MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY, BHOPAL

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAJOR PROJECT

(SESSION 2015 - 2016)

MOVIES AND TV SHOWS RECOMMENDATION BASED ON FACEBOOK PROFILE

PROJECT GUIDE:

Dr. MEENU CHAWLA

PROJECT TEAM:

SOURABH KULKARNI	121112002
ADITYA SRIVASTAVA	121112098
LEKHA BANSOD	121112052
MOHD. WARIS ANSARI	121112104

DECLARATION

We hereby declare that the work which is presented in the Major Project Report titled **MOVIES AND TV SHOWS RECOMMENDATION BASED ON FACEBOOK PROFILE** in partial fulfillment of the requirements of our final year Major Project during the degree of Bachelor of Technology in Computer Science & Engineering submitted in the Department of Computer Science and Engineering, MANIT Bhopal is an authentic record of our own work carried out from July, 2015 to April 2016 under the noble guidance of our project guide Dr. Meenu Chawla.

The work has been carried out in Maulana Azad National Institute of Technology, Bhopal and has not been submitted for any other degree or diploma. This is to certify that above declaration is correct to the best of our knowledge.

NAME	SCHOLAR NO.	SIGNATURE
SOURABH KULKARNI	121112002	
ADITYA SRIVASTAVA	121112098	
LEKHA BANSOD	121112052	
MOHD. WARIS ANSARI	121112104	

MAULANA AZAD NATIONAL INSTITUTE OF TECHNOLOGY,BHOPAL

(An Institute Of National Importance)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CERTIFICATE

This is to certify that Sourabh Kulkarni, Aditya Srivastava, Lekha Bansod and Mohd. Waris Ansari students of B.Tech 4th Year (COMPUTER SCIENCE & ENGINEERING), have successfully completed Movie And TV Show Recommendation Based On Facebook Profile in partial fulfillment of their major project in Computer Science & Engineering.

Dr. Meenu Chawla
(Project Guide)

Prof. Manish Pandey
(Project Co-ordinator)

ACKNOWLEDGEMENT

My deepest thanks to Dr. Meenu Chawla, the Guide of the project for guiding and correcting me at all stages of development of this project with attention and care. He has taken pain to go through the project and make necessary correction as and when needed.

I express my thanks to Dr. N.S. Chaudhari, Director of Maulana Azad National Institute of Technology and Dr. R.K. Pateriya, Associate Professor & Head of Department Computer Science & Engineering, for extending his support and providing us necessary facilities.

Besides our guides we would also like to thank Prof.Manish Pandey (our project coordinator) and our faculty members for providing us the much needed support and encouragement.

Lastly we would also like to express our deep appreciation towards our classmates without whom this project would have been a distant reality.

ABSTRACT

A recommender system aims to provide personalized recommendations to users for various items such as movies, music, books, online shopping etc. In this project we introduce a system for movie and TV show recommendation by using the data from a user's Facebook profile. We seek to use Facebook data to solve the cold-start problem in recommending movies and TV shows. We use a person's "likes" on Facebook to predict what TV shows and movies he/she may like. Basically, in a cold-start problem when a new user comes, the system does not have information about their preferences in order to make recommendations. So, this project provides an interesting solution to this problem. The moment when a new user logs into our application via Facebook, the application collects data from the user's Facebook profile which acts as the basis for the recommendation and the user gets personalized recommendations right from the beginning. We have worked on the widely known dataset provided by the IMDb.

This recommendation system uses three attributes of a movie which are actor, genre, rating. We assign appropriate weights to each of these attribute and calculate score for the movies. Top 10 scored movies are then recommended to the user.

CONTENT

1. INTRODUCTION

Consumers today in the world with the internet and its associated information explosion are faced with the problem of too much choice. Right from looking for a

restaurant to looking for good investment options, there is too much information available. To help the consumers cope with this information explosion, companies have deployed recommendation systems to guide the consumer. A number of

such online recommendation systems are implemented and are in use such as the recommendation system for books at Amazon.com and Libra, for movies at MovieLens.org, CDs at CDNow.com (from Amazon.com), etc.

Recommendation systems takes a huge data set of movies and tv shows and breaks these individual movies into long lists of attributes.The attributes used in this project our - Movie ID, Actors , Genres and ratings. Now recommendation is a simple matter of decoding an individual's tastes and matching those tastes to the relevant movies ex If a person enjoyed comedy and mystery movie, then the recommendation engine will find similar movies.

When a user first signs up for a Movie/TV recommendation service, the recommendation system has no information about what movies or TV shows the person has already watched and liked/disliked. This is called the cold-start situation. Despite not knowing anything about the user, the recommendation engine has to give good recommendations in order to retain the user' s

interest.This project provides a new solution to the cold-start problem.In this project, any user either a new user or a old user, can get recommendations just by logging into our app which uses of Facebook data such as 'Liked' movies, TV Shows and actors, and provides a customised recommendation to the user.

