**ADBMS MINI-PROJECT**

Group members:

Simran Gambani – 2015140019

Disha Gandhi - 2015140020

Rajvi Gandhi - 2015140021

Hrithik Gavali – 2015140023

**Movie Recommender System**

**Aim**: To develop a recommender system for movies based on two recommender models:

1. Popularity based
2. Collaborative filtering

**Methodology used:**

**Popularity model** - No personalization, user history is not considered, shows what is trending and what everyone has liked.

Using classifier algorithm - parametric classification here based on genre and rating.

**Collaborative filtering** - based on past behaviour

For example if user has previously watched a movie of genre comedy with rating 3.5 it is highly possible he would like other comedy movies with rating 3.5+ too

Item-item collaborative filtering was used here (classification being made using rating and genre)

Movie lens dataset studied through Python using libraries like pandas and graphlab.

Data studied, and divided into test and train to build recommender models.

**Python Code for analysis:**

import pandas as pd

import graphlab

# pass in column names for each CSV and read them using pandas.

# Column names available in the readme file

#Reading users file:

u\_cols = ['user\_id', 'age', 'sex', 'occupation', 'zip\_code']

users = pd.read\_csv('ml-100k/u.user', sep='|', names=u\_cols,

encoding='latin-1')

#Reading ratings file:

r\_cols = ['user\_id', 'movie\_id', 'rating', 'unix\_timestamp']

ratings = pd.read\_csv('ml-100k/u.data', sep='\t', names=r\_cols,

encoding='latin-1')

#Reading items file:

i\_cols = ['movie id', 'movie title' ,'release date','video release date', 'IMDb URL', 'unknown', 'Action', 'Adventure',

'Animation', 'Children\'s', 'Comedy', 'Crime', 'Documentary', 'Drama', 'Fantasy',

'Film-Noir', 'Horror', 'Musical', 'Mystery', 'Romance', 'Sci-Fi', 'Thriller', 'War', 'Western']

items = pd.read\_csv('ml-100k/u.item', sep='|', names=i\_cols,

encoding='latin-1')

print("done")

print(users.shape)

print(users.head(10))

print(ratings.shape)

print(ratings.head(10))

print(items.shape)

print(items.head(10))

r\_cols = ['user\_id', 'movie\_id', 'rating', 'unix\_timestamp']

ratings\_base = pd.read\_csv('ml-100k/ua.base', sep='\t', names=r\_cols, encoding='latin-1')

ratings\_test = pd.read\_csv('ml-100k/ua.test', sep='\t', names=r\_cols, encoding='latin-1')

print(ratings\_base.shape)

print(ratings\_test.shape)

train\_data = graphlab.SFrame(ratings\_base)

test\_data = graphlab.SFrame(ratings\_test)

popularity\_model = graphlab.popularity\_recommender.create(train\_data, user\_id='user\_id', item\_id='movie\_id', target='rating')

#Get recommendations for first 5 users and print them

#users = range(1,6) specifies user ID of first 5 users

#k=5 specifies top 5 recommendations to be given

popularity\_recomm = popularity\_model.recommend(users=range(1,6),k=5)

popularity\_recomm.print\_rows(num\_rows=25)

ratings\_base.groupby(by='movie\_id')['rating'].mean().sort\_values(ascending=False).head(20)

#recommending using collaborative filtering

#Train Model

item\_sim\_model = graphlab.item\_similarity\_recommender.create(train\_data, user\_id='user\_id', item\_id='movie\_id', target='rating', similarity\_type='pearson')

#Make Recommendations:

item\_sim\_recomm = item\_sim\_model.recommend(users=range(1,6),k=5)

item\_sim\_recomm.print\_rows(num\_rows=25)

