

Age and Context Sensitive Entertainment Recommendation System for Families

Abhinav Yadav Aman Kumar Dhruv Mishra Hunar Kaur Neelabh Kumar Srivastava
Sadhvi Bhan

Literature Review

In this section, we first discuss what we intend to derive from a particular paper, that is how the paper may be relevant to our project and then proceed to explain the findings in the literature review.

[1] Machine learning based recommendation systems

This study used collaborative filtering technique to recommend movies. Collaborative filtering is making movie recommendations based on user ratings and the past interactions of users and items resulting in prescribing movies to users who have similar tastes. It is a web application that allows users to rate movies as well as recommend them appropriate movies based on other's ratings. The web application implemented in this study recommends movies to the user based on his/her previous ratings to other movies. If the user is new, it is required for the user to rate a certain amount of movies before the application recommends movies to them. In this way the users can explore more by reviewing movies and providing recommendations for others. Whereas in content based filtering is based on an individual user's rating hence preventing the user better and more relevant recommendations.

[2] Information Retrieval based recommendation systems

This paper is regarding the implementation of a MovME recommendation system using user ratings and user comments. It is a more personalized recommender implemented using TF-IDF techniques. A MovMe ranking score is calculated that takes into consideration the term frequency of the terms in the movie titles and collaborative filtering for calculating similarities. This paper shows that user comments contain helpful information that can be used to improve movie searches. This study found that user comments are similar to movie descriptions. 60.05% of words in the two were shared. They then created a movie index by combining conventional TF-IDF, rating scores, and user similarities. A personalized user index table is created and used.

[3] The authors discuss how, when the input prefix is very tiny, query auto completion offers subpar predictions of the user's inquiry. They demonstrate how adding context, such as the user's most recent inquiries, can greatly enhance the prediction. They suggest the NearestCompletion query auto-completion algorithm, which returns the user input completions that are most comparable to the context queries. They use cosine similarity to gauge similarity by modeling questions and contexts as high-dimensional term-weighted vectors. They present a new method for mapping queries to vectors that expands a query by iterating through the query suggestion tree with the query as its root.

NearestCompletion's MRR is practically nil when the context is irrelevant. They suggest HybridCompletion, a mix of NearestCompletion and MostPopularCompletion, as a solution to this issue. With a cumulative MRR improvement of 31.5% over MostPopularCompletion, HybridCompletion is demonstrated to outperform both NearestCompletion and MostPopularCompletion.

Updated Problem Formulation:

Note: Our problem formulation stays the same as the baseline report.

We aimed to design a full-fledged movie recommender application that recommends movies based on the age of the users. The initially decided features of our recommender system were a search bar with age context-sensitive spell check, a search filter in a movie review, and a ranking of movie recommendations. However, researching more, we found out that using filtering techniques and then implementing a ranking model will lead to redundancy; hence we decided not to implement it.

We implemented the search bar feature by implementing boolean and phrase query searching using unigram inverted index and positional inverted indexing techniques respectively. The age-sensitive filter for the same is yet to be added. For recommending similar movies, we have implemented a content-based filtering technique that recommends movies based on the user's likes. To implement content-based filtering, we converted the description of the user's preferred movie and other movies in the database into TF-IDF vectors and then compared the vectors using cosine similarity. Higher similarity constant means that the movies are similar. The age-sensitive filter has also been added.

A quick recap to the problem formulation:

1. No need for separate ranking models
We initially researched various ranking models and algorithms. For example, RankingModel() is a model which we had considered using. However, we found that we may not require it. We will be finding cosine similarity between movie descriptions and use it directly to rank the movies.
2. Incorporation of genre and description in search bar: While implementing the search bar feature we found out that instead of just searching on the basis of title, more

relevant and vast suggestions were made when searching was done on the basis of title, genre and description.

3. Use of Google USE -

Universal Sentence Encoder (USE) is a pre-trained Encoder which can be used in a variety of tasks (sentimental analysis, classification and so on). We aim to use this encoder to make our information retrieval more inclusive and context aware. Using this, we can map similar phrases and make our system better. For example, 'wizard' and 'magic' are words which may appear in similar movies, so if a movie description contains the word 'wizard' then another movie with 'magic' in its description will be recommended.

