CS108 Project - Movie Mania

by

Tejas Narendra Chaudhari
23B0932

28 April, 2024

Contents

1 Objective

This project aims to scrap movie details from leading websites like IMDB and Metacritic and display them on a website which we created.

2 Project Setup

- To run the code for scrapping, your device should have a chromedriver installed.
- There should be support for Jupyter on your system
- You should have the following python modules:
  - selenium
  - bs4
  - requests
  - re
  - time
  - random
  - django
  - json
  - django-cors-headers
  - fuzzywuzzy
- To run the react app, you should have node.js and npm configured on your device.
- After unzipping the foder, you wil have two sub-folders.
  - WebScrapping: This folder has the code for webscrapping in the form of metacritic.ipynb and imdb.ipynb
  - MovieMania
- To run the server and the react app, first create a new terminal and navigate to the Backend folder. Now run the command "python manage.py runserver"
- To run the react app, in the terminal inside the MovieMania folder run the commands
  - npm install
  - npm start

3 Development Process

3.1 Web Scrapping

First step was to learn the concept of HTTP requests and python libraries that help in web scrapping, namely:

- Requests
- Beautiful Soup
- Selenium

Apart from this, I also had to learn about proxies.

After getting versed adequately with these modules, I selected the IMDB Top 250, which is a page for the top 250 movie s on IMDB based on popularity. I scrapped this page using Selenium and stored a list of links to the individual movie pages on IMDB.[1]. I also created a JSON database for the same using JSON library.

Later on, I went to the Metacritic page to extract critic reviews and movie trailers. On both the IMDB page and the Metacritic page, I faced problems in extracting the Movie Trailer data using Beautiful Soup, probably due to the possibility that videos on these pages are dynamically rendered. So, I had to do the job with Selenium, which was slightly unfavourable because Selenium Webdriver is too slow even with adequate internet and computational facilities. I identified the pattern of URLs on the Metacritic pages, which enabled me to directly go to each of the movie pages and review pages.[2].

For the database, I initially decided to go with plain JSON. I thought I could change the database in the future as per needs. But luckily, for all the functionalities that I used in future, JSON sufficed.

3.1.1 Challenges Faced And Solutions

The requests sent by the Python code using the requests module were getting rejected [Response 403]. As suggested by Mr. Sabyasachi, who was the T.A. for this project, I used scrappy.io and set up an API to generate proxies.

Initially, I was using Selenium to scrape the individual page of the IMDB Movies. I did not know about XPATHs then. So I was trying to uniquely locate elements purely based on their tags and classes, in which I was unsuccessful because Selenium allows only one class name as a filter in the find_element / find_elements function. Later, I realised that there is also the XPATH method. But, I could not successfully extract all the data using this method due to some unidentified error which said "No Element Exception".So I switched to doing the task using BS, which does not have the XPath feature as a downside but allows you to file using multiple class names and attributes.

I was getting NoElementFound exception when scrapping Metacritic website using Beautiful Soup. So I used Selenium for the same and it worked.

Web scrapping using selenium used a lot of time. It cumulatively takes about 7.5 hours of time to run both the source code files. But the code works and the wait is worth it.

[1]Refer to imdb.ipynb in the Scrapping folder

[2]Refer to metacritic.ipynb

Figure 1: 162 min runtime

Figure 2: 172 min runtime

3.2 Front End

I used the Bootstrap framework and the snippets available on their webpage to style my webpage and create components like navbars, footers, cards, etc. I have used the React Library of JavaScript. I decided to use React because of the following reasons:

- Component-based architecture
- Smooth user experience as a result of Virtual DOM.
- Easier to organise.

I used the react app to create template HTML pages for:

- Home Page: It is the page where you land on. It also has a Recommendation component, which recommends Movies to all users (whether the user is logged in or not). However, the recommendations get personalised for users who are logged in.
- Login cum SignIn Page: It is the page which allows the user to log in. This page has a single form for login and sign-in, which functions differently depending on whether the username exists in the model.
- Movie Page: This page is populated by the Movie Details and allows you to rate the movie.