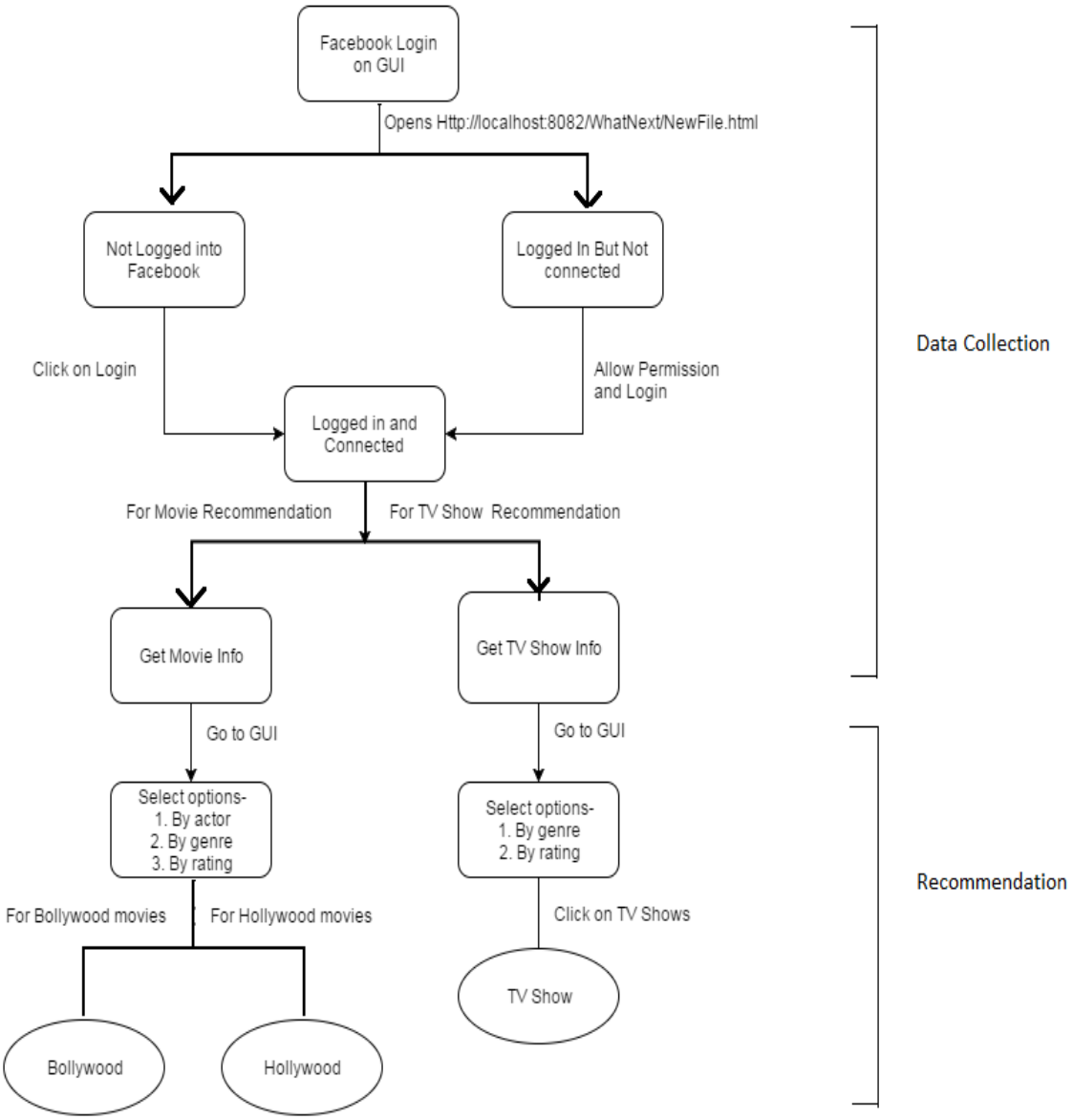
Advantage for user :

- Find things that are interesting
- Narrow down the set of choices
- Help me explore the space of options
- Discover new things
- Entertainment

Value for the provider:

- Additional and probably unique personalized service for the customer
- Increase trust and customer loyalty
- Increase sales, click trough rates, conversion etc.
- Opportunities for promotion, persuasion
- Obtain more knowledge about customers

1. FLOW CHART


Facebook Login on GUI
Opens Http://localhost:8082/WhatNext/NewFile.html
Not Logged into Facebook
Logged In But Not connected
Click on Login
Allow Permission and Login
Logged in and Connected
For Movie Recommendation
For TV Show Recommendation
Get Movie Info
Get TV Show Info
Go to GUI
Go to GUI
Select options-
1. By actor
2. By genre
3. By rating
Select options-
1. By genre
2. By rating
For Bollywood movies
For Hollywood movies
Click on TV Shows
TV Show
Bollywood
Hollywood
Data Collection
Recommendation

3. DATA COLLECTION

The basic module of our work is data collection. Data collection means extracting the required data from which an accurate recommendation system can be generated. Backbone of the recommendation system is the list of movies or tv shows a user likes on his/her facebook profile.

To grab data from facebook we use facebook graph api integrated into facebook application named "WHATNEXT!!!".

User visits the page http://localhost:8082/WhatNext/NewFile.html and login using his/her facebook account and then provides access permission to our application. This step gives us the permission to extract the user likes list from the profile. This like list is generated using the facebook graph api and then is saved as .csv file on the server side.

Apart from the above data collected we have our own defined database to support the recommendation system :

- *Movie_Database.csv* : it contains over 1000 Bollywood movies along with an id associated with each movie.
- *Actor_Database.csv* : it contains over 300 bollywood actors along with an id associated with each actor.
- *Genre_Database.csv* : it contains over 20 genres like action, drama, comedy, etc along with an id associated with each genre.
- *Result_Database.csv* : it contains data in format :-

[movie_id : actor_id : genre_id : rating]

3.1. Facebook SDK for Javascript

Facebook SDK for javasrcipt enables us to use facebook login to lower the barrier for people to sign up on to our site. It also makes it easy to call into Facebook′ s Graph API.

The Facebook SDK for JavaScript doesn't have any standalone files that need to be downloaded or installed, instead you simply need to include a short piece of regular JavaScript in your HTML that will asynchronously load the SDK into your pages. The async load means that it does not block loading other elements of your page.

3.2. Facebook Graph API

The Graph API is the primary way to get data in and out of Facebook's platform. It's a low-level HTTP-based API that you can use to query data, post new stories, manage ads, upload photos and a variety of other tasks that an app might need to do.