The Proposed Method:

1. Content - Based Filtering

Content Based Filtering recommends movies based on an individual user's preference and the features of the movie like its description, genre etc. We have implemented content based filtering by calculating the tf idf vectors for all the descriptions of movies in our data set and then finding their cosine similarity with the query tf idf vectors.

2. Collaborative Filtering

With the collaborative filtering method, we suggest movies to a user based on their shared interests with other users. We search for users who share the same preferences and tastes as the present user. We then suggest to the present user the movies that the matched users enjoy. For this method, we don't need movie metadata to suggest movies. Instead, it needs the user's ratings for the films. The dataset we used contained the movie id, user id, and ratings that the user id had given to a certain movie id.

3. Demographic Filtering

Demographic refers to the statistical characteristics of the human population. Content and collaborative filtering are ineffective when we don't have information about a user's choices. This situation is termed a 'Cold Start'. In such situations, we recommend the top-rated movies of all time to the user.

4. Development of Age-Filters

There are multiple different categories like TV-MA: strictly for adults, TV-14: for 14 and above etc which exist to categorize the movies according to age. We have thoroughly researched the categories and given a query, we update our dataset so that movies with unsuitable ratings are removed. Suitable ratings depend on the min and max age in the group. Min-age adult movies are removed, and max-age movies with parental guidance(PG) are allowed.

5. Description based similarity searching(Search Bar):

We have implemented a description based searching model at the moment. We will aim to increase model performance by reducing the running time of the model and tuning it without sacrificing the relevance of the results.

6. Integration with Google Encoder

To be implemented in the future deadline.

7. Spell Check

We have implemented spell check in our current model. We will take a step further and incorporate context sensitive spell correction in the future deadline. Also we will ensure the spell check is tuned according to our dataset.

8. Auto-complete

This is in continuation to spell check, so once we get decent results in contextual spell correction, we will look into implementing context sensitive auto completion.

9. Front - End

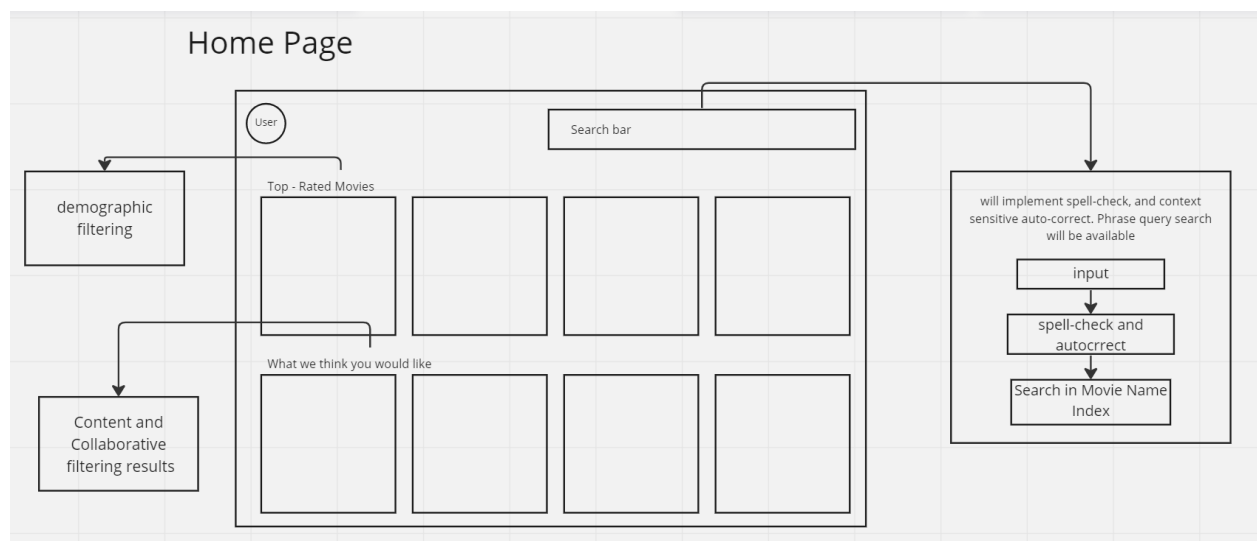
We have made a working UI right now. It still needs more work in order to be appealing to the user. We will accomplish this before the final deadline

10. Integration of all the code and dependencies

This will obviously be done once we are through with our project. Expected to be done before the final deadline.

