Dharmsinh Desai University

Faculty of Technology

Department of Computer Engineering

B.Tech CE Semester – VI

System Design Practice

Movie Review Sentiment Analysis

2019-2020

Swar Patel (CE-096)
(17CEUOG132)

Priyank Chaudhari (CE-104)
(17CEUTG028)

April 23, 2020

Guided By: Prof. Hariom Pandya
Assistant Professor of
Department of Computer Engineering
Dharmsinh Desai University
Faculty of Technology
Nadiad.

Dharmsinh Desai University, Nadiad

Faculty of Technology, Department of Computer Engineering

CERTIFICATE

This is to certify that Software System Design Practice's project entitled "Movie Review Sentiment Analysis" is the bonafied report of work carried out by

1) Patel Swar (17CEUOG132)
2) Priyank Chaudhari (17CEUTG028)

Of Department of Computer Engineering, Semester VI, academic year 2019-20,

under our supervision and guidance.

Guide	HOD
Prof. Hariom Pandya	Dr. C. K. Bhensdadia
Assistant Professor of Department of Computer Engineering, Dharmsinh Desai University, Nadiad.	Head of the Department of Department of Computer Engineering, Dharmsinh Desai University, Nadiad.

Table of Contents

Abstract 4
Introduction 4
Software Requirement Specification 4
Database Design 6
Use Case Diagram 7
Activity Diagram 8
Sequence Diagram 11
Implementation Details 12
Scaping review from google. 12
Cleaning Review 12
Labelling our scraped reviews. 14
Training A Model 15
Cost function intuition 16
Purpose of TF – IDF 17
How TF-IDF is calculated? 17
What is epoch? How is it helpful? 19
Word2vec !!! 20
Creating WebApi to achieve Interoperability 22
Website Component Structure in React Framework 22
Testing 23
Screenshot 23
Conclusion 31
Limitation 31
Future Extension 31
Bibliography 31

Abstract

Our goal is to create website which analyse user's reviews and give rating to that reviews and according to that reviews system gives rating to the movies.

Introduction

People always express their views and feelings about movies in the form of writing reviews on some of the well-known websites like imdb, rottontomatoes, google.
Our task is to gather that review data, clean that data and on that cleaned data train machine learning model and then predict review is it positive or negative? Based on number of total positive and negative review give ratings to movie from total 10 stars. We
Also need to create our website to display our predicated rating and also users can write reviews so that no more we need to fetch reviews from other websites 😅 .

Software Requirement Specification

R1.Manage User:

DESCRIPTION: Users can login and can create new account and view others users by searching using the username and using email address and delete his/her account. User can follow other users. User can update his/her profile details.

R1.1)Sign Up User:

DESCRIPTION: Users can register to the application by providing his/her details like password, username, email address and date of birth information.

INPUT: Users information.
OUTPUT: Redirect to Login page.

R1.2)Sign In User:

DESCRIPTION: Users can sign-in to his/her account using his/her email and password or using his/her username.

INPUT: User information.

OUTPUT: Sign-In to respective user profile.

R2.Manage Movies:

DESCRIPTION: User can search movies and add movies by providing appropriate details

R2.1) Add Review

DESCRIPTION: Review of movie gets saved.

INPUT: Review title, content and stars.
OUTPUT: Redirect to Same page.

R2.2) Add into watch later

DESCRIPTION: Movie will get added into watch later list.

INPUT: Movie detail and user detail
OUTPUT: Movie gets added into watch later list.

R2.3) Fetch Movie

DESCRIPTION: Movie list as per specified requirement is obtained.

INPUT: Page number, limit, genre
OUTPUT: Movies as per requirement.
STATE: Admin

R2.4) Delete Movie

INPUT: Movie id
OUTPUT: Movies gets deleted
STATE: Admin

R2.5) Predict Review's Sentiment

INPUT : User enter movie review
OUTPUT: System predicts sentiment of review.

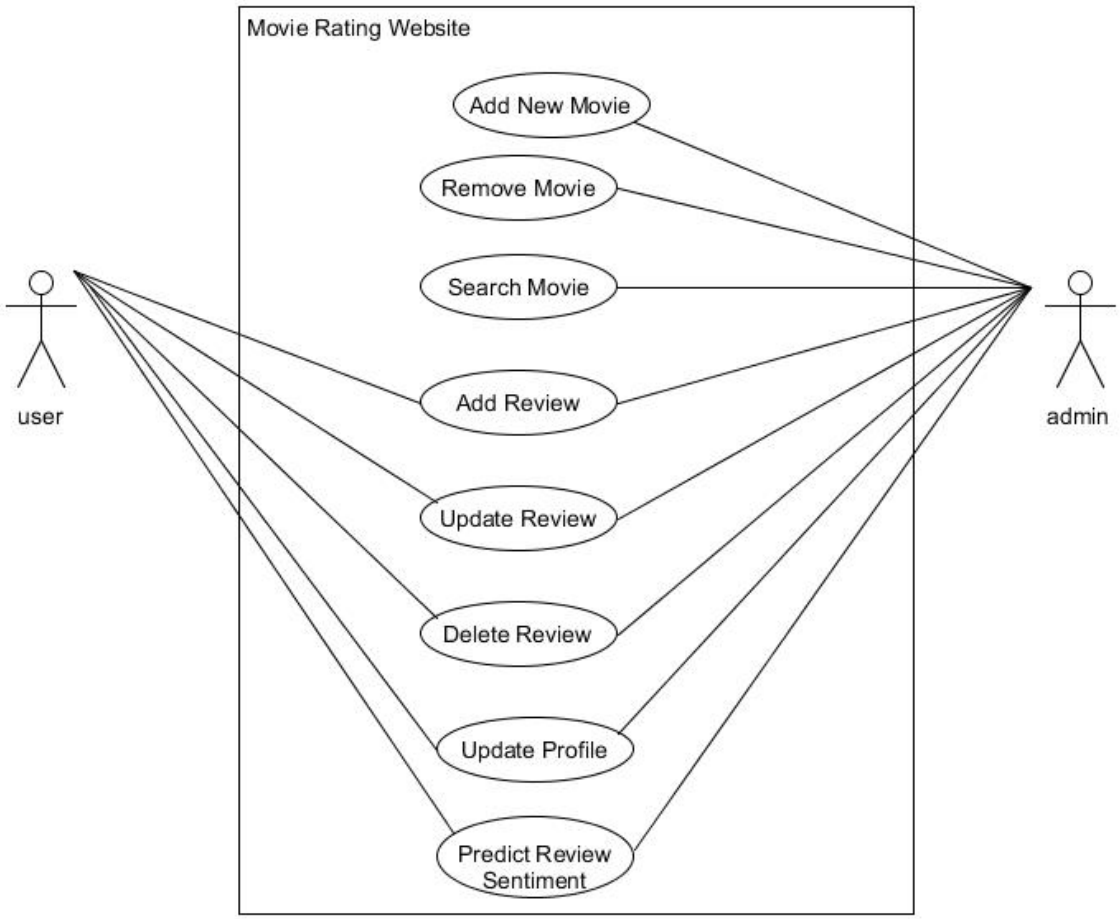
Database Design

We have only one main table which have all data related to movie.

```
const MovieSchema = new mongoose.Schema({
    imdbId: String,
    name: String,
    release_date: {type:Date},
    release_year: Number,
    release_info: mongoose.Schema.Types.Mixed,
    description: { type: String, default: null },
    genre: [String],
    poster: { type: String, default: null },
    posterimageid: String,
    stars: { type: Number, default: 0 },
    rated: { type: String, default: 'unrated' },
    running_time: mongoose.Schema.Types.Mixed,
    trailer_link: { type: String, default: null },
    images:[{
        id:String,
        h:{ type: Number, default: 0 },
        w:{ type: Number, default: 0 },
        msrc:{ type: String, default: null },
        src:{ type: String, default: null },
        altText:{ type: String, default: null }
    }],
    cast: [{
        title: String,
        persons:[{
            image: { type: String, default: null },
            link: { type: String, default: null },
            name: { type: String, default: null },
            role: { type: String, default: 'Not Specified' }
        }]
    }],
    reviews: [{
        heading:String,
        review:String,
        imdb_rating:{ type: Number, default: 0 },
        our_rating:{ type: Number, default: 0 },
        aho_label:{ type: Number, default: 0 },
    }],
    updated: { type: Date, default: Date.now }
});
```

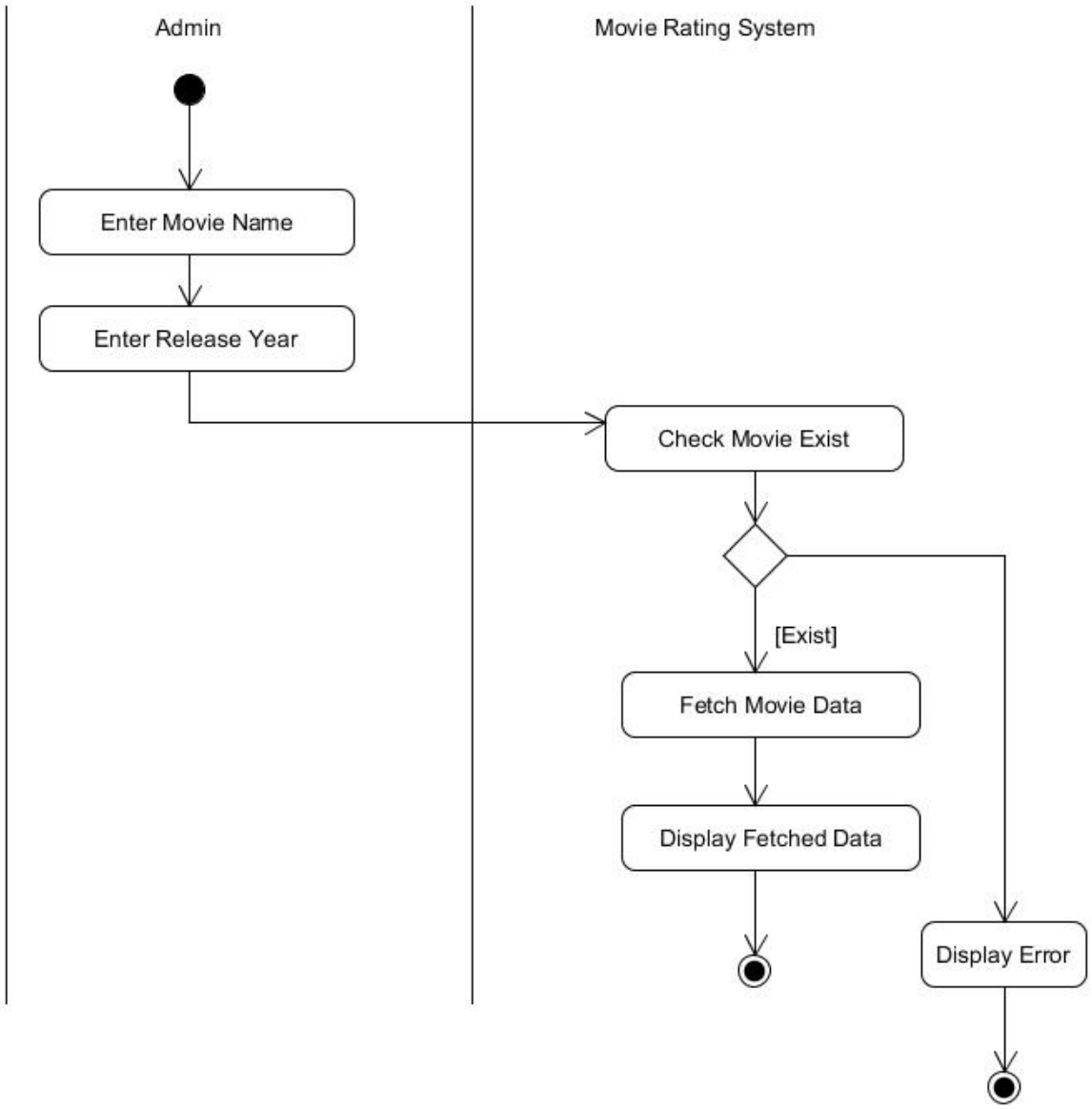
```
const userSchema = new Schema({
    username: String,
    email: String,
    password: String,
    favourite_list: [String]

})
```