The Graph API is named after the idea of a 'social graph' - a representation of the information on Facebook composed of:

- **nodes** - basically "things" such as a User, a Photo, a Page, a Comment
- **edges** - the connections between those "things", such as a Page's Photos, or a Photo's Comments

- **fields** - info about those "things", such as a person's birthday, or the name of a Page.

  Most Graph API requests require the use of access tokens which our app can generate by implementing Facebook Login explained above.

  All nodes and edges in the Graph API can be read simply with an HTTP GET request to the relevant endpoint. For example, if you wanted to retrieve information about the current user, you would make an HTTP GET request as below:

```
GET /v2.5/me HTTP/1.1

Host: graph.facebook.com

GET graph.facebook.com

  /{node-id}
```

  Most API calls must be signed with an access token. You can determine which permissions are needed in this access token by looking at the Graph API reference for the node or edge that you wish to read. Access token is generated from the Facebook app created.

  Thus through the above two component we get the list of movies liked by the user. Now we save this list as *.csv* file on server side using websocket. Webpage is hosted on the local host Apache Tomcat Server.

3.3. Websocket

WebSocket is a protocol which allows for communication between the client and the server/endpoint using a single TCP connection. Advantage WebSocket has over HTTP is that the protocol is full-duplex (allows for simultaneous two-way communication) and it′ s header is much smaller than that of a HTTP header, allowing for more efficient communication even over small packets of data.

The life cycle of a WebSocket is :

Client sends the Server a handshake request in the form of a HTTP upgrade header with data about the WebSocket it′ s attempting to connect to.

The Server responds to the request with another HTTP header, this is the last time a HTTP header gets used in the WebSocket connection. If the handshake was successful, the server sends a HTTP header telling the client it′ s switching to the WebSocket protocol.

Now a constant connection is opened and the client and server can send any number of messages to each other until the connection is closed. These messages only have about 2 bytes of overhead.

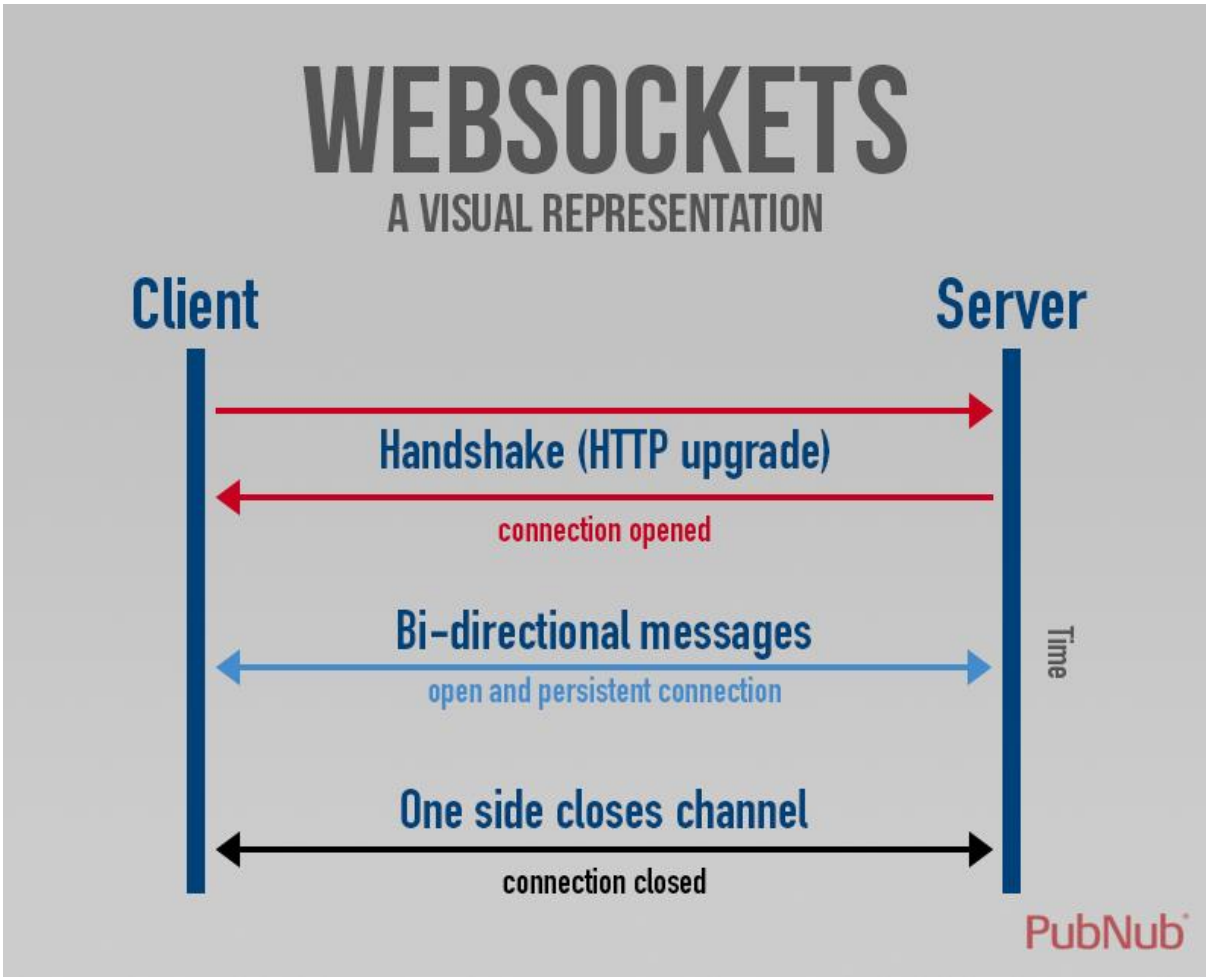


Fig1: Diagram showing working of a websocket.

In our work the movie name generated by Facebook SDK JavaScript is sent to server side one by one, which is waiting with a port open and then writes it into a *.csv* file. This *.csv* file acts as input to the recommender system.