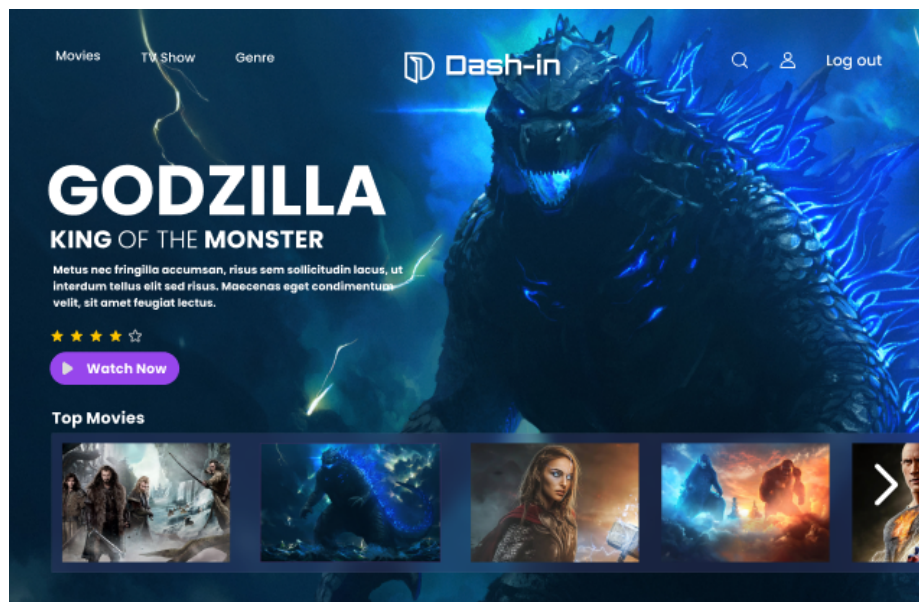
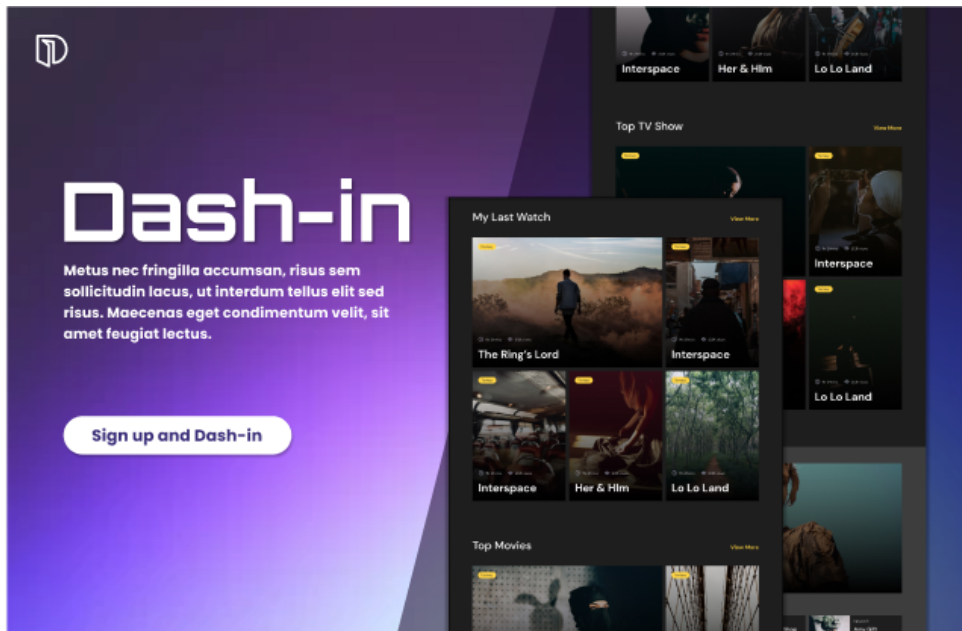
Low-Fi Prototype Design:

We designed some low fidelity prototypes for the frontend of our application. This is the Home Page of our application.