Entire code for the front-end can be viewed in the Front End Folder.

3.3 Backend

I used the Django framework to handle the backend. I chose Django for the following reasons:

- Django is a Python library. Whereas node.js, which I could have used, is based on Java Script. I feel more comfortable coding in Python.
- High-level framework comes with a lot of built-in features.

3.4 Integration

I used Fetch API for integration. I have only used GET and POST requests for this project. I also used the FuzzyWuzzy library in Python for implementing the Search Bar. The FuzzyWuzzy library in Python provides functionalities for approximate string matching. It helps you identify strings that are similar but not necessarily identical.

4 Working Of The Website

4.1 Home

On hosting the website, you will land on the **landing page**. The landing page has

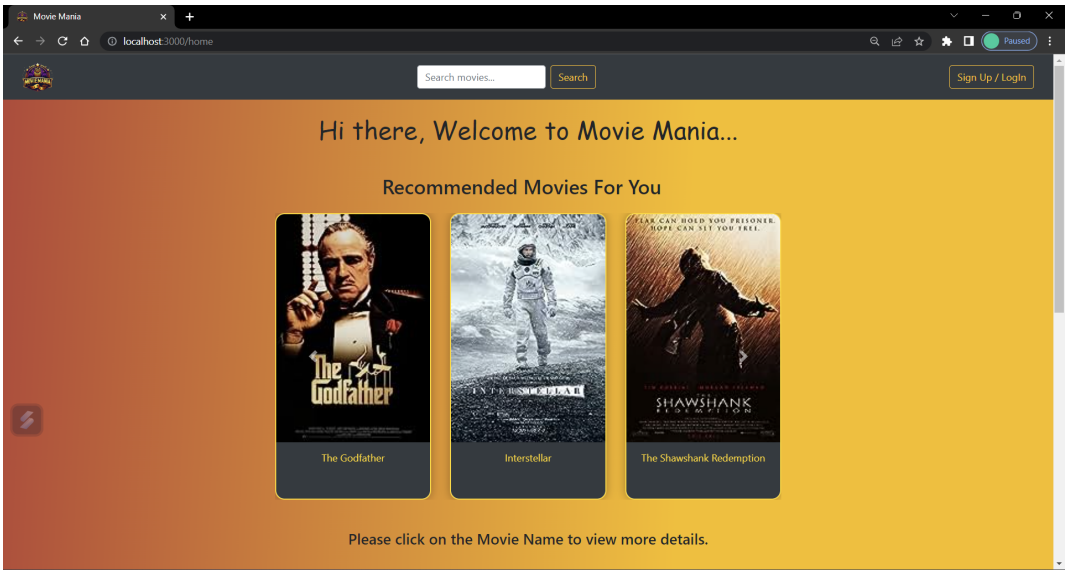


Figure 3: Landing Page

following features:

- **Navbar**: it consists of
  - Home Buttom: Top Left
  - Search Bar: Middle
  - Singin/Login Button: Top Right
- **Recommendation Carousel**: It is a caurosel of six movies. Since the user is not logged in, this carousel is not personalised. It just shows the six most popular movies as per IMDB. You can click on the Movie Name to goto the Movie Page.
- **Footer**

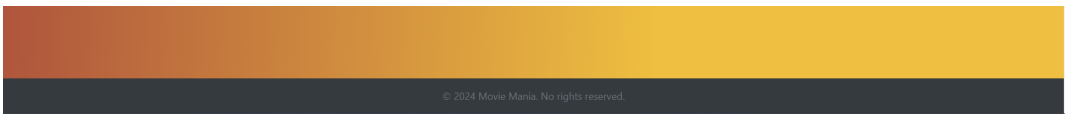


Figure 4: Footer

4.1.1 Home Button

On clicking on the **Home Button** on the top left, you can return to the website's home page from any page.

4.1.2 Search Bar

You can write the name of any movie on the search bar and land on the movie's page whose name matches the most with the given string.

