ABSTRACT

My Movie List is a movie recommendation system that recommends movies and tv shows to the user based on their watch history and personal preferences. It recommends movies and series based on collaborative filtering. The need for such a system is in demand due to the increase in streaming services during the pandemic. In this age of digitized media, a recommendation system that caters to a user's preference is highly needed. The main aim of the project is to provide a one-stop solution for the user's entertainment needs. The project is able to serve recommendations to the user based on the user's input as well as the user's past behavior and also is able to stream most of the recommendations.

Keywords: *recommendation system, collaborative filtering, cosine similarity, movie database, API, web application, KNN*

Table of Content

List of Figures

List of Tables

List of Abbreviations

API: Application Programming Interface

EDA: Exploratory Data Analysis

IDE: Integrated development environment

IMDB: Internet Movie Database

KNN: k-Nearest Neighbors

TMDb: The Movie Database

Chapter 1: Introduction

1.1. Introduction of the Project

My Movie List: Movie Recommendation System is a web-based application that recommends movies and shows to watch depending upon the user's preferences. The preference of the user is determined by their choices, watch history, and reviews from similar users.

Streaming has been a hot topic during this pandemic. Several streaming services have seen an increase in popularity as a result of the outbreak. There are a lot of movies and series available on a variety of different services. Different genres of films appeal to different people. It's difficult to find recommendations that cover all platforms. The web application presents users with specific movies they prefer from all the services and websites implemented through a collective database. All of the social media platforms used on a daily basis employ the same strategy to feed people with various advertisements depending on things previously liked and disliked.

The initial start of the application provides an interface to the user where they can see the trending movies, top movies as well as some movie recommendations. A user can search for some movies and then see the details about that movie as well as see recommendations. In the final version, the user will be able to log in, and the application will use the user's past watch history and server recommendations based on it. There will also be a dashboard containing a statistical analysis of the different attributes of the movies.

The main goal of this project is to provide users with information regarding different movies and tv shows and recommend movies/tv-shows that meet their preferences.

1.2. Problem Statement

The time we are living in right now is very different. It's an era of "abundance". We are surviving through a pandemic were going to theaters to watch movies is not feasible at the moment. Due to this reason, people have started to consume content through digital platforms like Netflix, Hulu, HBOMAX, Amazon Prime, etc. Currently, some existing systems do not have highly accurate recommendations, or their UI is not easily navigable. There is no single platform that incorporates and lists all the movies from all the different

streaming services nor do they give the user the choice to view a detailed breakdown of their preferences and recommendations. This system aims to address these problems.

1.3. Objectives

- To develop a web-based movie recommendation system based on users' historical data by using collaborative filtering.

1.4. Scope and Limitation

The aim of our project was to make it easier for people to choose movies based on user reviews, and watch history, current trends, and preferences. Our project delivers an interface for users to easily look for movies based on their preferences. Users are able to get recommendations based on previous ratings as well as based on a movie. Our web application will be the go-to stop for a user to meet their entertainment needs.

One of the main limitations is that the user will have to manually add information related to their watch history which does not allow user-specific recommendations to new users. Due to the large size of the data, performance is hampered. Only movie recommendations are served because serving shows recommendations are affected by individual episodes which makes it a bit more complex. Moreover, datasets catering to show recommendations are hard to come by. This app will mainly cover Hollywood movies due to the lack of information regarding Kollywood and Bollywood.

1.5. Development Methodology

The project used evolutionary prototyping. By using this approach the team was able to start small and then continuously improve and add functionality based on the project requirements. Daily standups were carried out which helped in building transparency among the team members and everyone stayed updated.

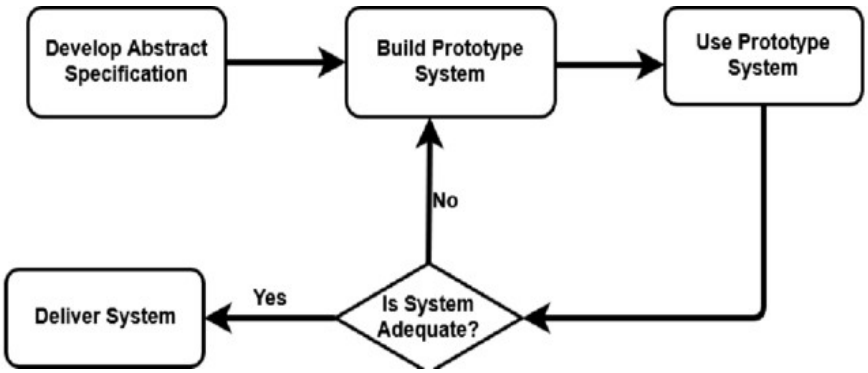


Figure 1.1: Evolutionary Prototyping

1.6. Report Organization

The report of the project is based on the following format.

Chapter 1: Introduction: In this section, the main points discussed are the overview, the background of the project, the objectives of the project, the scopes, and the limitations of the project.

Chapter 2: Background Study and Literature Review: In this section, a description of fundamental theories and general concepts related to the project has been discussed.

Chapter 3: System Analysis: This section describes the general architecture of the system. It also describes various entities working on the system. It provides the overall working mechanism of the system. It also contains the requirements and feasibility analysis for the system.

Chapter 4: System Design: System design describes the diagrammatic description of the system. Sequence diagram, Activity diagram, Use-Case diagram, and State diagram describe the entities involved in the system, their activities in the system, and the relation between the entities. This section also contains the pseudocode for the algorithm used.

Chapter 5: Implementation and Testing: This section describes the tools used and the implementation details of the project. It also consists of different test cases used to test different units of the project.

Chapter 6: Conclusion and Future Recommendations: This section describes the progress till now and what the plans are for the future.

Chapter 2: Background Study and Literature Review

2.1. Background Study

This project was chosen due to the rise in demand for streaming services due to the pandemic. As this project uses modern algorithms and frameworks for its construction, it helped in understanding the current market demand of technologies used to create similar web apps.

2.2. Literature Review

Anshu Sang [1] has represented the recommendation system. Using the rating and similarity among the two users; the system recommends an item to the user for the decision making. Then separate the movie data set into an unrated and rated sample set with the help of the KNN model. It can recommend the movies to the unseen users via user registration information, and it can create new and not popular movie recommendations according to the film's history and score. The database in this approach is the MYSQL database. The registration system for a user will snap the user's external and internal behavioral characteristics, and these characteristics are stored in the user database via a login module for the user. It depicts their effective way of approach for a collaborative filtering approach using KNN.

Anurag Banerjee in [2] used a weighting technique for the text retrieval system for an item-based collaborative recommender system. Their proposed scheme has been used for effective movie recommendations. The empirical analysis on the benchmark MovieLens 100K dataset has shown improvement over state-of-the-art recommender system algorithms. Also, the performance of the proposed technique needs to be tested on different other applications of the recommender system.

Meysam Shamshiri[3] proposed a better movie recommendation system that uses a group method for handling neural networks. In the proposed method, their approach is used to solve the collaborative filtering problem. The network's trust for the users is used to decrease the prediction error of the precise user-oriented collaborative filtering algorithm. The Prediction results of the proposed model are saved in terms of precision and error, and that is compared with many standard algorithms like MLP, and Bayesian network. Their main goal was to develop a system that has better accuracy than other models. They have

divided the methodology into three parts. Preparation of the data, Preprocessing on data, and GMDH model to give the perfect output for the desired user. In the GMDH algorithm, two or more neurons are connected via a polynomial layer in which the subsequent layer is generated. Also, this can be used in modeling mapping inputs to desired outputs. The purpose of this algorithm is to find the unknown coefficients in the Voltra function series. For the evaluation purpose, they have used the Root Mean Square Error (RMSE) and Mean Square Error (MSE) methods, respectively. In the final results, their proposed method outperforms all the state-of-art methods

Mukesh Kumar Kharita [4] has implemented item-based movie recommendations. For item-based recommendations in the paper, they have used the ratings of those movies that are highly similar to the rating of the movie, which is provided by a proper user from using the item similarity weights in the item similarity weight matrix. Moreover, recommend these movies to the specific user by choosing the K most similar items with higher ratings. In the paper, the accuracy of the model is a bit decreased in comparison with contemporary recommendation models. Also, they are using movie rating as the essential element in the whole evaluation process, which can be enhanced in the future by selecting other relevant variables as well.

Rahul Katarya and Om Prakash Verma [5] came up with a new strategy for the collaborative movie recommendation system. They have used k-means and cuckoo algorithms in order to improve performance. The users have been combined into the clusters, and then the centroid is checked. Users who have lower centroid values are connected more closely. After this stage, the cuckoo algorithms are applied to find the best fitness function based upon the previous and current best solutions. This approach applied to the Movie lens dataset, which contains 100,00 ratings of the users. Furthermore, 943 users have rated around 1682 movies. For evaluation purposes, it has been shown that they have overcome all the state-of-art methods by using standard deviation(SD), Root Mean Square Error(RMSE), Mean Square Error(MSE), and Mean Absolute Error(MAE). The main drawback of this approach is in the initial stage; if the users are not clustered well, then it may affect the final evaluation.

Arpita Jain [6] has implemented the usual movie recommender system using RapidMiner. The sparsity problem has been neglected by using the rapidminer platform. Also, it permits the person's innovation to convert an accessible introduction to the system and achieve a

much-organized arrangement that is working to the datasets. They have used the Root Mean Square Method(RMSE) method to evaluate the model for the movie recommendation.

Chapter 3: System Analysis

System analysis was done to completely understand the application that is to be developed. The gathered information helped to create logical models which in turn helped in coding. The large complex project was broken into small manageable parts so each may be designed, studied, and analyzed in detail. The tools used help to transform requirement specification into implementation.

3.1. Requirement Analysis

Following are the requirements for this project.

3.1.1. Functional Requirement

System functional requirements describe activities and services that the application provides.

- Users are able to search and view details regarding any movie/show.
- Users are able to access the dashboard and see different visualizations.
- Movie recommendations are served to the user based on their liking and item ratings as well as ratings given by other users.
- Users are able to see links to where they can view the movie.
- Users are able to see visualizations based on different attributes of movies.

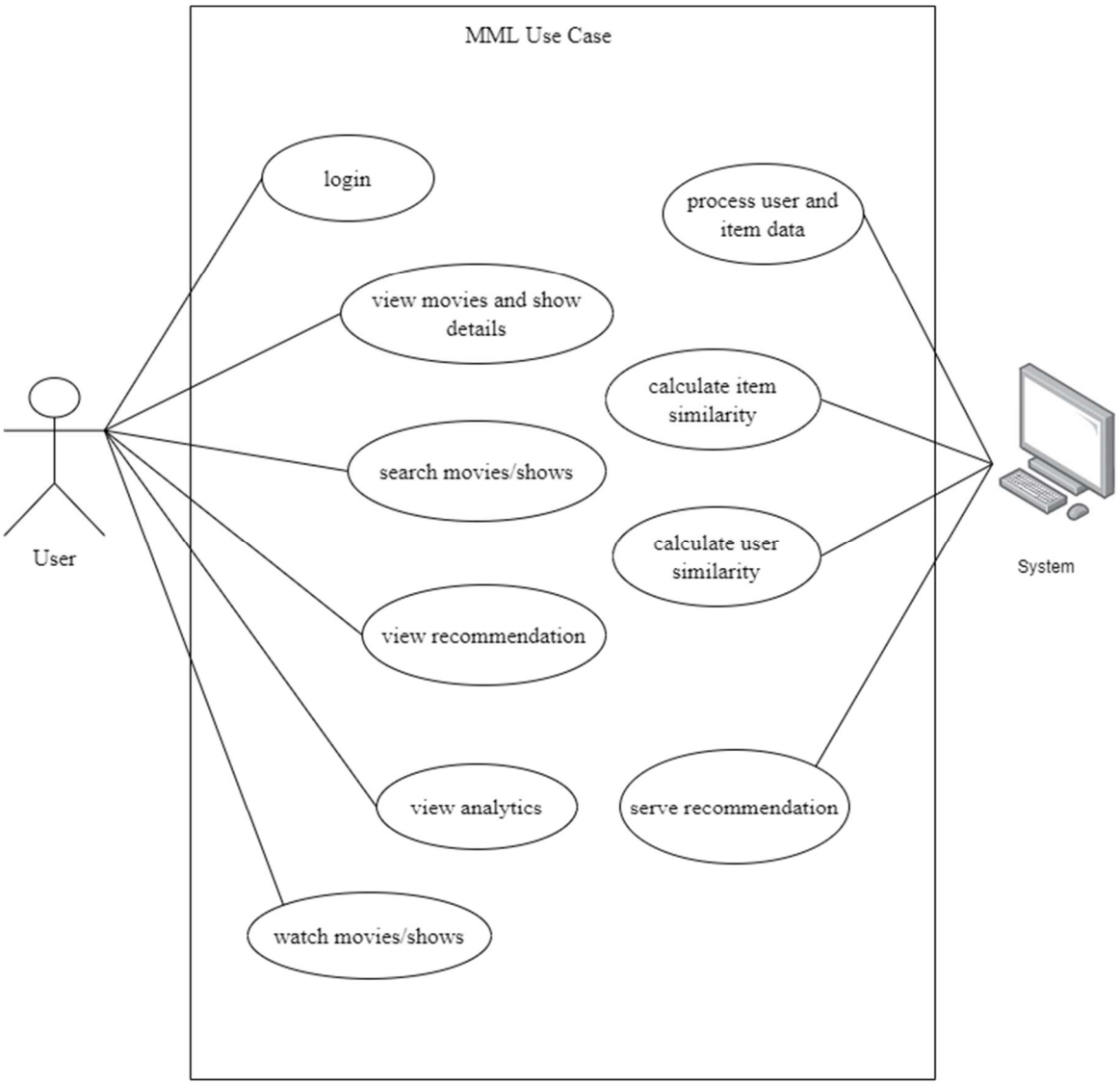


Figure 3.1: Use Case Diagram of the System

The above use case diagram describes the typical use case of the system. The users are able to login to the platform, search for movies and shows and view details such as cast, platforms to watch, get trailers, clips, etc The users are able to get recommendations based on their input parameters. They are also able to view analytics and watch recommended movies. The system is able to process user and item data, calculate similarities and show recommendations.