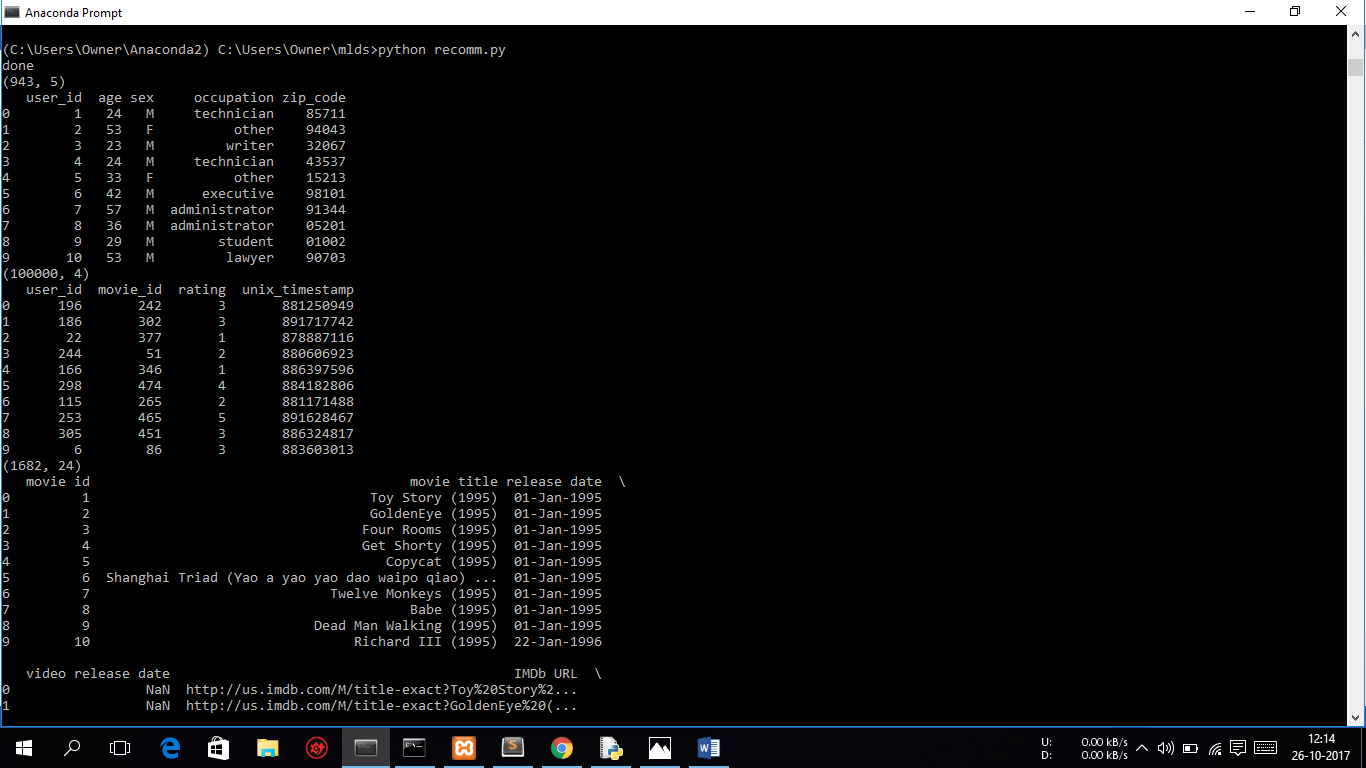
#precision-recall

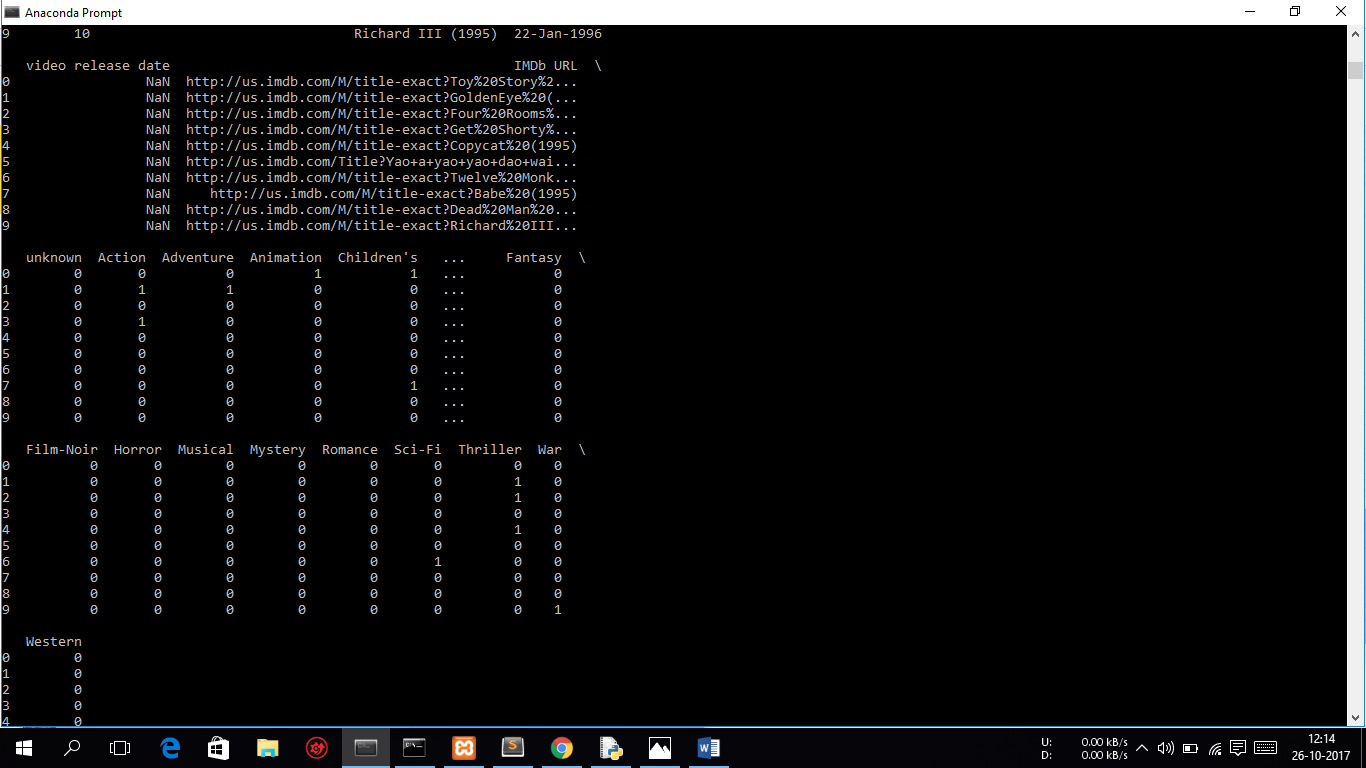
model\_performance = graphlab.compare(test\_data, [popularity\_model, item\_sim\_model])

graphlab.show\_comparison(model\_performance,[popularity\_model, item\_sim\_model])