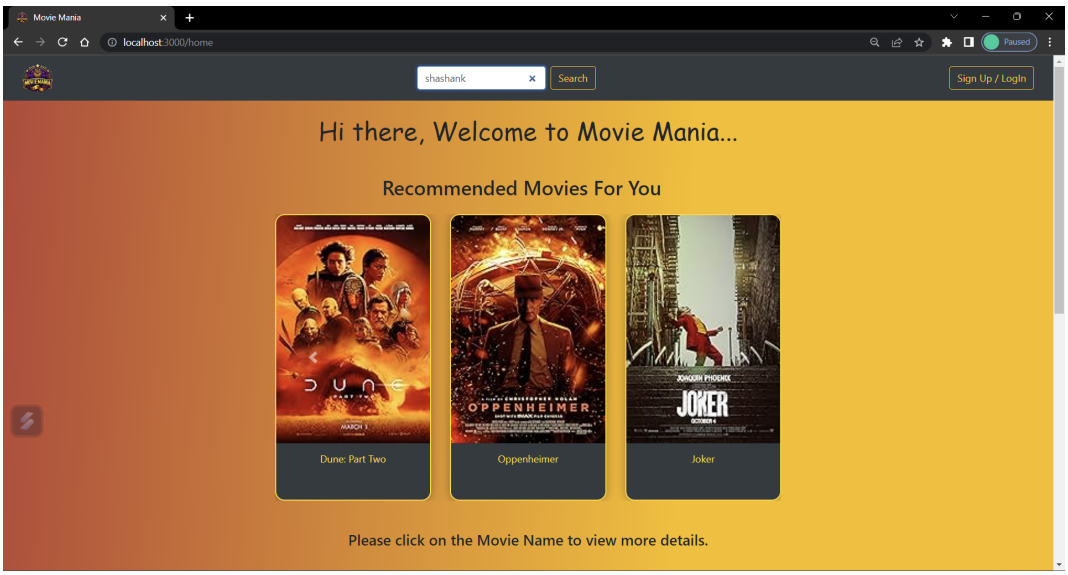


Figure 5: Input given As "shashank"

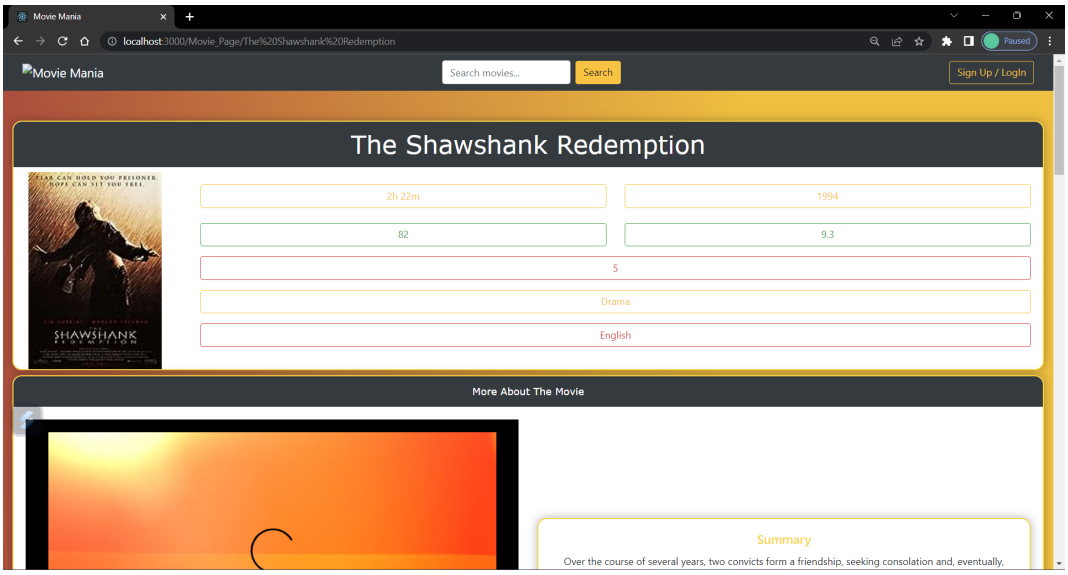


Figure 6: We get redirected to the page of "The Shawshank Redemption", whose name sounds like the given input.

