>
                   <p class="text">There are many variations of passages of Lorem Ipsum available, but the majority have suffered.</p>
            </div>
            <div class="footer_box">
              <div class="col_1_of_3 span_1_of_3">
                <h3>Follow Us</h3>
                <ul class="first">
                  <li><a href="https://www.facebook.com">Facebook</a></li>
                  <li><a href="https://www.twitter.com"">Twitter</a></li>
                  <li><a href="htts://www.instagram.com/manz.ar">Instagram</a></li>
                </ul>
                <div class="copy">
                            <p>&copy; Made by <a href="manzars.github.io" target="_blank"> Team ERRORist</a></p>
                </div>
              </div>
              <div class="clearfix"> </div>
            </div>
            <div class="clearfix"> </div>
          </div>
        </div>
      </footer>
    </div>
  </body>
</html>
```

4.1.2 Movies

4.1.2.1 View.py

```
from django.shortcuts import render, redirect
import os
import numpy as np
import random
from surprise import NormalPredictor
from movies.MovieLens import MovieLens
from movies.ContentKNNAlgorithm import ContentKNNAlgorithm
from movies.Evaluator import Evaluator
from joblib import load
#from movierec.settings import algo, dataset
from bs4 import BeautifulSoup
import requests
import re
import random
import pandas as pd
from .forms import UserForm
from django.views import generic
from django.contrib.auth import authenticate, login

 class counter():
        count = 0
        def inc(self):
                self.count = (self.count + 1) % 5
                return self.count
        def count_print(self):
                print(self.count)



recommendations = [1,2,3,4,5,6,7,8,9]
ml = MovieLens()
links = [1,2,3,4,5,6,7,8,9]
ratings = [1,2,3,4,5,6,7,8,9]
years = [1,2,3,4,5,6,7,8,9]
movies_ids = [1,2,3,4,5,6,7,8,9]


 def getYears(movie_name):
        p = re.compile(r"(?:\((\d{4})\))?\s*$")
        m = p.search(movie_name)
        return m.group(1)

 def LoadMovieLensData():

        data            = ml.loadMovieLensLatestSmall()
```

```
        rankings        = ml.getPopularityRanks()
        return (ml, data, rankings)

def Scraping(movie_id):
        rating = pd.read_csv('/home/manzars/Downloads/movie/src/movies/ml-latest-small/ratings.csv')
        p = rating.loc[(rating.movieId == movie_id)].iloc[:, 2:3].values
        total = 0
        for i in range(len(p)):
           total = total + float(p[i])
        avg = total/len(p)
        rating = round(avg, 2)


        link = pd.read_csv('/home/manzars/Downloads/movie/src/movies/ml-latest-small/links.csv')
        p = link.loc[(link.movieId == movie_id)].iloc[:, 1:2].values
        x = str(p[0][0])
        req     = requests.get('http://www.imdb.com/title/tt' + x)
        soup    = BeautifulSoup(req.text, 'lxml')
        tag     = soup.findAll('div', {'class': 'poster'})

        try:
                link    = tag[0].img.attrs['src']
        except:
                link = 'https://user-images.githubusercontent.com/24848110/33519396-7e56363c-d79d-11e7-
969b-09782f5ccbab.png'
        return link, rating


def generate_recommendation(user_id):
        np.random.seed(0)
        random.seed(0)

        (ml, evaluationData, rankings) = LoadMovieLensData()

        testset = dataset.GetAntiTestSetForUser(user_id)
        predictions = algo.GetAlgorithm().test(testset)

        recommendations = []
        for userID, movieID, actualRating, estimatedRating, _ in predictions:
                intMovieID = int(movieID)
                recommendations.append((intMovieID, estimatedRating))

        recommendations.sort(key=lambda x: x[1], reverse=True)
        recom = []

        for ratings in recommendations[:20]:
                recom.append(ml.getMovieName(ratings[0]))

        return recom
```

```
def index_view(request,*args,**kwargs):

    if(not recommendations):

        recom = generate_recommendation(609)

        movie_id = []
        for rec in recom:
            movie_id.append(ml.getMovieID(rec))



        year = []

        for movie in recom:
            year.append(getYears(movie))

        link, rating = [], []

        for ids in movie_id:
            link.append(Scraping(ids)[0])
            rating.append(Scraping(ids)[1])

            links = link
            ratings = rating
            movies_ids = movie_id
            recommendations = recom
            years = year

            contex = {
                'name': recommendations,
                'link': links,
                'rating': ratings,
                'year': year
            }
    else:



        contex = {
            'name': recommendations,
            'link': links,
            'rating': ratings,
            'year': years,
            'ids': movies_ids
        }
```

```
        return render(request,"web/index.html",contex)


def login_view(request,*args,**kwargs):
        return render(request,"web/login.html",{})


def movie_view(request,*args,**kwargs):
        return render(request,"web/movie.html",{})


def register_view(request,*args,**kwargs):
        return render(request,"web/register.html",{})


def single_view(request,*args,**kwargs):
        query = request.GET.get('id')
        if(query):
                print(query)
        else:
                print('not found')
        message = "{}".format(query)
        contex = {
                'message': query
        }
        return render(request,"web/single.html",{})


class UserFormView(View):
        form_class = UserForm
        template = 'web/register.html'

        def get(self, request):
                form = self.form_class(None)
                return render(request, self.template_name, {'form': form})


        def post(self, request):
                form = self.form_class(request.POST)
                if form.is_valid():
                        user = form.save(commit = False)
                        username = form.cleaned_data['username']
                        password = form.cleaned_data['password']
                        user.set_password(password)
                        user.save()
```

4.1.2.2 forms.py

```
from django.contrib.auth.models import User
from django import forms

class UserForm(forms.ModelForm):
        pa ssword = forms.CharField(widget = forms.PasswordInput  )
        class Meta:
                model = User
                fields = ['username', 'email', 'password']
```

4.1.3 Movierec
4.1.3.1 settings.py

```
 """
Django settings for movierec project.

Generated by 'django-admin startproject' using Django 2.0.7.

For more information on this file, see
https://docs.djangoproject.com/en/2.0/topics/settings/

For the full list of settings and their values, see
https://docs.djangoproject.com/en/2.0/ref/settings/
"""

import os
from joblib import load


from django.shortcuts import render
import os
import numpy as np
import random
from surprise import NormalPredictor
from movies.MovieLens import MovieLens
from movies.ContentKNNAlgorithm import ContentKNNAlgorithm
from movies.Evaluator import Evaluator



algo = load('algo.pkl')
dataset = load('dataset.pkl')




# Build paths inside the project like this: os.path.join(BASE_DIR, ...)
```

```
BASE_DIR = os.path.dirname(os.path.dirname(os.path.abspath(__file__)))



# Quick-start development settings - unsuitable for production
# See https://docs.djangoproject.com/en/2.0/howto/deployment/checklist/

# SECURITY WARNING: keep the secret key used in production secret!
SECRET_KEY = 'gxsddef9u0$me8o5h17e9f=18-k#eysqjap#6dtwei0etjb4lq'

# SECURITY WARNING: don't run with debug turned on in production!
DEBUG = True

  ALLOWED_HOSTS = ['192.168.1.102', '192.168.43.225', '172.16.104.9', '172.16.106.19', '127.0.0.1',
'192.168.1.109']


# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'movies',
]

MIDDLEWARE = [
    'django.middleware.security.SecurityMiddleware',
    'django.contrib.sessions.middleware.SessionMiddleware',
    'django.middleware.common.CommonMiddleware',
    'django.middleware.csrf.CsrfViewMiddleware',
    'django.contrib.auth.middleware.AuthenticationMiddleware',
    'django.contrib.messages.middleware.MessageMiddleware',
    'django.middleware.clickjacking.XFrameOptionsMiddleware',
]

ROOT_URLCONF = 'movierec.urls'