3.4. Apache Tomcat Server

Apache Tomcat, often referred to as Tomcat, is an open-source web server developed by the Apache Software Foundation (ASF). Tomcat implements several Java EE specifications including Java Servlet, Java Server Pages(JSP), Java EL, and WebSocket, and provides a "pure Java" HTTP web server environment for Java code to run in.

In our project it is used to host the webpage locally through which user signs up to the Facebook account.

Web page generated has address: http://localhost:8082/WhatNext/NewFile.html.

3.5 Given below is the code used to get Data from user Facebook:

FrontEnd : This code generates the webpage which extracts the list of liked movies.

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<style>
.div1 {
    width: 300px;
    height: 50px;
    border: 1px solid blue;
background-color:grey;
margin-top:50px;

}
#header {
    background-color:black;
    color:white;
    text-align:center;
    padding:5px;
}
#status {
background-color:grey;
color:white;
text-align:center;
padding:3px;



}
#login {
align:center;
}
#footer {
background-color:black;
    color:white;
    clear:both;
    text-align:center;
    padding:5px;
    margin-top:480px;
}
</style>
<body background="bg.jpg">
<div id="header">
<h1>WhatNext !!!</h1>
</div>
```

```
<script>
var ws= new WebSocket("ws://127.0.0.1:8080");
  window.fbAsyncInit = function() {
    FB.init({
      appId      : '127256997673241',
      xfbml      : true,
      version    : 'v2.5'
    });
    FB.getLoginStatus(function(response){
     if (response.status === 'connected'){
   document.getElementById('status').innerHTML = 'We are connected';
      }

elseif(response.status==='not_authorized'){document.getElementById('
status').innerHTML = 'We are not connected';
      }
      else{
            document.getElementById('status').innerHTML  =  'you  are
not logged into facebook';
      }
    });
  };

  (function(d, s, id){
     var js, fjs = d.getElementsByTagName(s)[0];
     if (d.getElementById(id)) {return;}
     js = d.createElement(s); js.id = id;
     js.src = "//connect.facebook.net/en_US/sdk.js";
     fjs.parentNode.insertBefore(js, fjs);
   }(document, 'script', 'facebook-jssdk'));
  function login() {
       FB.login(function(response){
            if (response.status === 'connected'){
               document.getElementById('status').innerHTML   =   'We
are connected';
           }
           else if(response.status === 'not_authorized'){
                document.getElementById('status').innerHTML   =   'We
are not connected';
           }
           else{
                document.getElementById('status').innerHTML  =  'You
are not logged into facebook';
           }
       },{scope:'user_likes'});
  }
  function getmovieinfo() {
//
FB.api('/me','GET',{fields:'first_name,last_name,name,id,likes'},fun
ction(response){
//
document.getElementById('status').innerHTML=console.log(response);
//      });
      FB.api('/me/movies',function(response) {
```

```
            if(response && !response.error){
      document.getElementById('status').innerHTML=response.paging;
                  for(var i=0;i<=25;i++)
                        {
                  console.log(JSON.stringify(response.data[i].name));
                  ws.send(response.data[i].name);
                        }
      }
      });

  }
  function gettvinfo() {
//
FB.api('/me','GET',{fields:'first_name,last_name,name,id,likes'},fun
ction(response){
//
document.getElementById('status').innerHTML=console.log(response);
//      });
      FB.api('/me/television',function(response) {
            if(response && !response.error){
//
      document.getElementById('status').innerHTML=response.paging;
                  for(var i=0;i<=25;i++)
                        {
                  console.log(JSON.stringify(response.data[i].name));
                  //ws.send(response.data[i].name);
                        }
                  }
      });
}

</script>
<center><div class=div1>
<div id="status" style="color:yellow"></div>
<button     id="getmovieinfo"     onclick=getmovieinfo()>Get     movie
info</button>
<button id="gettvinfo" onclick=gettvinfo()>Get TV info</button>
<button id="login" onclick=login()>Login</button>
</div></center>
<div id="footer">
Copyright © Aditya Sourabh Lekha Waris
</div>
</body>
</html>
```

BackEnd : This code sends the above generated movie list onto the server side to be stored in a csv file using websocket.

Websocket.java

```
import org.eclipse.jetty.server.Server;
import org.eclipse.jetty.websocket.server.WebSocketHandler;
import org.eclipse.jetty.websocket.servlet.WebSocketServletFactory;


public class Websocket {
     public static void main(String[] args) throws Exception {

       Server server = new Server(8080);
       WebSocketHandler wsHandler = new WebSocketHandler() {
           @Override
           public void configure(WebSocketServletFactory factory) {
          factory.register(CopyOfWebsocket1.class);
        //  factory.register(CopyOfWebsocket2.class);
           }
       };
       server.setHandler(wsHandler);
      server.start();
       server.join();
   }
}
```

CopyOfWebsocket1.class

```
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStream;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.URL;
import java.net.URLConnection;
import java.net.URLEncoder;

import org.eclipse.jetty.websocket.api.Session;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketClose;
import
org.eclipse.jetty.websocket.api.annotations.OnWebSocketConnect;
import org.eclipse.jetty.websocket.api.annotations.OnWebSocketError;
import
org.eclipse.jetty.websocket.api.annotations.OnWebSocketMessage;
import org.eclipse.jetty.websocket.api.annotations.WebSocket;

@WebSocket
```

```
public class CopyOfWebsocket1 {
      public static int j=0;
      public static InetAddress ip=null;
    File file = new
File("C:\\Users\\Aditya\\Documents\\NetBeansProjects\\Demo_code\\lik
e1.csv");
      @OnWebSocketClose
    public void onClose(int statusCode, String reason) {
      //check=1;
         System.out.println("Close: statusCode=" + statusCode + ",
reason=" + reason);
         //i=0;
         j=0;
    }
      @OnWebSocketError
    public void onError(Throwable t) {
        System.out.println("Error: " + t.getMessage());
    }
       @OnWebSocketMessage
          public void onText(Session session,String message) throws
IOException {
           System.out.println(message);