3.1.2. Non Functional Requirement

Non-functional Requirements are characteristics or attributes of the system that can judge its operation. The following points clarify them:

- Accuracy and Precision: The system was able to perform its process with accuracy and precision and was able to avoid problems.
- Flexibility: The system was easy to modify and errors were easily fixed.
- Security: The system is secured and has preserved the user's information.
- Usability: The system is made so that it is easily understandable and dealt with.
- Maintainability: The maintenance process can cope with any problem that arises suddenly.
- Speed and Responsiveness: The system was quick to respond to the user's requests.
- Scalability: The system is able to scale according to the needs.

3.2. Feasibility Study

Feasibility studies aim to uncover the strengths and weakness of an existing or proposed system, opportunities and threats objectively and rationally as presented by the environment, the resources required to carry through, and ultimately the prospects for the success. The feasibility study of this application had been carried out which is as follows:

3.2.1. Technical Feasibility

There is not any technical constraints that hampered our project development and resulting software. All the technologies and databases used in the making can be accessed for free. A simple prototype can be easily hosted until and unless we need a large server to handle massive requests which can be added once the full version is completed.

3.2.2. Operational Feasibility

The project was implemented in a way that allowed the functioning of recommendations smoothly. It also provides a user-friendly interface in a modular fashion. The operation of the project runs smoothly in a controlled environment.

3.2.3. Economic Feasibility

Development of this application is safest from economic hardships as our project is free from financial contribution. The technologies used can be easily accessed for free to launch the first prototype. But this advantage is a short-term one as further modification of the

application (addition of new servers, data storage, scaling of the application) would increase the price exponentially.

3.2.4. Schedule Feasibility

Our proposal document for this project had plans for developing the software by the start of the seventh semester. But University examinations hampered the predesigned timeline for our project. We completed this project by the end of March 2022. Only small changes to strategic plans had been made for the completion of this project.

3.3. Analysis

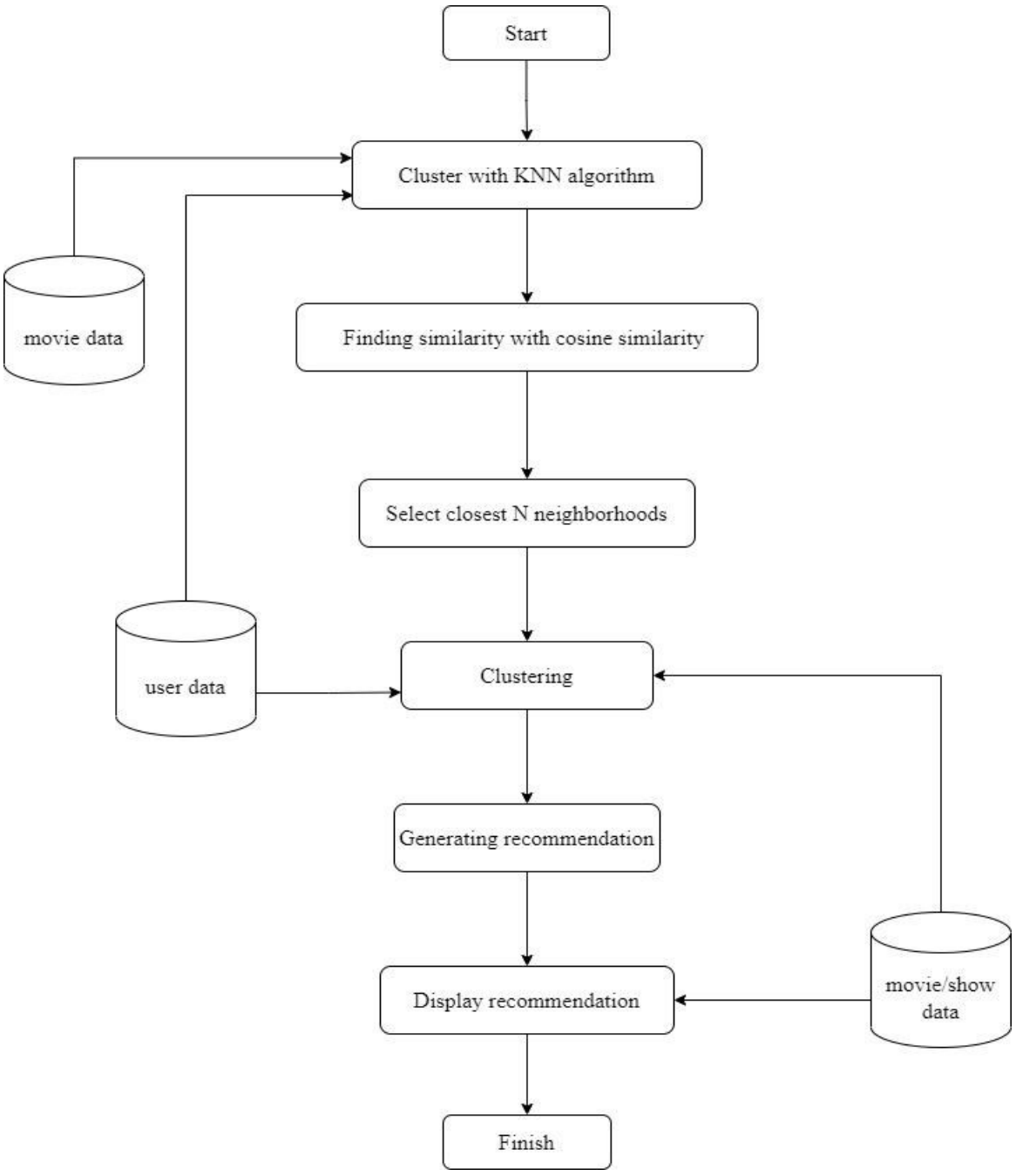


Figure 3.2: Process Model of Recommendation system

The above figure describes the process that would be ongoing on the system. User data and movie data contain the data needed to calculate the user similarity using cosine similarity. KNN algorithm-generated clusters based on similarity. The Closest N neighborhoods are selected and clustered accordingly. Recommendations are generated and are then served to the user based on those clusters.

Chapter 4: System Design

4.1. Design

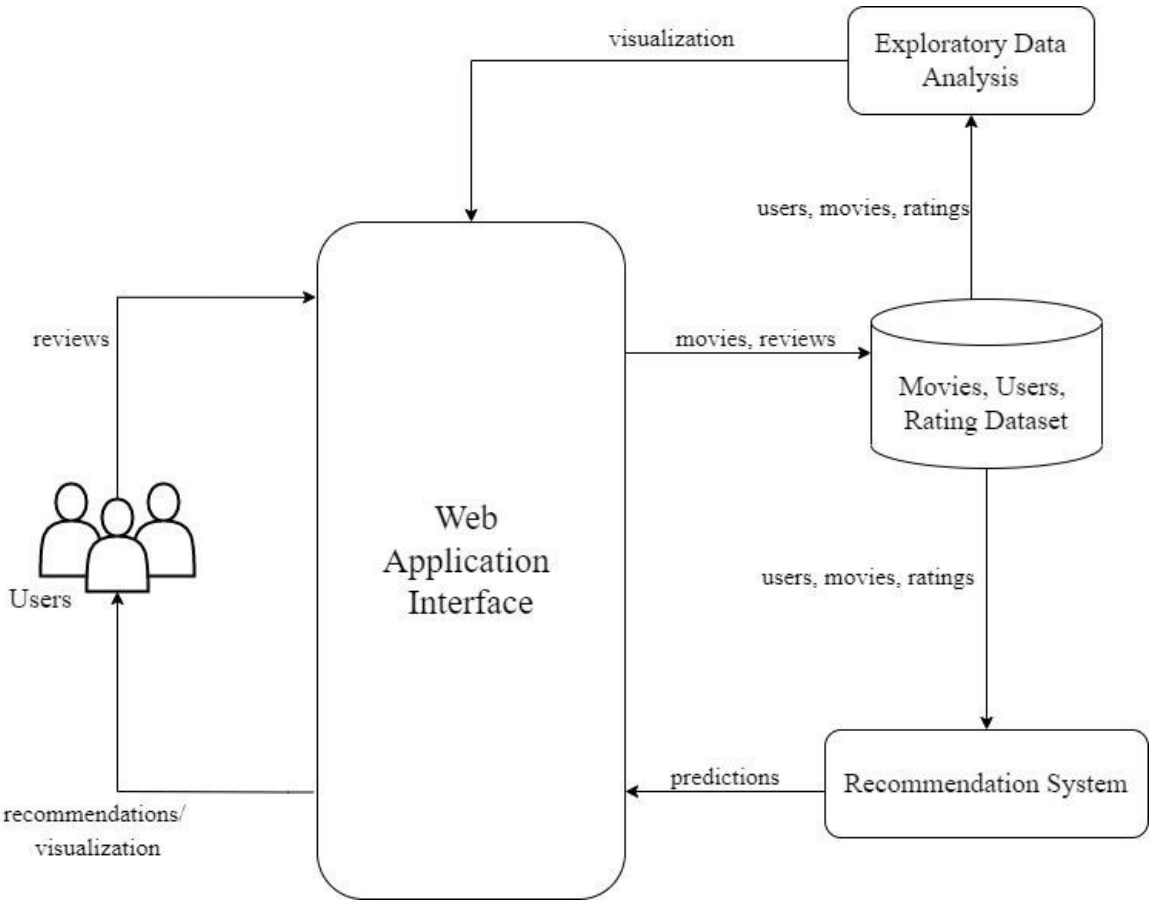


Figure 4.1: System Architecture of the Application

The figure describes the system architecture of the application. It consists of a web application interface where users interact on, a movie and review database(TMDb) which consists of data about the movies and their reviews. The user details, movies, and ratings are then fed into the recommendation system to provide predictions to the user via the web interface. The web interface will also show visualizations based on the dataset used on the project.

4.2. Interface Design

Following are some screenshots of the application in different phases.