**OUTPUT:**

Dimensions of the data are studied here

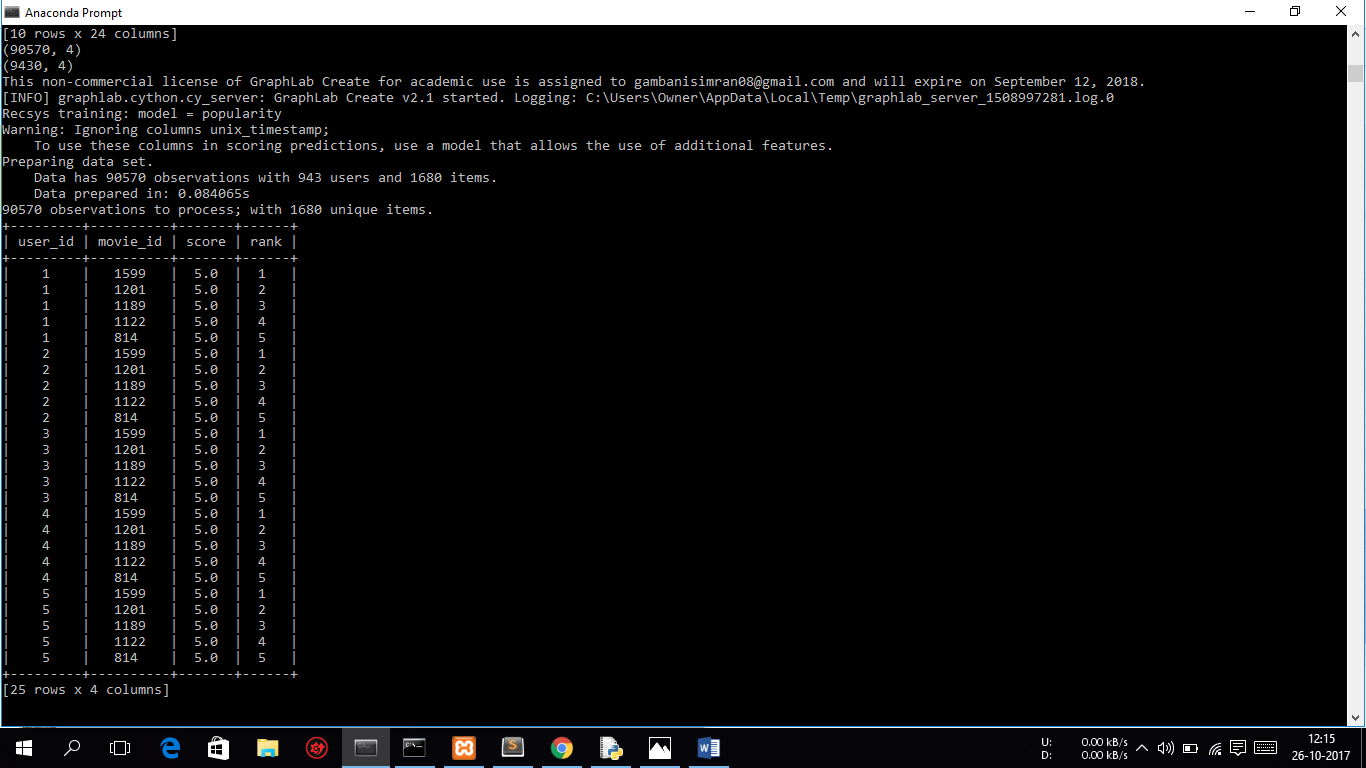


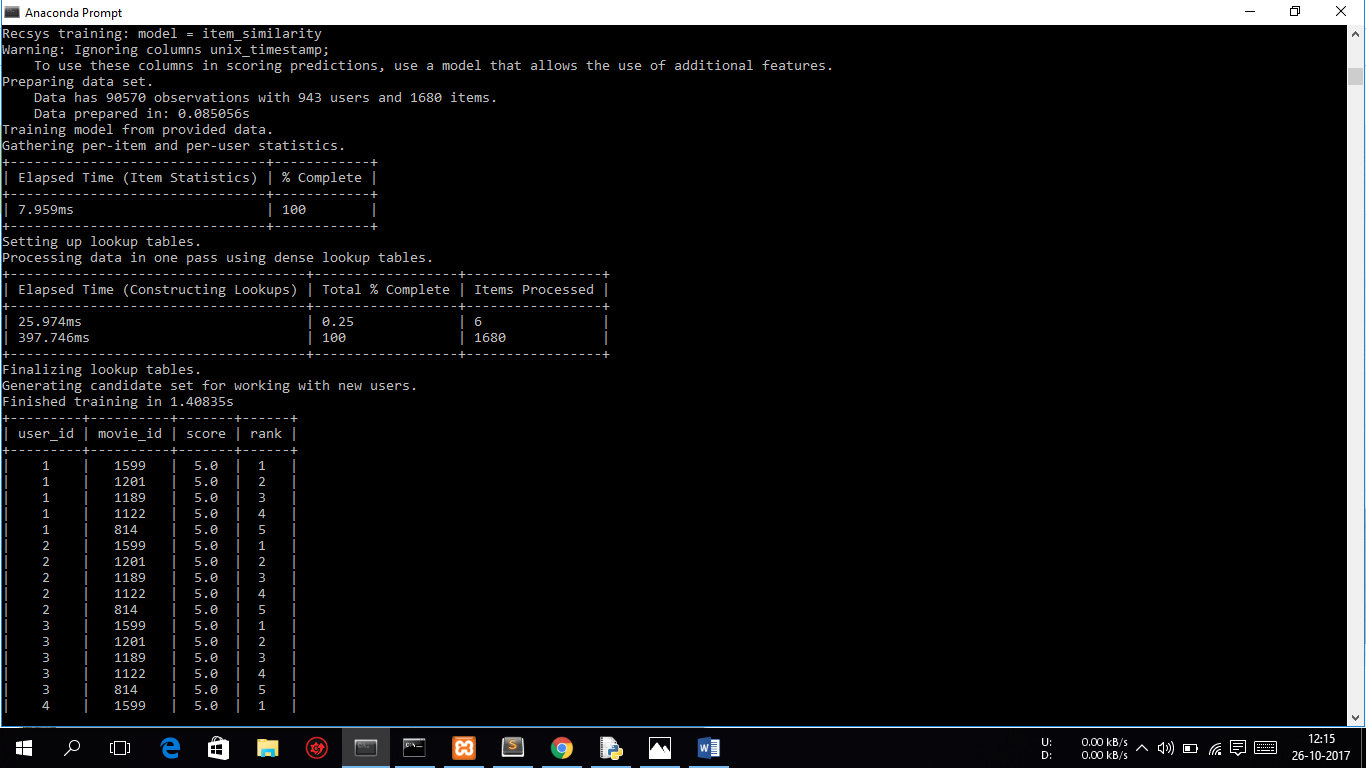


First model – Popularity based

Test data used to make top 5 recommendations for first 5 users