Use Case Diagram

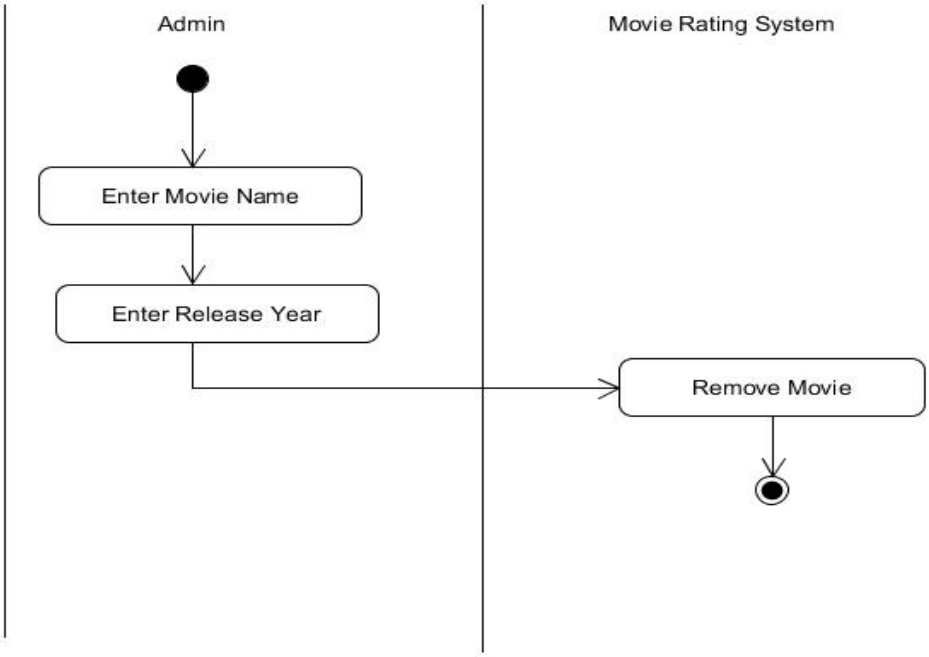


Activity Diagram

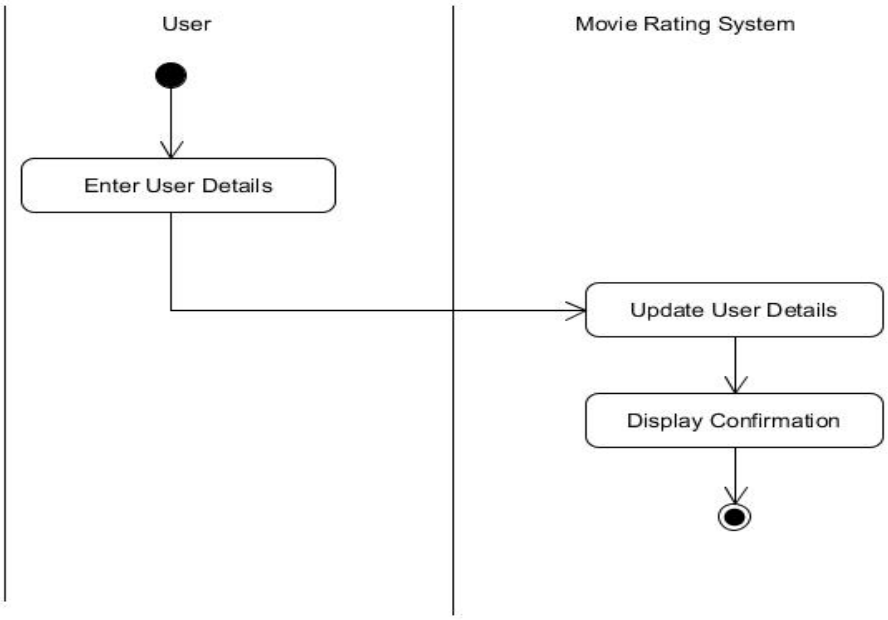
Add Movie :



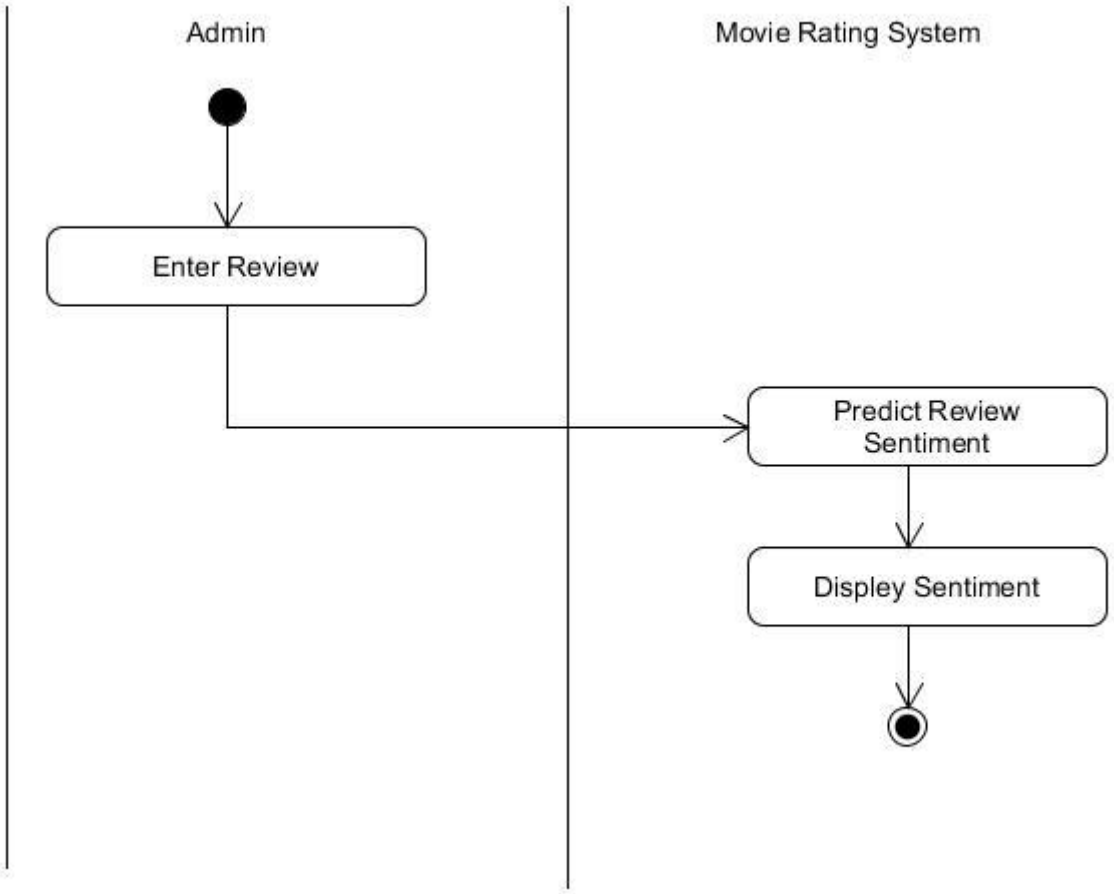
Remove Movie :



Update Profile :

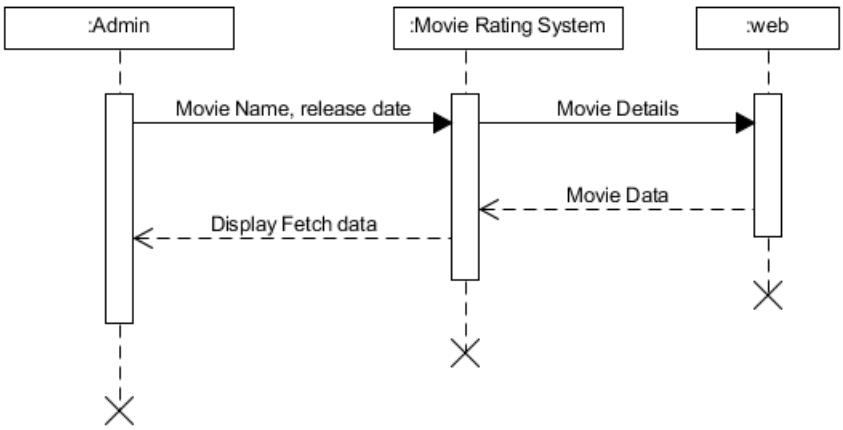


Predict Review Sentiment :

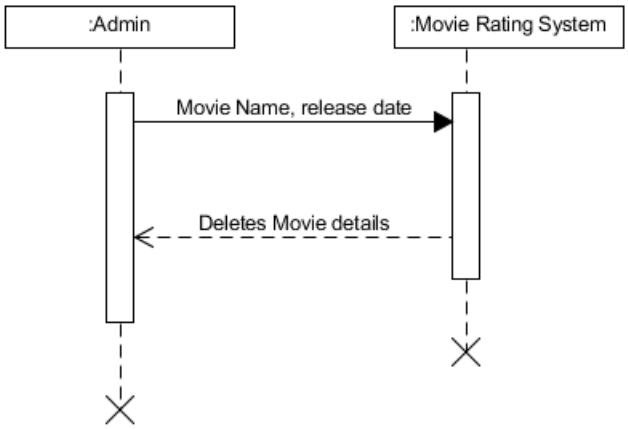


Sequence Diagram

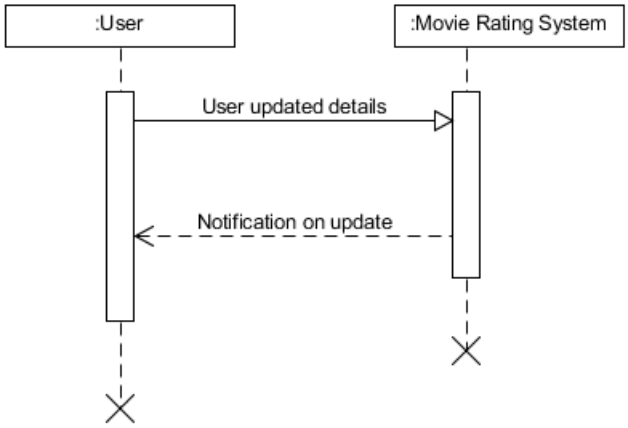
Add Movie :



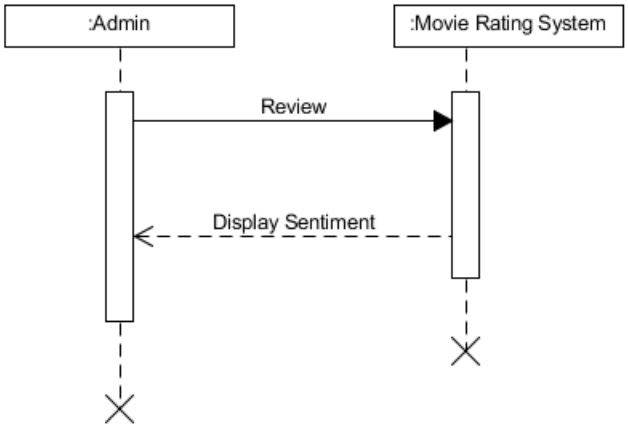
Remove Movie :



Update Profile:



Predict Review Sentiment:



Implementation Details

Our Workflow for Creating model.
Steps:

Scaping review from google.

- We find already created good dataset of imdb movie reviews. But we don't want use that so…… we scraped reviews from google using NodeJS with puppeteer library (we wasted our so much precious internet data and it take so much time to scape more than 1000 movie's reviews.).

- How we did scraping?

  We download csv file which contains list of Bollywood movies.
  We read csv file and then combine movie name with it's release year
  and used that for searching in google (only movie name is sometimes not working). after loading movie page we search for particular html tag which gives us list of reviews and then we store that reviews in json file. Hey we know that you have question about labelling review it's in 3rd step but don't skip step 2.

Cleaning Review

- It's most important step for training your model. Why? Because raw text contains maybe spelling mistakes, punctuation marks, emojis, words like

goooood. and if we don't clean that then our model is trained on this misspelled word and emojis and punctuation marks that we don't want.