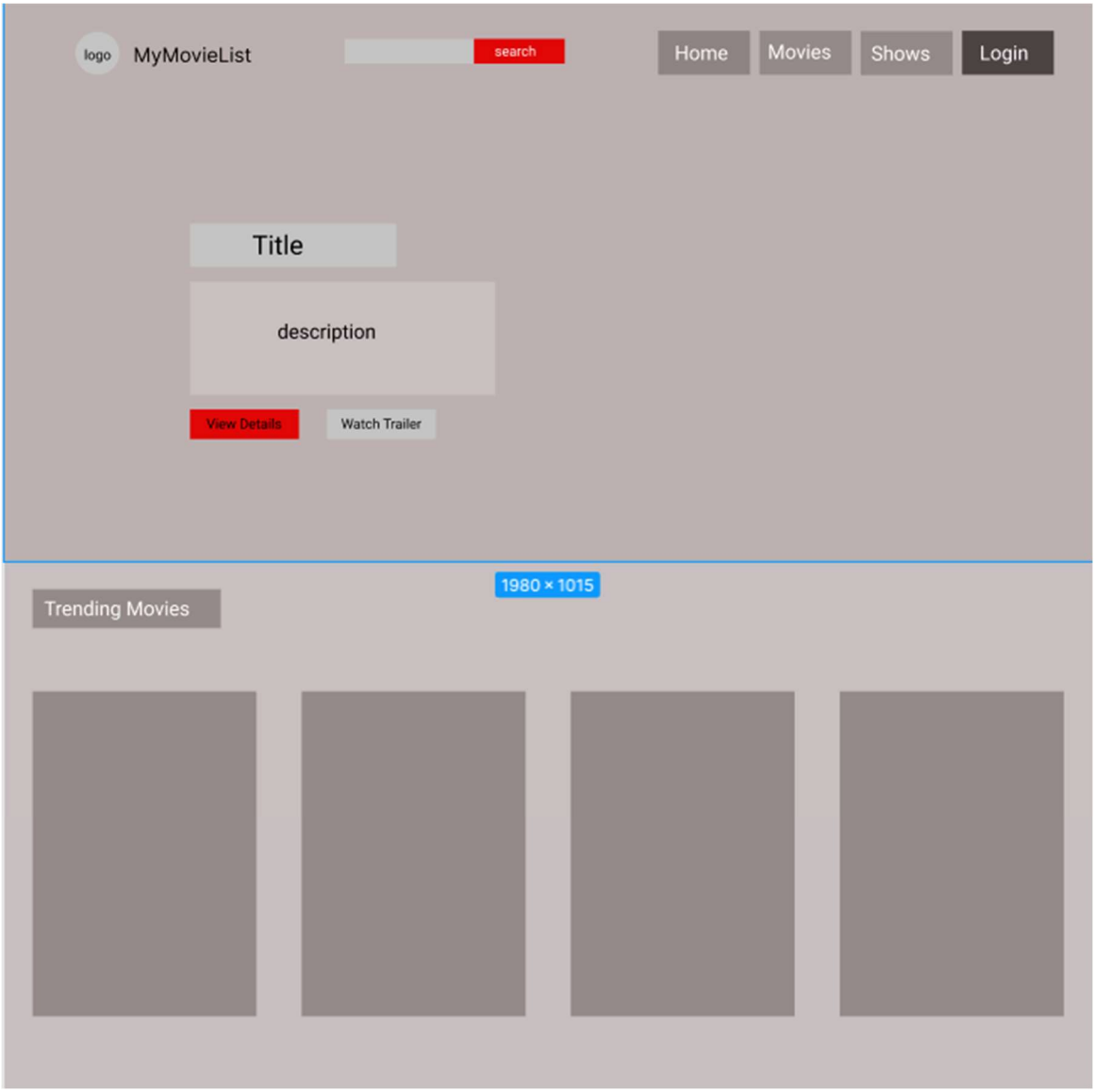


Figure 4.2: Wireframe UI for Homepage

This is the UI for the home page of our application. Here, a user can log in via google, search for movies/tv shows, view details such as cast, and trailers, access the dashboard, and see trending and popular movies/shows list.

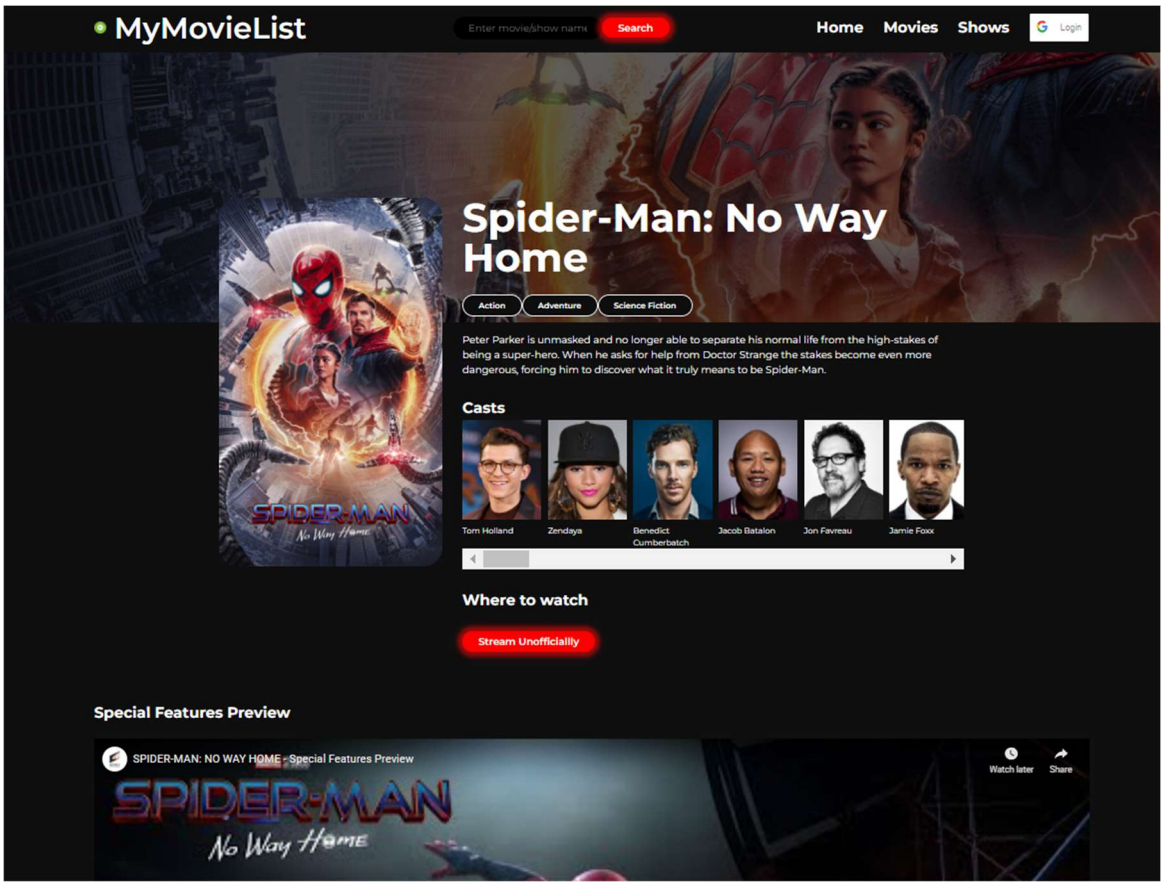


Figure 4.3: UI of Details Page

This is the details page of a movie/show. Here a user can see the cast list, trailers, popular videos, and clips. The user can also stream the desired movie/show through our platform.

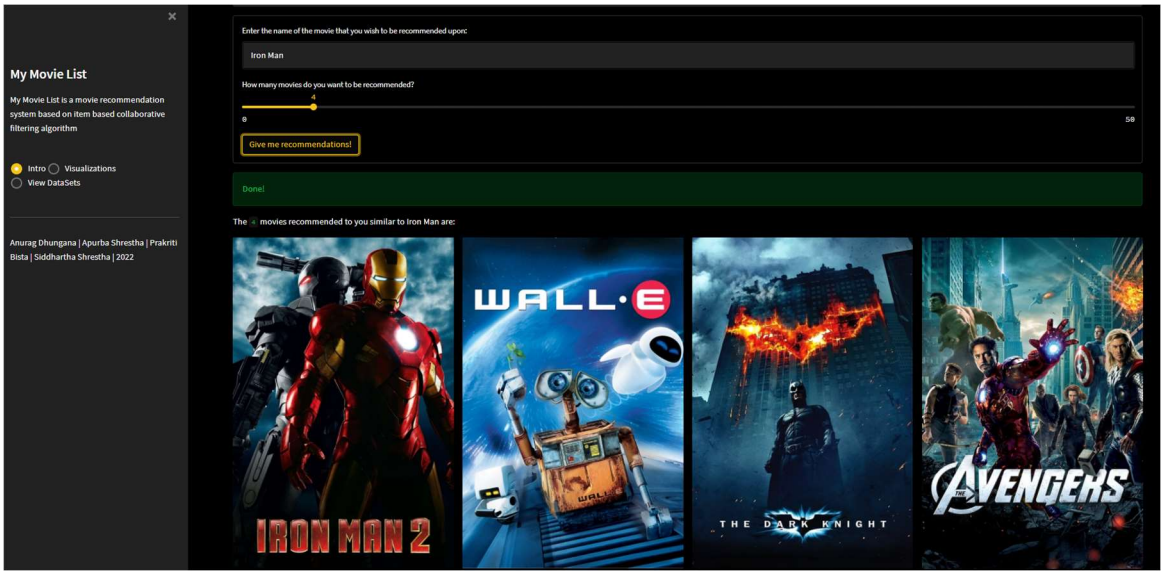


Figure **4.4: UI for Recommendation Model**

Here a user can get recommendations based on an item or their user id. The recommended movies can be clicked upon to view their details as well as to stream them.

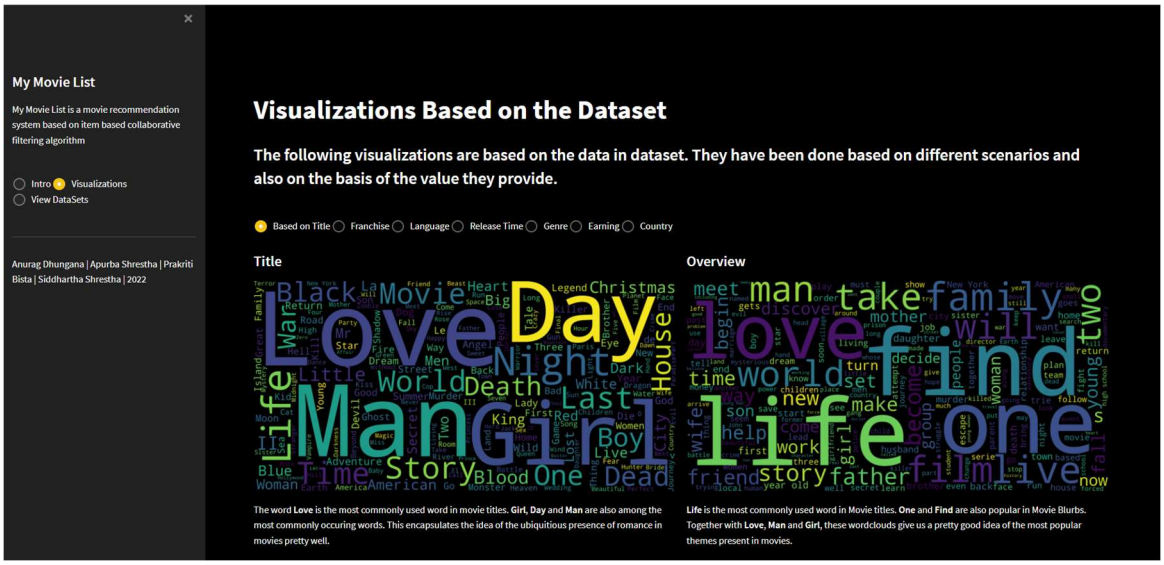


Figure 4.5: UI for Data Visualization

This page contains different EDA done on the dataset that we have used. It shows graphs and visualizations based on the movies and their details that we have in our dataset.

4.3 Algorithm Details

This section contains the details regarding the algorithm used in this project.

4.3.1. K-Nearest Neighbor Algorithm

The k-nearest neighbors (KNN) algorithm is a simple, easy-to-implement supervised machine learning algorithm that can be used to solve both classification and regression problems. The KNN algorithm assumes that similar things exist in close proximity.

We can implement a KNN model by following the below steps(7):

1. Load the data
2. Initialize the value of k
3. For getting the predicted class, iterate from 1 to the total number of training data points

1. Calculate the distance between test data and each row of training data. Here we use cosine distance as our distance metric.
2. Sort the calculated distances in ascending order based on distance values
3. Get top k rows from the sorted array
4. Get the most frequent class of these rows
5. Return the predicted class

4.3.2. Cosine-based similarity

Cosine similarity is a metric used to measure how similar two items are. Mathematically, it measures the cosine of the angle between two vectors projected in a multi-dimensional space. The output value ranges from 0–to 1. 0 means no similarity, whereas 1 means that both the items are 100% similar.

Also known as vector-based similarity, this formulation views two items and their ratings as vectors and defines the similarity between them as the angle between these vectors(8):

$$similarity(A,B) = \frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Table 4.1: Similarity Between different users

userId	**1**	**2**	**3**	**4**	**5**	**6**
userId						
1	**0.000000**	**0.988283**	**0.978406**	**0.964220**	**0.986819**	**0.970456**
2	**0.988283**	**0.000000**	**0.987141**	**0.971166**	**0.995793**	**0.979893**
3	**0.978406**	**0.987141**	**0.000000**	**0.961237**	**0.985179**	**0.970773**
4	**0.966819**	**0.995793**	**0.985179**	**0.000000**	**0.968638**	**0.955187**
5	**0.986419**	**0.995793**	**0.985179**	**0.968638**	**0.000000**	**0.978368**

Chapter 5: Implementation and Testing

5.1. Implementation

The following section details how the project was implemented.

5.1.1. Tools Used

Following are the tools and frameworks currently being used for the accomplishment of the project:

1. ReactJS: It was used to create the frontend of our web application based on the Figma design.
2. TMDb API: It was used to access the data required to display in the front end of the application. The data called by the API ranged from names to cast lists, episode links as well as links to clips from YouTube.
3. Python: It was used to create the recommendation engine as well as the dashboard for the user to interact on. It was also used to perform EDA on the dataset and build visualizations.
4. Git: It was used for version control and distributing the work to the team members.
5. Visual Studio Code: It was the primary IDE used for the development of the project.
6. Figma: It was used to create the initial wireframe for our frontend design.
7. Draw.io: It was used to create the different diagrams for this report.