TEMPLATES = [
    {
        'BACKEND': 'django.template.backends.django.DjangoTemplates',
        'DIRS': [os.path.join(BASE_DIR,"templates")],
        'APP_DIRS': True,
        'OPTIONS': {
            'context_processors': [
                'django.template.context_processors.debug',
                'django.template.context_processors.request',
```

```
                'django.contrib.auth.context_processors.auth',
                'django.contrib.messages.context_processors.messages',
            ],
        },
    },
]

WSGI_APPLICATION = 'movierec.wsgi.application'


# Database
# https://docs.djangoproject.com/en/2.0/ref/settings/#databases

DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.sqlite3',
        'NAME': os.path.join(BASE_DIR, 'db.sqlite3'),
    }
}


# Password validation
# https://docs.djangoproject.com/en/2.0/ref/settings/#auth-password-validators

AUTH_PASSWORD_VALIDATORS = [
    {
        'NAME': 'django.contrib.auth.password_validation.UserAttributeSimilarityValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.MinimumLengthValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.CommonPasswordValidator',
    },
    {
        'NAME': 'django.contrib.auth.password_validation.NumericPasswordValidator',
    },
]


# Internationalization
# https://docs.djangoproject.com/en/2.0/topics/i18n/

LANGUAGE_CODE = 'en-us'

TIME_ZONE = 'UTC'

USE_I18N = True

USE_L10N = True
```

```
    USE_TZ = True


    # Static files (CSS, JavaScript, Images)
    # https://docs.djangoproject.com/en/2.0/howto/static-files/

    STATIC_URL = '/static/'
    MEDIA_URL='/media/'
    STATICFILES_DIRS=[
        os.path.join(BASE_DIR,"static"),
        #'/var/www/static/',
    ]

    STATIC_ROOT=os.path.join(os.path.dirname(BASE_DIR),"static_cdn")
    MEDIA_ROOT=os.path.join(os.path.dirname(BASE_DIR),"media_cdn")
```

4.1.3.2 urls.py

```
"""movierec URL Configuration

The `urlpatterns` list routes URLs to views. For more information please see:
    https://docs.djangoproject.com/en/2.0/topics/http/urls/
Examples:
Function views
    1. Add an import:  from my_app import views
    2. Add a URL to urlpatterns:  path('', views.home, name='home')
Class-based views
    1. Add an import:  from other_app.views import Home
    2. Add a URL to urlpatterns:  path('', Home.as_view(), name='home')
Including another URLconf
    1. Import the include() function: from django.urls import include, path
    2. Add a URL to urlpatterns:  path('blog/', include('blog.urls'))
"""
from django.conf import settings
from django.conf.urls.static import static
from django.contrib import admin
from django.urls import path
from movies.views import *

urlpatterns = [
    path('admin/', admin.site.urls),
    path('', index_view, name = 'index'),
    path('login/', login_view, name = 'login'),
    path('register/', register_view, name = 'register'),
    path('single/', single_view, name = 'single'),
    path('movie/', movie_view, name = 'movie')
]
if settings.DEBUG:
    urlpatterns += static(settings.STATIC_URL,document_root=settings.STATIC_ROOT)
```

4.1.4 Machine Learning

4.1.4.1 Movielens.py

```
import os
import csv
import sys
import re

from surprise import Dataset
from surprise import Reader

from collections import defaultdict
import numpy as np

class MovieLens:

    movieID_to_name = {}
    name_to_movieID = {}
    ratingsPath = '/home/manzars/Downloads/movie/src/movies/ml-latest-small/ratings.csv'
    moviesPath = '/home/manzars/Downloads/movie/src/movies/ml-latest-small/movies.csv'

    def loadMovieLensLatestSmall(self):

        # Look for files relative to the directory we are running from
        #os.chdir(os.path.dirname(sys.argv[0]))

        ratingsDataset = 0
        self.movieID_to_name = {}
        self.name_to_movieID = {}

        reader = Reader(line_format='user item rating timestamp', sep=',', skip_lines=1)

        ratingsDataset = Dataset.load_from_file(self.ratingsPath, reader=reader)

        with open(self.moviesPath, newline='', encoding='ISO-8859-1') as csvfile:
                movieReader = csv.reader(csvfile)
                next(movieReader)  #Skip header line
                for row in movieReader:
                    movieID = int(row[0])
                    movieName = row[1]
                    self.movieID_to_name[movieID] = movieName
                    self.name_to_movieID[movieName] = movieID

        return ratingsDataset

    def getUserRatings(self, user):
        userRatings = []
        hitUser = False
        with open(self.ratingsPath, newline='') as csvfile:
            ratingReader = csv.reader(csvfile)
```

```
            next(ratingReader)
            for row in ratingReader:
                userID = int(row[0])
                if (user == userID):
                    movieID = int(row[1])
                    rating = float(row[2])
                    userRatings.append((movieID, rating))
                    hitUser = True
                if (hitUser and (user != userID)):
                    break

        return userRatings

    def getPopularityRanks(self):
        ratings = defaultdict(int)
        rankings = defaultdict(int)
        with open(self.ratingsPath, newline='') as csvfile:
            ratingReader = csv.reader(csvfile)
            next(ratingReader)
            for row in ratingReader:
                movieID = int(row[1])
                ratings[movieID] += 1
        rank = 1
        for movieID, ratingCount in sorted(ratings.items(), key=lambda x: x[1], reverse=True):
            rankings[movieID] = rank
            rank += 1
        return rankings

    def getGenres(self):
        genres = defaultdict(list)
        genreIDs = {}
        maxGenreID = 0
        with open(self.moviesPath, newline='', encoding='ISO-8859-1') as csvfile:
            movieReader = csv.reader(csvfile)
            next(movieReader)  #Skip header line
            for row in movieReader:
                movieID = int(row[0])
                genreList = row[2].split('|')
                genreIDList = []
                for genre in genreList:
                    if genre in genreIDs:
                        genreID = genreIDs[genre]
                    else:
                        genreID = maxGenreID
                        genreIDs[genre] = genreID
                        maxGenreID += 1
                    genreIDList.append(genreID)
                genres[movieID] = genreIDList
        # Convert integer-encoded genre lists to bitfields that we can treat as vectors
        for (movieID, genreIDList) in genres.items():
            bitfield = [0] * maxGenreID
```

```
            for genreID in genreIDList:
                bitfield[genreID] = 1
            genres[movieID] = bitfield

        return genres

    def getYears(self):
        p = re.compile(r"(?:\((\d{4})\))?\s*$")
        years = defaultdict(int)
        with open(self.moviesPath, newline='', encoding='ISO-8859-1') as csvfile:
            movieReader = csv.reader(csvfile)
            next(movieReader)
            for row in movieReader:
                movieID = int(row[0])
                title = row[1]
                m = p.search(title)
                year = m.group(1)
                if year:
                    years[movieID] = int(year)
        return years

    def getMiseEnScene(self):
        mes = defaultdict(list)
         with open("/home/manzars/Downloads/movie/src/movies/LLVisualFeatures13K_Log.csv", newline='')
as csvfile:
            mesReader = csv.reader(csvfile)
            next(mesReader)
            for row in mesReader:
                movieID = int(row[0])
                avgShotLength = float(row[1])
                meanColorVariance = float(row[2])
                stddevColorVariance = float(row[3])
                meanMotion = float(row[4])
                stddevMotion = float(row[5])
                meanLightingKey = float(row[6])
                numShots = float(row[7])
                mes[movieID] = [avgShotLength, meanColorVariance, stddevColorVariance,
                  meanMotion, stddevMotion, meanLightingKey, numShots]
        return mes

    def getMovieName(self, movieID):
        if movieID in self.movieID_to_name:
            return self.movieID_to_name[movieID]
        else:
            return ""

    def getMovieID(self, movieName):
        if movieName in self.name_to_movieID:
            return self.name_to_movieID[movieName]
        else:
            return 0
```

4.1.4.2 Evaluator.py