High-Fi Prototype:

<https://www.figma.com/file/1eWPW9ADCo2TVV1G4B784N/Dash-In?node-id=0%3A1&t=h5Jc5RB9Sp01IAgH-1>



Updated Baseline Results:

1. Query Processing using Boolean Search - Search Bar

This is our prototype for the search bar. Whenever a user enters a word, probably looking for a particular movie, we want to display all movie titles in which the query word/phrase occurs in the movie name or description. Sometimes, a user may not recall the complete movie name, and may rather enter a description for the movie, hence description has also been considered.

In this, for each movie we first tokenized the movie name, description and genre. Using these, a unique cumulative token set was generated for each movie. Now whenever the user inputs a word/phrase query, it is converted to a boolean query and movie names corresponding to matching sets are returned.

```
Original Input Query: jurassic
```

```
Number of Documents: 4
```

```
Movie Names:
```

```
LEGO Jurassic World: The Indominus Escape
```

```
Jurassic World Camp Cretaceous
```

```
LEGO Jurassic World: Legend of Isla Nublar
```

```
LEGO Jurassic World: Secret Exhibit
```

search results for keyword "jurassic

```
+++++
Original Keywords: jurassic world
Generated Query: jurassic or world
Number of Documents: 764
Movie Names:
LEGO Jurassic World: The Indominus Escape
Jurassic World Camp Cretaceous
LEGO Jurassic World: Legend of Isla Nublar
LEGO Jurassic World: Secret Exhibit
The Most Assassinated Woman in the World
Marc Maron: Too Real
The World We Make
Christiane Amanpour: Sex & Love Around the World
We Bare Bears
Animal World
Skylines
The Grandmaster
Birders
USS Indianapolis: Men of Courage
20 Feet From Stardom
SMOSH: The Movie
Mad World
Clive Davis: The Soundtrack of Our Lives
Scott Pilgrim vs. the World
+++++
```

Success

search result for keyword "jurassic world"

```

+++++
Original Keywords: action adventure fantasy
Generated Query: action or adventure or fantasy
Number of Documents: 2564
Movie Names:
Automata
Good People
Kidnapping Mr. Heineken
Moonwalkers
Next Gen
Black Panther
Animal World
Hell and Back
Black Panther
In the Shadow of the Moon
Norm of the North: King Sized Adventure
Good People
Kidnapping Mr. Heineken
Moonwalkers
Next Gen
Archibald's Next Big Thing
Hell and Back
Black Panther
Sturgill Simpson Presents Sound & Fury
...
Top Grier
Used Goods
Viking Destiny
+++++

```

Search results for genre

2. Movie Recommender:

We implemented content-based filtering for recommending movies. The idea behind content filtering is that, given a movie, the program must find and recommend a similar movie to the user. To compare similarities between the plots of the movies, we planned to use the movie description column of our dataset.

We first converted our string description into a vector. We did this by calculating the Term Frequency-Inverse Document Frequency (TF-IDF) vectors of the description of the movies. We did this for all the movies in our dataset.

However, before calculating the TDIF matrix, we enforced a strict age rating filter on the dataset and filtered out all inappropriate entries. We did this to ensure that the content is family-friendly and that no viewer is exposed to potentially inappropriate content.