5.1.2. Understanding the dataset

The data used in this project has been obtained from two sources: The Movie Database (TMDB) and MovieLens. MovieLens has a publicly available dataset that contains 26 million ratings and 750,000 tag applications applied to 45,000 movies by 270,000 users. It also includes tag genome data with 12 million relevance scores across 1,100 tags. A small subset of this dataset, containing 10,000 ratings for 9000 movies from 700 users is also available. One of the files contains the TMDB ID of every movie listed in the MovieLens dataset. Using this ID, the metadata, credits, and keywords of all 45,000 movies were obtained by running a script that requested and parsed data from TMDB Open API. The

data collected was initially in the JSON format but was converted into CSV files using Python's Pandas Library. The dataset has a lot of features that are explained below.

5.1.3. Features

- Adult: Indicates if the movie is X-Rated or Adult.
- Belongs_to_collection: A stringified dictionary that gives information on the movie series to the particular film belongs.
- budget: The budget of the movie in dollars.
- Genres: A stringified list of dictionaries that list out all the genres associated with the movie.
- Homepage: The Official Homepage of the movie.
- Id: The ID of the move.
- Imdb_id: The IMDB ID of the movie.
- Original_language: The language in which the movie was originally shot.
- Original_title: The original title of the movie.
- Overview: A brief blurb of the movie.
- Popularity: The Popularity Score assigned by TMDB.
- Poster_path: The URL of the poster image.
- Production_companies: A stringified list of production companies involved with the making of the movie.
- Production_countries: A stringified list of countries where the movie was shot/produced in.
- Release_date: Theatrical Release Date of the movie.
- Revenue: The total revenue of the movie in dollars.
- Runtime: The runtime of the movie in minutes.

- Spoken_languages: A stringified list of spoken languages in the film.
- Status: The status of the movie (Released, To Be Released, Announced, etc.)
- Tagline: The tagline of the movie.
- Title: The Official Title of the movie.
- Video: Indicates if there is a video presentation of the movie with TMDB.
- Vote_average: The average rating of the movie.
- Vote_count: The number of votes by users, as counted by TMDB.

Here are a total of 45,466 movies with 24 features. The dataset had a lot of features that had 0s for values it did not possess. These values were converted to NaN. Some features were still in the form of a Stringified JSON Object. They were converted into Python Dictionaries using Python's ast library. These were further reduced into lists since we did not have a need for ID, timestamp, and other attributes. The data frame exploded wherever the analysis demanded it (for instance, genres and production countries). Finally, most of the features were converted into a Python basic type (integer, string, float) by removing all the unclean values. The date string was converted into a Pandas Datetime and from it, we extracted the month, year, and day of release of every movie.

For the recommendation system, a small subset of the above data was used. This dataset contained movie details and ratings.

The rating dataset had:

- userId - unique for each user
- movieId - using this feature, we take the title of the movie from movies dataset
- rating - Ratings given by each user to all the movies using this we are going to predict the top 10 similar movies

Table 5.1 : Rating Dataset

	userId	movieId	rating	timestamp

0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	984982931

The movie dataset had movieId which was linked to the rating dataset which was used to extract movie titles after the recommendation was done.

Table 5.2: Movie Dataset

	movie Id	**title**	**genres**
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy

5.1.4. Implementation Details:

This system used Kaggle as the initial data source for the implementation and the testing phase of the project and IMDB was the primary data source for the production phase. The front end uses the data from TMDB to show details.

Kaggle provides a curated set of highly cleaned data sets and a community of data scientists and enthusiasts who are continuously adding and managing the data along with helping each other out. Thus, this project used Kaggle's datasets to build the recommender. Unlike Kaggle, data from TMDb API are continuously updated and collected over various periods of time and are verified by many independent users. Specifically, the MovieLens Dataset, the IMDB dataset, and the TMDB dataset are used for serving recommendations and performing EDA.

Collaborative filtering is used for the recommendation system. Collaborative filtering methods are based on collecting and analyzing a large amount of information on user behaviors, activities, or preferences and predicting what users will like based on their similarity to other users and items.

For this project, Collaborative filtering is used to serve recommendations to the user. With collaborative filtering, the system is based on past interactions between users and movies. The fundamental assumption behind the collaborative filtering technique is that similar user preferences over the items could be exploited to recommend those items to a user who has not seen or used them before. We assume that users who agreed in the past (purchased the same product or viewed the same movie) will agree in the future. With this in mind, the input for a collaborative filtering system is made up of past data of user interactions with the movies they watch. Also instead of users, the input can also be items such as movie names.

Specifically, item-based collaborative filtering and user-based collaborative filtering are being tested on the application. For the item-based collaborative filtering, the similarities between different items are calculated by using cosine similarity and clustered KNN algorithm. In the real world, ratings are very sparse and data points are mostly collected from very popular movies and highly engaged users. So we reduced the noise by adding some filters and qualified the movies for the final dataset.

- To qualify for a movie, a minimum of 10 users should have voted for a movie.
- To qualify a user, a minimum of 50 movies should have been voted by the user.

Our final dataset has dimensions of 2121 * 378.

The similarity values between items are measured by observing all the users who have rated both the items. As shown in the diagram below, the similarity between two items is dependent upon the ratings given to the items by users who have rated both of them:

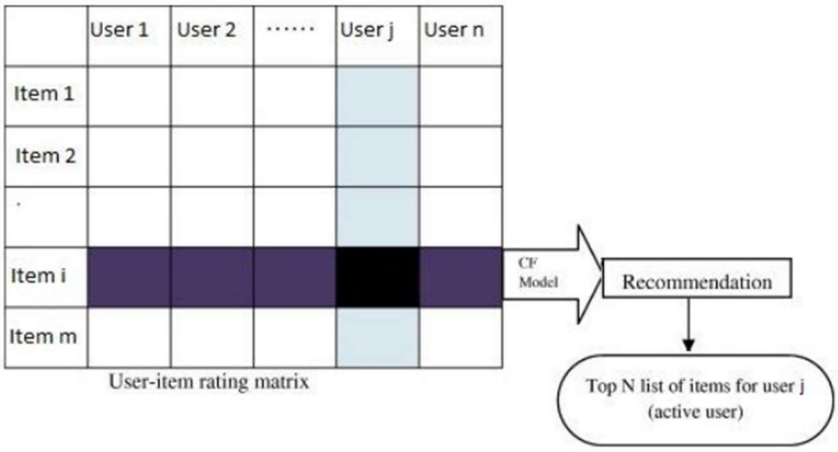


Figure 5.1: Finding Similarity Between Items

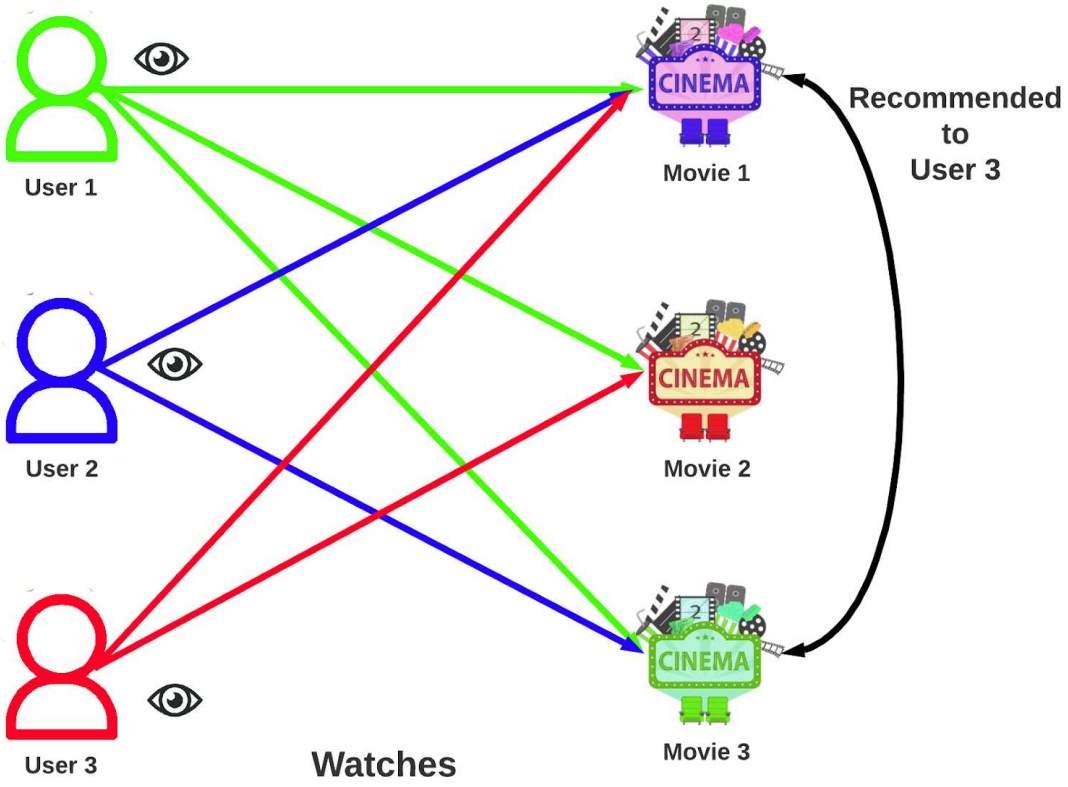


Figure 5.2: Item-based Collaborative Filtering

For user-based clustering, the movies, ratings, and tags dataset was taken. Each movie has a unique id, as well as every user has a unique id which is then linked to the rating given by the user. Then the rating average for each user was calculated.

Table 5.3 : Rating Average

userId	movie Id	1	4	6	7	10	11	15	16	17	…
0	1	4.0	0.0	0.0	4.5	0.0	0.0	2.5	0.0	4.5	…
1	2	0.0	0.0	4.0	0.0	0.0	0.0	0.0	0.0	0.0	…
2	3	4.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	…
3	5	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	…
4	6	4.0	0.0	4.0	0.0	0.0	5.0	0.0	0.0	0.0	
…	…	…	…	…	…	…	…	…	…	…	…

The data was then cleaned and used to create clusters. Top 30 Neighbors were created for each user and each item.

```
#top 30 neighbours for each user
sim_user_30_u = find_n_neighbours(similarity_with_user,30)
sim_user_30_u.head()
```

MagicPython

userId	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10	...	top21	top22	top23	top24	top25	top26	top27	top28
1	301	597	414	477	57	369	206	535	590	418	...	484	469	72	593	44	297	434	483
2	189	246	378	209	227	326	393	332	196	528	...	114	153	596	495	407	567	93	308
3	441	496	549	231	527	537	313	518	244	246	...	309	586	230	303	34	161	39	47
4	75	137	590	391	43	128	462	250	290	85	...	472	593	299	32	349	268	159	199
5	145	35	565	134	58	444	446	347	530	142	...	94	569	411	588	584	404	498	323

5 rows × 30 columns

Figure 5.3: Top 30 Neighbors for each user

```
#top 30 neighbours for each movie
sim_user_30_m = find_n_neighbours(similarity_with_movie,30)
sim_user_30_m.head()
```

MagicPython

userId	top1	top2	top3	top4	top5	top6	top7	top8	top9	top10	...	top21	top22	top23	top24	top25	top26	top27	top28
1	49	54	72	53	515	550	189	133	26	513	...	145	60	595	574	556	92	435	163
2	49	189	515	53	25	145	54	26	87	194	...	65	163	251	150	299	440	519	507
3	515	49	25	53	496	54	442	72	26	87	...	92	595	556	65	463	251	180	150
4	581	54	189	49	25	515	300	53	472	251	...	26	289	463	609	214	206	507	574
5	145	49	515	53	609	26	189	550	35	87	...	293	142	72	92	556	439	478	163

Figure 5.4: Top 30 Neighbors for each movie

This project uses the K-Nearest Neighbors algorithm with cosine distance metric to find which items are near to each other. The concept, in this case, is to find similar movies

instead of similar users and then recommend similar movies to that '1' has had in his/her past preferences. This is executed by finding every pair of items that were rated by the same user, then measuring the similarity of those rated across all users who rated both, and finally recommending them based on similarity scores.

5.2. Testing

System testing is actually a series of different tests whose primary purpose is to fully exercise the computer-based system. Although each test has a different purpose, all work to verify that all the system elements have been properly integrated and performed allocated functions. The testing process is actually carried out to make sure that the product exactly does the same thing it is supposed to do. Testing is the final verification and validation activity within the organization itself. In the testing stage following goals are to achieve: -

- To find and eliminate any residual errors from previous stages.
- To validate the software as a solution to the original problem.

5.2.1. Unit Testing

Unit testing is a software testing method by which individual units of source code, sets of one or more computer modules together with associated control data, usage procedures, and operating procedures, are tested to determine whether they are fit for use. In this movie recommendation system, the individual units are individually tested by providing one or few inputs and expected for a single output.