```
from movies.EvaluationData import EvaluationData
from movies.EvaluatedAlgorithm import EvaluatedAlgorithm

class Evaluator:

    algorithms = []

    def __init__(self, dataset, rankings):
        ed = EvaluationData(dataset, rankings)
        self.dataset = ed

    def AddAlgorithm(self, algorithm, name):
        alg = EvaluatedAlgorithm(algorithm, name)
        self.algorithms.append(alg)

    def Evaluate(self, doTopN):
        results = {}
        for algorithm in self.algorithms:
            print("Evaluating ", algorithm.GetName(), "...")
            results[algorithm.GetName()] = algorithm.Evaluate(self.dataset, doTopN)

        # Print results
        print("\n")

        if (doTopN):
            print("{:<10} {:<10} {:<10} {:<10} {:<10} {:<10} {:<10} {:<10} {:<10}".format(
                "Algorithm", "RMSE", "MAE", "HR", "cHR", "ARHR", "Coverage", "Diversity", "Novelty"))
            for (name, metrics) in results.items():
                print("{:<10} {:<10.4f} {:<10.4f} {:<10.4f} {:<10.4f} {:<10.4f} {:<10.4f} {:<10.4f}
{:<10.4f}".format(
                    name, metrics["RMSE"], metrics["MAE"], metrics["HR"], metrics["cHR"],
metrics["ARHR"],
                    metrics["Coverage"], metrics["Diversity"], metrics["Novelty"]))
        else:
            print("{:<10} {:<10} {:<10}".format("Algorithm", "RMSE", "MAE"))
            for (name, metrics) in results.items():
                print("{:<10} {:<10.4f} {:<10.4f}".format(name, metrics["RMSE"], metrics["MAE"]))

        print("\nLegend:\n")
        print("RMSE:      Root Mean Squared Error. Lower values mean better accuracy.")
        print("MAE:       Mean Absolute Error. Lower values mean better accuracy.")
        if (doTopN):
            print("HR:        Hit Rate; how often we are able to recommend a left-out rating. Higher is better.")
            print("cHR:       Cumulative Hit Rate; hit rate, confined to ratings above a certain threshold. Higher is
better.")
            print("ARHR:      Average Reciprocal Hit Rank - Hit rate that takes the ranking into account. Higher
is better." )
```

```
        print("Coverage:  Ratio of users for whom recommendations above a certain threshold exist. Higher is better.")
        print("Diversity: 1-S, where S is the average similarity score between every possible pair of recommendations")
        print("           for a given user. Higher means more diverse.")
        print("Novelty:   Average popularity rank of recommended items. Higher means more novel.")

    def SampleTopNRecs(self, ml, testSubject=85, k=10):

        for algo in self.algorithms:
            print("\nUsing recommender ", algo.GetName())

            print("\nBuilding recommendation model...")
            trainSet = self.dataset.GetFullTrainSet()
            algo.GetAlgorithm().fit(trainSet)

            print("Computing recommendations...")
            testSet = self.dataset.GetAntiTestSetForUser(testSubject)

            return (algo, self.dataset)
            '''
            predictions = algo.GetAlgorithm().test(testSet)

            recommendations = []

            print ("\nWe recommend:")
            for userID, movieID, actualRating, estimatedRating, _ in predictions:
                intMovieID = int(movieID)
                recommendations.append((intMovieID, estimatedRating))

            recommendations.sort(key=lambda x: x[1], reverse=True)

            for ratings in recommendations[:10]:
                print(ml.getMovieName(ratings[0]), ratings[1])
            '''
```

4.1.4.3 EvaluationData.py

```
from surprise.model_selection import train_test_split
from surprise.model_selection import LeaveOneOut
from surprise import KNNBaseline

class EvaluationData:

    def __init__(self, data, popularityRankings):

        self.rankings = popularityRankings

        #Build a full training set for evaluating overall properties
        self.fullTrainSet = data.build_full_trainset()
        self.fullAntiTestSet = self.fullTrainSet.build_anti_testset()

        #Build a 75/25 train/test split for measuring accuracy
        self.trainSet, self.testSet = train_test_split(data, test_size=.25, random_state=1)

        #Build a "leave one out" train/test split for evaluating top-N recommenders
        #And build an anti-test-set for building predictions
        LOOCV = LeaveOneOut(n_splits=1, random_state=1)
        for train, test in LOOCV.split(data):
            self.LOOCVTrain = train
            self.LOOCVTest = test

        self.LOOCVAntiTestSet = self.LOOCVTrain.build_anti_testset()

        #Compute similarty matrix between items so we can measure diversity
        sim_options = {'name': 'cosine', 'user_based': False}
        self.simsAlgo = KNNBaseline(sim_options=sim_options)
        self.simsAlgo.fit(self.fullTrainSet)

    def GetFullTrainSet(self):
        return self.fullTrainSet

    def GetFullAntiTestSet(self):
        return self.fullAntiTestSet

    def GetAntiTestSetForUser(self, testSubject):
        trainset = self.fullTrainSet
        fill = trainset.global_mean
        anti_testset = []
        u = trainset.to_inner_uid(str(testSubject))
        user_items = set([j for (j, _) in trainset.ur[u]])
        anti_testset += [(trainset.to_raw_uid(u), trainset.to_raw_iid(i), fill) for
                         i in trainset.all_items() if
                         i not in user_items]
        return anti_testset

    def GetTrainSet(self):
        return self.trainSet
```

```
    def GetTestSet(self):
        return self.testSet

    def GetLOOCVTrainSet(self):
        return self.LOOCVTrain

    def GetLOOCVTestSet(self):
        return self.LOOCVTest

    def GetLOOCVAntiTestSet(self):
        return self.LOOCVAntiTestSet

    def GetSimilarities(self):
        return self.simsAlgo

    def GetPopularityRankings(self):
        return self.rankings
```

4.1.4.3 ContentRec.py

```
from movies.MovieLens import MovieLens
from movies.ContentKNNAlgorithm import ContentKNNAlgorithm
from movies.Evaluator import Evaluator
#from surprise import NormalPredictor
from joblib import dump, load
import random
import numpy as np

class ContentRecs:

    def LoadMovieLensData():
        ml = MovieLens()
        print("Loading movie ratings...")
        data = ml.loadMovieLensLatestSmall()
        print("\nComputing movie popularity ranks so we can measure novelty later...")
        rankings = ml.getPopularityRanks()
        return (ml, data, rankings)

    np.random.seed(0)
    random.seed(0)

    # Load up common data set for the recommender algorithms
    (ml, evaluationData, rankings) = LoadMovieLensData()

    # Construct an Evaluator to, you know, evaluate them
    evaluator = Evaluator(evaluationData, rankings)

    contentKNN = ContentKNNAlgorithm()
```

```
evaluator.AddAlgorithm(contentKNN, "ContentKNN")

# Just make random recommendations
#Random = NormalPredictor()
#evaluator.AddAlgorithm(Random, "Random")

evaluator.Evaluate(False)

(algo, dataset) = evaluator.SampleTopNRecs(ml)



testset = dataset.GetAntiTestSetForUser(609)
predictions = algo.GetAlgorithm().test(testset)

dump(algo, 'algo.pkl')
dump(dataset, 'dataset.pkl')


'''
algo = load('algo.pkl')

dataset = load('dataset.pkl')

recommendations = []

print ("\nWe recommend:")
for userID, movieID, actualRating, estimatedRating, _ in predictions:
    intMovieID = int(movieID)
    recommendations.append((intMovieID, estimatedRating))

recommendations.sort(key=lambda x: x[1], reverse=True)

for ratings in recommendations[:20]:
    print(ml.getMovieName(ratings[0]), ratings[1])
'''
```

4.1.4.3 ContentKNNAlgorithm.py

```
from surprise import AlgoBase
from surprise import PredictionImpossible
from movies.MovieLens import MovieLens
import math
import numpy as np
import heapq

class ContentKNNAlgorithm(AlgoBase):

    def __init__(self, k=40, sim_options={}):
        AlgoBase.__init__(self)
        self.k = k

    def fit(self, trainset):
        AlgoBase.fit(self, trainset)

        # Compute item similarity matrix based on content attributes

        # Load up genre vectors for every movie
        ml = MovieLens()
        genres = ml.getGenres()
        years = ml.getYears()
        mes = ml.getMiseEnScene()

        print("Computing content-based similarity matrix...")