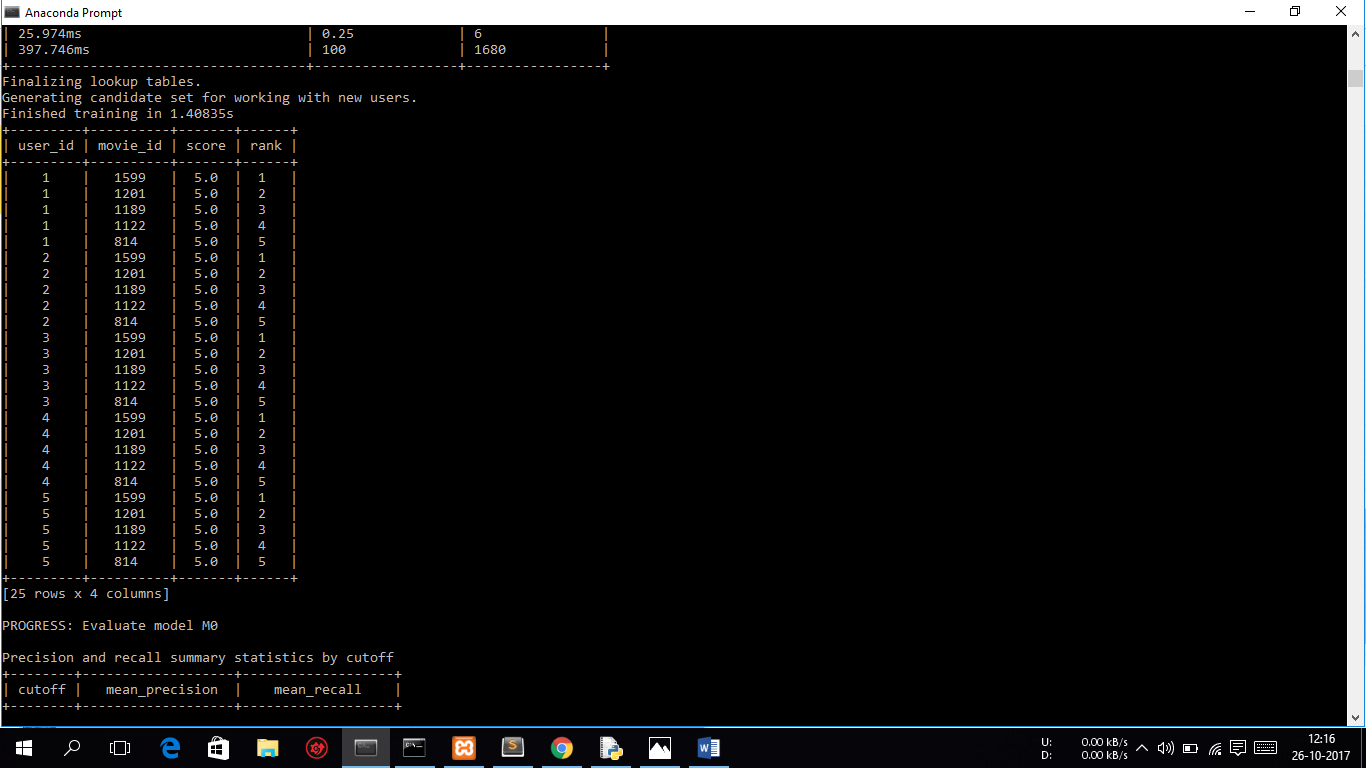
Observation - all are recommended same movies which have rating of 4.5/5





In item-item collaborative filtering model

Based on rating. Each user is recommended different movies based on his/her history.