For quantifying similarity, we compared different similarity metrics(Cosine, Pearson, and Euclidean) and found relatively indifferent rankings across the three. We finally settled for Cosine Similarity as it is independent of magnitude and is much faster to calculate than the other two metrics without significantly affecting our final results.

```
Initializing Recommender...
Recommender Initialized

Now Cleaning The Dataset...
Dataset Cleaned...

Now Ready To Recommend...
Enter the Total Number of Users:2
Enter Your Age:15
Enter Your Movie Preference:#realityhigh
Enter Your Age:16
Enter Your Movie Preference:Care of Kancharapalem
Recalculating the Entries To Suit Your Preferences
Now searching from a catalogue of over 8800 Movies and TV Shows...
Here's What We Think You'll Like:
    1 Mr. Young
    2 Big Stone Gap
    3 A Very Special Love
    4 Kocan Kadar Konus
    5 Just Friends

Now Ready To Recommend...
Enter the Total Number of Users:3
Enter Your Age:19
Enter Your Movie Preference:Shutter
Enter Your Age:29
Enter Your Movie Preference:Phobia 2
Enter Your Age:23
Enter Your Movie Preference:Death of Me
Recalculating the Entries To Suit Your Preferences
Now searching from a catalogue of over 8809 Movies and TV Shows.
Here's What We Think You'll Like:
    1 Unsolved Mysteries
    2 13 Reasons Why
    3 Fear Files... Har Mod Pe Darr
    4 Malevolent
    5 The Unborn Child
```

Content Based Filtering for a groups of teenagers vs Content Based Filtering for an average urban household

3. Spell Check and Correct

We have used TextBlob Library to implement spelling correction. This will later be merged with the search bar to auto-correct movie names in case the user enters a wrong spelling. This will prevent the search bar from returning blank results.

The current version demonstrates that while this library works, there are still some loopholes. For example, if the user enters: Pirates of Caribbean (incorrect spelling of

Caribbean) then this library fails as Caribbean is a proper noun. For this, we plan to incorporate the proper nouns present in movie names of our dataset.

```
[6] from textblob import TextBlob

[8] def correct_sentence_spelling(sentence):

    sentence = TextBlob(sentence)
    result = sentence.correct()

    print(result)

correct_sentence_spelling("A sentencee tto demonstratt spell carraction !")
A sentence to demonstrate spell correction !
```

```
[6] from textblob import TextBlob

[8] def correct_sentence_spelling(sentence):

    sentence = TextBlob(sentence)
    result = sentence.correct()

    print(result)

correct_sentence_spelling("prrate ot Carribbean")
pirate of Caribbean
```

4. Description Similarity Based Searching:

This searching technique relies on tokenizing the queries, and then searching them in the description of movies in our dataset through information retrieval techniques. We obtain the results from these searches and select the first few relevant results. Then we read the description of these results and return movies which have the highest similarity/correlation with the movies from our search. This is what we finally return to the user.

5. Collaborative filtering:

In the collaborative filtering approach, we recommend the movies to a user based on other users having similar interests. We look for users with similar tastes and preferences to the current user. Then we recommend the movies favored by the matched users to the current user. We don't require movie metadata to recommend movies for this technique. Instead, it requires