        # Compute genre distance for every movie combination as a 2x2 matrix
        self.similarities = np.zeros((self.trainset.n_items, self.trainset.n_items))

        for thisRating in range(self.trainset.n_items):
            if (thisRating % 100 == 0):
                print(thisRating, " of ", self.trainset.n_items)
            for otherRating in range(thisRating+1, self.trainset.n_items):
                thisMovieID = int(self.trainset.to_raw_iid(thisRating))
                otherMovieID = int(self.trainset.to_raw_iid(otherRating))
                genreSimilarity = self.computeGenreSimilarity(thisMovieID, otherMovieID, genres)
                yearSimilarity = self.computeYearSimilarity(thisMovieID, otherMovieID, years)
                #mesSimilarity = self.computeMiseEnSceneSimilarity(thisMovieID, otherMovieID, mes)
                self.similarities[thisRating, otherRating] = genreSimilarity * yearSimilarity
                self.similarities[otherRating, thisRating] = self.similarities[thisRating, otherRating]

        print("...done.")

        return self

    def computeGenreSimilarity(self, movie1, movie2, genres):
        genres1 = genres[movie1]
        genres2 = genres[movie2]
        sumxx, sumxy, sumyy = 0, 0, 0
        for i in range(len(genres1)):
            x = genres1[i]
```

```
            y = genres2[i]
            sumxx += x * x
            sumyy += y * y
            sumxy += x * y

        return sumxy/math.sqrt(sumxx*sumyy)

    def computeYearSimilarity(self, movie1, movie2, years):
        diff = abs(years[movie1] - years[movie2])
        sim = math.exp(-diff / 10.0)
        return sim

    def computeMiseEnSceneSimilarity(self, movie1, movie2, mes):
        mes1 = mes[movie1]
        mes2 = mes[movie2]
        if (mes1 and mes2):
            shotLengthDiff = math.fabs(mes1[0] - mes2[0])
            colorVarianceDiff = math.fabs(mes1[1] - mes2[1])
            motionDiff = math.fabs(mes1[3] - mes2[3])
            lightingDiff = math.fabs(mes1[5] - mes2[5])
            numShotsDiff = math.fabs(mes1[6] - mes2[6])
            return shotLengthDiff * colorVarianceDiff * motionDiff * lightingDiff * numShotsDiff
        else:
            return 0
    def estimate(self, u, i):
        if not (self.trainset.knows_user(u) and self.trainset.knows_item(i)):
            raise PredictionImpossible('User and/or item is unkown.')
        # Build up similarity scores between this item and everything the user rated
        neighbors = []
        for rating in self.trainset.ur[u]:
            genreSimilarity = self.similarities[i,rating[0]]
            neighbors.append( (genreSimilarity, rating[1]) )

        # Extract the top-K most-similar ratings
        k_neighbors = heapq.nlargest(self.k, neighbors, key=lambda t: t[0])

        # Compute average sim score of K neighbors weighted by user ratings
        simTotal = weightedSum = 0
        for (simScore, rating) in k_neighbors:
            if (simScore > 0):
                simTotal += simScore
                weightedSum += simScore * rating
        if (simTotal == 0):
            raise PredictionImpossible('No neighbors')