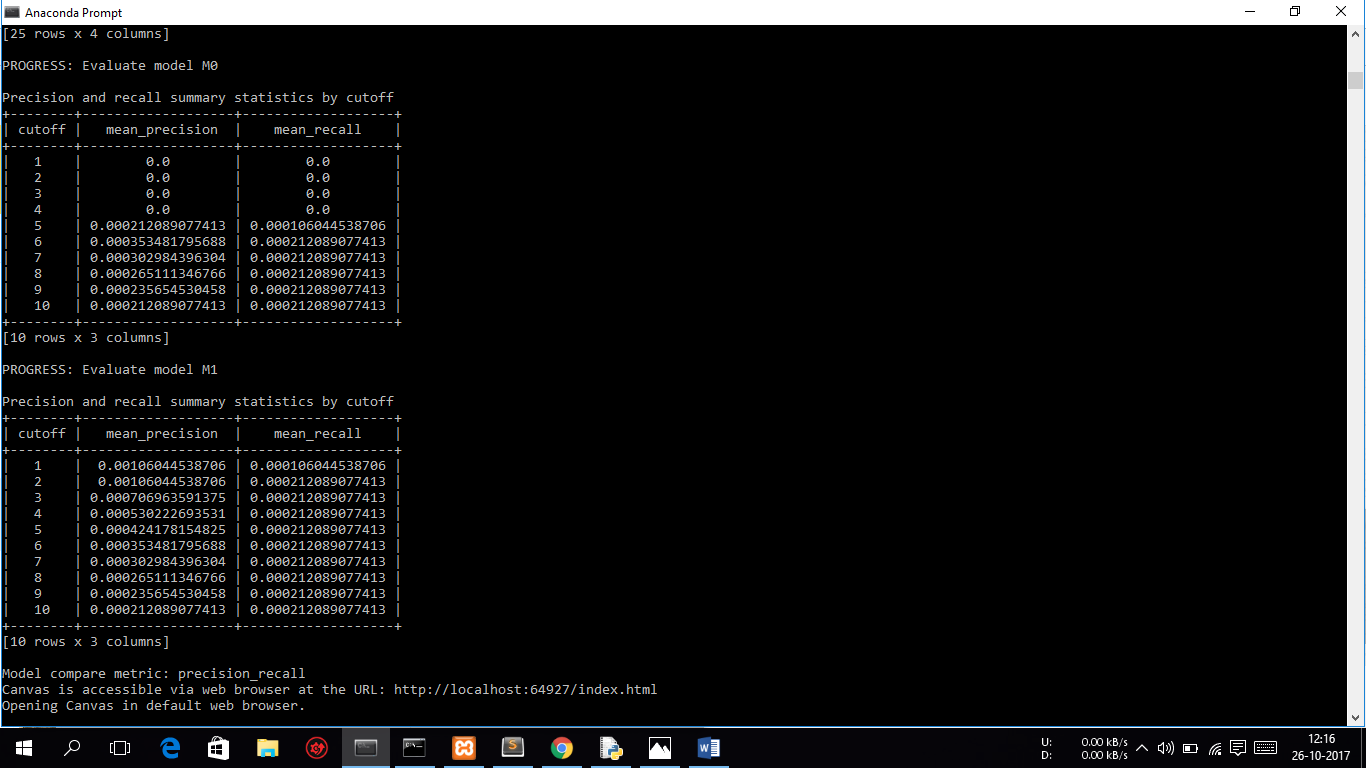
Comparison of both models

Precision - Out of recommended how many the user actually liked

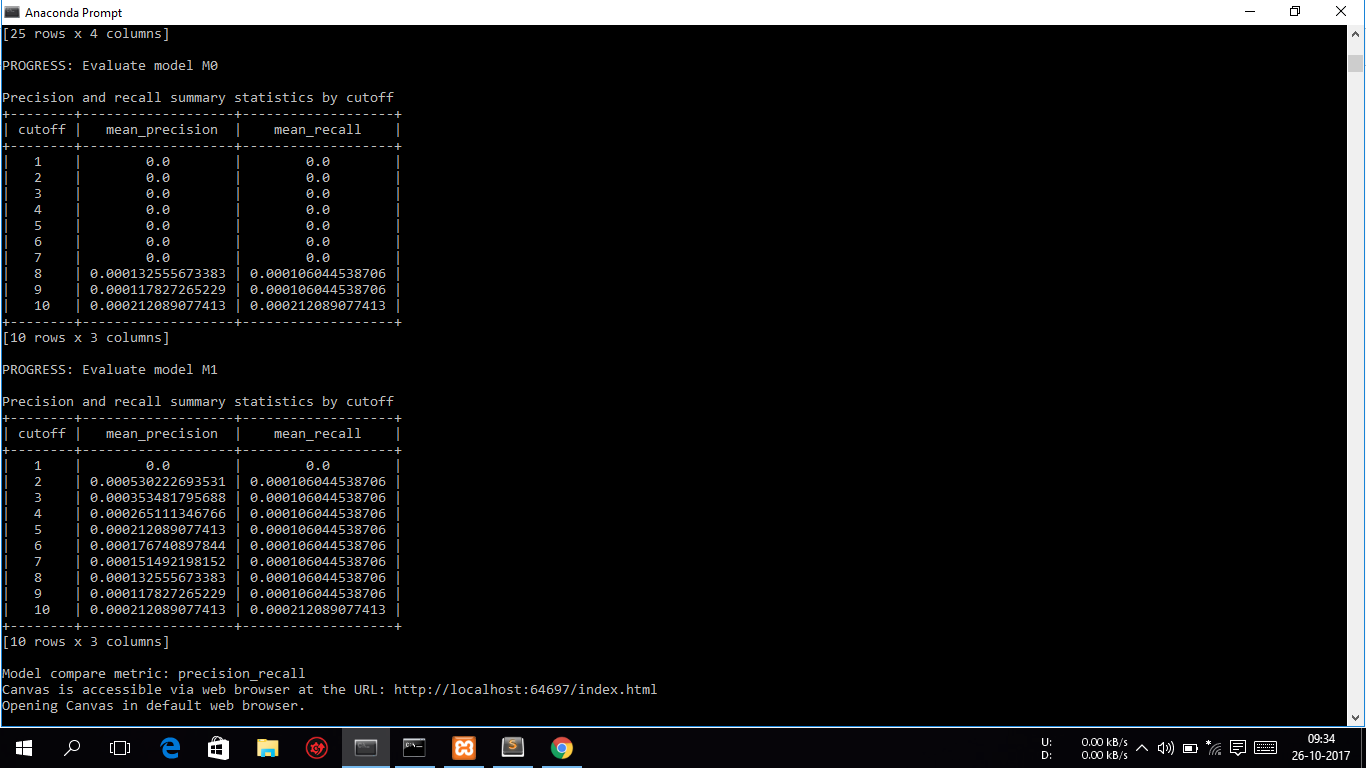
Recall - Out of the likes how many were actually recommended