- How we cleaned our data?
  We first observed some of our scarped reviews and we found that…….

1. People uses emojis so much frequently. 6 out of 10 reviews has emojis so we can't remove emojis from raw text because it's important factor for identify positivity or negativity of review. So what we did? We created list of positive emojis and list of negative emojis and then replace positive emoji in our raw text with "emo_pos" and negative emoji with "emo_neg".
   emo_pos - 😊 👌 😍 😇 😔 😉 and so many more…
   emo_neg - 😡 😤 😢 😒 👎 and so many more….
   We know that there are so many emojis but we repleced only few of them which we know that it is conveying meaning positivity or negativity.

2. In raw text few words are in uppercase and others are in lowercase. We have to convert all words into lower case why? Because "good" and "Good" have same meaning for us but model treat them differently.

3. There is unnecessary space in raw text so we removed them. There is puncution marks, symbols and digits we also removed that because at the end our model is only going to work with words.

4. STOPWORDS - Stop words are generally the most common words in a language. e.g. the, is, a ,an, but, again, some, there, once, of, am, for, do, yours,…….. we have to remove them because they are not going to convey any positivity or negativity and it's also reduce our dataset file size.

5. Stemming - stemming is the process of reducing inflected (or sometimes derived) words to their word stem.
   There is Two type of popular stemming algorithm. (i) Porter (ii) Lancaster.
   We used Porter stemmer which is less strict and faster.
   Lemmatization - Lemmatization usually refers to the morphological analysis of words, which aims to remove inflectional(related) endings. It helps in returning the base or dictionary form of a word, which is known as the lemma.
   e.g.
   text:

   Stemming is funnier than a bummer says the sushi loving computer scientist movie moved best beautiful moves studies study studying cries

   After applying porter stemmer:

   Stem is funnier than a bummer say the sushi love comput scientist movi Move best beauty studi studi studi cri.

   After applying lemmatization :

   Stemming is funnier a bummer say the sushi loving computer scientist movie moved best beautiful move study study studying cry.

What is difference between lemmatization and stemming?

> Somewhat similar but there is difference that a stemmer operates on single word without knowledge of context and therefore cannot discriminate between words which have different meaning while lemmatization preserve the meaning. e.g. we know that “marketing“ and “market” has different meaning but if we do stemming then “marketing” is replaced with “market”
> While in lemmatization “marketing” remain same.

Why stemming and lemmatization?

> It reduces the word density in the given text and helps in preparing the accurate features for training machine. Cleaner the data, the more intelligent and accurate your machine learning model, will be.
>
> We used “text-miner” and “lemmatizer” npm package to do cleaning.
> Check this website to do online lemmatization and stemming.

Labelling our scraped reviews.

> So we first downloaded list of positive and negative words. And then based on count of positive and negative words in reviews we labelled them as 1 (positive) and 0 (negative). Way we did this is wrong for some of the reviews. why….? Idiot is negative word. And we know that 3 idiots is great movie with 8+ imdb rating. But most of the reviews contains idiot word which not referencing negativity but it referring to movie name. so due to the idiot word-count half of reviews of that movie are labelled as negative. So what’s solution…? Whatever the idea we used for labelling review is not good but it happening with very few reviews and after the training model we again predicted reviews of 3 idiots most of are predicted as positive as expected. Good way to label the movie review data is based on the stars given by each user(more than 7 then positive, less than 4 then negative, for 5 6 neutral). On google every movie don’t have reviews with stars but imdb have that. So good way is to fetch reviews and stars from imdb.

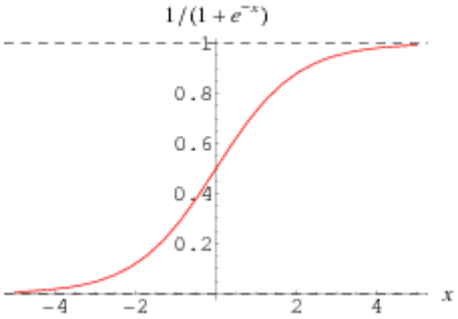
What we used for finding count of positive and negative words?

> Popular string matching algorithm rabin-karp, knuth-morris-pratt takes O(n) (where n is length of text) for single word and we have list of 1000+ words of positive and 1000+ words of negative and total 25000 reviews. so we need to run kmp or rabinkarp for 2000 times for one review. Looks terrible…….
> But our boat is not going to sink because there is great aho-corasick string matching algorithm which runs with time complexity of O(N + L + Z) where Z is number of pattern, N is length of text and L is total number of characters in all words(patterns). This algorithm search whole 1000 words in 1 go. This algorithm uses Trie data structure for all pattern words. For this we used aho-corasick npm package. See this visualization of this algorithm.

Training A Model.

What type of algorithm can be used for classification?

As we have to do classification linear regression is not going to work because it is used to determine continuous value(numeric). Unlike linear regression, the dependent variable is categorical (in our case positive and negative). If we take the weighted sum of inputs as the output as we do in Linear Regression, the value can be more than 1 but we want a value between 0 and 1. That's why Linear Regression can't be used for classification tasks.



Sigmoid function

(credit: http://mathworld.wolfram.com/SigmoidFunction.html)

Logistic Regression is a generalized Linear Regression in the sense that we don't output the weighted sum of inputs directly, but we pass it through a function that can map any real value between 0 and 1. That function is called sigmoid function. check out this online sigmoid calculator.

A Linear Regression model can be represented by the equation.

$$h(x) = \theta^T x$$

We then apply the sigmoid function to the output of the linear regression

$$h(x) = \sigma(\theta^T x)$$

where the sigmoid function is represented by,

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

$$h(x) = \frac{1}{1 + e^{-\theta^T x}}$$

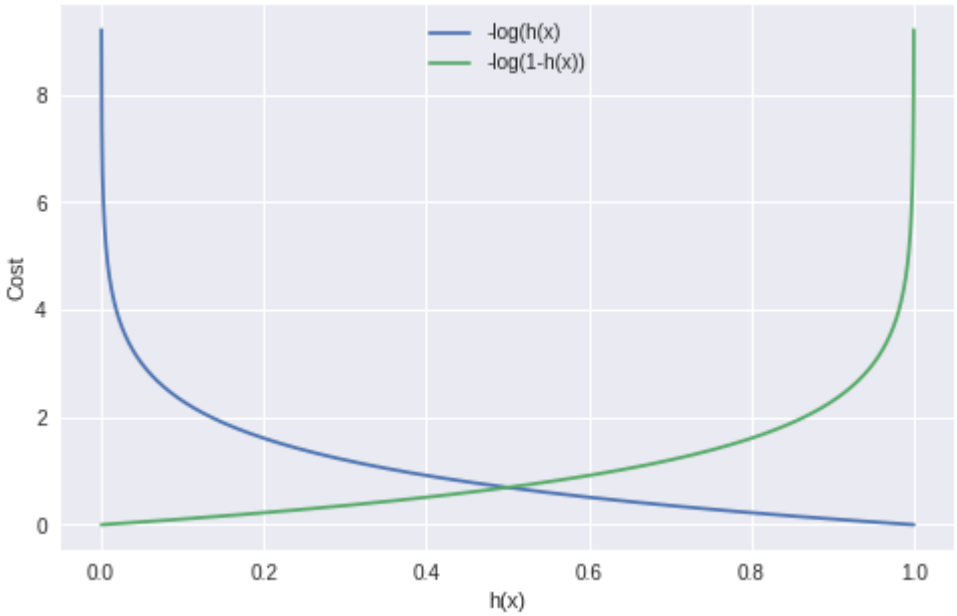
$$h(x) = \begin{cases} > 0.5, & \text{if } \theta^T x > 0 \\ < 0.5, & \text{if } \theta^T x < 0 \end{cases}$$

The cost function for a single training example can be given by:

$$cost = \begin{cases} -log(h(x), & \text{if } y = 1 \\ -log(1 - h(x)), & \text{if } y = 0 \end{cases}$$

Cost function intuition

If the actual class is 1 and the model predicts 0, we should highly penalize it and vice-versa. As you can see from the below picture, for the plot -log(h(x)) as h(x) approaches 1, the cost is 0 and as h(x) nears 0, the cost is infinity(that is we penalize the model heavily). Similarly for the plot -log(1-h(x)) when the actual value is 0 and the model predicts 0, the cost is 0 and the cost becomes infinity as h(x) approaches 1.



(credit : https://towardsdatascience.com/building-a-logistic-regression-in-python-301d27367c24)

For minimizing cost gradient descent is used. Gradient descent is algorithm to find local minimum of function. If we take small step in opposite direction of gradient then we reach to local minimum. (Gradient is derivation of cost function & we know that dy/dx is used for finding minima and maxima).

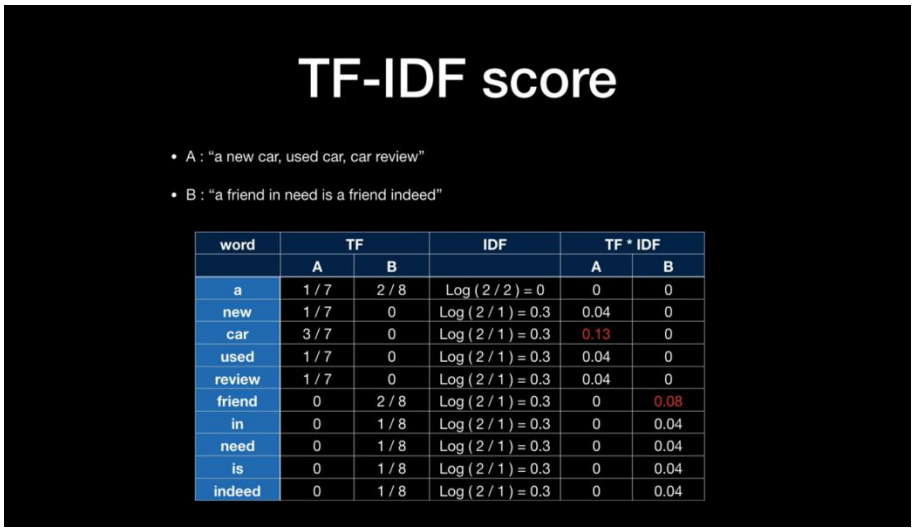
But now what to pass to this logistic regression model? model cannot directly work with text (reviews). it only works with numeric value so some how we have to convert our reviews to numeric values.

Purpose of TF – IDF

TF-IDF help us to find correlation between reviews.
TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document in a collection of documents. This is done by multiplying two metrics: how many times a word appears in a document, and the inverse document frequency of the word across a set of documents.

TF-IDF was invented for document search and information retrieval. It works by increasing proportionally to the number of times a word appears in a document, but is offset by the number of documents that contain the word. So, words that are common in every document, such as this, what, and if, rank low even though they may appear many times, since they don't mean much to that document in particular.