             InetAddress
ip1=session.getRemoteAddress().getAddress();
            String h=ip1.toString();
            System.out.println(h);
            BufferedWriter out = new BufferedWriter(new FileWriter

("C:\\Users\\Aditya\\Documents\\NetBeansProjects\\Demo_code\\like1.c
sv",true));
                     out.write(message+"\n");
                     out.close();


          }
       @OnWebSocketConnect
          public void onConnect(Session session) throws IOException
{

              System.out.println("Connect: " +
session.getRemoteAddress().getAddress());
               ip=session.getRemoteAddress().getAddress();


               String h=ip.toString();
}

}
```

4. RECOMMENDATION SYSYTEM

The second part of the project is the Recommendation System. This is the module which takes in the input file containing the 'Liked' movies of the user and processes it and produces recommendations based on user's interests. In this module, for each movie a score is calculated based on the attributes present in the movies liked by the user. This part involves various steps:

Getting the list of 'Liked' movies and TV shows from the Data Collection part and assigning them proper ID's by matching them against movies present in the Movie Database.

After ID's have been assigned, a 'actor_count' is calculated based on the 'Actors' present in the user 'Liked' movies and a 'genre_count' is also calculated based on the 'Genre' of the 'Liked' movies.

Next step is to calculate the score of each movie present in our database based on the *actor_count* and *genre_count* ,along with taking into account the choice of user i.e. how the user wants the recommendation to be - by actor or by genre or by ratings.

After the scores have been calculated, the movies having the highest score are recommended to the user.

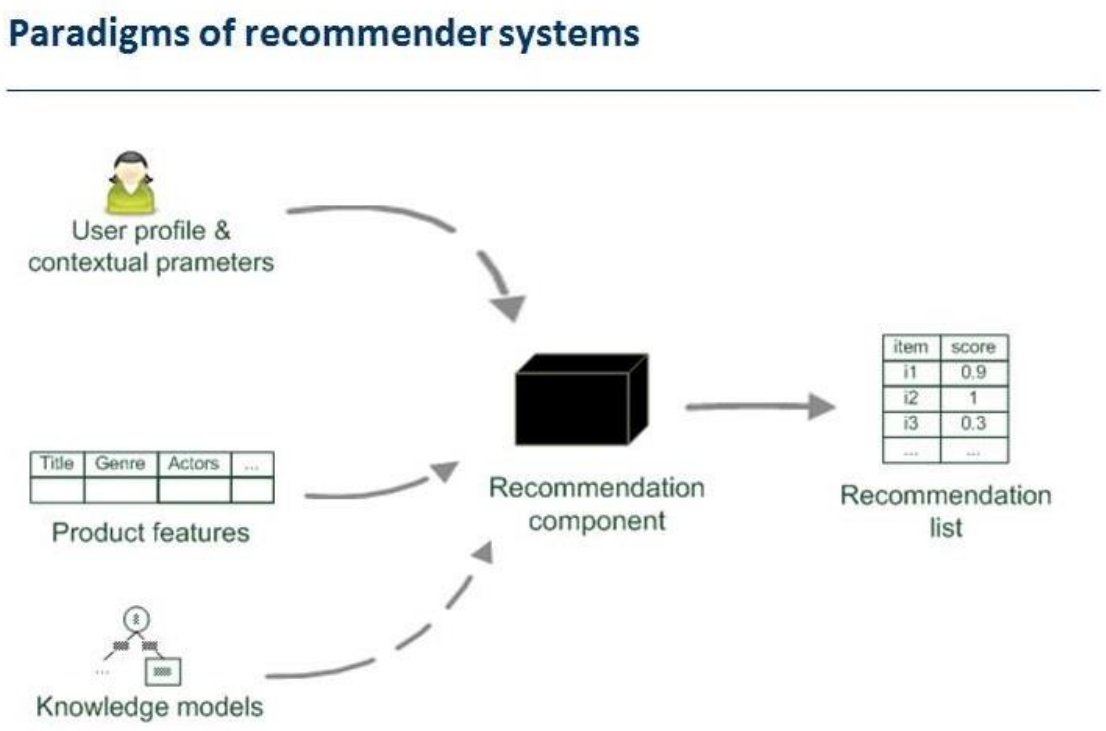


Fig. 2 : Working of a Recommendation System

In our recommendation system, the data collected from the user' s Facebook account is used to build a Profile of the user. This profile is calculated based on the features or attributes of the 'Liked' movies. Attributes used in this project are *Title, Actors, Genre* and *Ratings*. Then these features are processed by the Recommender Component and it produces a Recommendation list .This list contains the score of each movie which is then sorted and is provided to the user as output.

PLATFORM USED - PYTHON 2.7.11

4.1 Assignment Of ID To Liked Movies

In this module, ID is assigned to each liked movie present in *liked_movies.csv* by matching it against all movies present in our database *Movie_Database.csv* . The output generated is saved in *output.csv.*

INPUT- *liked_movies.csv , results.csv*

OUTPUT – *output.csv*

```
import csv

f2 = file('liked_movies.csv', 'r')
f3 = file('output.csv', 'w')
c3 = csv.writer(f3)

for user in csv.reader(f2):
        f1 = file('Movie_Database.csv', 'r')
        for admin in csv.reader(f1):
                try:
                        if user[0]==admin[0]:
                                #print admin[1]
                                f3.write(admin[1]+"\n")
                except IndexError:
                        pass

f1.close()
f2.close()
f3.close()
```

4.2 Calculation Of Actor_Count And Genre_Count

In this module, actor_count and genre_count is calculated. The attributes of movies used in this project are Actors, Genre and ratings.On the basis of these attributes, we get to know interests of the user. Actor_count and genre_count are calculated using the below code snippet:

INPUT - *output.csv*

OUTPUT - *actor_count.csv , genre_count.csv*