Both the precision are compared

M0 is popularity which has much less precision than M1 which is collaborative filtering model

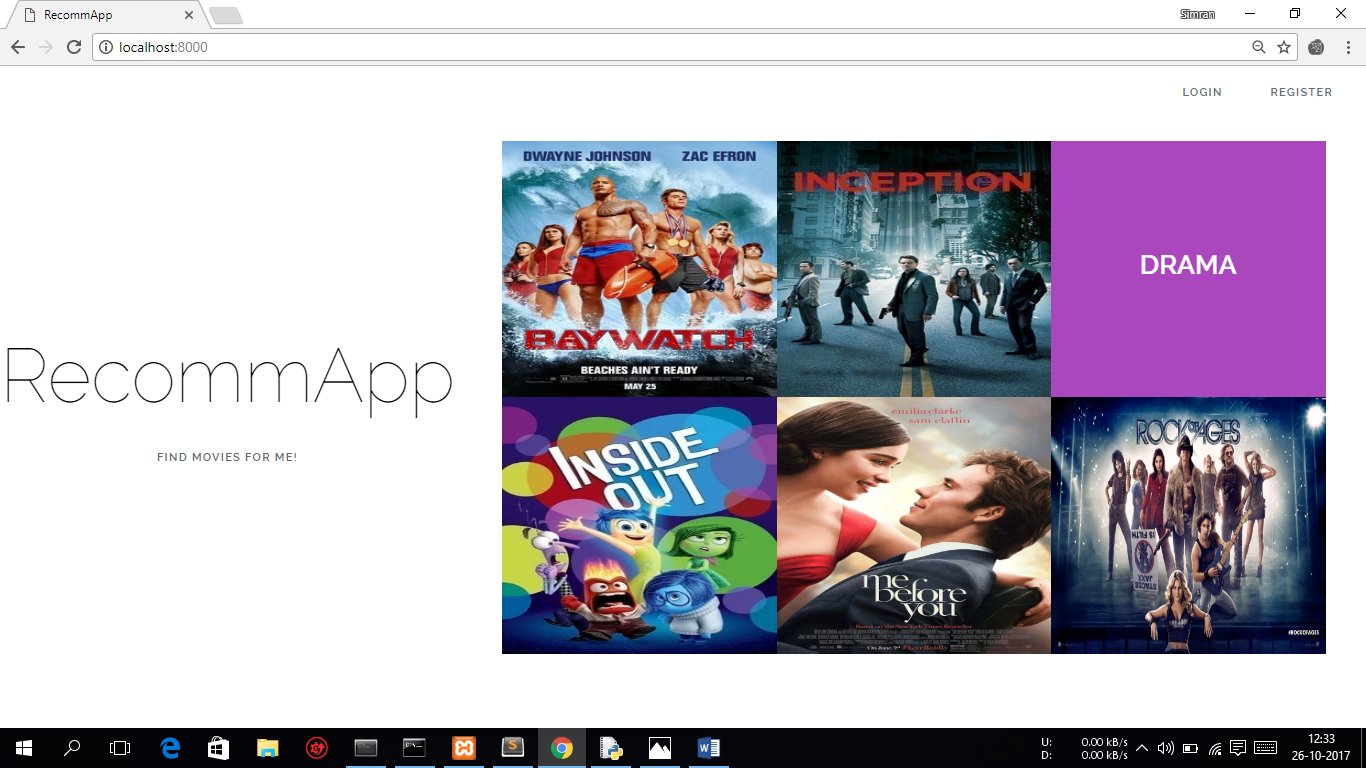


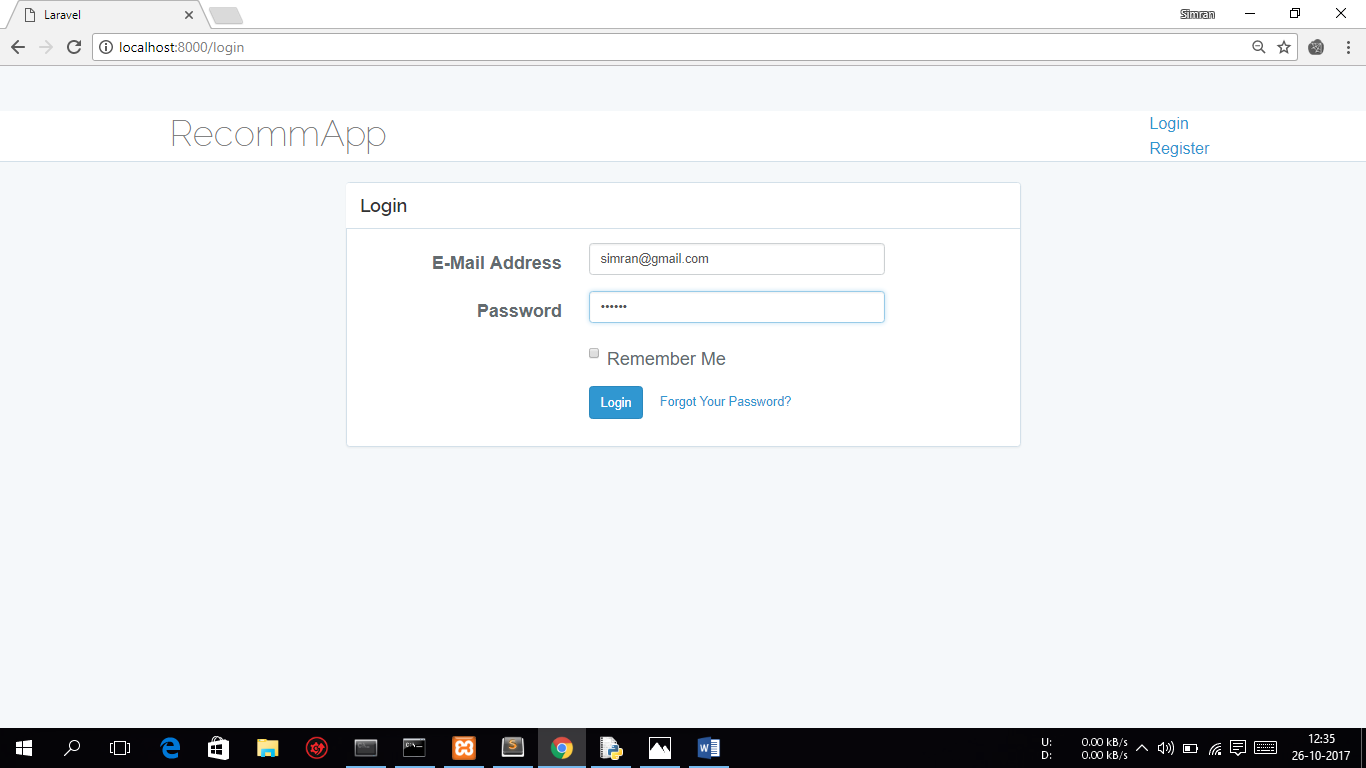
Another comparison

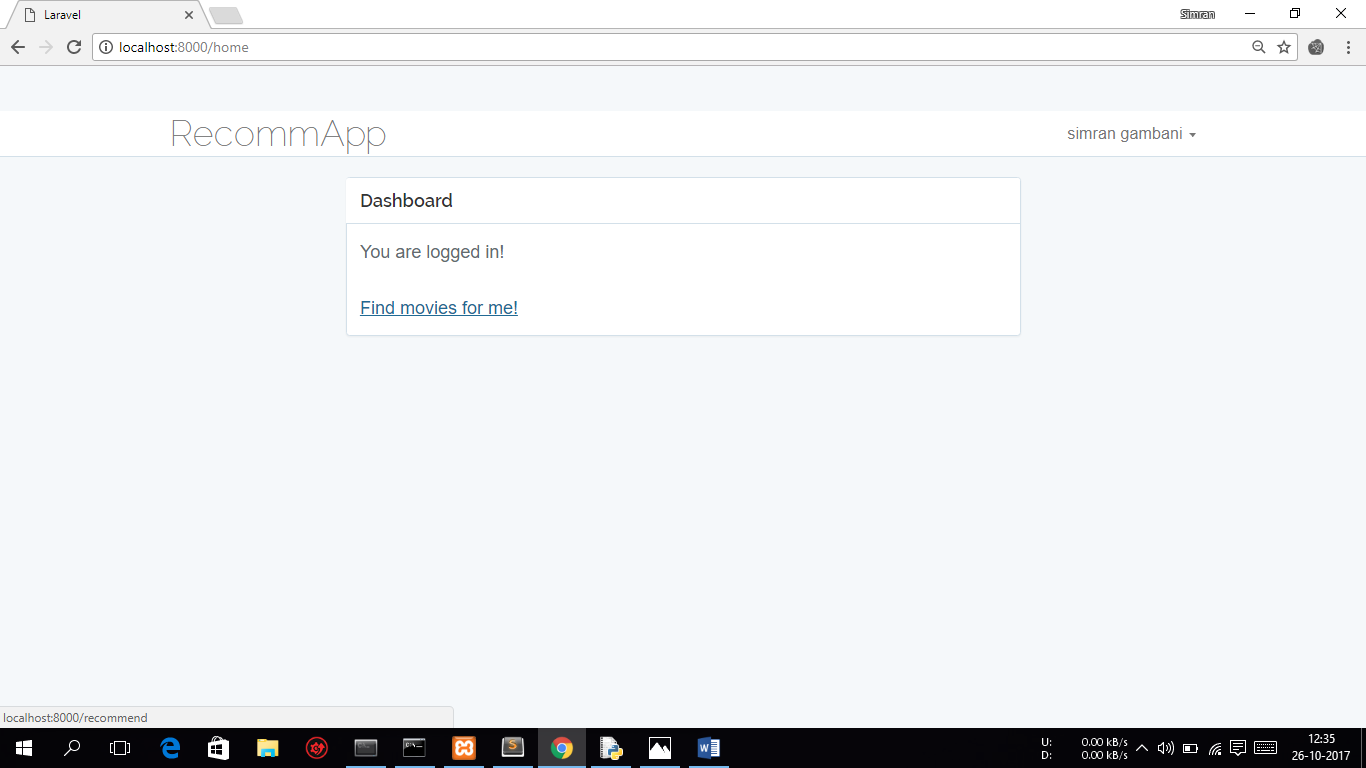


Comparing both models, collaborative filtering model has better performance.