How TF-IDF is calculated?



word	TF		IDF	TF * IDF	
	A	B		A	B
a	1 / 7	2 / 8	Log (2 / 2) = 0	0	0
new	1 / 7	0	Log (2 / 1) = 0.3	0.04	0
car	3 / 7	0	Log (2 / 1) = 0.3	0.13	0
used	1 / 7	0	Log (2 / 1) = 0.3	0.04	0
review	1 / 7	0	Log (2 / 1) = 0.3	0.04	0
friend	0	2 / 8	Log (2 / 1) = 0.3	0	0.08
in	0	1 / 8	Log (2 / 1) = 0.3	0	0.04
need	0	1 / 8	Log (2 / 1) = 0.3	0	0.04
is	0	1 / 8	Log (2 / 1) = 0.3	0	0.04
indeed	0	1 / 8	Log (2 / 1) = 0.3	0	0.04

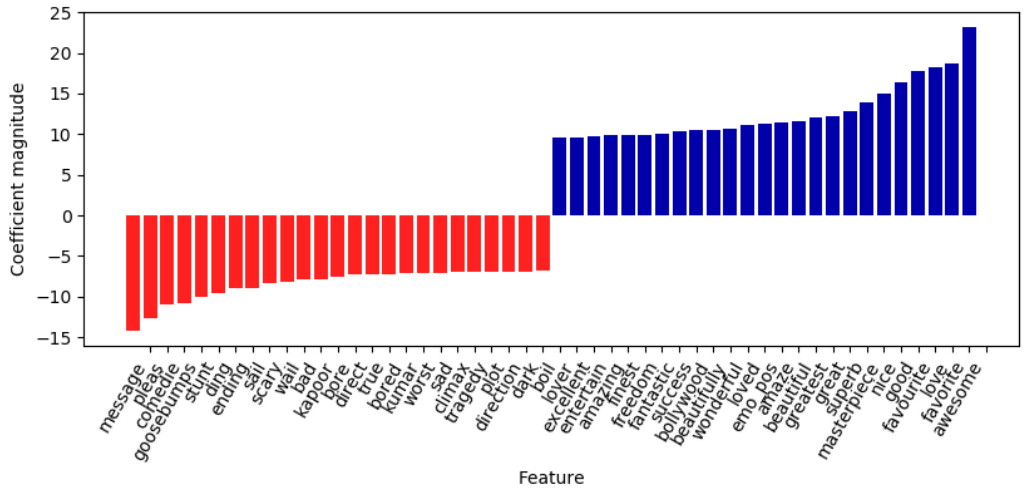
(credit : https://www.youtube.com/watch?v=G1bof7UL9RU&list=PLwyIYCshlcyHtMTL8dvUoeM_CF2h0X2JC&index=4)

The **term frequency** of a word in a document. There are several ways of calculating this frequency, with the simplest being a raw count of instances a word appears in a document. Then, there are ways to adjust the frequency, by length of a document, or by the raw frequency of the most frequent word in a document.

The **inverse document frequency** of the word across a set of documents. This means, how common or rare a word is in the entire document set. The closer it is to 0, the more common a word is. This metric can be calculated by taking the total number of documents, dividing it by the number of documents that contain a word, and calculating the logarithm. So, if the word is very common and appears in many documents, this number will approach 0. Otherwise, it will approach 1.

Why TF-IDF works here?

Vectorizing a document is taking the text and creating one of these vectors, and the numbers of the vectors somehow represent the content of the text. TF-IDF enables us to gives us a way to associate each word in a document with a number that represents how relevant each word is in that document. Then, documents with similar, relevant words will have similar vectors, which is what we are looking for in a machine learning algorithm.



After Doing TF-IDF, in our vocabulary which top 25 words in positive words has highest magnitude and which 25 Words in negative words has lowest highest magnitude.

Vocabulary created based on our positive reviews and negative review corpus we also considered n-gram of size 2.
Our vocabulary size has total 9462 words.

Why only 9642 words……?

We have dataset of size approx 25000 reviews which contains so many words. Vocabulary size is reduced because we(tfidfvectorizer) removing words which occurring so much frequently and also the word which occurring so much rarely. Our focusing on medium frequency words. So Each review has vector size of 9642.

We trained our model using logistic regression which gives us accuracy of 92%.
We also trained tfidf with Neural Network which give accuracy of 89%.
(for Neural Network Keras library is used).

What is epoch? How is it helpful?

An epoch describes the number of times the algorithm sees the entire data set. So, each time the algorithm has seen all samples in the dataset, an epoch has completed.
We tried different epochs we get best accuracy at 3-epoch. When used more than 6-epoch it try to overfit the data and accuracy is reduced.

What is confusion matrix, precision and recall and f1 score?

Confusion Matrix:

	Class 1 Predicted	*Class 2 Predicted*
Class 1 Actual	TP	FN
Class 2 Actual	FP	TN

True Positive (TP) : Observation is positive, and is predicted to be positive.
False Negative (FN) : Observation is positive, but is predicted negative.
True Negative (TN) : Observation is negative, and is predicted to be negative.
False Positive (FP) : Observation is negative, but is predicted positive.

(credit : https://www.geeksforgeeks.org/confusion-matrix-machine-learning/)

A confusion matrix is a summary of prediction results on a classification problem. The number of correct and incorrect predictions are summarized with count values and broken down by each class. This is the key to the confusion matrix. The confusion matrix shows the ways in which your classification model is confused when it makes predictions. It gives us insight not only into the errors being made by a classifier but more importantly the types of errors that are being made.

$$Recall = \frac{TP}{TP + FN}$$

Recall can be defined as the ratio of the total number of correctly classified positive examples divide to the total number of positive examples. High Recall indicates the class is correctly recognized (a small number of FN).

Precision:

$$Precision = \frac{TP}{TP + FP}$$

To get the value of precision we divide the total number of correctly classified positive examples by the total number of predicted positive examples. High Precision indicates an example labelled as positive is indeed positive (a small number of FP).

F-measure:
Since we have two measures (Precision and Recall) it helps to have a measurement that represents both of them. We calculate an F-measure which uses Harmonic Mean in place of Arithmetic Mean as it punishes the extreme values more. The F-Measure will always be nearer to the smaller value of Precision or Recall.

$$F\text{ - measure} = \frac{2*\text{Recall}*\text{Precision}}{\text{Recall} + \text{Precision}}$$

Why it's important?

Assume we are doing classification for cancer detection problem. In our dataset we have data of 10000 people. In that 9980 people don't have cancer and only 20 people have cancer. Now after training our model predict all 10000 people don't have cancer then what is accuracy 9980/10000 = 99.80%.
But it's not actually not good model, it cannot detect person has cancer. We calculate f1-score for person who have cancer than we get 0. so f1-score reflects real accuracy.

Below is Classification report of our model which trained using tfidf + logistic regression.

	precision	recall	f1-score	support
0	0.89	0.90	0.90	3386
1	0.94	0.93	0.93	5164
accuracy			0.92	8550
macro avg	0.91	0.91	0.91	8550
weighted avg	0.92	0.92	0.92	8550

- **Other Technique we used for classification:**

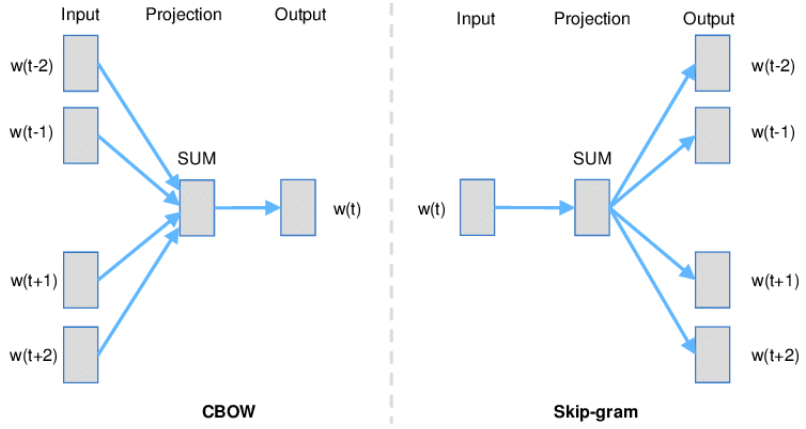
Word2vec !!!

Word2vec is also used for finding correlation between words. What is correlation? E.g "good" and "awesome" two words has same meaning so there vector should be nearer to each other.

But How this vectors are generated?
There are 2 main algorithm for word2vec
(i) Continuous Bag of word
(ii) Skip-gram

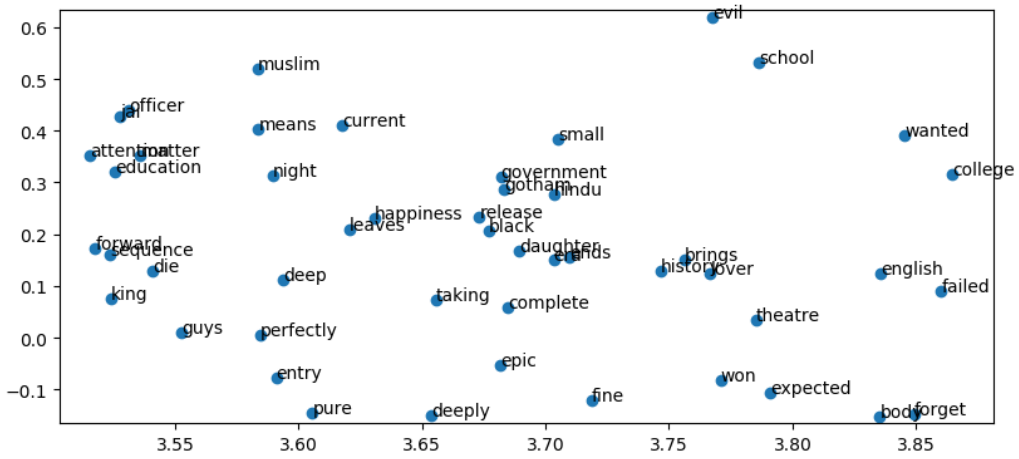
The major difference between these two methods is that CBOW is using context to predict a target word while skip-gram is using a word to predict a target context. Generally, the skip-gram method can have a better performance compared with CBOW method, for it can capture two semantics for a single word. For instance, it

will have two vector representations for Apple, one for the company and another for the fruit.



(credit : https://www.researchgate.net/figure/Continuous-Bag-of-words-CBOW-CB-and-Skip-gram-SG-training-model-illustrations_fig1_326588219)

We used genism library to get word embedding and to generate doc2vec.
If we sum vector of words of review then we get doc2vec.
Using logistic regression and doc2vec we trained our model and we get accuracy of 60.12%.
After that we used neural network & doc2vec to train model which gives improvement. It gives accuracy of 67%. For neural Network we used Keras library.



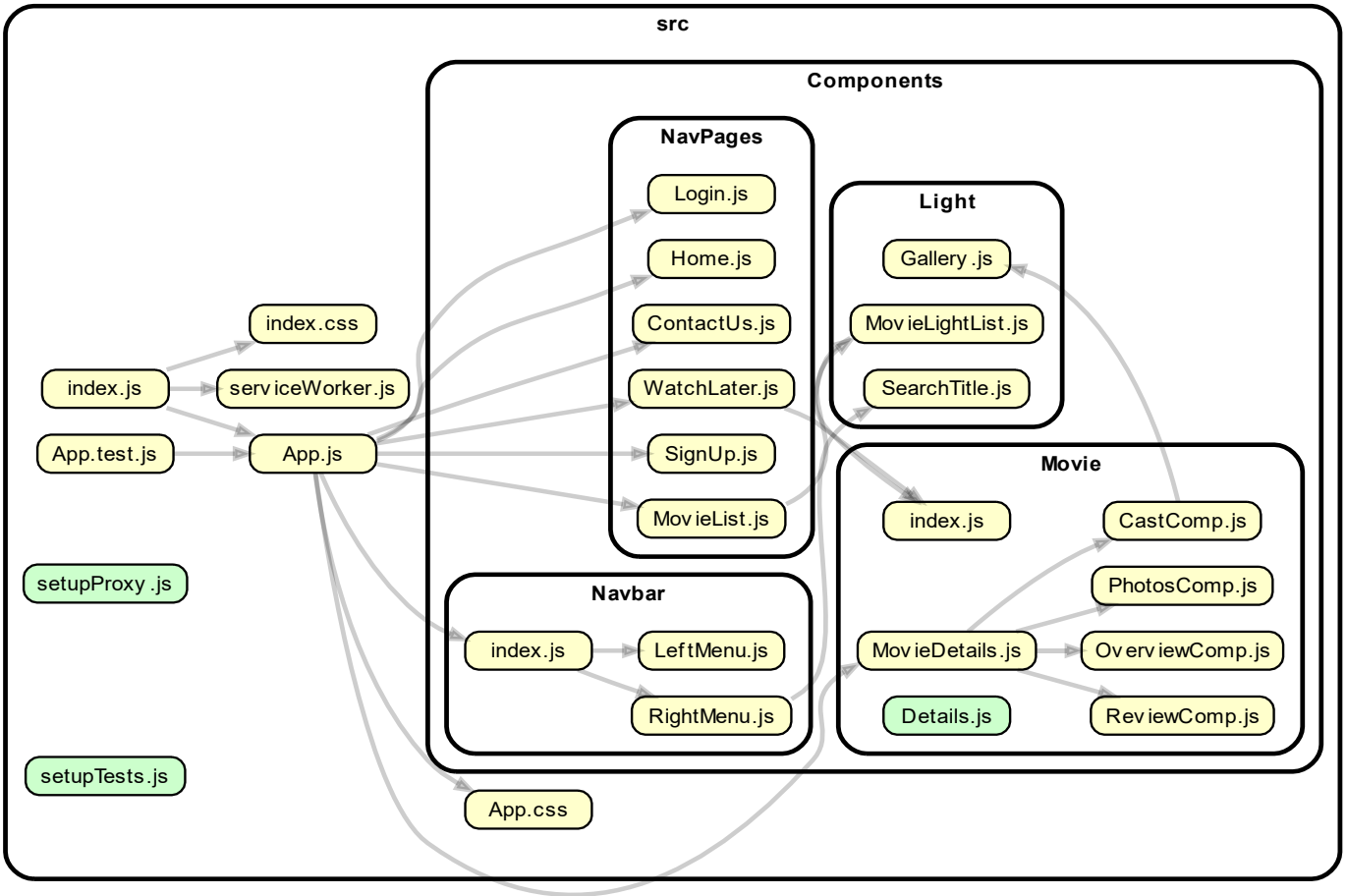
(using PCA vectors dimension is reduced to 2)
PCA – principal component analysis – it used for dimensionality reduction.
We have one word vector size is 8000. Using PCA we reduced it to 2 dimension for plot the words.