        predictedRating = weightedSum / simTotal

        return predictedRating
```

SCREENSHOT :

- **Home Page**

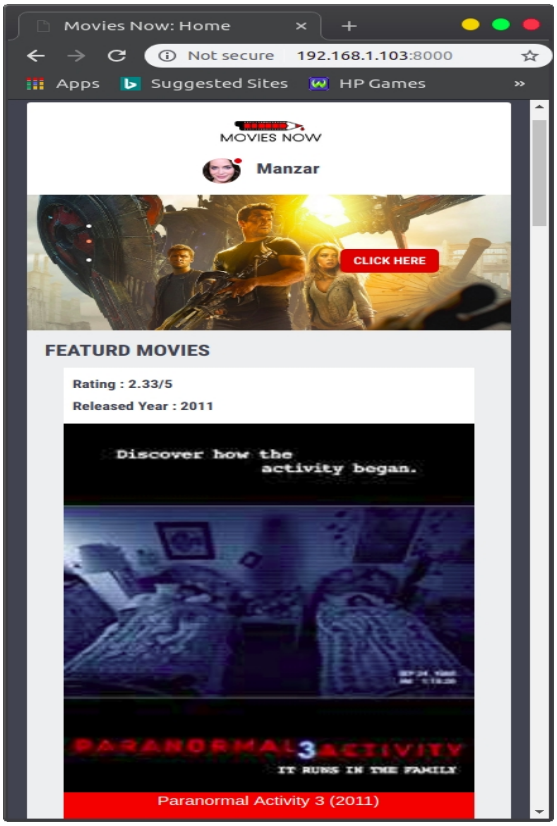


Figure 1.1 HomePage1

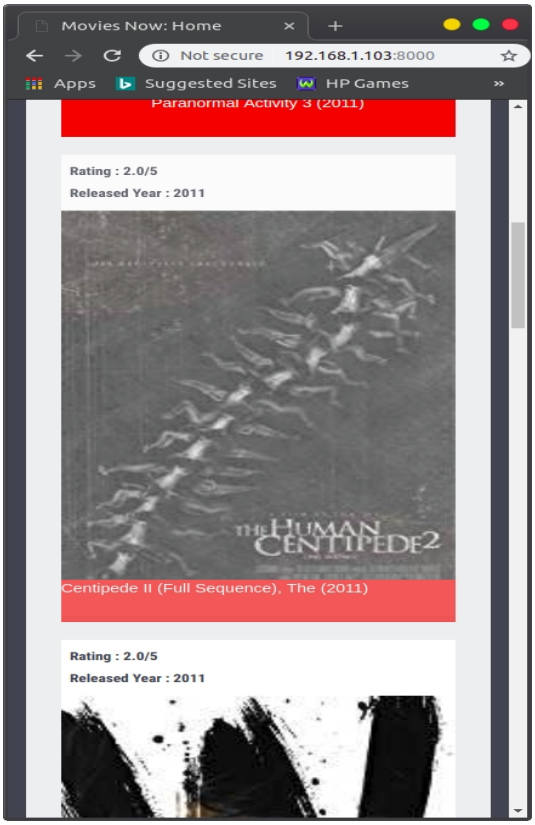
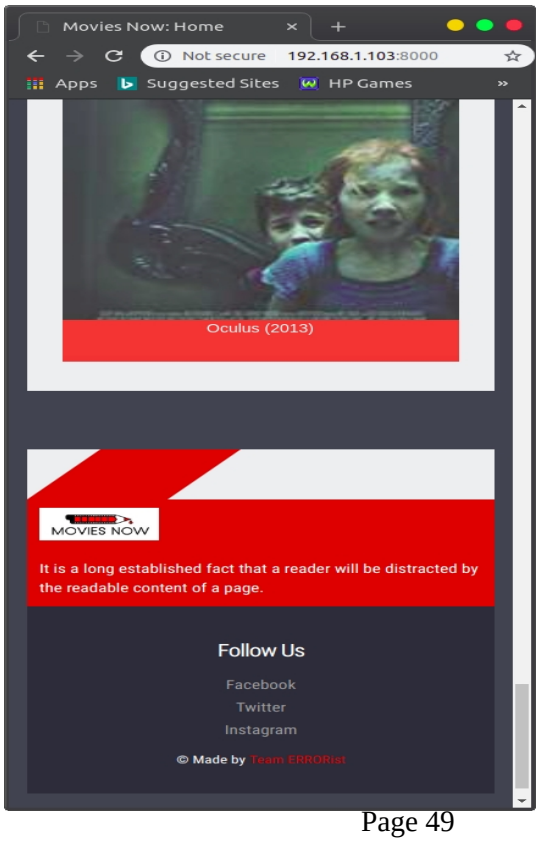


Figure 1.2 HomePage2



- **Single movie page**

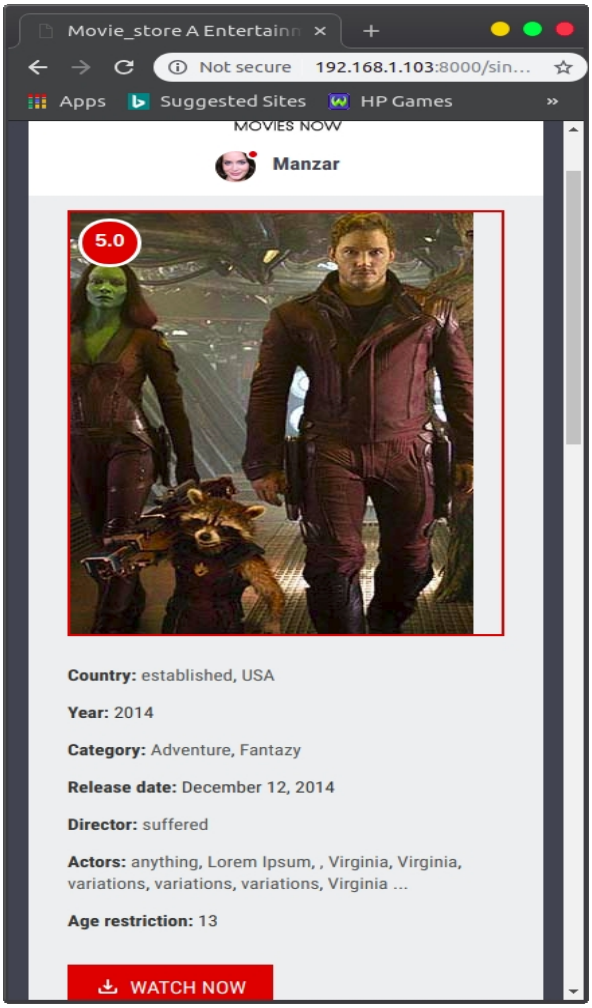


Figure 2.1 Singlemovie1

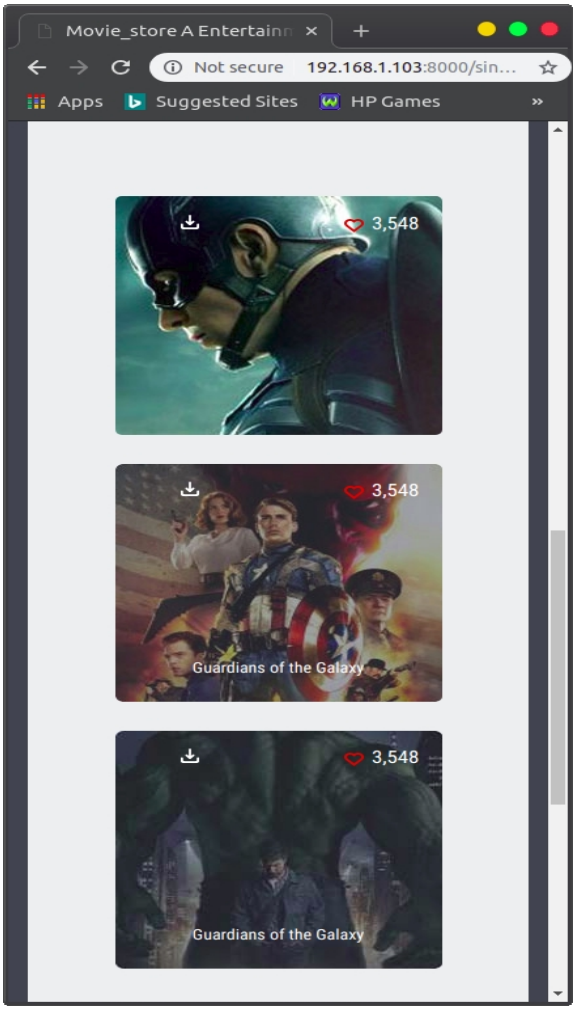


Figure 2.1 Singlemovie1

- **Login & Registration Page**

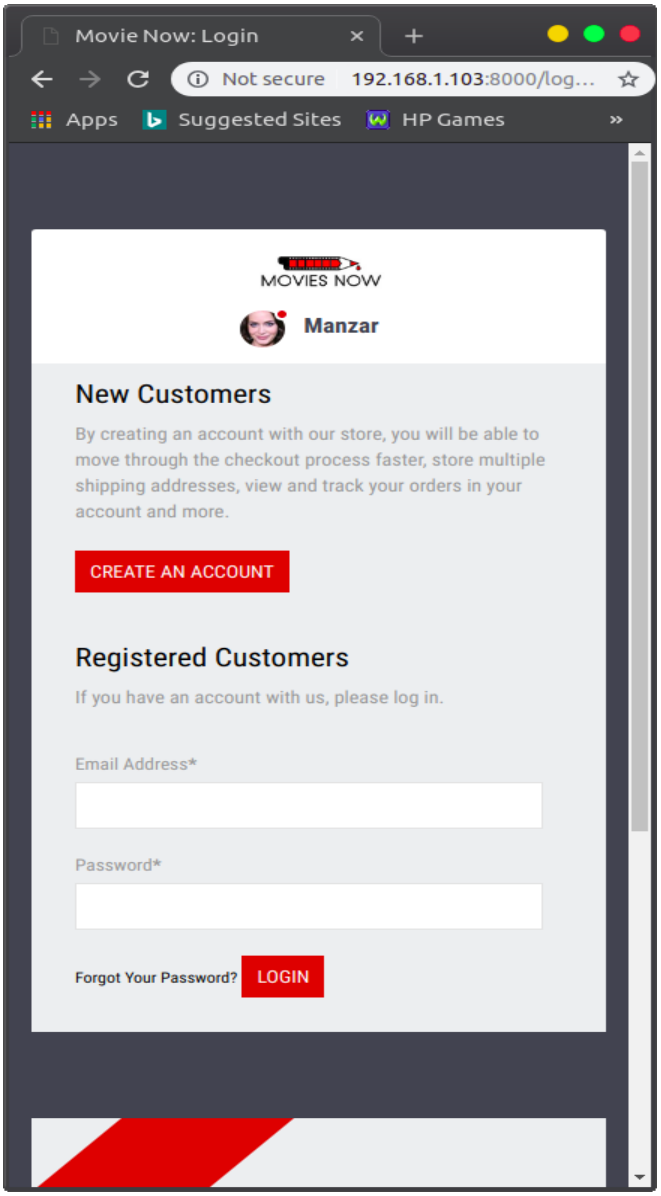


Figure 3.1 login page

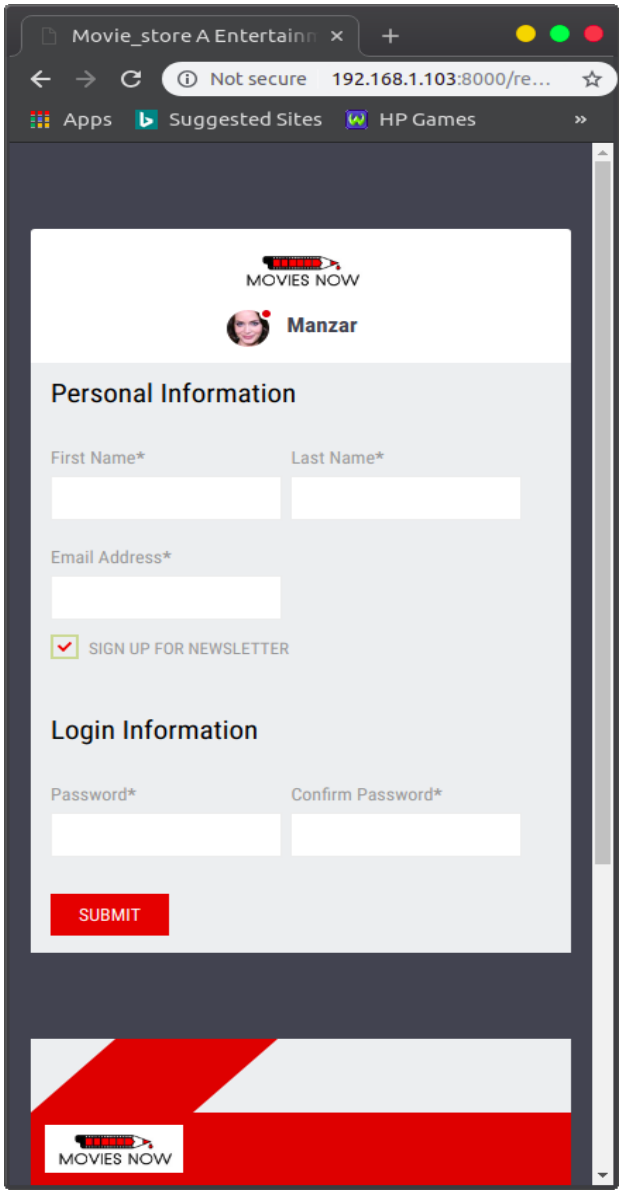


Figure 3.2 Register page

- **Home Page(Desktop view)**