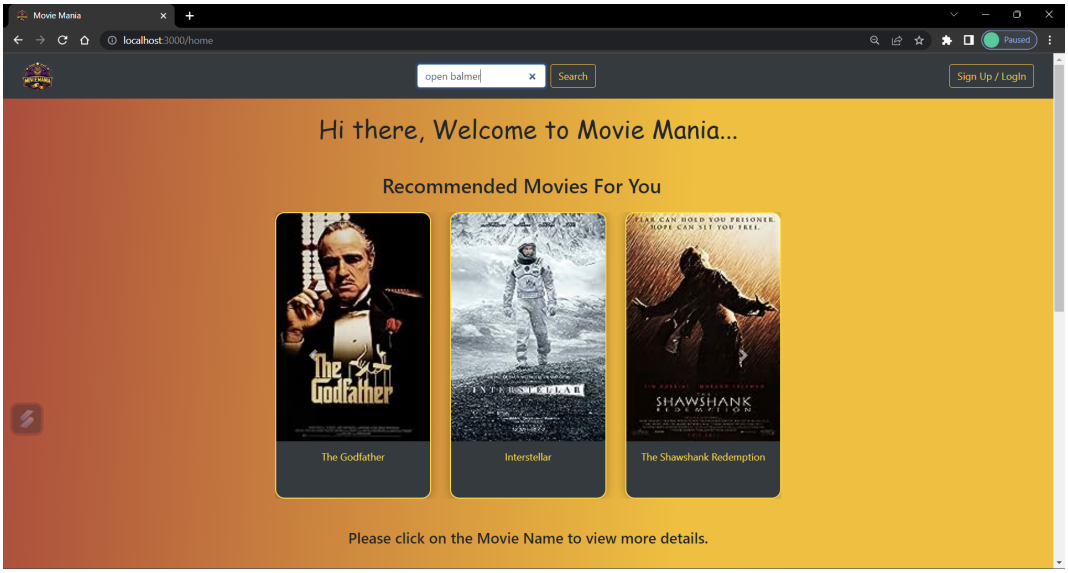


Figure 7: Input given As "open balmer"

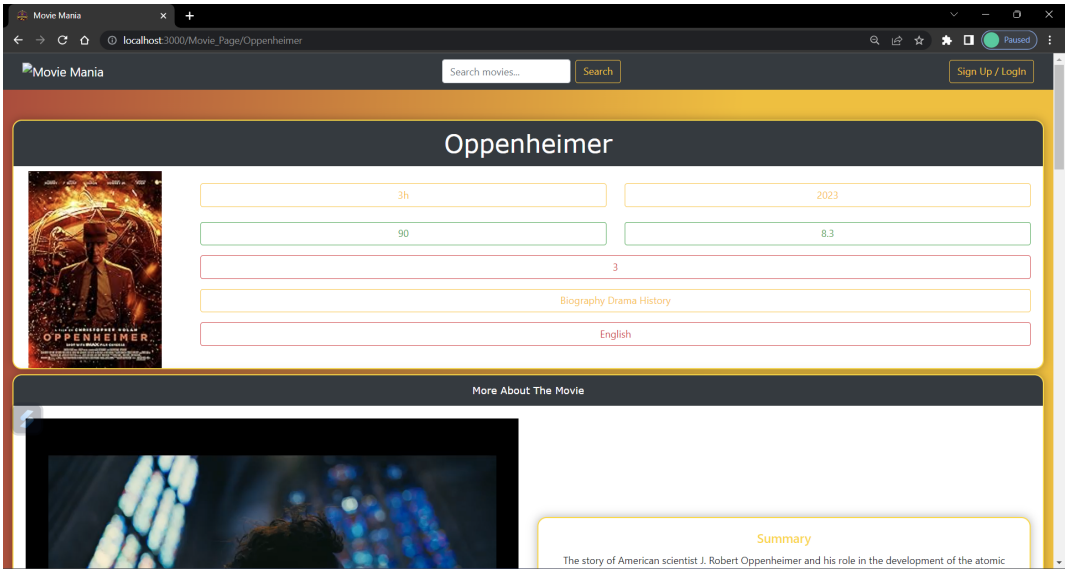


Figure 8: We get redirected to Oppenheimer movie page

4.2 Movie Page

The movie page of each movie is populated by the details of each movie. It has the following details:

- Movie Name
- Movie Release Year
- Duration
- IMDB Rating

- Metacritic Metascore
- Movie Mania Rating
- Genre(s)
- Language
- Summary of the movie
- Movie Trailer
- Directors
- Writers
- Cast
- Critic Review with the name of Critic

The movie page also has a rating bar through which you can rate the movie. Users can rate movies only when they are logged in. If the user is not logged in and tries to give a rating, the user will be taken to the **Login Page**.

Also, if the user has already rated the movie and later updates the rating, only the latest rating of the user is considered.

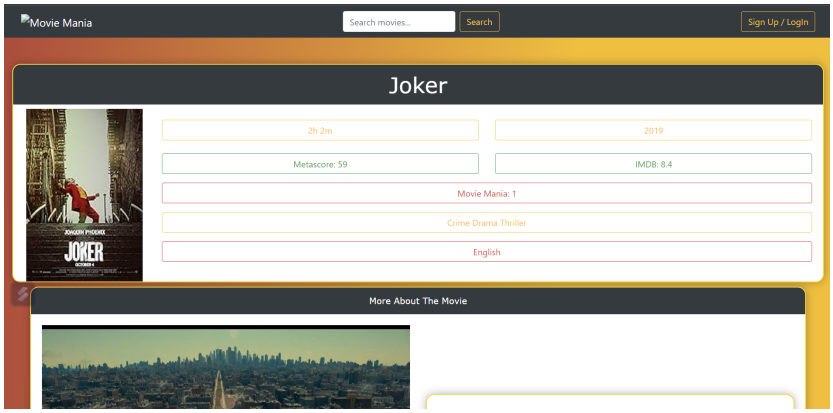


Figure 9: Basic Details

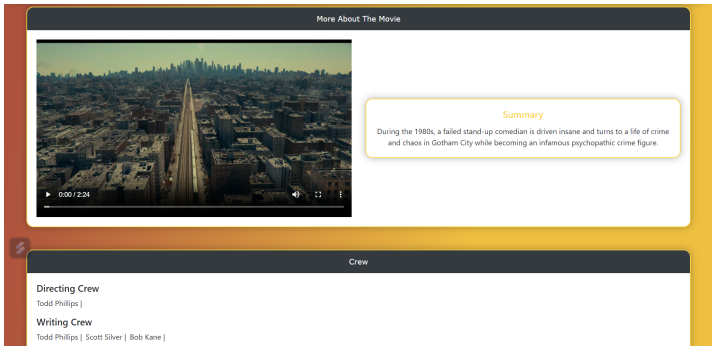


Figure 10: Trailer and Summary

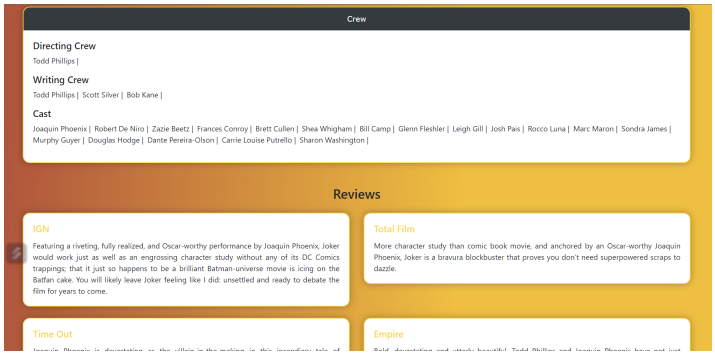


Figure 11: Crew

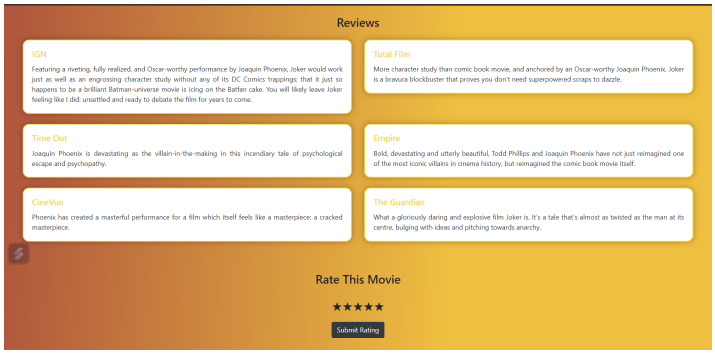


Figure 12: Reviews And Rating Bar

4.3 Login Page

It has a single form which allows you to log into the database. You can go back to

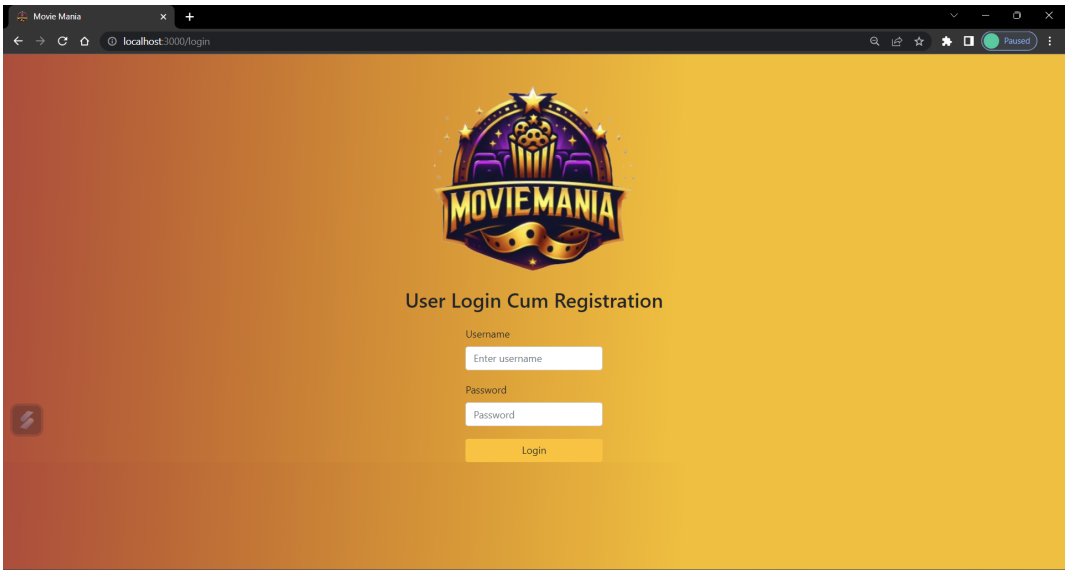


Figure 13: Login Page

home using the Movie Mania logo button.
After you login successfully, you are directed to the Home page. Notice that the recommendations changed.