the ratings the user gave to the movies. The dataset we used had the movie id, user id, and the ratings assigned by the user id to the specific movie id.

```
Initializing Recommender...
Recommender Initialized

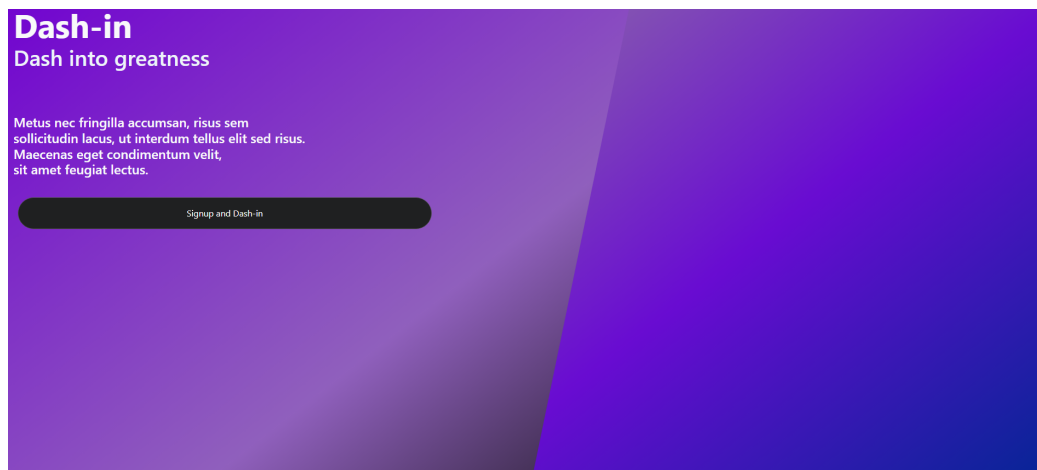
Now Cleaning The Dataset...
Dataset Cleaned...

Now Ready To Recommend...
Successfully Added 2 New Users
Recalculating the Entries To Suit Your Preferences
Now searching from a catalogue of over 8809 Movies and TV Shows...
Here's What We Think You'll Like:
    1 Sardaar ji
    2 Insidious
    3 Vivah
    4 Night of Knots
    5 Death of Me

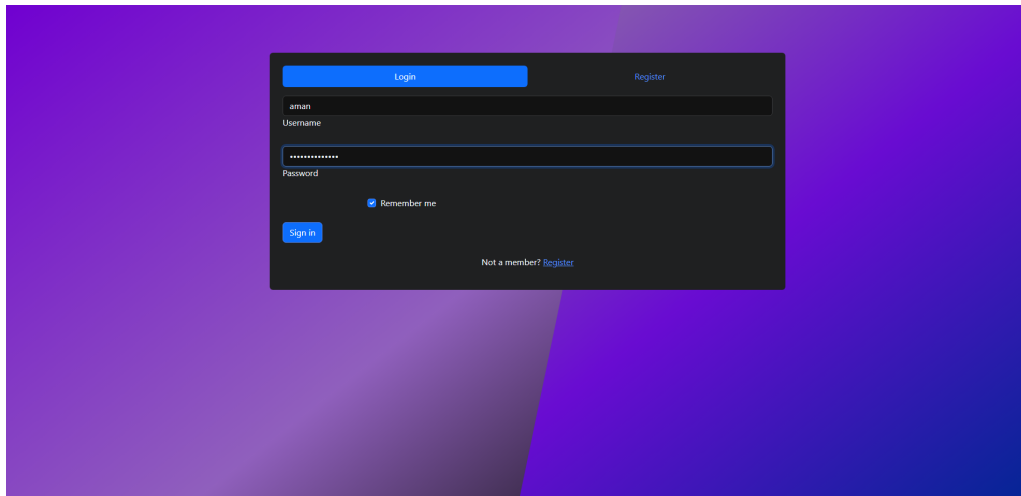
Others Also Watched:
    1 Flight of the Navigator (1986)
    2 My Fair Lady (1964)
    3 Driving Miss Daisy (1989)
    4 Inspector Gadget (1999)
    5 M*A*S*H (a.k.a. MASH) (1970)
```

Current Status of the User Interface with Description Similarity Based Searching:

Index:

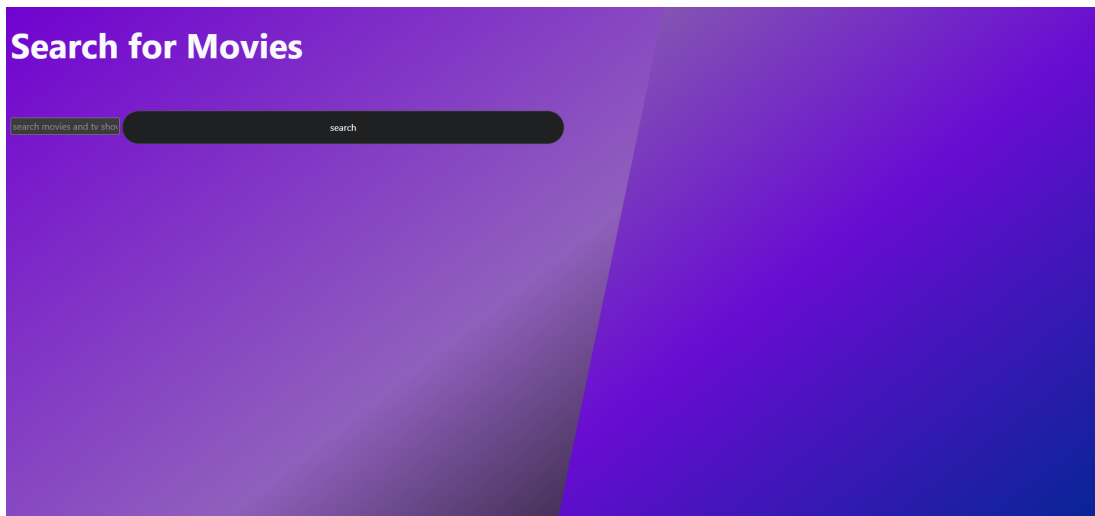


Login/Signup Page:



A login/signup page with a dark purple gradient background. A dark gray modal box is centered, containing a 'Login' button and a 'Register' link. Below these are input fields for 'Username' (containing 'aman') and 'Password' (masked with dots). A 'Remember me' checkbox is checked. A 'Sign in' button is at the bottom left, and a 'Not a member? Register' link is at the bottom right.

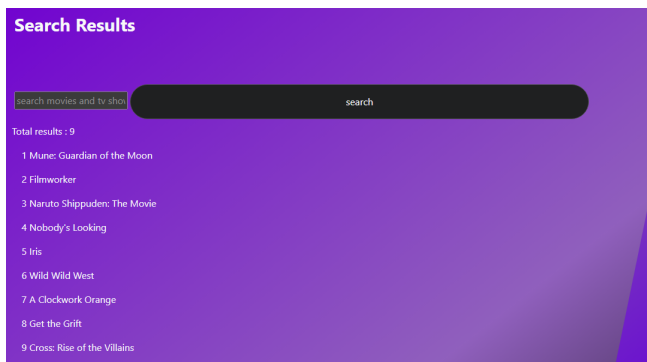
Homepage:



A homepage with a dark purple gradient background. At the top left, it says 'Search for Movies'. Below this is a search bar with a placeholder 'search movies and tv shows' and a 'search' button.

Predictive Search Based on the query:

For query “war”:

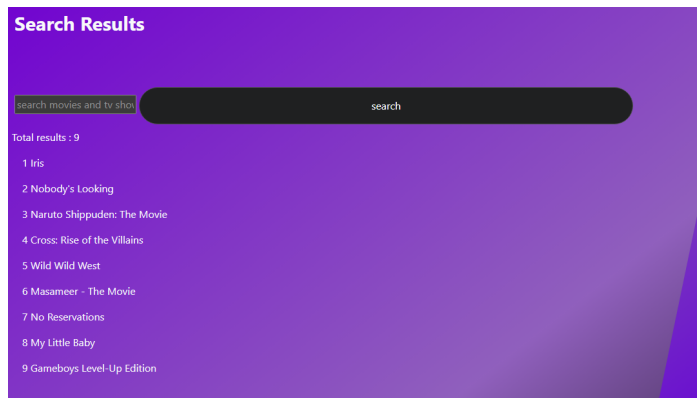


A search results page with a dark purple gradient background. At the top left, it says 'Search Results'. Below this is a search bar with a placeholder 'search movies and tv shows' and a 'search' button. The results are listed below the search bar:

Total results : 9

- 1 Mune: Guardian of the Moon
- 2 Filmworker
- 3 Naruto Shippuden: The Movie
- 4 Nobody's Looking
- 5 Iris
- 6 Wild Wild West
- 7 A Clockwork Orange
- 8 Get the Gift
- 9 Cross: Rise of the Villains

For query “lego jurassic”:



References

[1] F. Furtado , A, Singh, *Movie Recommendation System Using Machine Learning 2022*

[2] *MovMe: Personalized Movie Information Retrieval* Hyung W. Kim, Kee J. Han, Minsam Ko, Mun Y. Yi

[3] Ziv Bar-Yossef and Naama Kraus. 2011. Context-sensitive query auto-completion. In Proceedings of the 20th international conference on World wide web (WWW '11). Association for Computing Machinery, New York, NY, USA, 107–116.
<https://doi.org/10.1145/1963405.1963424>