```
import csv
f1=file('output.csv','r')

actors_count=[]
   for i in range(270):
       actors_count.append(0)
   for row in csv.reader(f1):
       f2=file('results.csv','r')
       for j in csv.reader(f2):
           try:
               if row[0]==j[0]:
                   new_row=[]
                   new_row=j[1].split(":")
                   for i in range(0,len(new_row)):
                     #  print new_row[i]
                       index=int(new_row[i])
                       #print new_row[0] +" "+new_row[1]
                       #index=int(new_row[int(i)])
                       #print index
```

```
actors_count[index]=actors_count[index]+1

            except IndexError:
                pass
    f4=file('actors_count.csv','w')
    l= len(actors_count)
    #for i in actors_count:
     #   print i
    for i in range(1,l):
        try:
            output=''
            output=str(i)+","
            output=output+str(actors_count[i])
            f4.write(output)
            #print output
            f4.write("\n")
        except IndexError:
            pass

    f1.close()
    f2.close()

    ### End of part 1, now part 2

    genre_count=[0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0]
    f1=file('output1.csv','r')
    for row in csv.reader(f1):
        f2=file('results.csv','r')
        for j in csv.reader(f2):
            try:
                if row[0]==j[0]:
                    new_row=[]
                    new_row=j[2].split(":")
```

```
                for i in range(0,len(new_row)):
                    index=int(new_row[i])

                    #print new_row[0] +" "+new_row[1]
                    #index=int(new_row[int(i)])
                    #print index
                    genre_count[index]=genre_count[index]+1

        except IndexError:
            pass
f5=file('genre_count.csv','w')
l= len(actors_count)
#for i in actors_count:
 #   print i
for i in range(1,l):
    try:
        output=''
        output=str(i)+","
        output=output+str(genre_count[i])
        f5.write(output)
       # print output
        f5.write("\n")
    except IndexError:
        pass


f1.close()
f2.close()
f4.close()
f5.close()
```

4.3 Top Actor And Top Genre

In this module, TOP ACTORS and TOP GENRES are displayed to the user. TOP ACTORS are those actors which are present in the liked movies of the user the most number of times and similarly TOP GENRES is the list of favorite genres of the movies liked by the user.

INPUT - *actors_count.csv , genre_count.csv*

OUTPUT – *top_actors.csv , top_genre.csv*

```
import csv
import operator
f1=file('actors_count.csv','r')
f5=file('top_actors.csv','w')
csv1=csv.reader(f1)
sortedcsv=sorted(csv1,key=operator.itemgetter(1),reverse=Tru
e)
for i in sortedcsv:
    f2=file('actors.csv','r')
    for j in csv.reader(f2):
        if i[0]==j[1]:
            output=''
            output=i[0]+","+i[1]
            #print output
            f5.write(output+"\n")
            #print j[0]+" = "+i[1]
            break
f1.close()
f2.close()
```

```
f5.close()

#print "\n"

f3=file('genre_count.csv','r')
f6=file('top_genre.csv','w')
csv2=csv.reader(f3)
sortedcsv2=sorted(csv2,key=operator.itemgetter(1),reverse=Tr
ue)
for i in sortedcsv2:
    f4=file('genre.csv','r')
    for j in csv.reader(f4):
        if i[0]==j[1]:
            output=''
            output=i[0]+","+i[1]
            #print output
            f6.write(output+"\n")
          #  print j[0]+" = "+i[1]
            break
f3.close()
f4.close()
f6.close()
```

4.4 Calculation Of Movie Score

In this module, each of the three attributes i.e. Actors, Genres and Rating are assigned some weights. These weights vary according to the choices filled by the user. The user is provided with 3 choices-

A) By Actor

B) By Genre

C) By Ratings

Let the weight for Actor attribute be *w_actor*, weight for Genre be *w_genre* and for Ratings be *w_rating*.

BY ACTOR	BY GENRE	BY RATING	w_actor	w_genre	w_rating
Yes	No	No	0.8	0	0.2
No	Yes	No	0	0.8	0.2
No	No	Yes	0.1	0.1	0.8
Yes	Yes	No	0.5	0.4	0.1
Yes	No	Yes	0.5	0.1	0.4
No	Yes	Yes	0.1	0.4	0.5
Yes	Yes	Yes	0.5	0.2	0.3

Table 1 : Weight distribution table

ASSUMPTIONS:

We took a survey and found out that a general user usually watches a movie giving higher preference to his/her Favorite Actor/Actresses then movie ratings and lastly movie genre.

So order of preference used by our recommendation system is :

Actor > Rating > Genre

FORMULA :

$$\text{Movie\_score} = (A_1+A_2+A_3+\cdots\cdots\cdots+A_N) * \text{w\_actor} + (G_1+G_2+G_3+\cdots\cdots\cdots+G_M) * \text{w\_genre} + \text{Rating} * \text{w\_rating}$$

where,

A_i --> Count of actor with ID i

G_j --> Count of genre with ID j

N --> Number of actors present in the movie

M --> Number of Genres present in the movie

Rating --> Rating of the movie from IMDb

w_actor --> weight of Actor attribute

w_genre --> weight of Genre attribute

w_rating --> weight of Rating attribute

Code Snippet for this module,

INPUT - *test1.csv, result.csv, output.csv*

OUTPUT - *movie_score.csv*