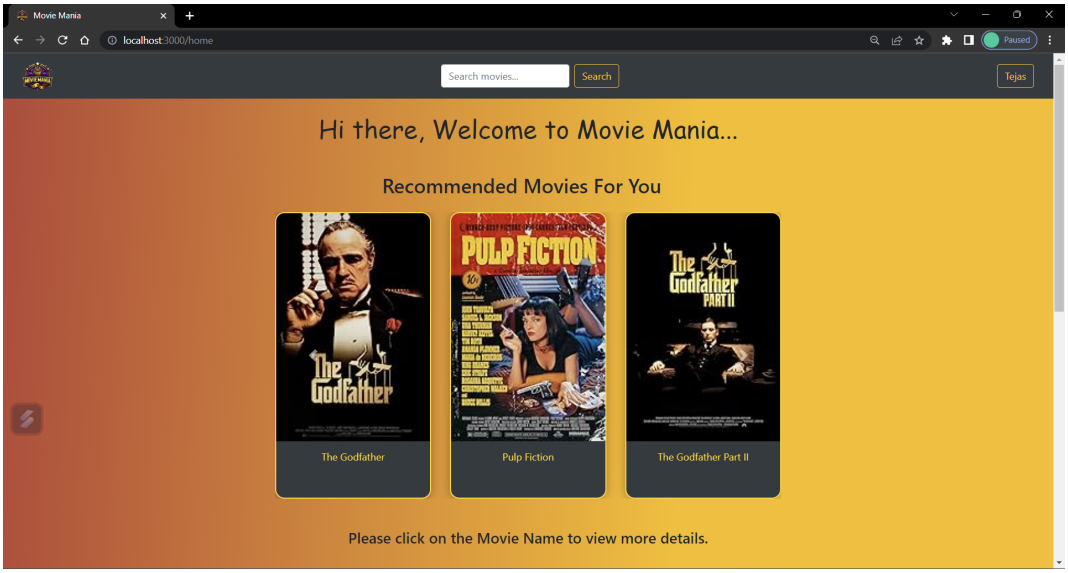


Figure 14: Home Page when User is Logged In

5 Database Structure And Recommendation Algorithm

The database that we use to render content to webpages it the json file which was created while scrapping i.e. **movies.json**.

Apart from this, there are 4 models created in Django:

- **Movie**: It has a column for the name and a column for a rating of the movie. It is being used to display the movie mania rating of each user
- **Rating**: It has three columns: userID, movieID and rating. This is used to keep a record of which user rated which movie. It has foreign key relationship with user and Movie models.
- **user**: It stores the username and password of each registered user.
- **genreRating**: It is used in the Recommendation algorithm. It has four columns, namely genre, rating, movieName and userId.

When a user rates a movie with genres, say Action and Comedy, a rating of 4, we make a new row for each genre with userID, movie name and rating i.e. 4.

Recommendation Algorithm Logic

When we have to recommend a movie to a user, we iterate through all the movies in the database and calculate a movie score.

$movieScore = 60 * genreScore + 20 * popularityScore + 20 * imdbRank$

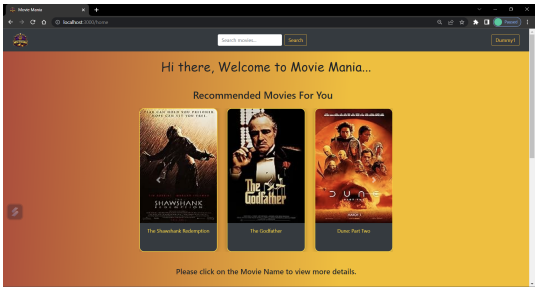
popularityScore is calculated on the basis of teh IMDB Rating. imdbRank is the rank of the movie on IMDB Top 250. To calculate the genreScore, we use the genreRating model. For a given user and a genre, we find to total sum of all the ratings from the table and divide it by 5 times the total number of ratings, lets call this quantity genreRatingValue. Now, we store the data of the user in a dictionary with key-value pairs as the genreName:genreRatingValue. (Default value of genreRatingValue is 0).

Now, when we are iterating through the list of movies, we add the genereRatingValue for each movie genre by looking up the value from the dictionary we created. We divide this sum by the number of genres and this gives us the genreScore.

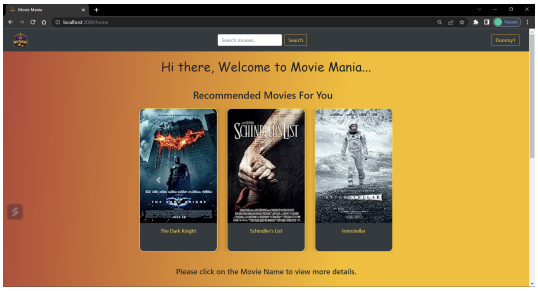
So now we have the movieScore for each movie for the given user and thus, we send the top 6 movies for getting recommend to the user.