Creating WebApi to achieve Interoperability.

Our website is created in React, Nodejs and our model is trained in python so that's why we created webapi using Flask Framework. For that we saved our model* and regularization parameter and loaded into webapi to serve request. Our webapi expecting review in request and in response we are sending 0(negative) or 1(positive).

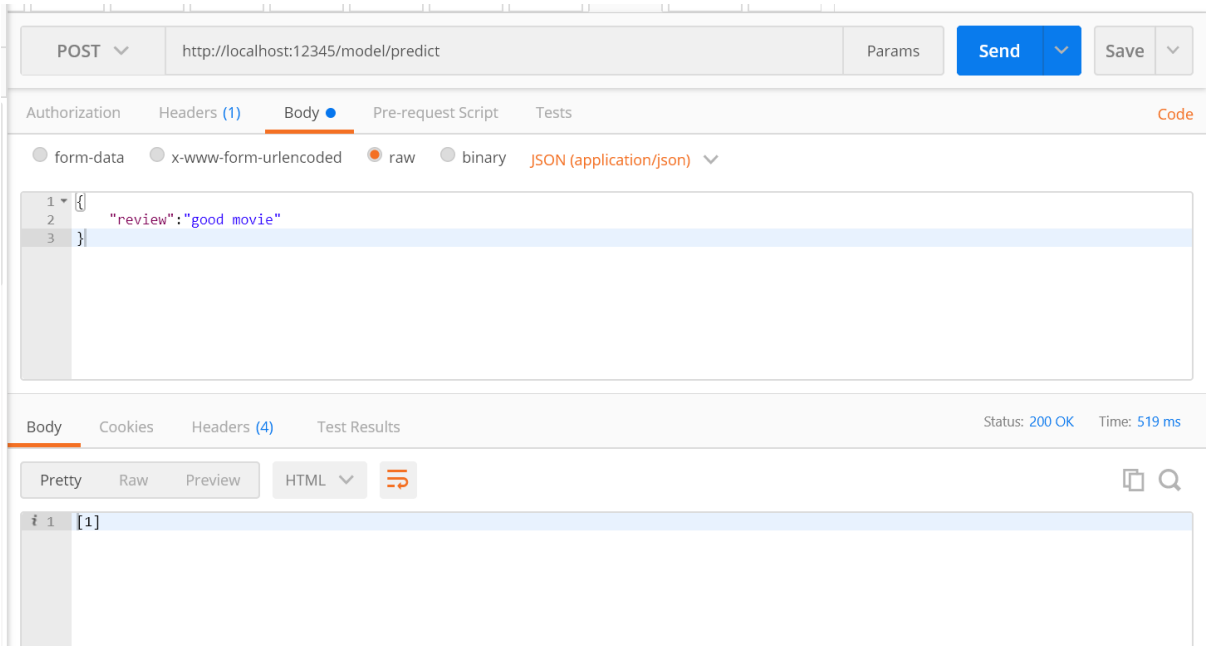
[* we are using logistic regression and tfidf model because it gives best accuracy of 92%]

Website Component Structure in React Framework:



Testing

We created our models in python and to consume that model for prediction we created flask web api which we tested using Postman app.

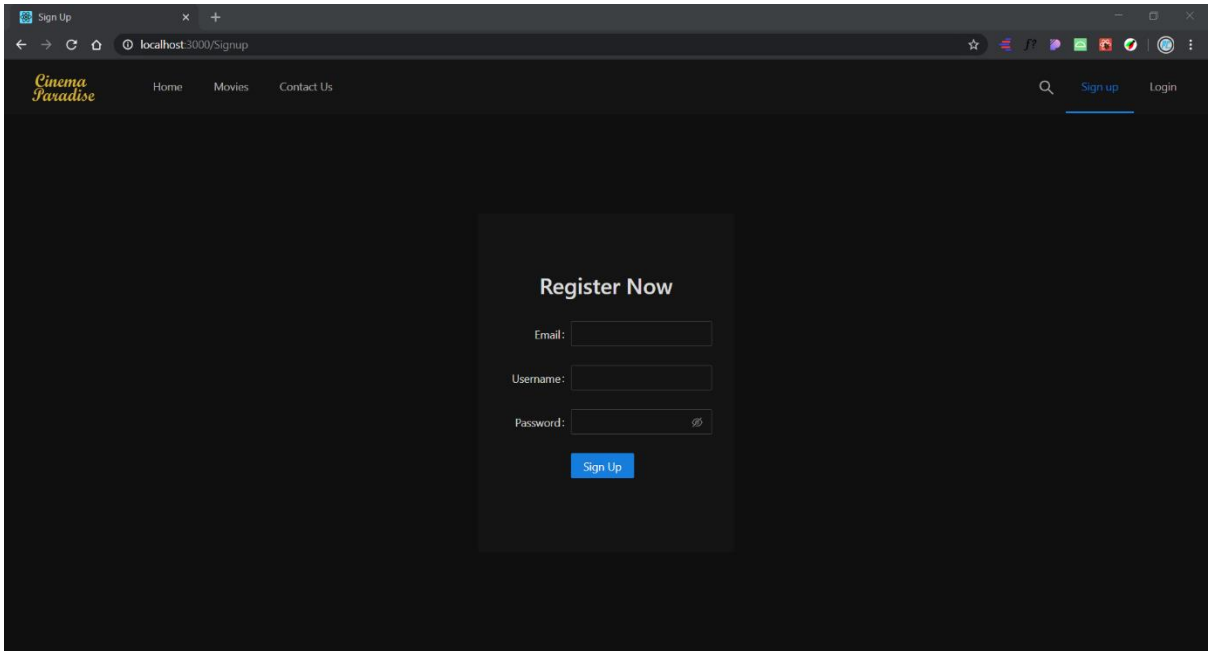


- We tested other endpoints using curl and postman.

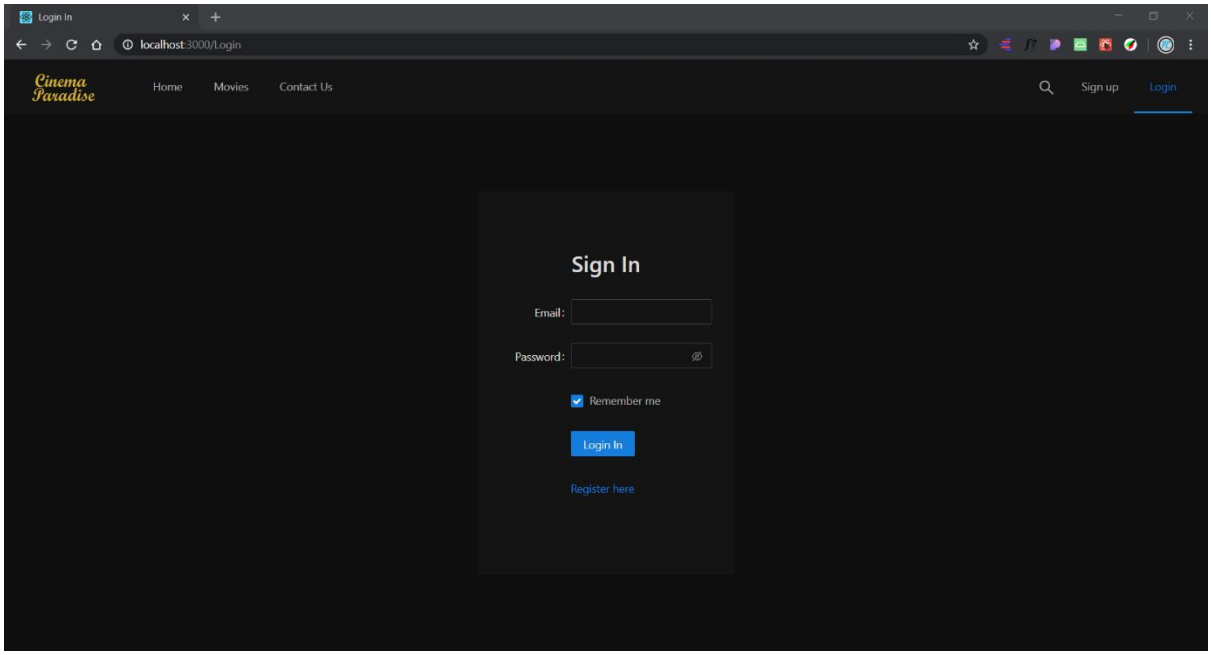
- Using Apache Jmeter we did stress testing & found how much request it can able to handle it concurrently.