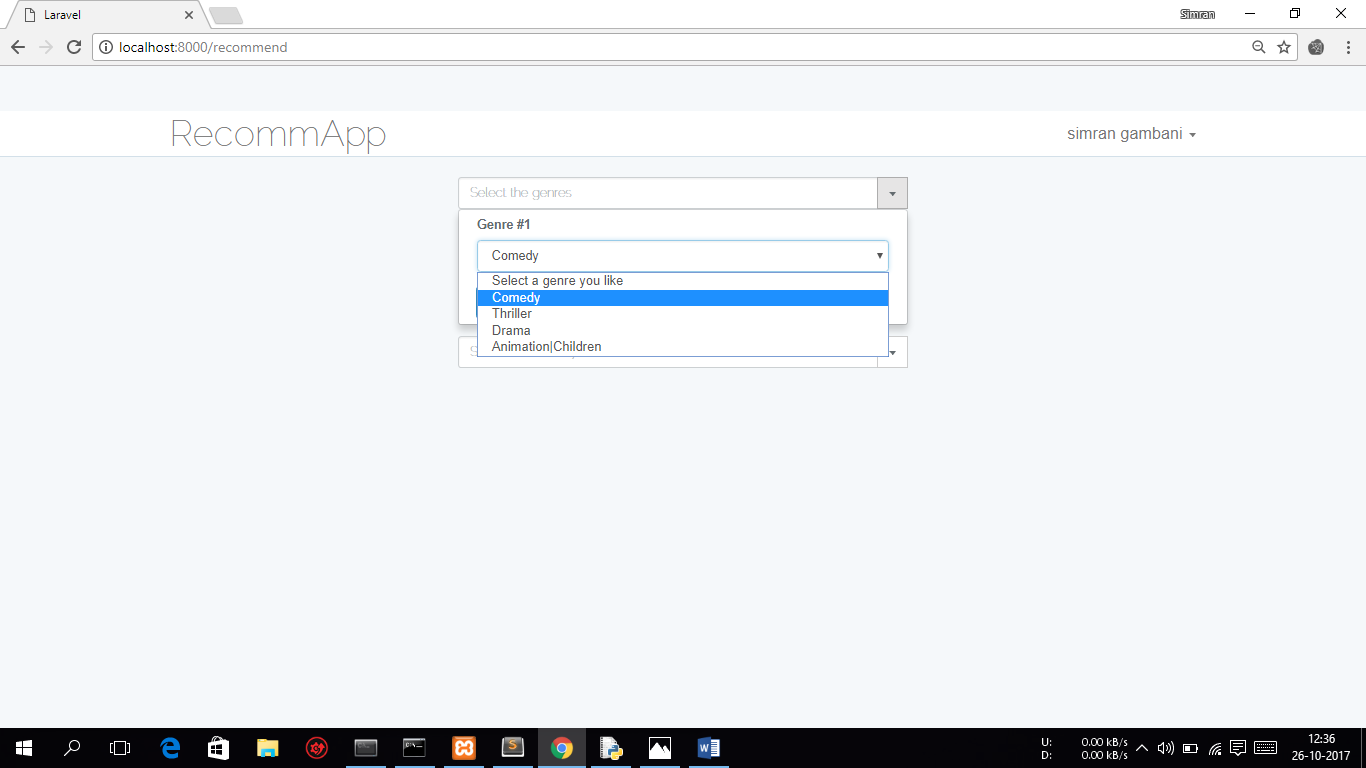
WEBSITE SCREENSHOTS

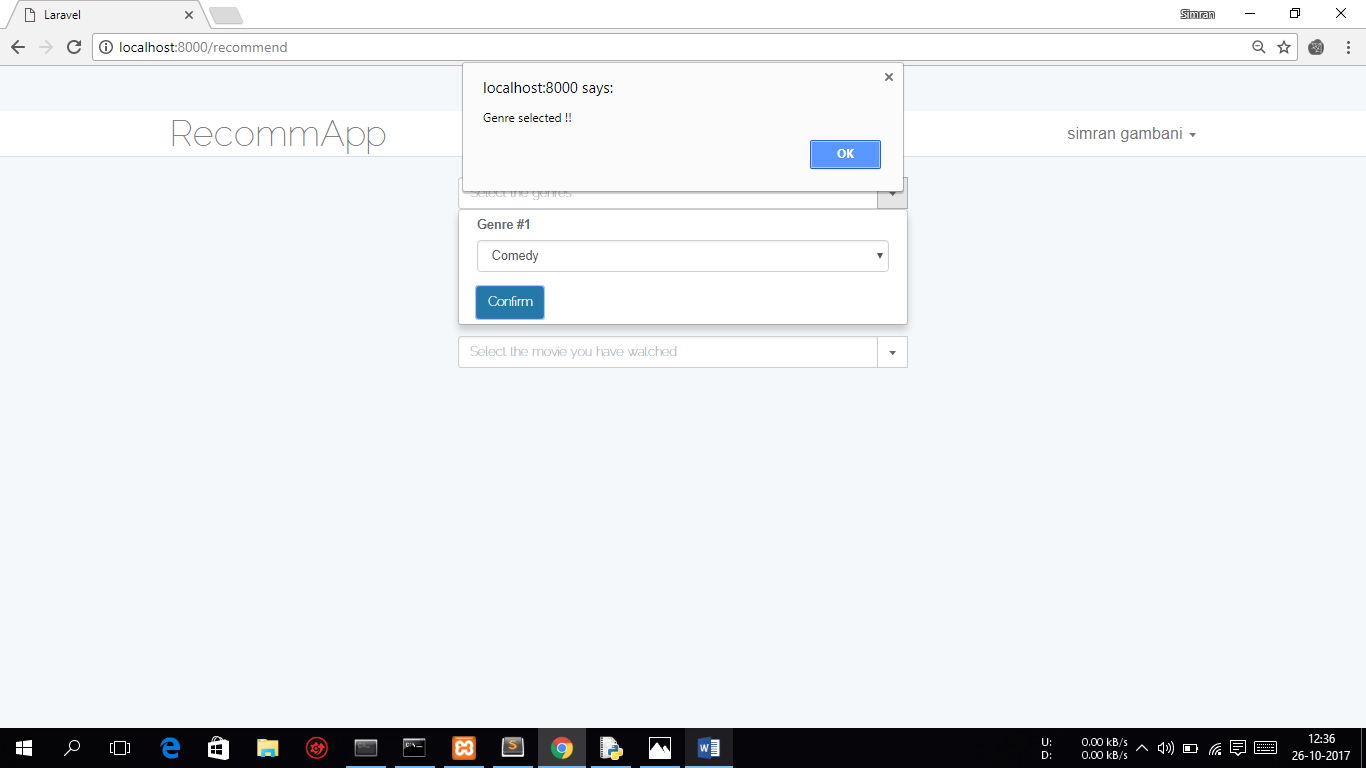




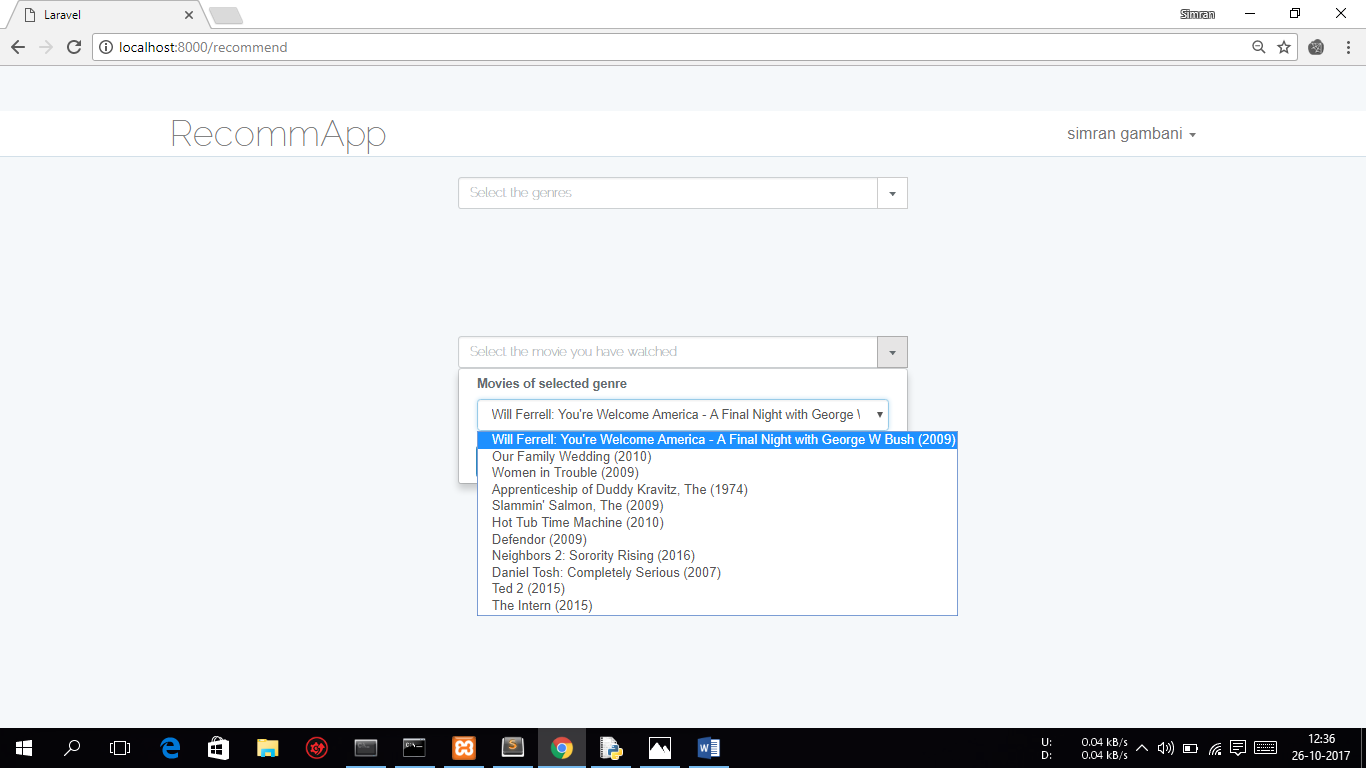


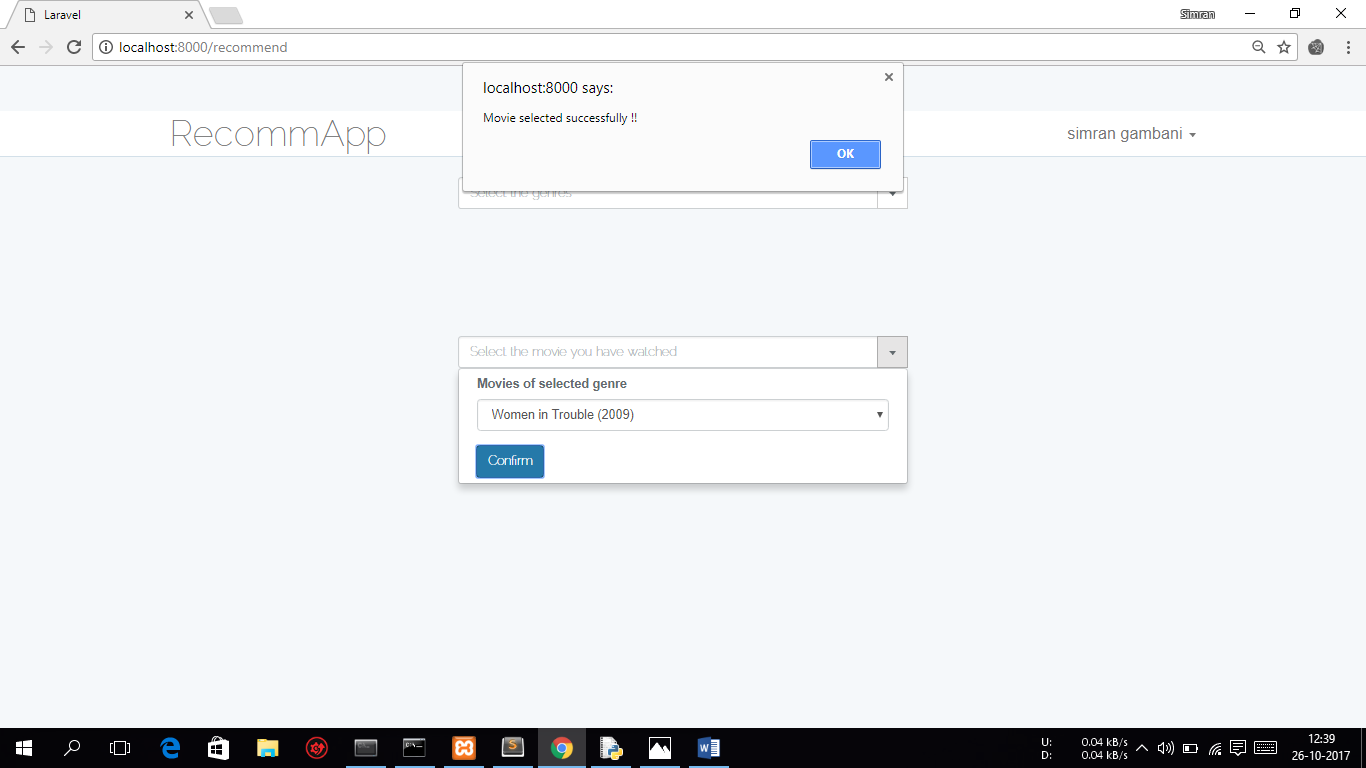
Genre selected



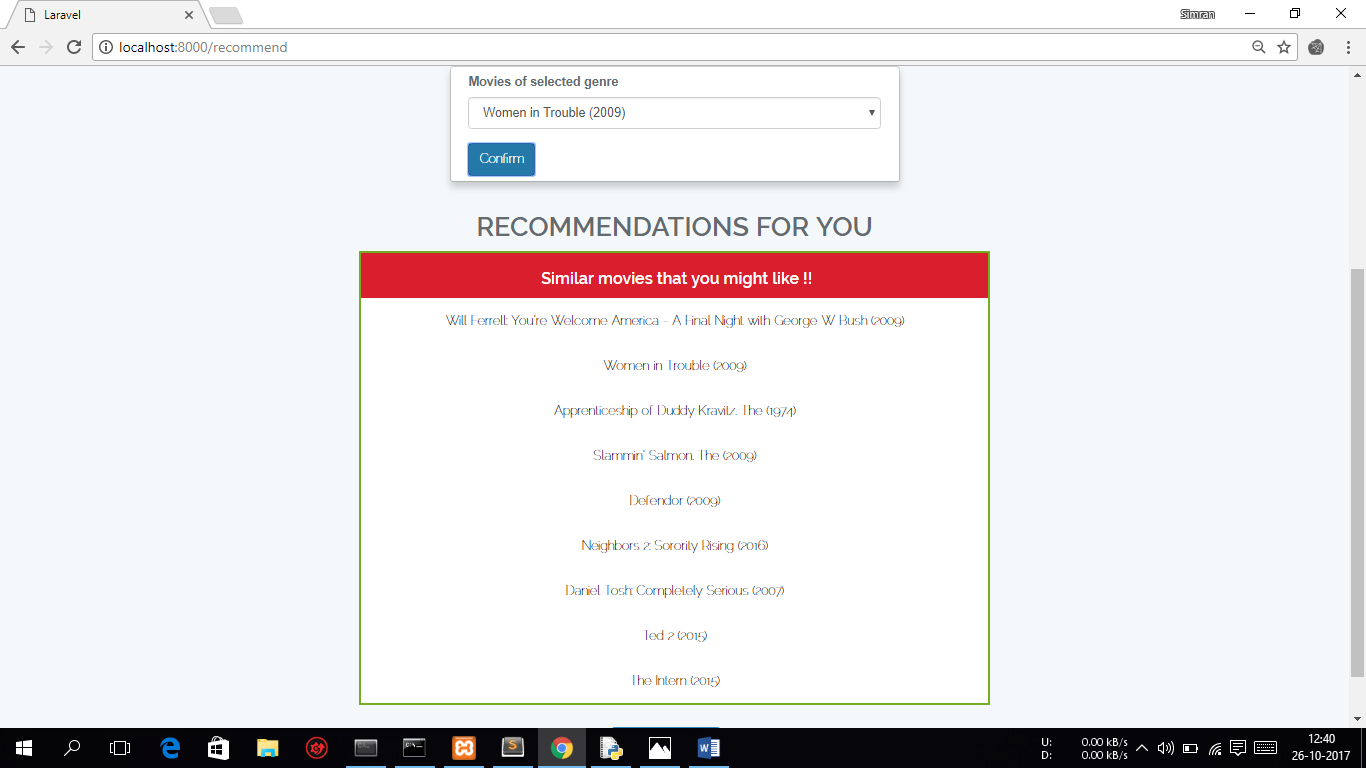


Only movies of selected genre are displayed





Recommendations of movies with same genre and having rating above the movie watched are displayed



Popular recommendations does not take user preferences into consideration. Only trending and movies with ratings>4 across all genres are displayed.