Table 5.4: Unit Testing on Recommendations

Test ID	Description	Input data	Expected Result	Actual Result	Pass/fail
1		Null	Same page should be displayed with no outputs	Same page is displayed with no outputs	Pass

	Movie Recommendation	Iron Man	Similar Types of movies should be recommended.	Similar Types of movies are recommended.	Pass

Table 5.5: Unit Testing on Movies Details

Test ID	Description	Input	Expected Result	Actual Result	Pass/Fail
3	Displaying details of a chosen movie.	GodFather	Display the data in reactJs frontend	Data is displayed by reactJs	Pass

5.2.2. Integration Testing

Integration testing is the phase in software testing in which individual software modules are combined and tested as a group. Integration testing is conducted to evaluate the compliance of a system or component with specified functional requirements. It was done after unit testing and before system testing.

Table 5.6: Integration testing between recommended movies and movie details

Test ID	Description	Input	Expected Result	Actual Result	Pass/Fail
4	When a user clicks on a recommended movie on the backend, they must be redirected to that movie's detail page in the front end.	List of recommended movies.	Redirect to the movie's detail page.	Redirected to the movie's detail page.	PASS

5.2.3 System Testing

By testing the system on its different functions, the system performs satisfactorily. But due to the traditional approach to the recommender, it cannot take a huge amount of data. The input to the recommender is also case-sensitive. The recommendations provided by the system based on the user's input are acceptable so it passes the basic system test.

5.3. Result Analysis

A user can get recommendations on our system by either their userID or Movie name.

A user enters the movie name "Iron Man" in the recommendation system. The input is used by the system to calculate the similarity between different movies. The top N number of movies that are most similar to "Iron Man" is then presented to the user.

	MovieID	Title	Genre	Distance	imdbId	tmdbId	Poster
0	68954	Up (2009)	Adventure|Animation|Children|Drama	0.368857	1049413	14160.0	https://image.tmdb.org/t/p/w500//eAdO0qa9m0NFS...
1	112852	Guardians of the Galaxy (2014)	Action|Adventure|Sci-Fi	0.368758	2015381	118340.0	https://image.tmdb.org/t/p/w500//r7vmZjiyZw9rp...
2	60684	Watchmen (2009)	Action|Drama|Mystery|Sci-Fi|Thriller|IMAX	0.368558	409459	13183.0	https://image.tmdb.org/t/p/w500//aZvOkdo203bm1...
3	68358	Star Trek (2009)	Action|Adventure|Sci-Fi|IMAX	0.366029	796366	13475.0	https://image.tmdb.org/t/p/w500//lV5OpzAss1z06...
4	33794	Batman Begins (2005)	Action|Crime|IMAX	0.362759	372784	272.0	https://image.tmdb.org/t/p/w500//8RW2runSEc34I...
5	72998	Avatar (2009)	Action|Adventure|Sci-Fi|IMAX	0.310893	499549	19995.0	https://image.tmdb.org/t/p/w500//jRXYjXNq0Cs2T...
6	77561	Iron Man 2 (2010)	Action|Adventure|Sci-Fi|Thriller|IMAX	0.307492	1228705	10138.0	https://image.tmdb.org/t/p/w500//6WBeq4fCfn7AN...
7	60069	WALLÂ·E (2008)	Adventure|Animation|Children|Romance|Sci-Fi	0.298138	910970	10681.0	https://image.tmdb.org/t/p/w500//hbhFnRzzg6ZDm...
8	58559	Dark Knight, The (2008)	Action|Crime|Drama|IMAX	0.285835	468569	155.0	https://image.tmdb.org/t/p/w500//qJ2tW6WMUDux9...
9	89745	Avengers, The (2012)	Action|Adventure|Sci-Fi|IMAX	0.285319	848228	24428.0	https://image.tmdb.org/t/p/w500//RYMX2wcKCBAr2...

Figure 5.5: Recommendations based on a user input

Here the metic "distance" shows the individual similarity between the input and the recommended movie. It is calculated by using cosine similarity. As seen here, the movie with the highest similarity is shown first, and so on. The movies recommended are similar to the input in terms of rating and are also diverse which helps the user to choose from a variety of options.

A user with userId "10" is recommended movies based on the user's past information such as the user's review and its similarity with other users. The user is recommended movies based on the clusters that were formed for the top 30 neighbors based on movies and users individually.

	movieId	score	title	genres	imdbId	tmdbId
0	177593	4.280987	Three Billboards Outside Ebbing, Missouri (2017)	Crime|Drama	5027774	359940.0
1	158966	4.277445	Captain Fantastic (2016)	Drama	3553976	334533.0
2	96430	4.156950	Odd Life of Timothy Green, The (2012)	Comedy|Drama|Fantasy	1462769	71864.0
3	1150	4.110627	Return of Martin Guerre, The (Retour de Martin...	Drama	84589	4483.0
4	750	3.969103	Dr. Strangelove or: How I Learned to Stop Worr...	Comedy|War	57012	935.0
5	858	3.894386	Godfather, The (1972)	Crime|Drama	68646	238.0
6	922	3.882507	Sunset Blvd. (a.k.a. Sunset Boulevard) (1950)	Drama|Film-Noir|Romance	43014	599.0
7	1673	3.881211	Boogie Nights (1997)	Drama	118749	4995.0
8	159817	3.872755	Planet Earth (2006)	Documentary	795176	192040.0
9	1136	3.870245	Monty Python and the Holy Grail (1975)	Adventure|Comedy|Fantasy	71853	762.0

Figure 5.6: Recommendations for a specific user

Here the metric “score” shows the similarity score between the user and the movie. The movie which has the highest similarity score is displayed first and so on. The user-based recommendations are relevant to the user’s preferences based on the previous ratings provided. Our system is successful in recommending accurate, relevant, and interesting movies to the user based on the KNN algorithm

Due to the huge amount of feature-rich data on our dataset we performed some analysis on our data. Some of the results of the analysis are presented below and the conclusions are very interesting.

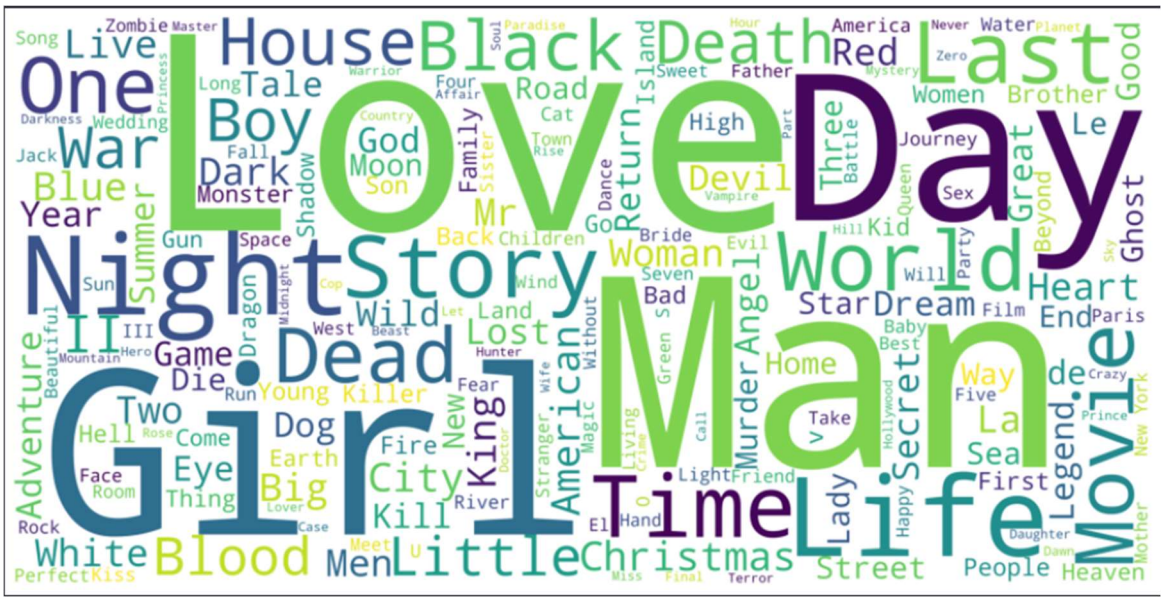


Figure 5.7: Analysis Done on The Titles of All The Movies.

The word **Love** was the most commonly used word in movie titles. **Girl**, **Day,** and **Man** were also among the most commonly occurring words. We thought this encapsulated the idea of the ubiquitous presence of romance in movies pretty well.

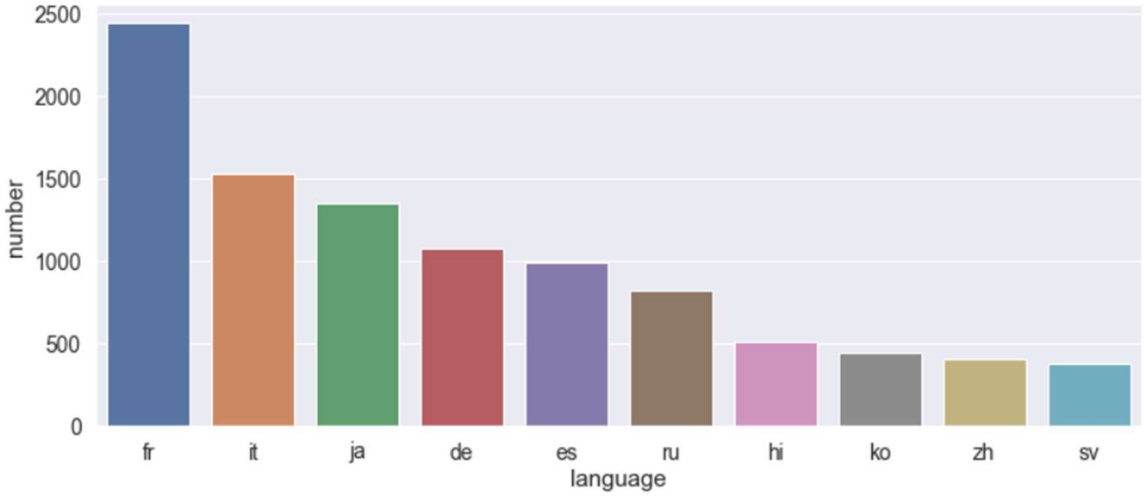


Figure 5.8: Analysis of Distribution of Different Languages(Except English)

French and Italian are the most commonly occurring languages after English. Japanese and Hindi form the majority as far as Asian Languages are concerned.

	title	popularity	year
30700	Minions	547.488298	2015
33356	Wonder Woman	294.337037	2017
42222	Beauty and the Beast	287.253654	2017
43644	Baby Driver	228.032744	2017
24455	Big Hero 6	213.849907	2014
26564	Deadpool	187.860492	2016
26566	Guardians of the Galaxy Vol. 2	185.330992	2017
14551	Avatar	185.070892	2009
24351	John Wick	183.870374	2014
23675	Gone Girl	154.801009	2014

Figure 5.9: Analysis Based on Popularity

Minions is the most popular movie by the TMDB Popularity Score. Wonder Woman and Beauty and the Beast, two extremely successful woman-centric movies come in second and third respectively.

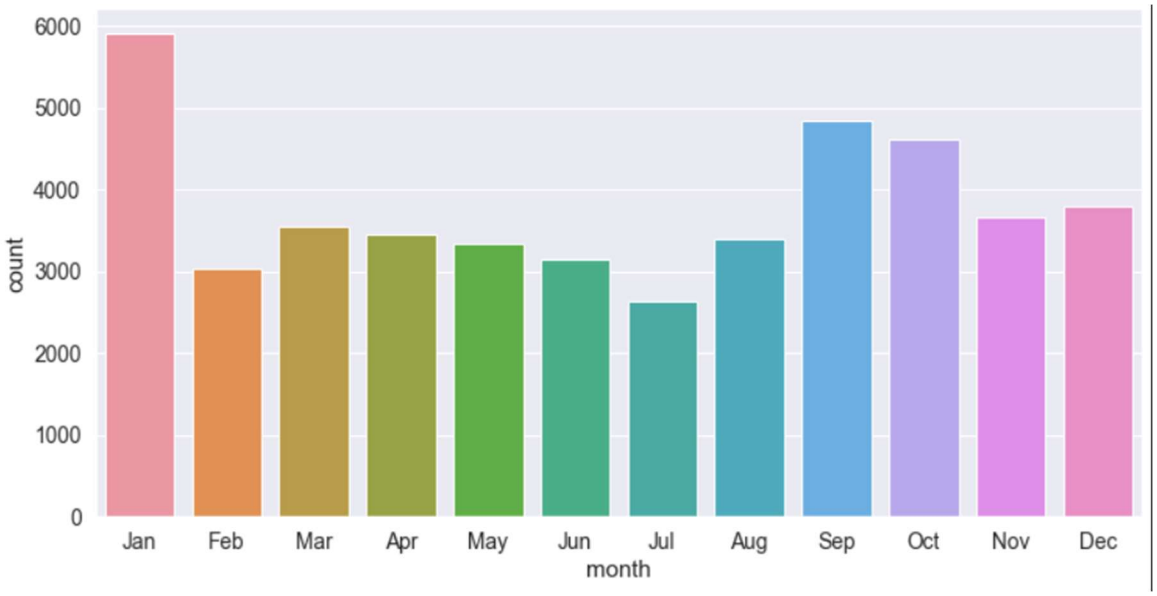


Figure 5.10: Analysis Based on Release Month.