Screenshot

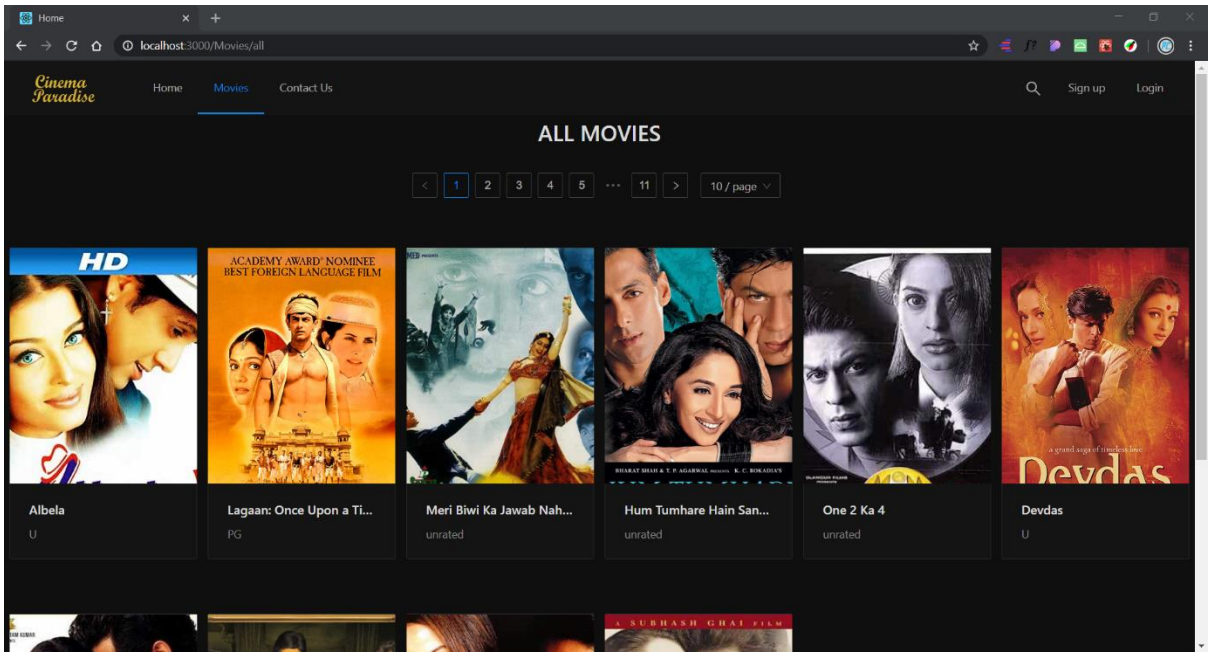
Sign Up Page



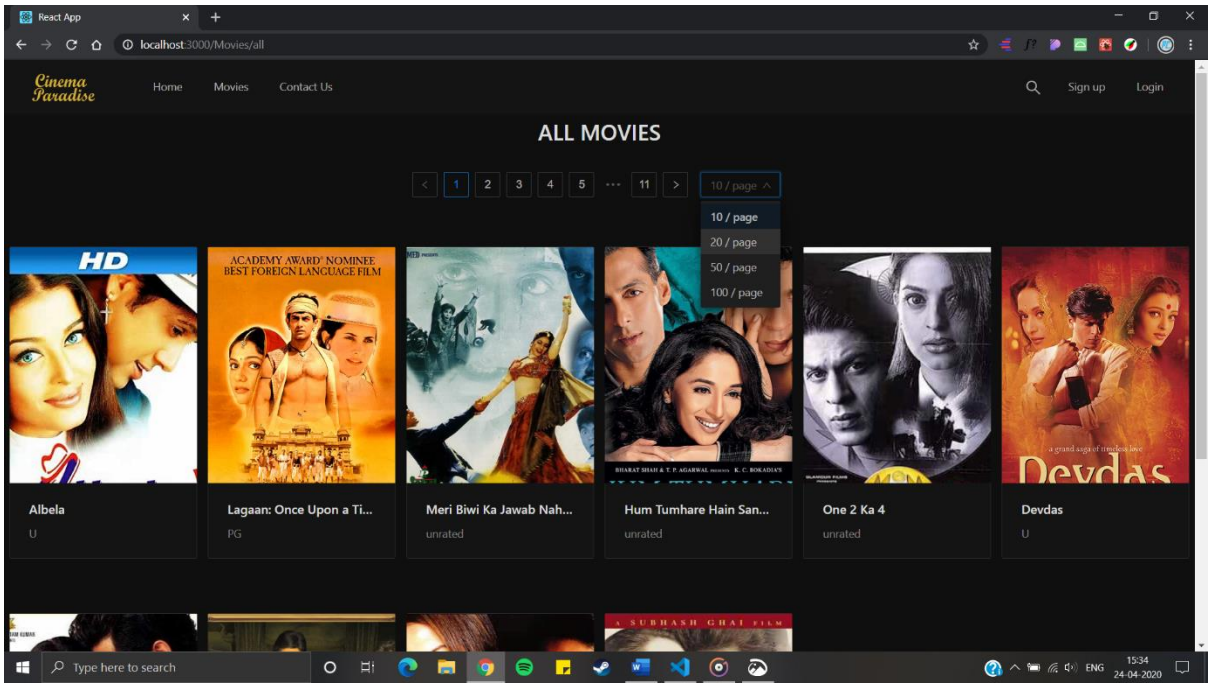
Sign In Page



Movie List page

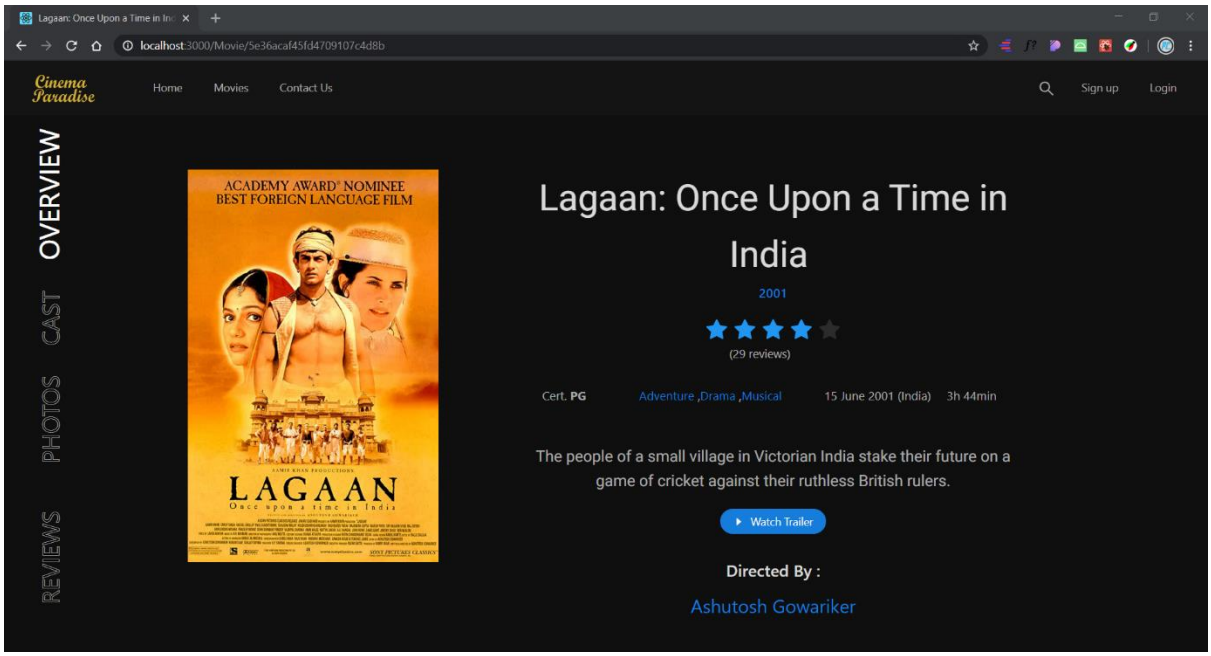


Movie list page where you can change results per page

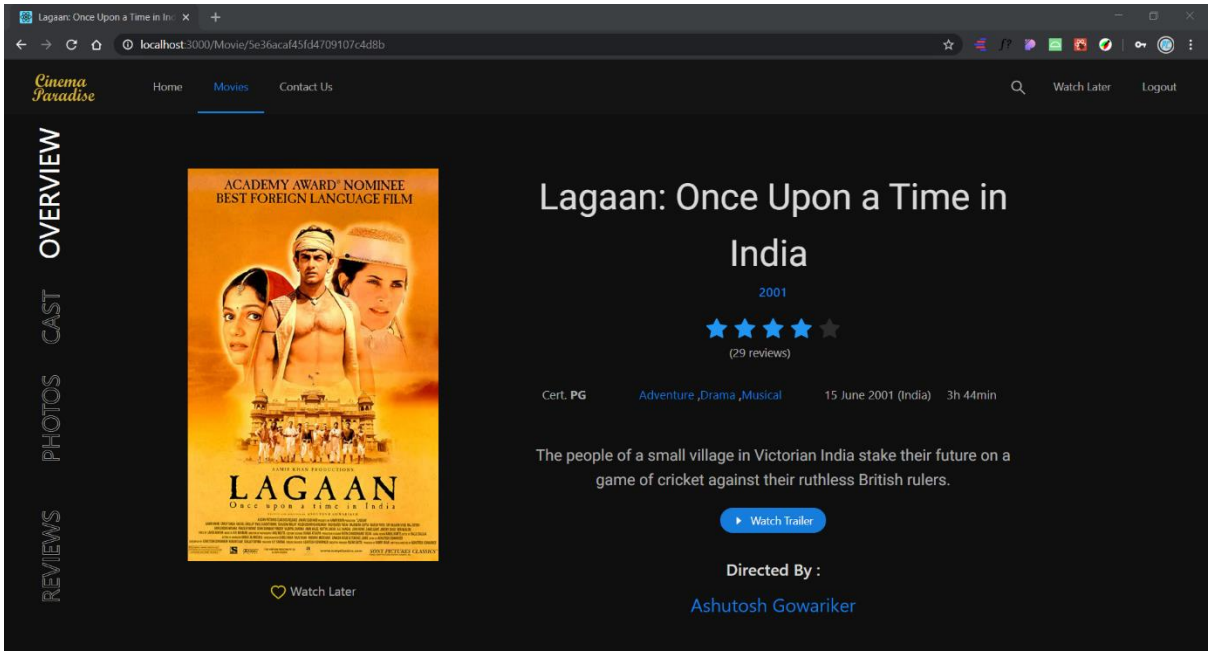


Movie detail overview page
(genre links are clickable where you will be navigated to movie list page with specific genre later in this screenshots)

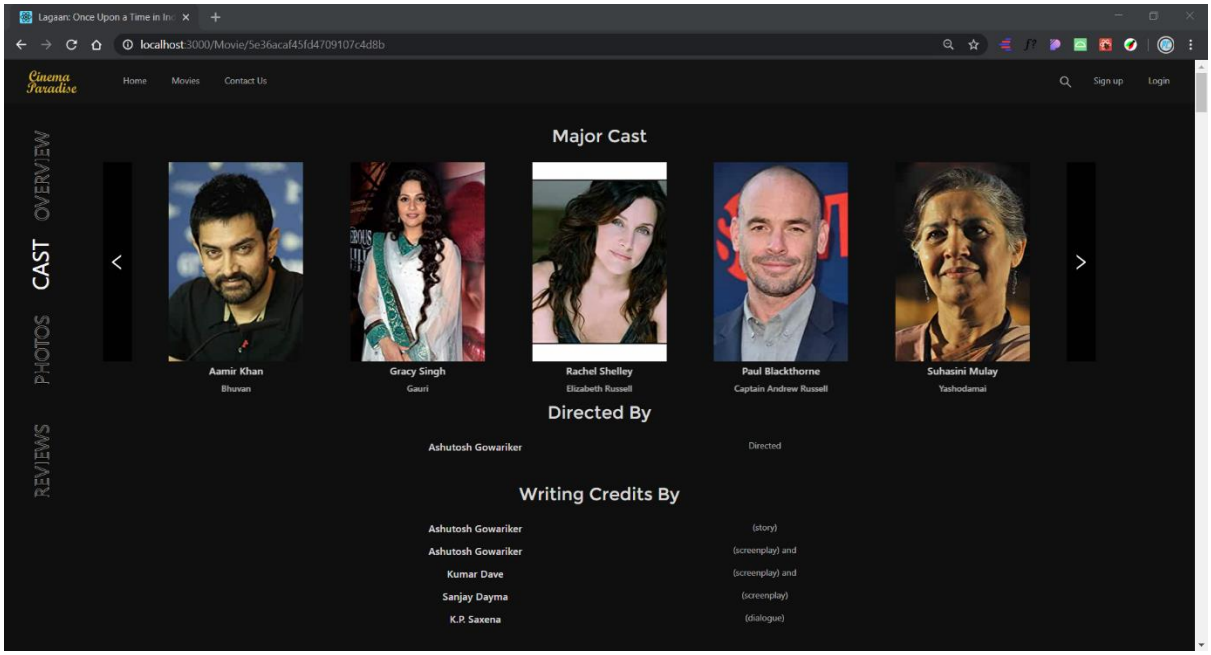
1.Without login

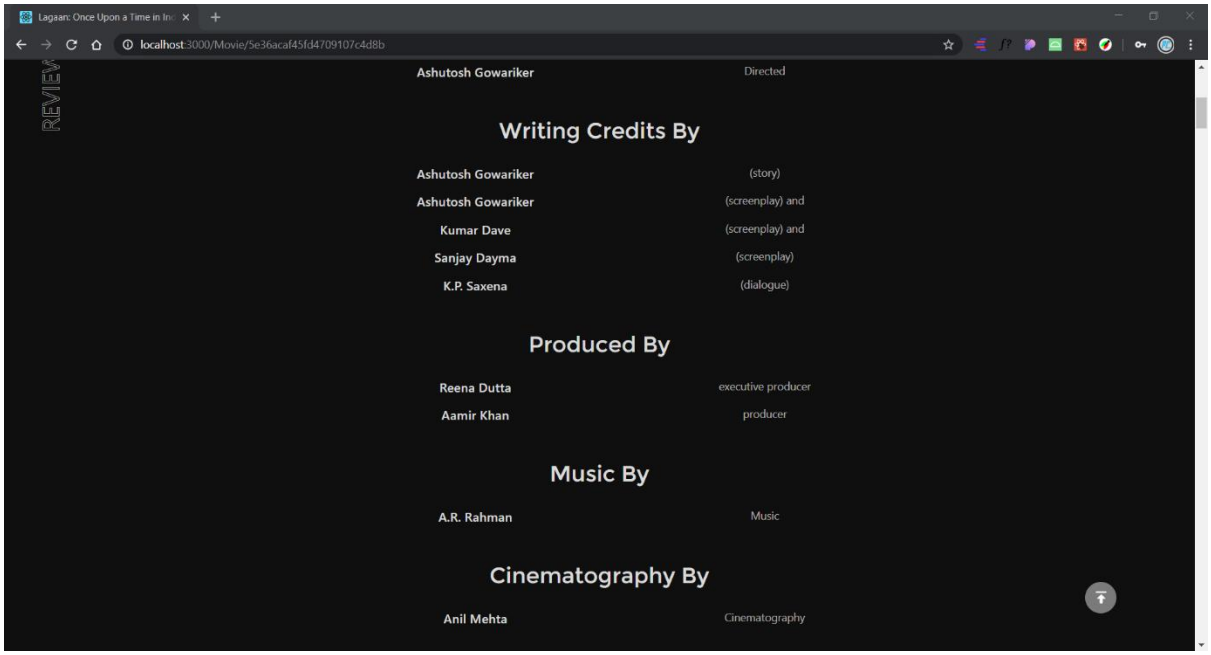


2.With Login (Watch later functionality)