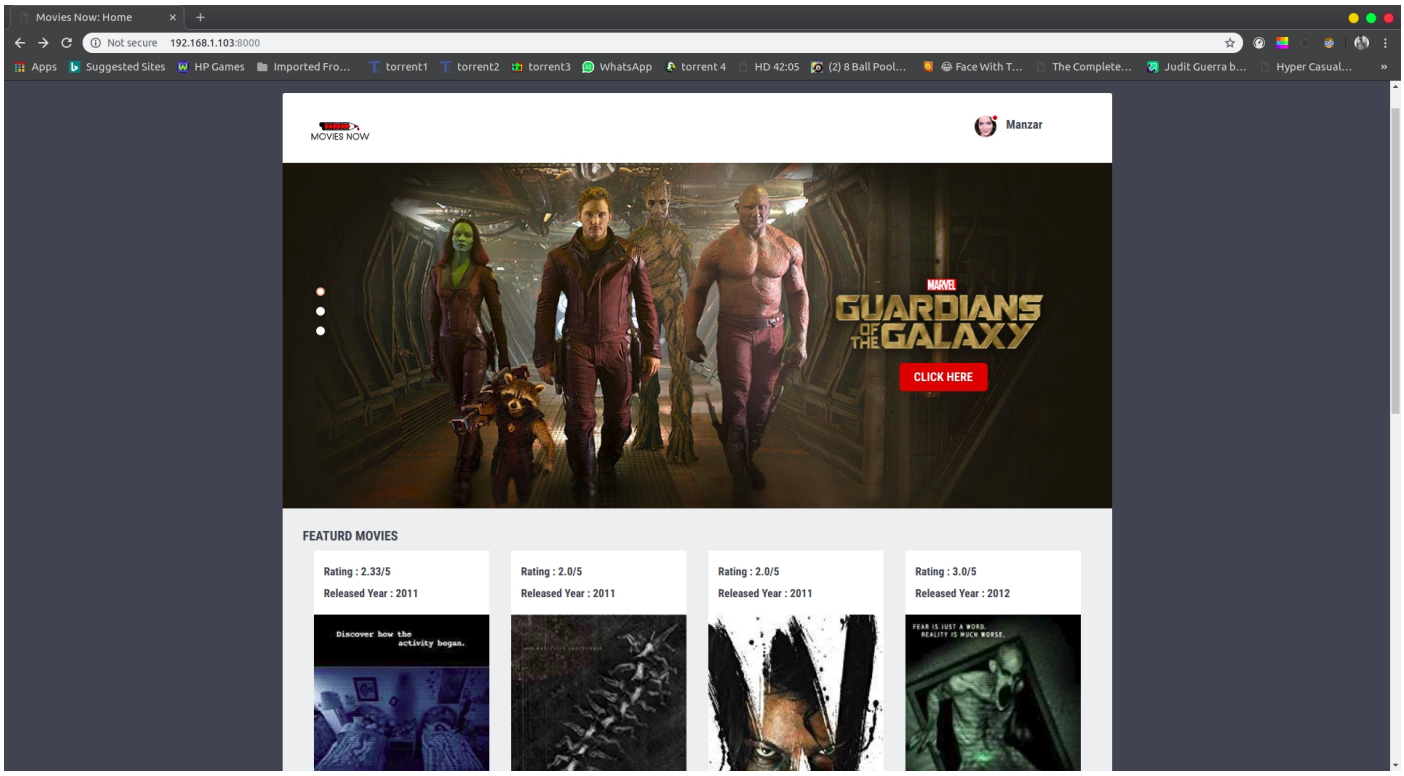


Figure 4.1 HomePage1

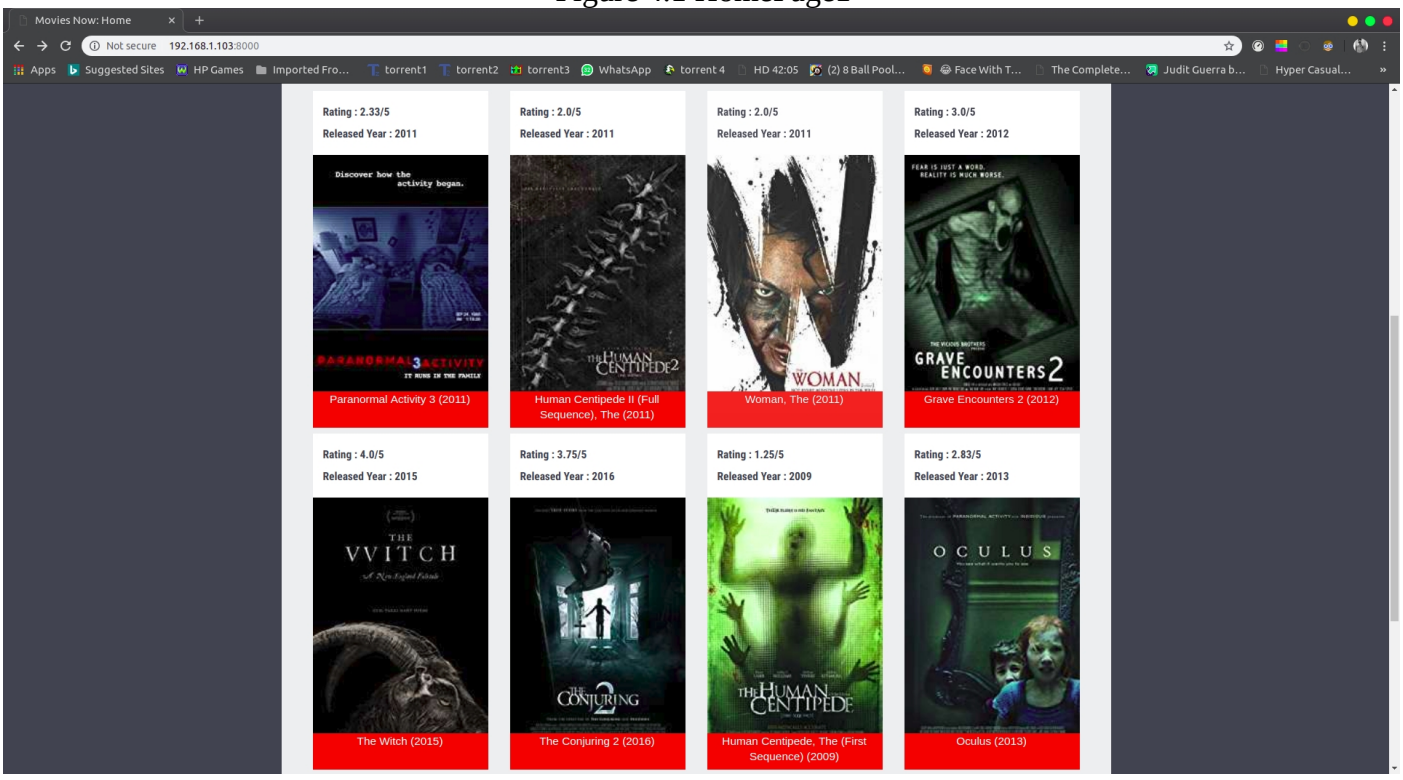


Figure 4.2 HomePage2

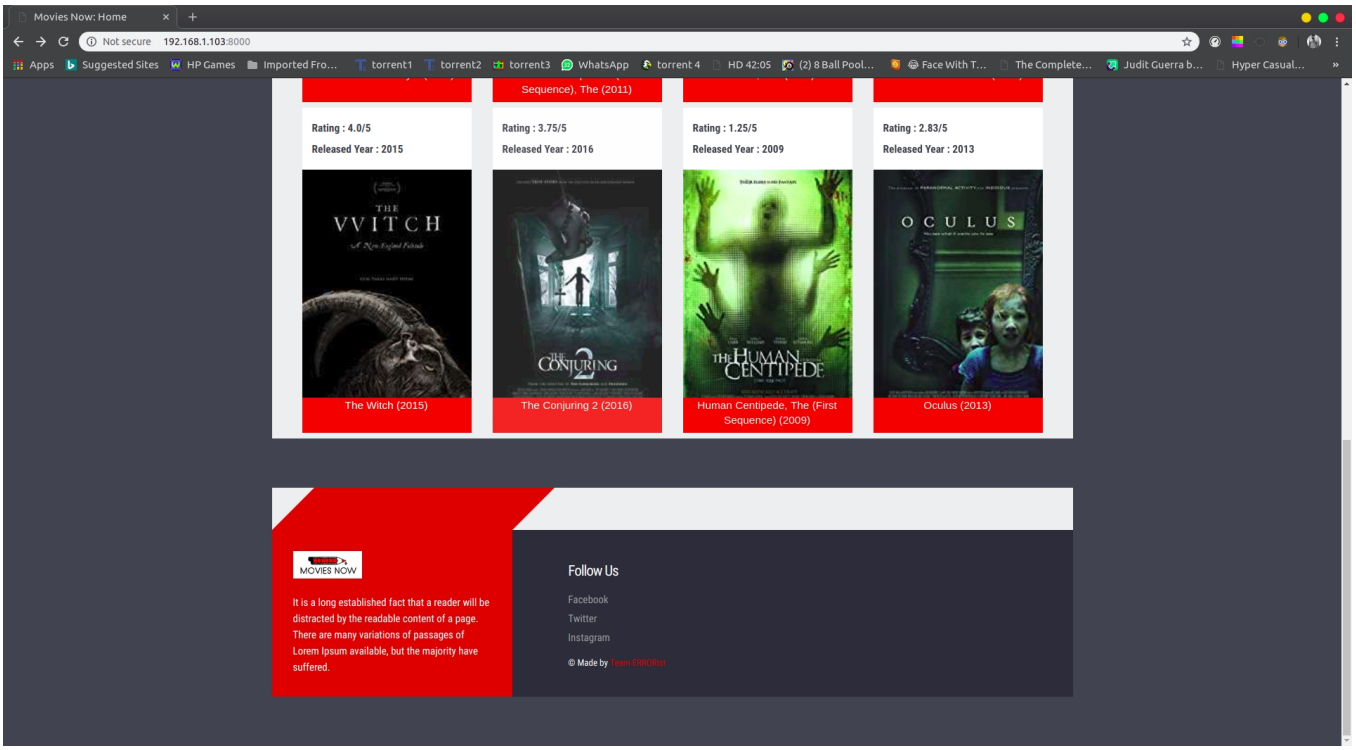


Figure 4.3 HomePage3

- **Single movie(Desktop View)**

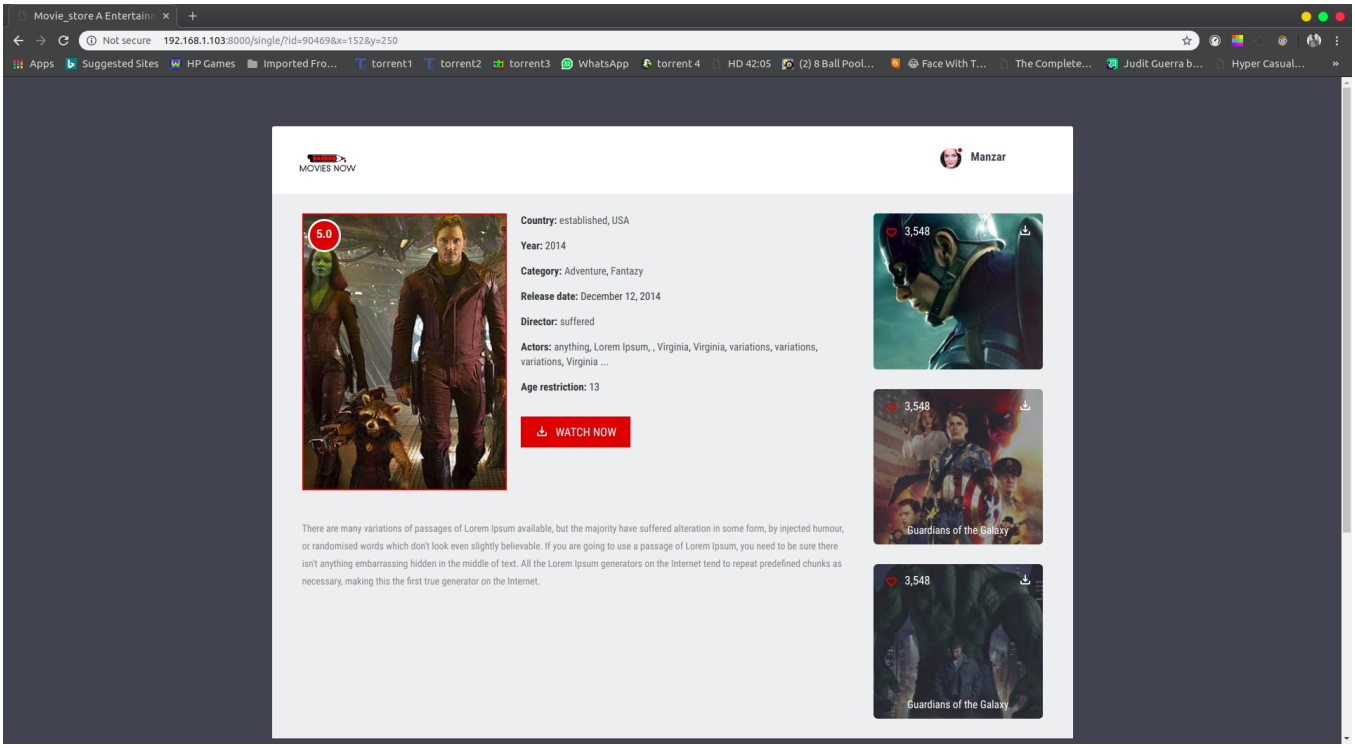


Figure 2.1 Single movie

- **Login & Register(Desktop view)**

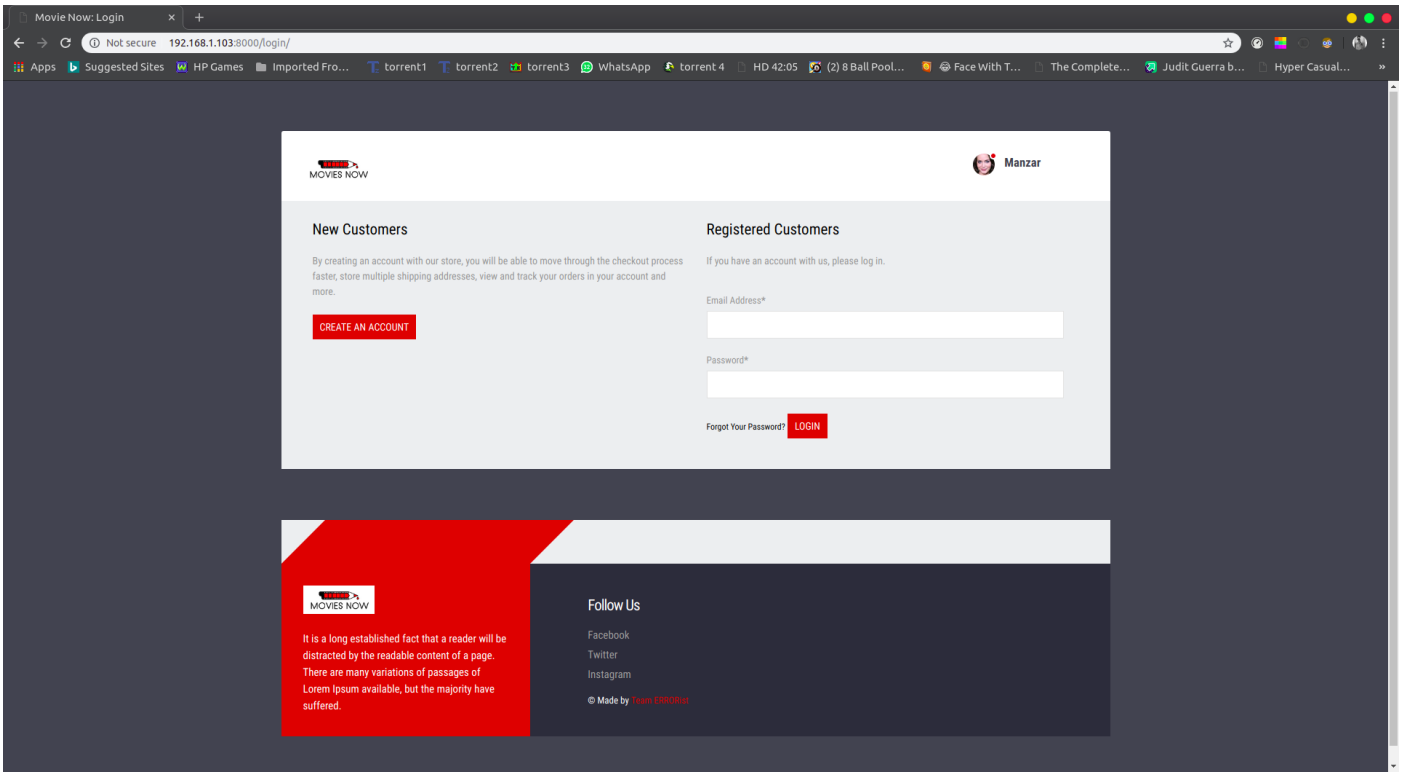


Figure 3.1 Login Page

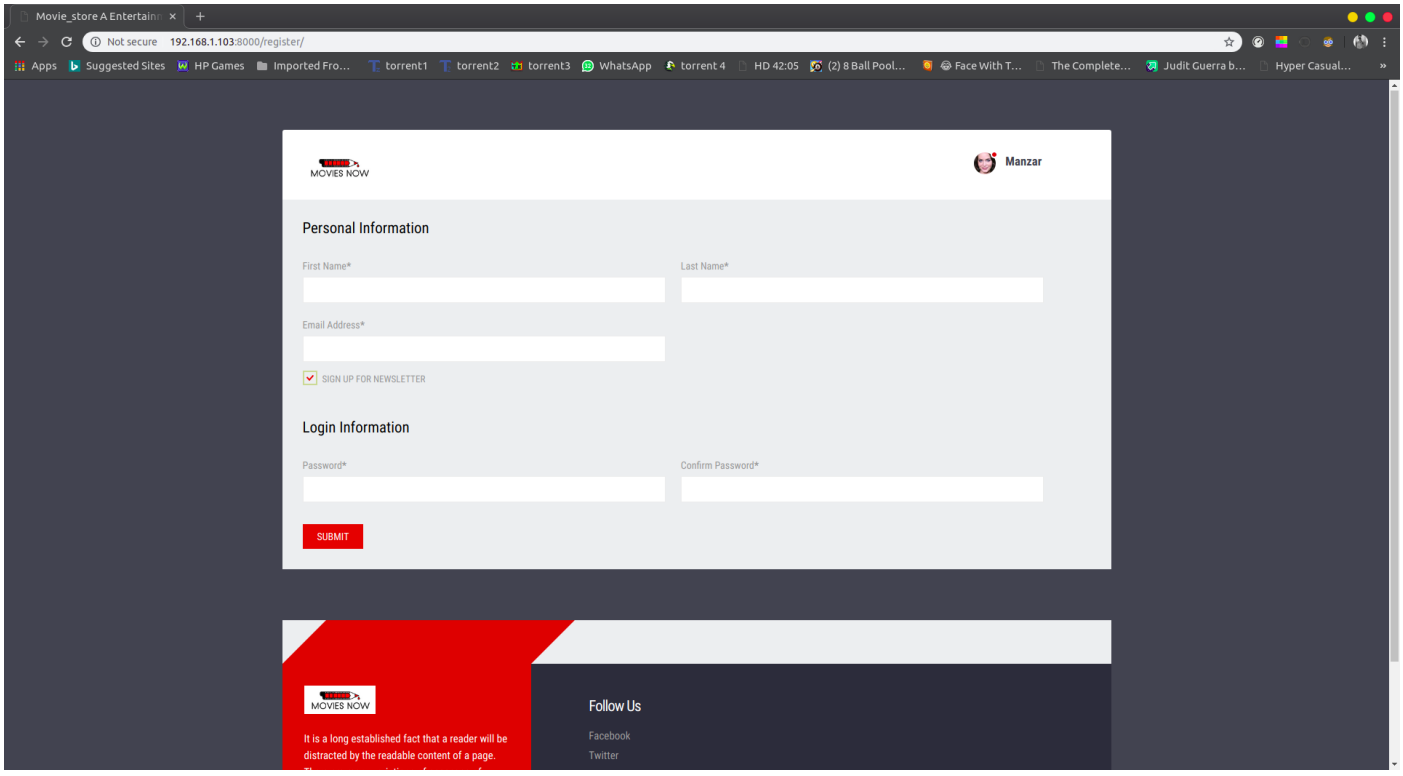


Figure 3.2 Registration page

CHAPTER 5

CONCLUSION

5.1 CONCLUSION :

A recommendation system has been implemented based on hybrid approach of collaborative filtering engine and context based engine. We have tried to combine the existing algorithms for recommendation to come up with a hybrid one. It improves the performance by overcoming the drawbacks of traditional recommendation systems. It describes the conventional Content, Collaborative Filtering and Context Filtering recommendation approaches along with their precision, recall and accuracy parameters. This paper has presented a number of utilized evaluation metrics, from which some were used to measure quality, while others to measure performance. Recommender systems make the selection process easier for the users. Hybrid recommendation engine is a competent system to recommend Movies for e-users, whereas the other recommender algorithms are quite slow with inaccuracies. This recommender system will assuredly be a great web application, which can be clubbed with today's high demanding online purchasing web sites. Our approach can be extended to various domains to recommend books, music, etc.

5.2 FUTURE SCOPE:

In Future the user will be able to search Movies directly into the website.Insist of linking the Movies to gets IMDB page we will implement our own Movies databases.

REFERENCE:

[1] IEEE paper related to Recommender System
"https://cseweb.ucsd.edu/classes/wi14/cse152-a/rrecommender_system.pdf"
[2] https://www.youtube.com
[3] https://surprise.org