```
import csv
import operator
f=file('test1.csv','r')

actor=0
genre=0
rating=0

for row in csv.reader(f):
    a=row[0]
    b=row[1]
    c=row[2]
if a=='1' and b=='1':       # By Actor , By Genre , By Rating
    if c=='1':
        actor=0.5
        genre=0.2
        rating=0.3
    else:                   # By Actor , By Genre
        actor=0.5
        genre=0.4
        rating=0.1
if a=='1' and c=='0' and b=='0':        # By Actor
    actor=0.8
    genre=0
    rating=0.2
if a=='0' and c=='0' and b=='1':        # By Genre
    actor=0
```

```
    genre=0.8
    rating=0.2
if a=='0' and c=='1' and b=='0':        # By Rating
    actor=0.1
    genre=0.1
    rating=0.8
if a=='0' and c=='1' and b=='1':        # By Genre , By Rating
    actor=0.1
    genre=0.4
    rating=0.5
if a=='1' and c=='1' and b=='0':        # By Actor , By Rating
    actor=0.5
    genre=0.1
    rating=0.4

#print actor
#print genre
#print rating
f.close()

f1=file('results.csv','r')
f5=file('movie_score.csv','w')


for row in csv.reader(f1):
    try:
        score=0
        flag=0
        new_row1=row[1].split(":")
        new_row2=row[2].split(":")
        #print new_row
        f4=file('output.csv','r')
        for col in csv.reader(f4):
            #print col[0]
```

```
            if row[0]==col[0]:
                flag=1
                #print col[0]
                break
        if flag==0:

            for i in range(0,len(new_row1)):
                index=new_row1[i]
                #print 'index'+" "+index
                f2=file('top_actors.csv','r')
                for j in csv.reader(f2):
                    if index==j[0]:
                   score=score+int(j[1])*(actor)
                            break
            for k in range(0,len(new_row2)):
                index=new_row2[k]
                f3=file('top_genre.csv','r')
                for l in csv.reader(f3):
                    if index==l[0]:
                        score=score+int(l[1])*(genre)
                        break
            score=score+float(row[3])*(rating)
            #print row[0]+" "+str(score)
            output=''
            output=row[0]+","+str(score)
            f5.write(output+"\n")

    except IndexError:
        pass
f4.close()
f5.close()
f1.close()
```

4.5 Final Recommendation List

This module provides the user with the final movie recommendation list. It takes the movie_score.csv file as input and display the top movies in descending order.

INPUT - *movie_score.csv*

OUTPUT - Final Recommendation List

```
import csv
import operator
f1=file('movie_score.csv','r')
csv1=csv.reader(f1)
sortedcsv=sorted(csv1,key=operator.itemgetter(1),reverse=True)
k=0
flag=0
for i in sortedcsv:
    f2=file('movy_id.csv','r')
    for j in csv.reader(f2):
        if i[0]==j[1]:
            print j[0]
            k=k+1
            if k==10:
                flag=1
                break
    if flag==1:
        break
```

5. SCREENSHOTS

Fig3 : Graphical User Interface

Fig 4 : What Next!!! Webpage

Fig 5 : Result_Database.csv

```
100,35:89:25,4:6:11,8.2
101,41:8,1:13,6.4
102,1:94:,6:11,8.3
103,49:155:156,3:5:6,8.6
104,132:157:,6:1,8.2
105,49:124,6:11,8.2
106,1:41:49,5:6,7.8
107,161:159:,6:11,7.4
108,89:9:25,6:11:9,7.4
109,132:53:162,6:11:4,7.3
110,135:81:164,6:1,7.5
111,7:14:68,6:1:5,7.4
112,137:166:167,6:11:,7.5
```

Fig 6 : Sample input