Demonstration of Recommendation Algorithm

User lands on home page: IMDB Top 6 recommended Now we gave 1 Star Rating



(a) Slide1

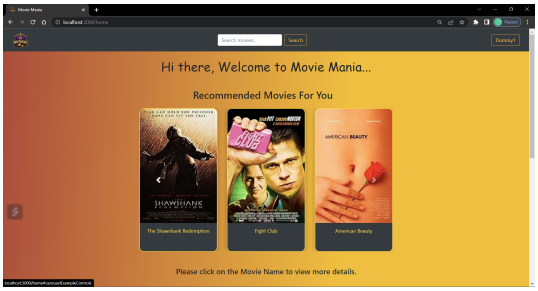


(b) Slide2

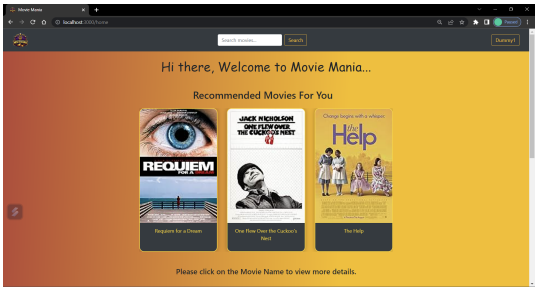
Figure 15: Step 1

to Dune 2. The genre of Dune 2 were SciFi Adventure And Drama. Now it stops recommending Intersteller which is of similar genre.

Now it recommends these movies:



(a) Slide1

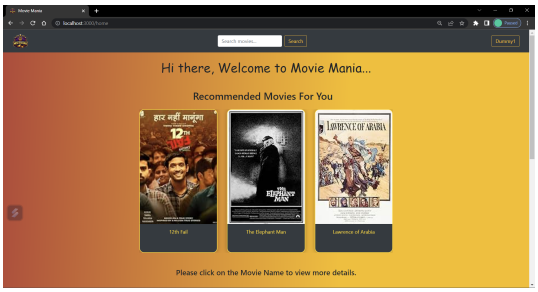


(b) Slide2

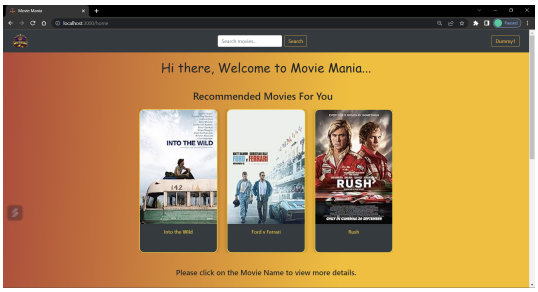
Figure 16: Step 2

Now we rate Dangal, whose genres are Action, Biography and Drama. But we also rated Dune 2 which had action and drama 1. So it will start recommending more movies of genre action and biography.

Here note that 12th fail has does not have genre action, yet it is recommended first as it has higher rank on IMDB and also more IMDB rating.



(a) Slide1



(b) Slide2

Figure 17: Step 3

6 Customisations

- **Scrapping**: Additional details movie trailer, critic reviews, name of writers, etc. were scrapped
- **User Registration And Personalized Recommendations**
- **Implemented Movie Manian rating**
- **Advanced Search Bar**
- **Enhanced UI using Bootsrap**

7 References

https://www.geeksforgeeks.org/implementing-web-scraping-python-beautiful-soup/
https://scrapeops.io/web-scraping-playbook/403-forbidden-error-web-scraping/
https://www.toptal.com/python/web-scraping-with-python
https://www.analyticsvidhya.com/blog/2022/07/scraping-imdb-reviews-in-python-using-se
https://github.com/sivasahukar95/CodeStore/blob/master/Scraping%20data%20from%
20imdb.ipynb
https://www.youtube.com/watch?v=flnSmUnOIRc&t=3255s
https://www.w3schools.com/REACT/DEFAULT.ASP
https://www.w3schools.com/django/
https://getbootstrap.com/
https://beautiful-soup-4.readthedocs.io/en/latest/
https://www.freecodecamp.org/news/javascript-fetch-api-for-beginners/#:~:
text=The%20Fetch%20API%20allows%20you,use%20the%20async%2Fawait%20syntax.
https://pypi.org/project/fuzzywuzzy/

8 Drive Link

https://drive.google.com/file/d/1QL9sg07DQoMQPNnL1dFHjo5UFdhG4vzd/view?usp=
sharing