It appears that January is the most popular month when it comes to movie releases. In Hollywood circles, this is also known as the dump month when subpar movies are released by the dozen.

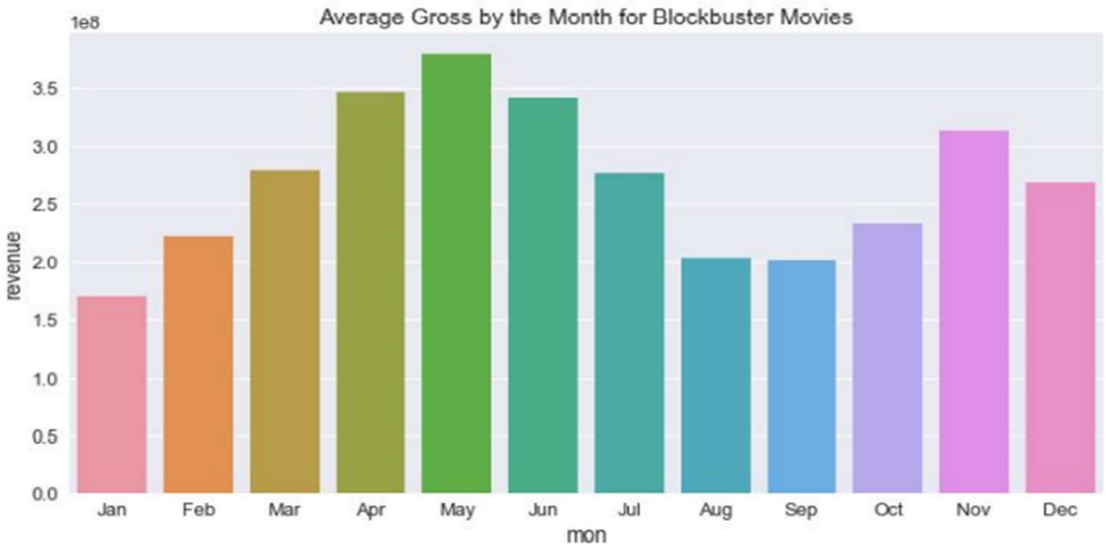


Figure 5.11: Analysis Based on Gross by Month

We see that the months of April, May, and June have the highest average gross among high grossing movies. This can be attributed to the fact that blockbuster movies are usually released in the summer when the kids are out of school and the parents are on vacation and therefore, the audience is more likely to spend their disposable income on entertainment.

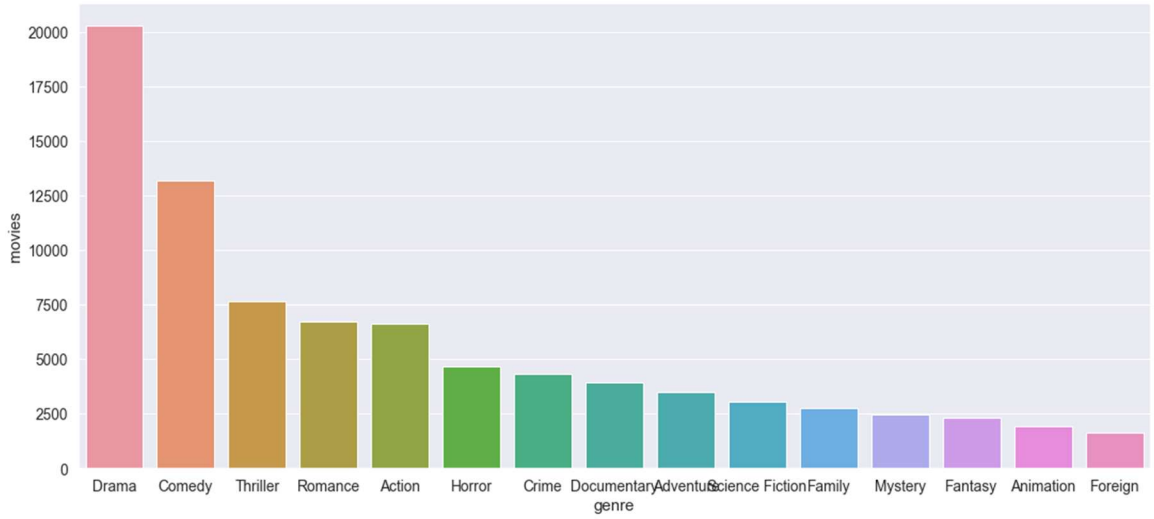


Figure 5.12: Analysis Based on Genres

Drama is the most commonly occurring genre with almost half the movies identifying themselves as drama film. Comedy comes in at a distant second with 25% of the movies having adequate doses of humor. Other major genres represented in the top 10 are Action, Horror, Crime, Mystery, Science Fiction, Animation, and Fantasy.

This concludes our result analysis.

Chapter 6: Conclusion and Future Recommendations

6.1. Evaluation

This report is composed of the guidelines of structure that the university provided. Every element included in the scheme is included with its full proficiency. However, some of the components of this report need serious skill and should invest enough time into it.

Analyzing the developed system, it is a web application, made with high reliability and precision. At some point, the libraries and the interpreter caused serious problems. There are a lot of issues regarding the loading of the graphs due to the huge amount of graphs involved. Sometimes we weren't able to predict the type of error that occurred without prior knowledge of that very same error.

Upon multiple dissections of journals, articles, research papers, websites, and books, the major backbone of this project was originated. Our findings greatly relied upon thesis papers and similar reports on movie recommendations. The time that our team invested in this particular field of 'getting-to-know movie recommendation' provided a satisfying outcome.

The process of working on this project was a mixed version of fun and challenges. The system-making process was an ultimate success with a bag of errors that were resolved to launch this platform. Report writing, on the other hand, seems to be a bit time-consuming if intended to be done with full proficiency. However, our investment of time returned us with a good knowledge of various fields, how they work, their use, future, and everything related to it. Although the project timeline was a bit disturbed due to the COVID-19 pandemic and the schedule of semester examinations, the prebuild mind map of the project was successfully delivered, all possible through the management skill learned throughout this process.

6.2. Future Recommendations

In the future we want to be able to allocate the resources and work evenly. We would like to be able to code the front end and the back end of the project in the same programming language. Working under a standardized framework is also something that we would like to do. We would also like to implement some sort of caching mechanism in order to load

the resources faster. We would also like to add more visualizations and perform more EDA on the dataset to extract much more information.

6.3. Conclusion

In a nutshell, this application is a result of thorough research and analysis through different movie recommendation platforms. Although this version might not be very accurate and efficient, this technology has a huge potential to grow and get implemented in every platform out there. The planning and coordination among team members for working on this project, while we were all staying inside during the project, was challenging. New thoughts, new ideas stay as an idea, until acted upon; and the successful presentation of this report has been the outcome of the same.

References

[1] Cui, Bei-Bei. (2017/2018). Design and Implementation of Movie Recommendation System Based on Knn Collaborative Filtering Algorithm. ITM Web of Conferences. 12. 04008. 10.1051/itmconf/20171204008.

[2] Banerjee, Anurag & Basu, Tanmay. (2018). Yet Another Weighting Scheme for Collaborative Filtering Towards Effective Movie Recommendation.

[3] Using collaborative filtering and k-means. DOI:10.19101/IJACR.2017.729004 M Shamshiri, GO Sing, YJ Kumar, International Journal of Computer Information Systems and Industrial Management Applications(2019).

[4] Kharita, M. K., Kumar, A., & Singh, P. (2018). ItemBased Collaborative Filtering in Movie Recommendation in Real-time. 2018 First International Conference on Secure Cyber Computing and Communication (ICSCCC). DOI:10.1109/icsccc.2018.8703362

[5] Katarya, R., & Verma, O. P. (2017). An effective collaborative movie recommender system with a cuckoo search. Egyptian Informatics Journal, 18(2), 105–112. DOI:10.1016/j.eij.2016.10.002

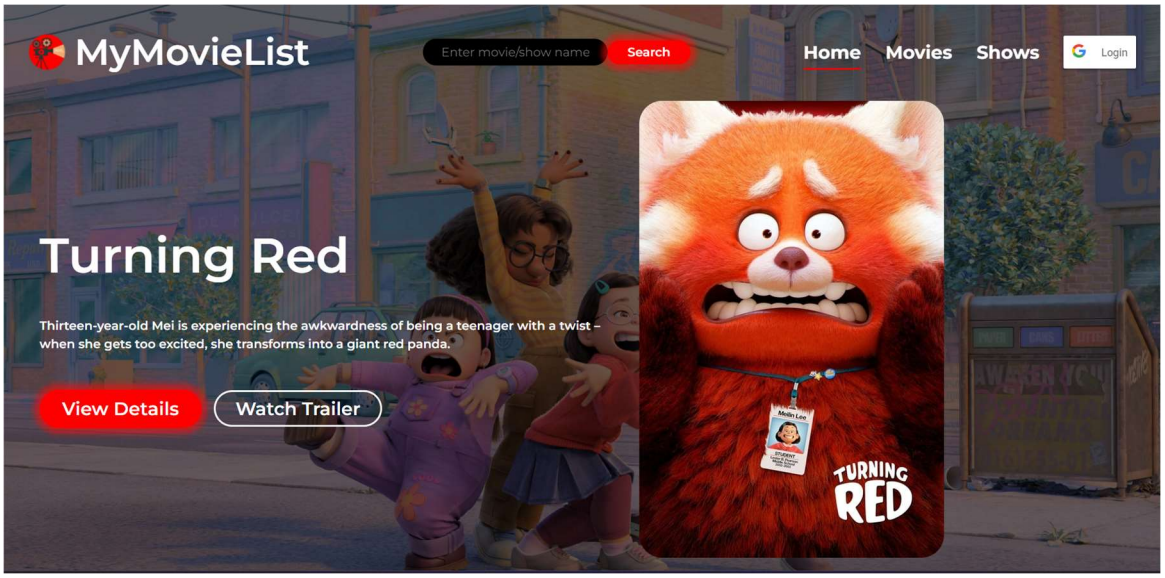
[6] Jain, A., & Vishwakarma, S. K. (2017/2018). Collaborative Filtering for Movie Recommendation using RapidMiner. International Journal of Computer Applications, 169(6), 0975-8887

[7] Onel Harrison (2018), Machine Learning Basics with the K-Nearest Neighbors Algorithm, Medium.com

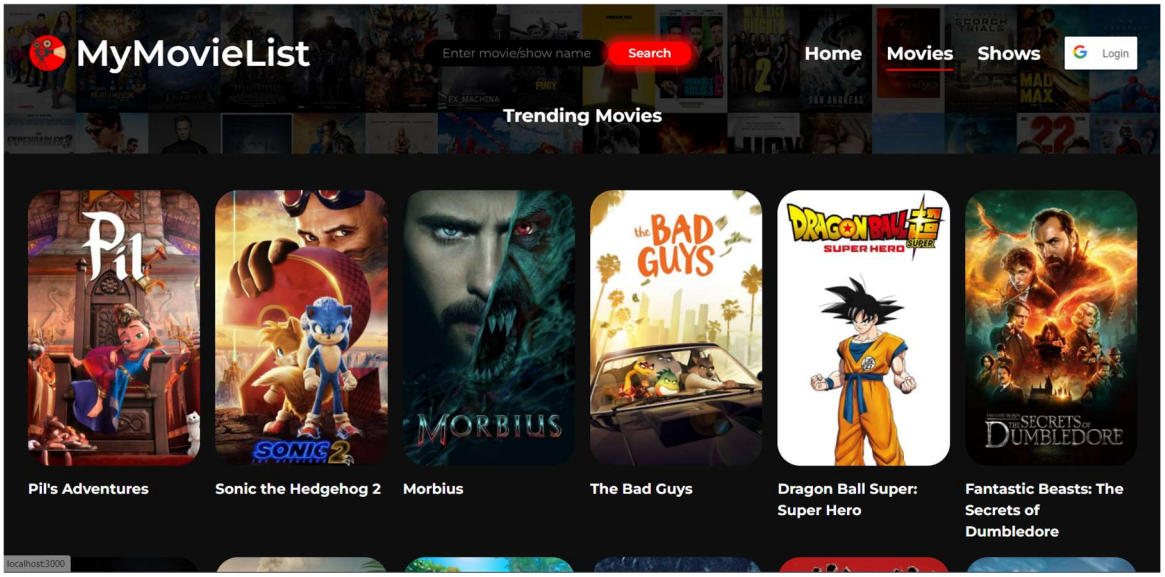
[8] Recommender Systems: A Computer Science Comprehensive Exercise Carleton College, Northfield, MN(2017).