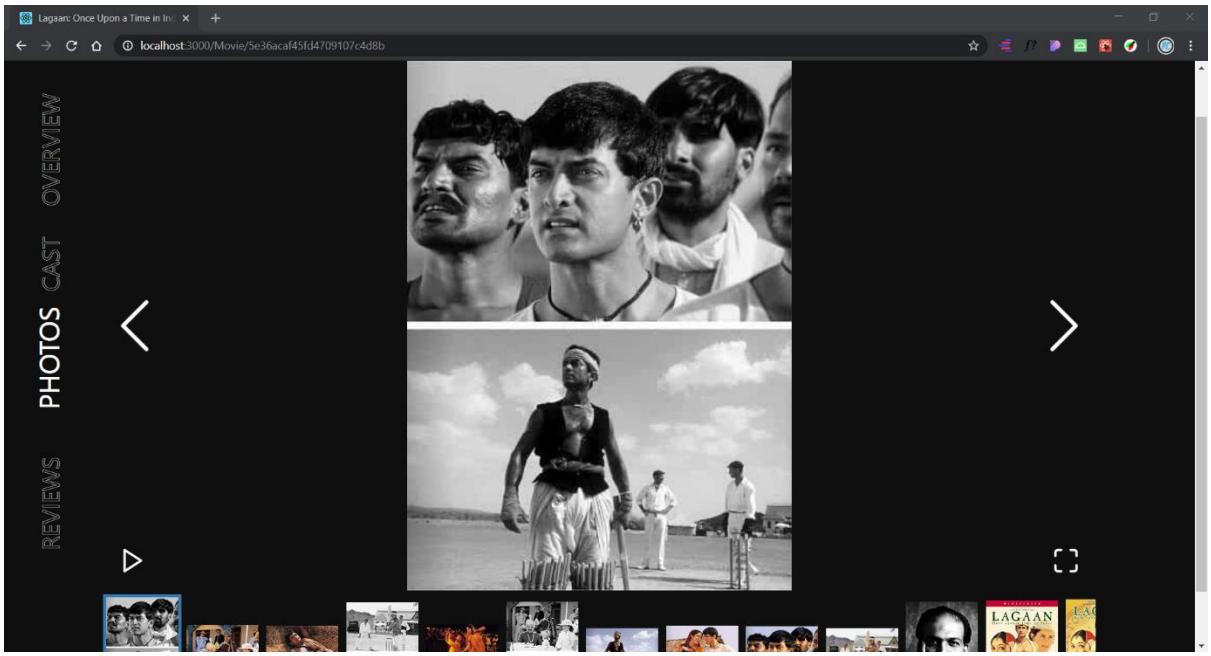


Movie Details Cast page

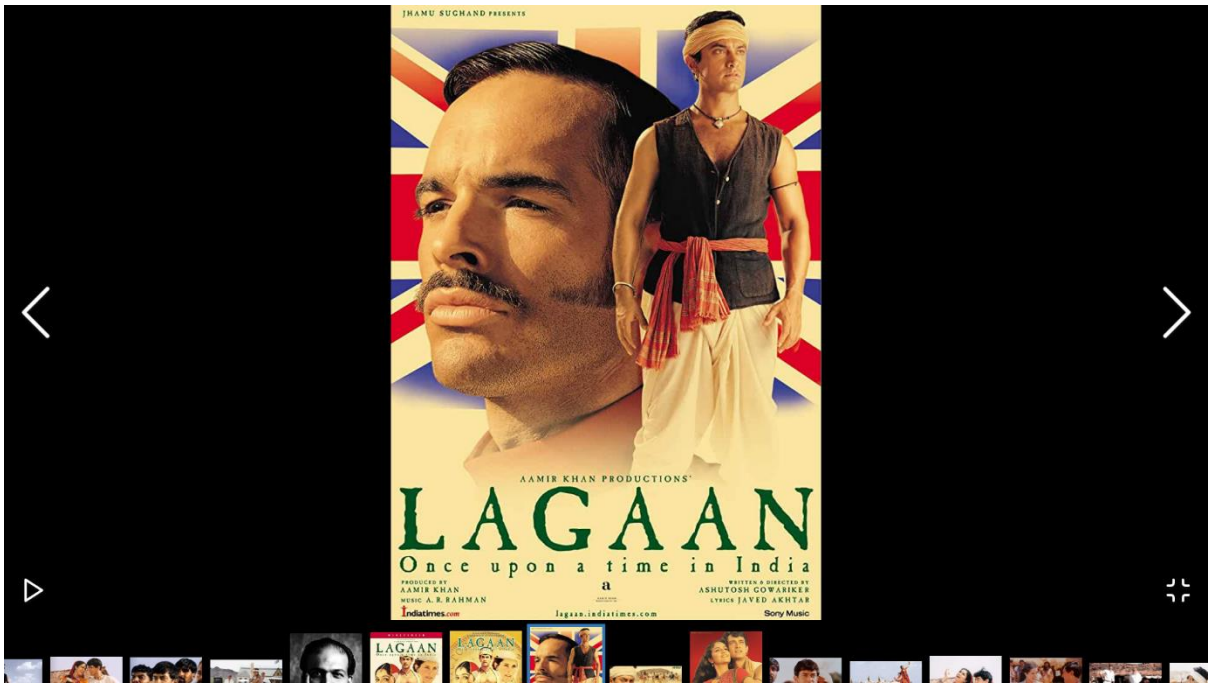




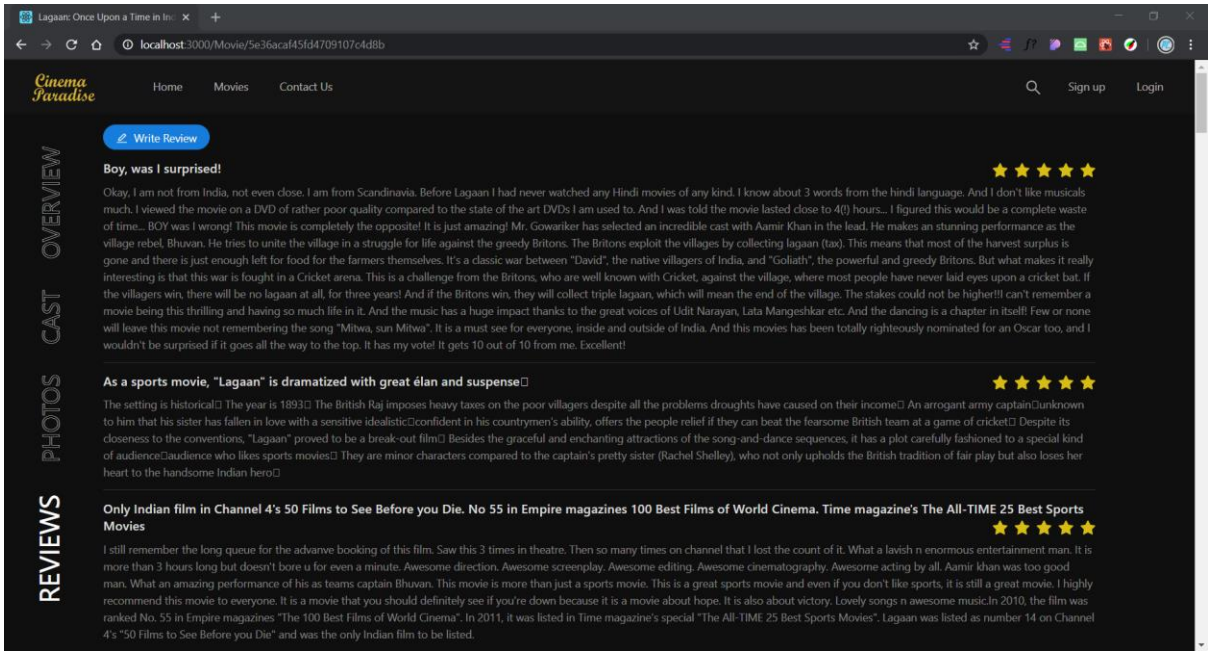
Movie details photos page which can be played as slideshow
1.without slideshow enable