```
1  Aligarh (film)
2  Commando (2013 film)
3  My Name Is Khan
4  Highway
5  Queen The Film
6  Haider
7  Golmaal - Fun Unlimited
8  Bajirao Mastani
9  Disney ABCD2
10 Oh My God (2012 film)
11 Devdas (soundtrack)
12 Holiday
13 Tare Zamee Par
14 Ki And Ka
15 Chennai Express
16 Gabbar Is Back
17 Bajrangi Bhaijaan
18 Kapoor And Sons
```

List of movies liked by user extracted from facebook

Fig 6 : Output Generated

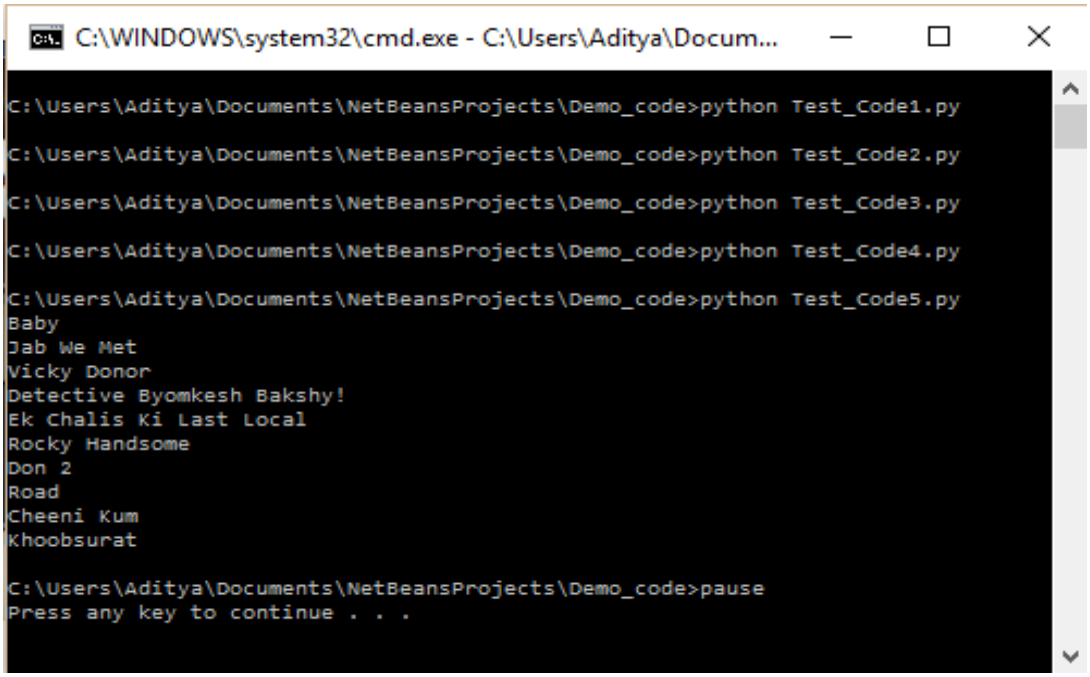
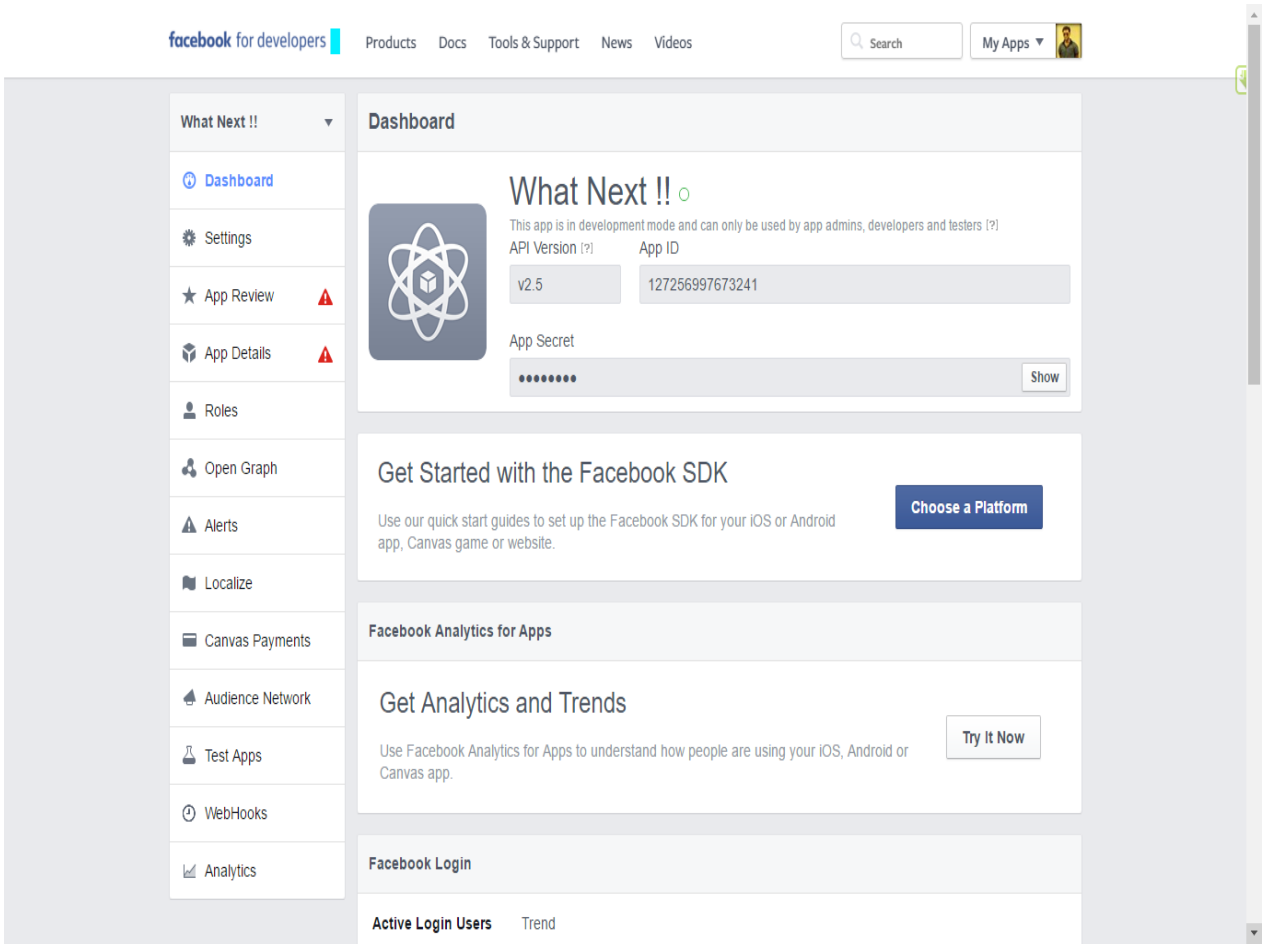


Fig 7 : Facebook Application Page



6. TECHNOLOGIES USED

1. Facebook SDK JavaScript

2. Facebook Graph API

3. WebSocket J2EE

4. Apache Tomcat

5. Python 2.7.11

6. Java

8. HTML, CSS

PLATFORMS – Netbeans IDE, Eclipse IDE , Python IDLE

7. CONCLUSION

Recommendation systems in e-commerce become increasingly important due to the large number of choices consumers face. Thus we have developed a recommendation system which recommends list of movies which the user may watch according to his like. Here by like we mean kind of movies he/she had liked on his/her facebook profile. "What Next!!!" recommender engine , fulfills this functionality .

Future extension to this project is that in current engine we recommend only Bollywood movies but we can extend it to Hollywood movies also. TV shows recommendation functionality is also likely to come.

With one or more user s using the system , we can then generate similarity criteria between them and then recommend movies of common interest.

8. REFERENCES

1. http://developers.facebook.com
2. https://developers.facebook.com/docs/graph-api/using-graph-api
3. https://blog.idrsolutions.com/2013/12/websockets-an-introduction
4. http://www.tutorialspoint.com/javascript/index.htm
5. http://www.tutorialspoint.com/jquery/index.htm
6. http://www.imdb.com/
7. http://www.facebook.com/
8. http://stackoverflow.com/