Appendices

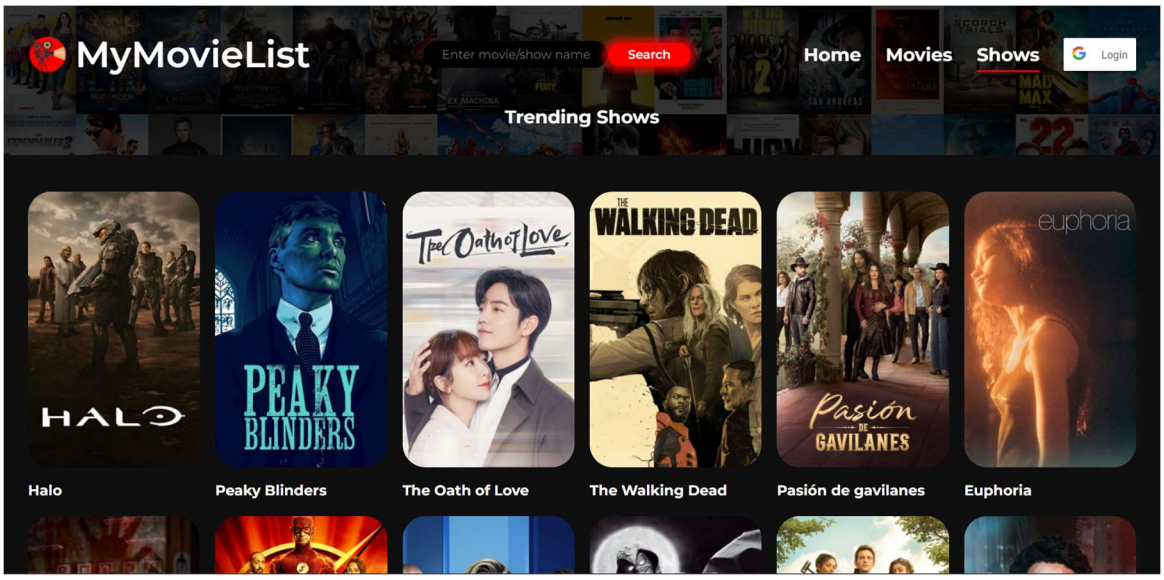
Snapshots of the UI of the Web Application