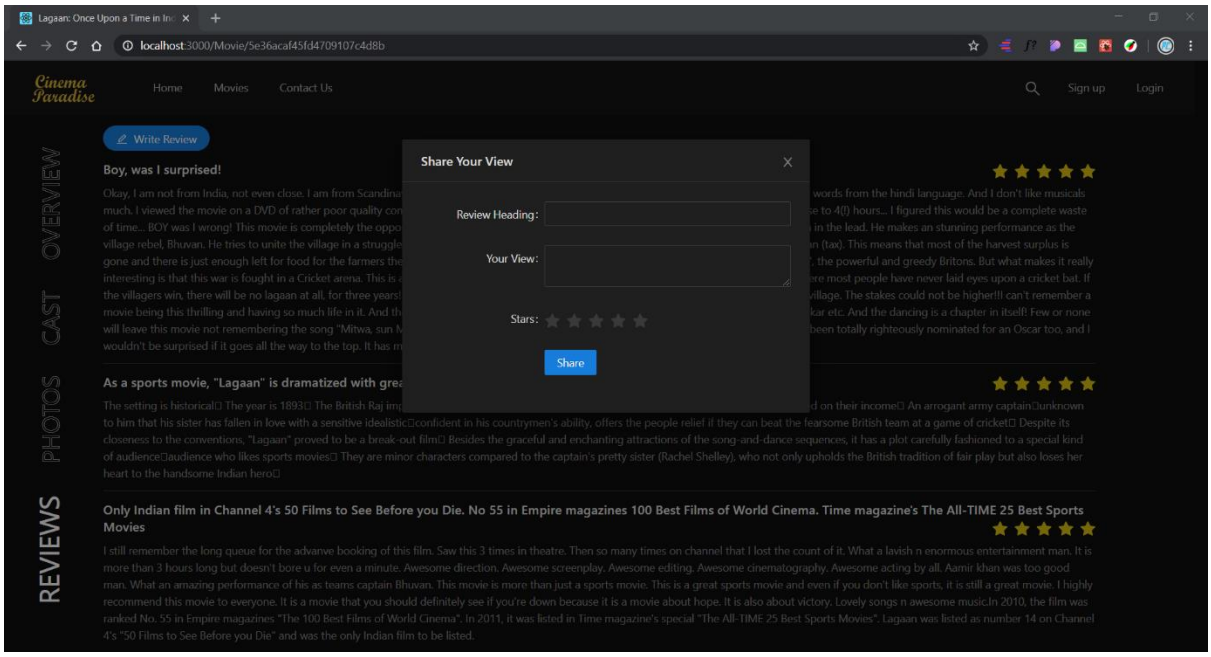
2.with slideshow enable



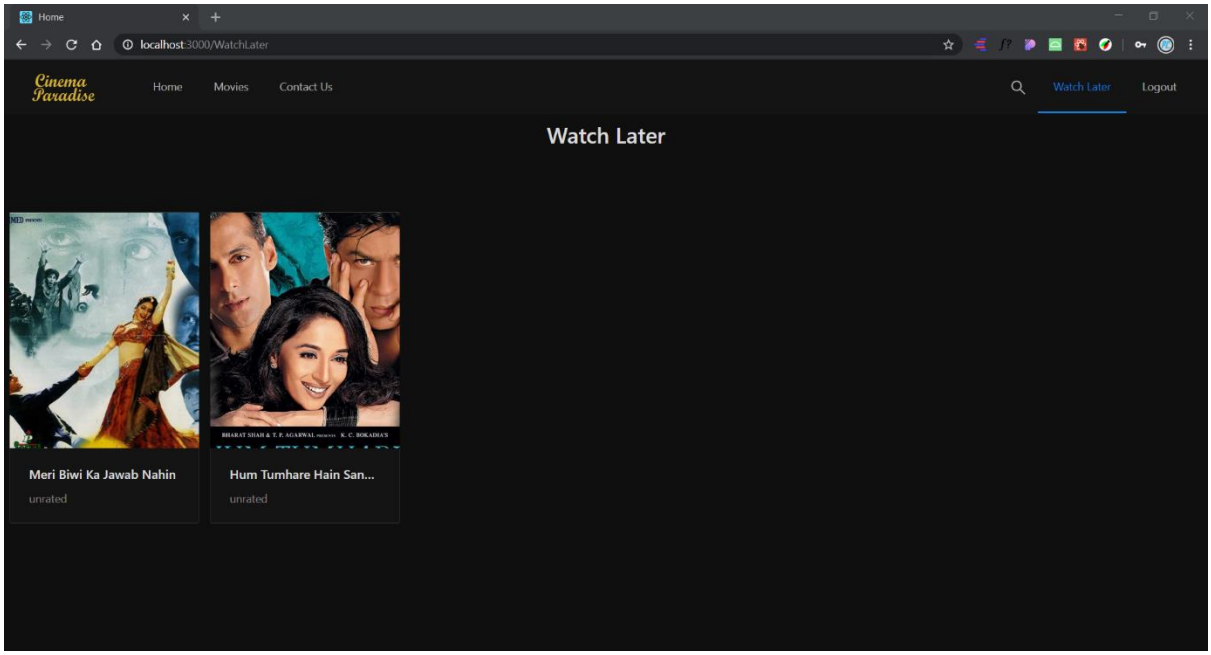
Movie details Reviews Page



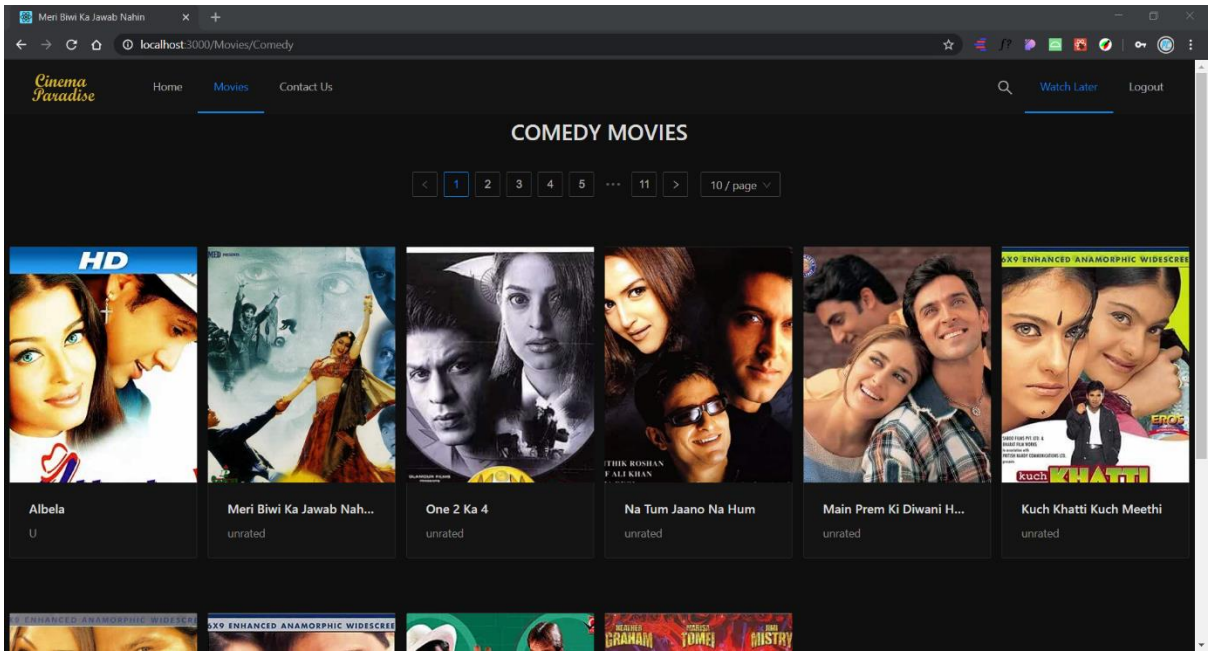
Movie details Reviews Page where you can post your review



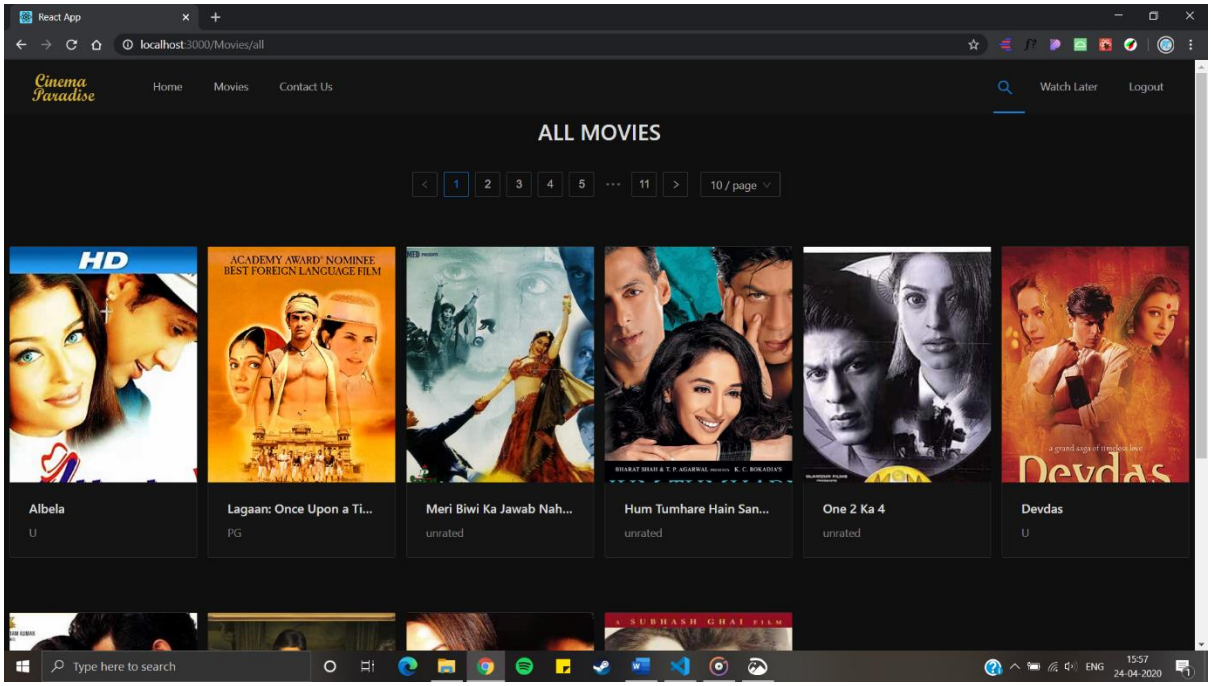
Watch Later List page when logged in

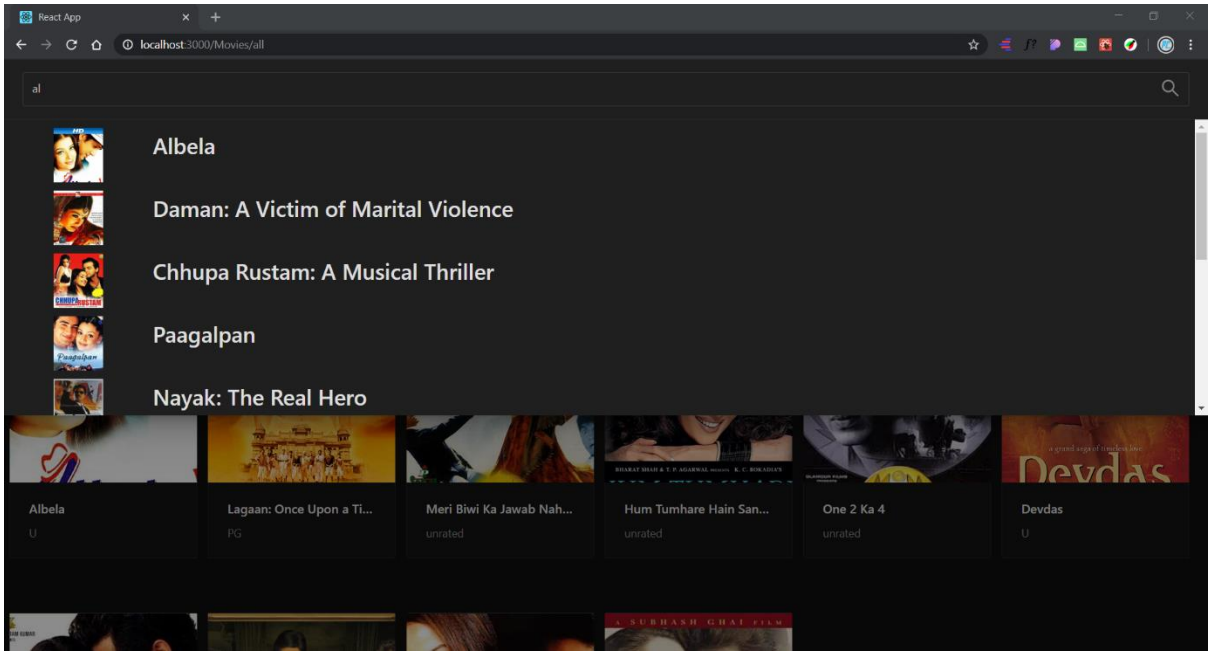


Movie Based on genre comedy



Searching Movie based on their name by clicking search icon on navigation





Conclusion

We successfully implemented movie review sentiment analysis website. While doing this project we learned so much about machine learning. We learned How to optimize our models and improve accuracy and also understood importance of data pre-processing. We learned how to integrate different technologies. We also learned most popular web framework react.

Limitation

- We created our own dataset by doing scraping and we labelled that scraped data using string matching algorithm which may label wrongly to very few document.

Future Extension

- Now we have users review data their likes and dislikes so in future we can add recommendation functionalities to our project.
- Deploying using Docker to spawn multiple nodes of server and achieve scalability.
-

Bibliography

- Machine Learning Course Stanford University, Coursera
  Instructor: Andrew ng
- https://www.researchgate.net/figure/Continuous-Bag-of-words-CBOW-CB-and-Skip-gram-SG-training-model-illustrations_fig1_326588219
- http://mathworld.wolfram.com/SigmoidFunction.html
- https://towardsdatascience.com/building-a-logistic-regression-in-python-301d27367c24
- https://www.youtube.com/watch?v=G1bof7UL9RU&list=PLwyIYCshlcyHtMTL8dvUoeM_CF2h0X2JC&index=4
- https://www.geeksforgeeks.org/confusion-matrix-machine-learning/