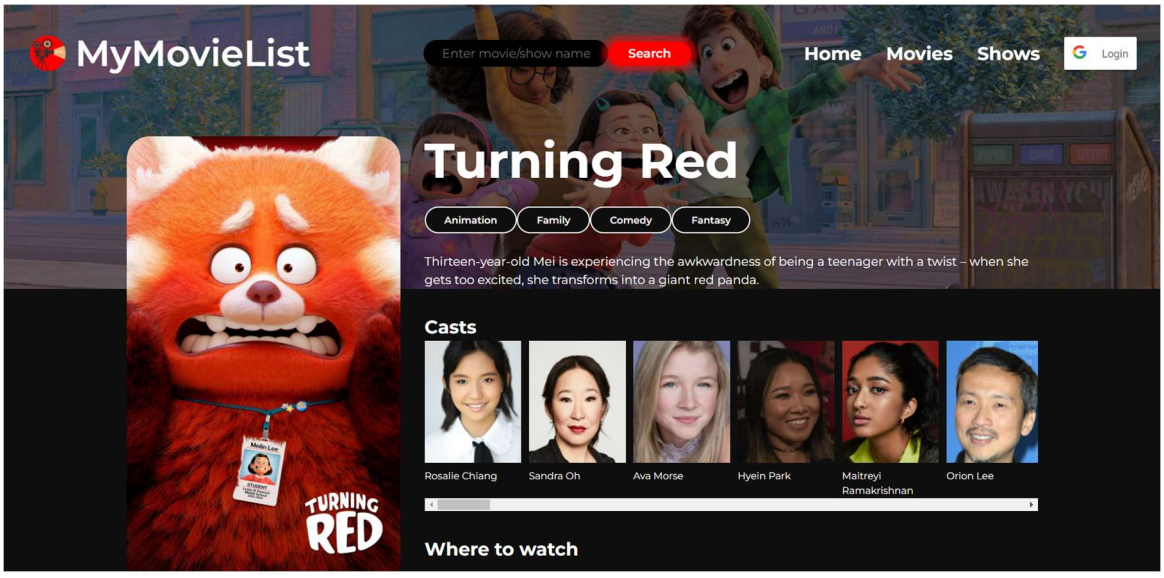
Home Page



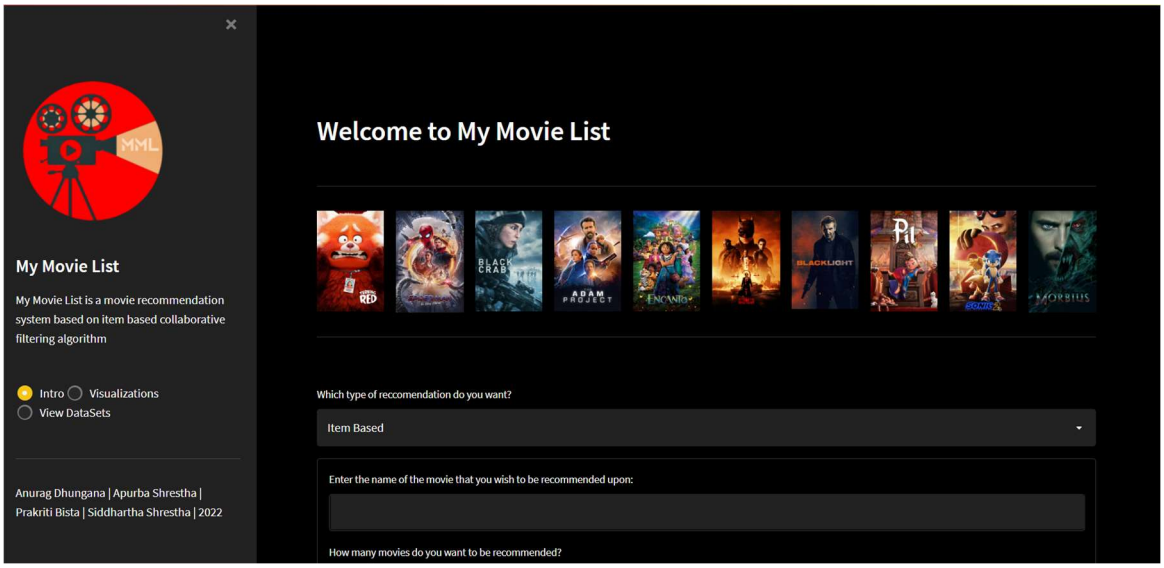
Trending movies page



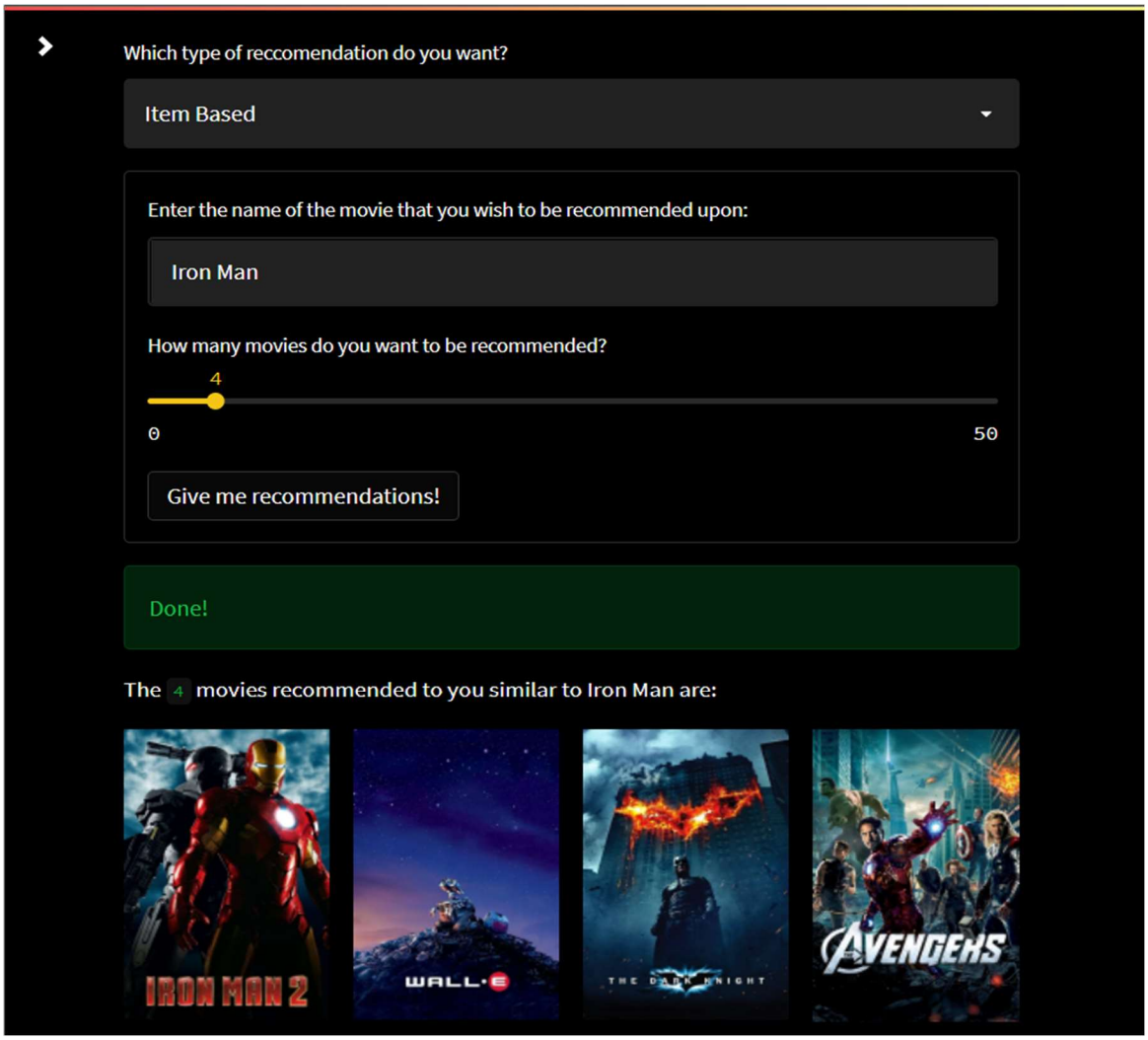
Trending shows page



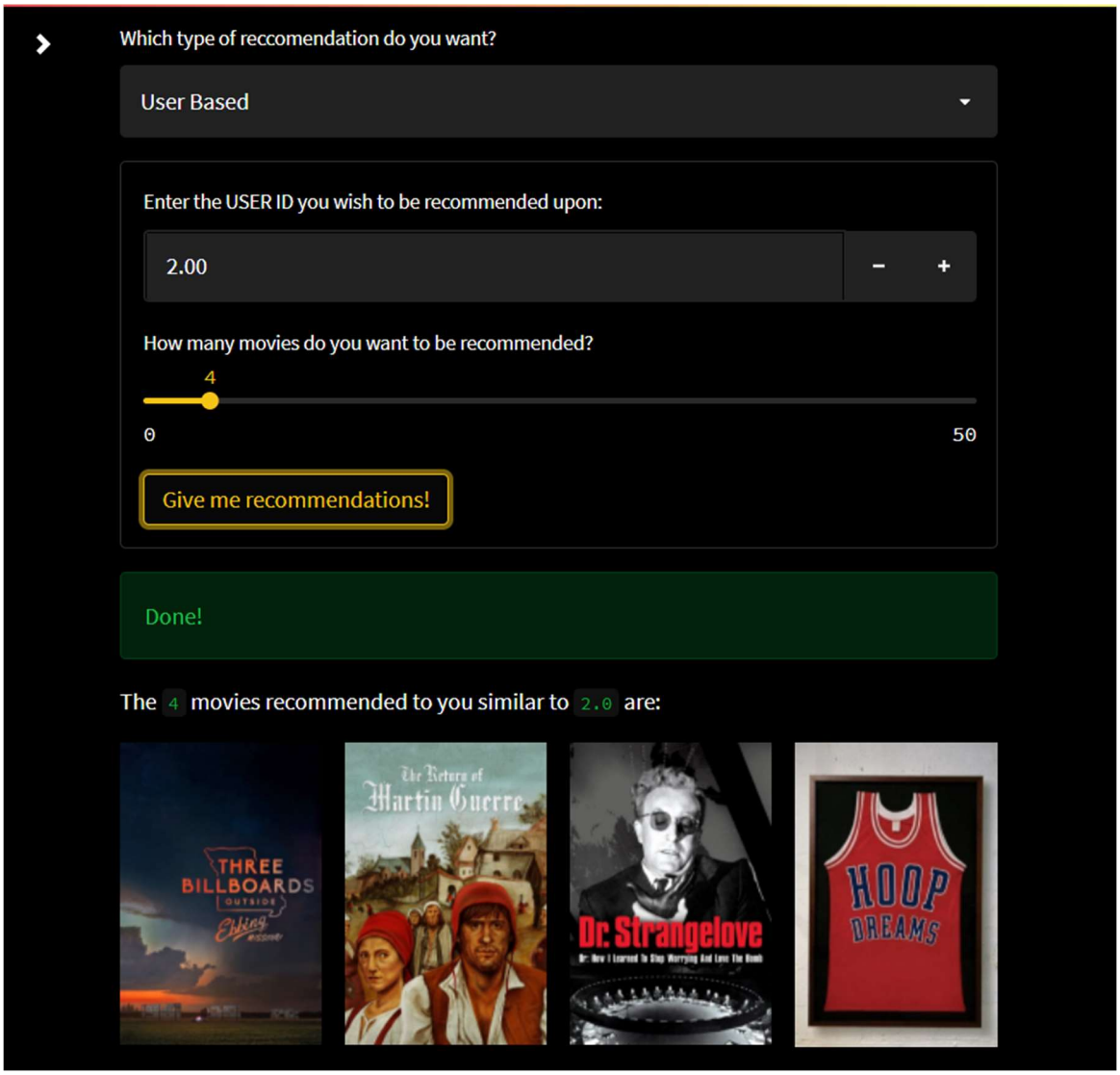
Movie Details page



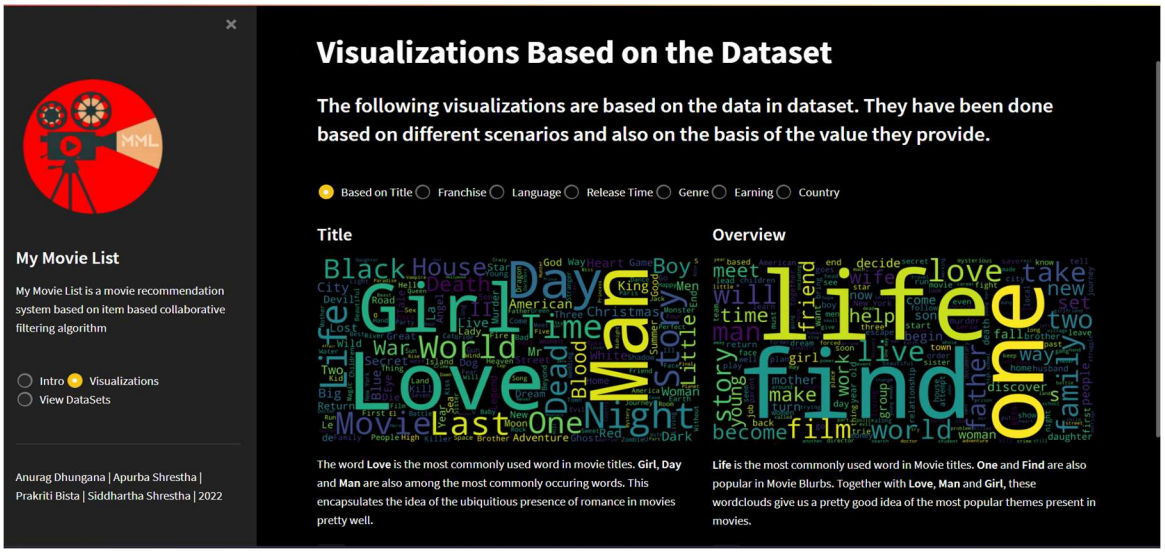
Recommender Dashboard

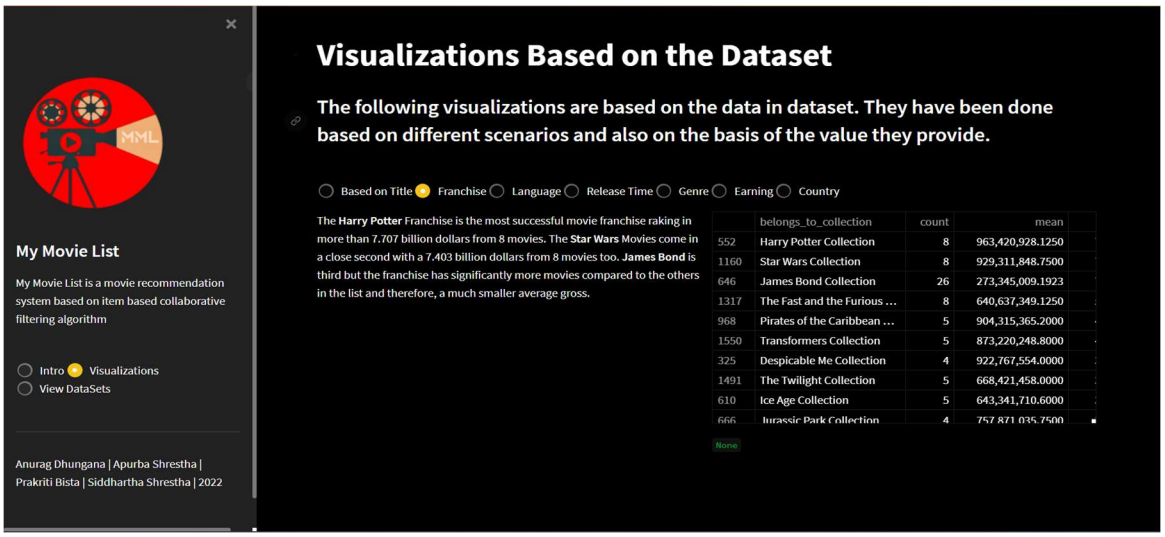


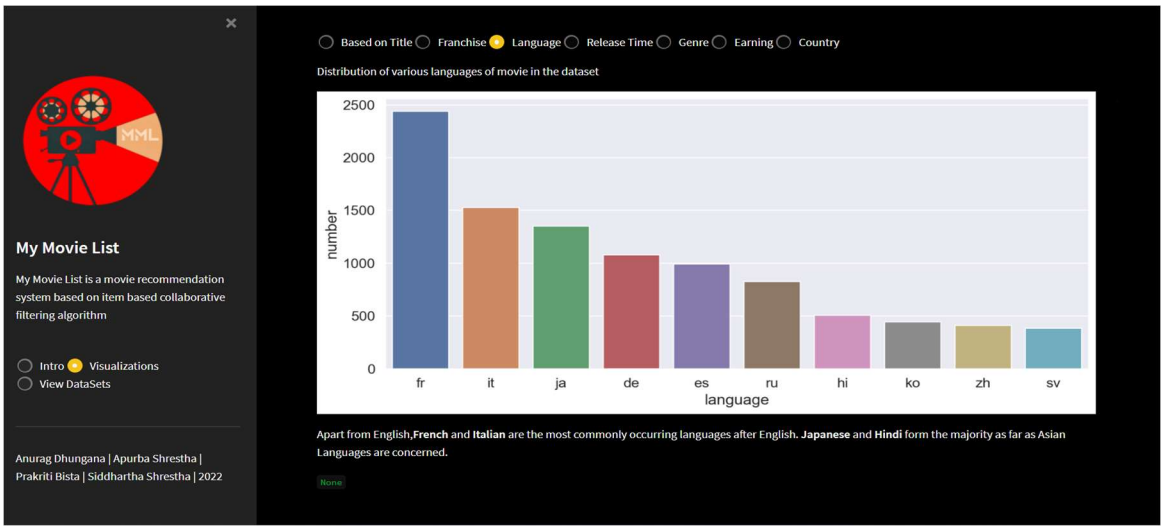
Item-based Recommendation



User-Based Recommendation

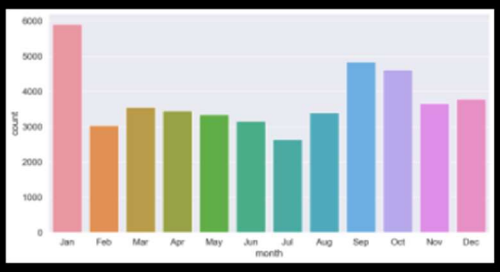






Different Visualizations on the dataset

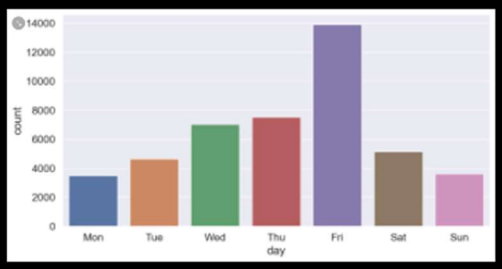
Number of Movies released in a particular month.



It appears that **January** is the most popular month when it comes to movie releases. In Hollywood circles, this is also known as the *the dump month* when sub par movies are released by the dozen.

None

Number of Movies released on a particular day.

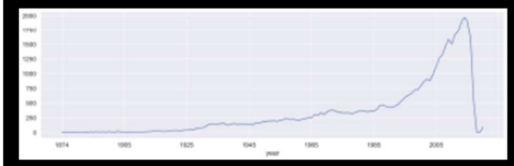


Friday is clearly the most popular day for movie releases. This is understandable considering the fact that it usually denotes the beginning of the weekend. **Sunday** and **Monday** are the least popular days and this can be attributed to the same aforementioned reason.

None

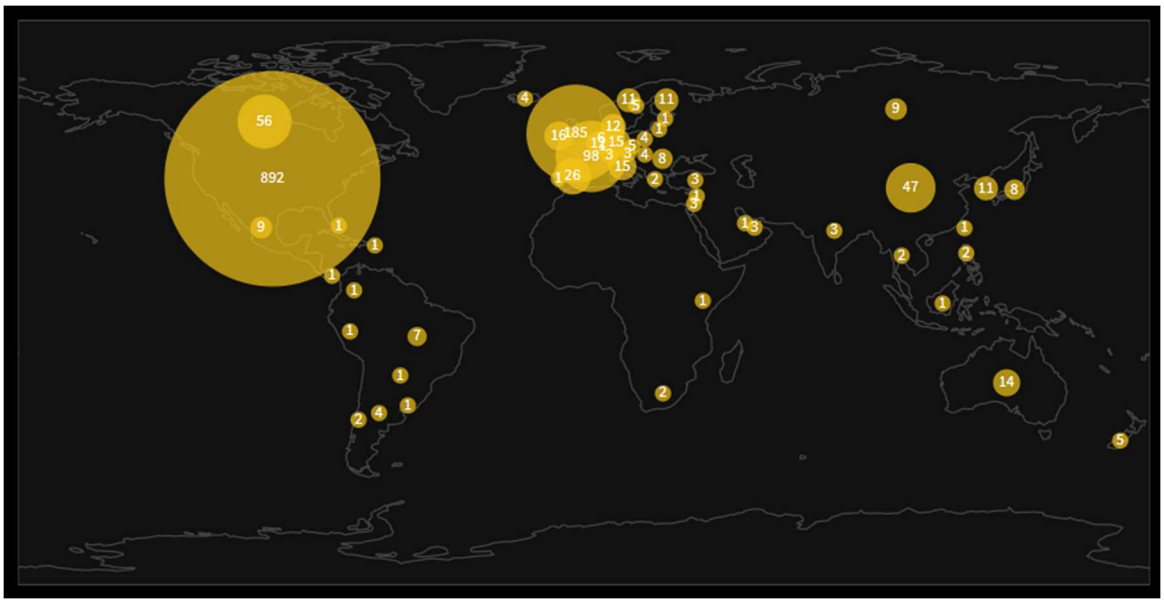
We see that the months of **April, May** and **June** have the highest average gross among high grossing movies. This can be attributed to the fact that blockbuster movies are usually released in the summer when the kids are out of school and the parents are on vacation and therefore, the audience is more likely to spend their disposable income on entertainment.

No of Movies Prodcyed by year



None

Visualizations based on release time



Visualization of the distribution of various countries of movies in the dataset

	adult	belongs_to_collection	budget	genres	homepage	id	imdb_id	original_lan
0	False	{'id': 10194, 'name': 'Toy ...	30000000	[{'id': 16, 'name': 'Animati...	http://toystory.disney.co...	862	tt0114709	en
1	False	<NA>	65000000	[{'id': 12, 'name': 'Advent...	<NA>	8844	tt0113497	en
2	False	{'id': 119050, 'name': 'Gru...	0	[{'id': 10749, 'name': 'Ro...	<NA>	15602	tt0113228	en
3	False	<NA>	16000000	[{'id': 35, 'name': 'Comed...	<NA>	31357	tt0114885	en
4	False	{'id': 96871, 'name': 'Fath...	0	[{'id': 35, 'name': 'Comedy'}]	<NA>	11862	tt0113041	en

Movies Dataset

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Chil...
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II ...	Comedy

Ratings Dataset

	userId	movieId	rating	timestamp
0	1	1	4.0000	964982703
1	1	3	4.0000	964981247
2	1	6	4.0000	964982224
3	1	47	5.0000	964983815
4	1	50	5.0000	964982931

Datasets used for recommendation engine and data visualization