Project Report on

MOVIE RECOMMENDATION ENGINE

By
Anirban Sen(20130450)
Debashish Sen(20130492)
Sumit Kumar(20130595)

In partial fulfillment of requirements for the award of degree in
Bachelor of Technology in Computer Science and Engineering
(2017)

Under the Project Guidance of
Mr. Abhijit Chanda,
(VP Technology, Gamma Analytics)

Internal reviewer
Mrs. Jhuma Sunuwar (Assistant Professor II),
Computer Science and Engineering Department

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SIKKIM MANIPAL INSTITUTE OF TECHNOLOGY
(A constituent college of Sikkim Manipal University)
MAJITAR, RANGPO, EAST SIKKIM – 737136

PROJECT REVIEW CERTIFICATE

This is to certify that the work recorded in this project report entitled **"Movie Recommendation Engine"** has been jointly carried out by **Mr. Anirban Sen (Reg 20130450), Mr. Debashish Sen (Reg 20130492) and Mr. Sumit Kumar (Reg 20130595)** of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering. This report has been duly reviewed by the undersigned and recommended for final submission for Major Project Viva Examination.

Mrs. Jhuma Sunuwar,
Assistant Professor II,
Department of Computer Science & Engineering
Sikkim Manipal Institute of Technology
Majitar, East Sikkim – 737136.

CERTIFICATE OF ACCEPTANCE

This is to certify that the below mentioned student(s) of Computer Science & Engineering Department of Sikkim Manipal Institute of Technology (SMIT) has / have worked under the supervision of **Mr. Abhijit Chanda** of **Gamma Analytics, Kolkata** from **9th January 2017 to 8th May 2017**on the project entitled **"Movie Recommendation Engine"**

The project is hereby accepted by the Department of Computer Science & Engineering, SMIT in partial fulfillment of the requirements for the award of Bachelor of Technology in Computer Science and Engineering.

University Registration No	Name of Student(s)	Project Venue
20130450	Anirban Sen	Gamma Analytics, Kolkata
20130492	Debashish Sen	Gamma Analytics, Kolkata
20130595	Sumit Kumar	Gamma Analytics, Kolkata

Dr. Kalpana Sharma
Professor & HOD
Computer Science & Engineering Department
Sikkim Manipal Institute of Technology
Majhitar, Sikkim - 737136

DECLARATION

We, the undersigned, hereby declare that the work recorded in this project report entitled **"Movie Recommendation Engine"** in partial fulfillment for the requirements of award of B.Tech (CSE) from Sikkim Manipal Institute of Technology (A constituent college of Sikkim Manipal University) is a faithful and bonafide project work carried out at **"Gamma Analytics, Kolkata"** under the supervision and guidance of **Mr. Abhijit Chanda of Gamma Analytics, VP (Technology).**

The results of this investigation reported in this project have so far not been reported for any other Degree / Diploma or any other Technical forum.

The assistance and help received during the course of the investigation have been duly acknowledged.

Anirban Sen (20130450)

Debashish Sen (20130492)

Sumit Kumar (20130595)

ACKNOWLEDGEMENT

We would like to extend our sincere thanks to our mentors without whose support this project could not be completed successfully. We are highly indebted to **Mr. Abhijit Chanda, Vice- President (Technology), Gamma Analytics, Kolkata** for his constant guidance and supervision and for providing necessary information required for the successful completion of the project.

We would like to express our gratitude towards **Mrs. Jhuma Sunuwar**, **Assistant Professor II, Department of Computer Science & Engineering, Sikkim Manipal Institute of Technology** for providing valuable guidance in completion of this project.

We would also like to extend our utmost gratitude towards **Prof (Dr.) Kalpana Sharma, HOD, Department of Computer Science and Engineering, Sikkim Manipal Institute of Technology** for her kind co-operation and encouragement which helped us in completion of this project.

We would like to express our special gratitude towards **Mr. Biswaraj Sen** and Major Project Coordinators for giving such attention and time and for organizing the presentations in time. Our thanks and appreciations also go to Computer Science and Engineering department of SMIT, Majhitar.

We would like to express our deep gratitude to **Mr. Tapojyoti Bhattacharjee, Mr. Rishav Kumar, Mr. Rahul Jotder, Mr. Ramraj Patel, Mr. Subhajyoti Mitra and Mr. Jeffin Ben** for their support & guidance.

Anirban Sen(20130450)

Debashish Sen (20130492)

Sumit kumar (20130595)

DOCUMENT CONTROL SHEET

1	Report No	CSE/Major Project/ External /B.Tech/73/2017
2	Title of the Report	Movie Recommendation Engine
3	Type of Report	Technical
4	Author(s)	1. Anirban Sen (Reg 20130450) 2. Debashish Sen (Reg 20130492) 3. Sumit Kumar (Reg 20130595)
5	Organizing Unit	Gamma Analytics, Kolkata
6	Language of the Document	English
7	Abstract	To develop a Movie Recommendation Engine
8	Security Classification	General
9	Distribution Statement	General

List of Figures

List of Tables

Content

ABSTRACT

Recommendation systems have become prevalent in recent years as they dealing with the information overload problem by suggesting users the most relevant products from a massive amount of data. For media product, online collaborative movie recommendations make attempts to assist users to access their preferred movies by capturing precisely similar neighbours among users or movies from their historical common ratings. However, due to the data sparsely, neighbour selecting is getting more difficult with the fast increasing of movies and users. It employs principal component analysis (PCA) data reduction technique to dense the movie population space which could reduce the computation complexity in intelligent movie recommendation as well. The project results on Movielens dataset indicate that the proposed approach can provide high performance in terms of accuracy, and generate more reliable and personalized movie recommendations when compared with the existing methods.

INTRODUCTION

Fast development of internet technology has resulted in explosive growth of available information over the last decade. Recommendation systems (RS), as one of the most successful information filtering applications, have become an efficient way to solve the information overload problem. The aim of Recommendation systems is to automatically generate suggested items movies, books, news, music, CDs, DVDs, WebPages) for users according to their historical preferences and save their searching time online by exacting useful data.

Movie recommendation is the most widely used application coupled with online multimedia platforms which aims to help customers to access preferred movies intelligently from a huge movie library. Lot of work has been done both in the academic and industry area in developing new movie recommendation algorithms and extensions. The majority of existing recommendation systems is based on collaborative filtering (CF) mechanism which has been successfully developed in the past few years. It first collects ratings of movies given by individuals and then recommends promising movies to target customer based on the "like-minded" individuals with similar tastes and preferences in the past. There have been many famous online multimedia platforms (e.g., youtube.com, Netflix.com, IMDB.com) incorporated with CF technique to suggest media products to their customers.

Some examples for recommendation engine:

Example 1:

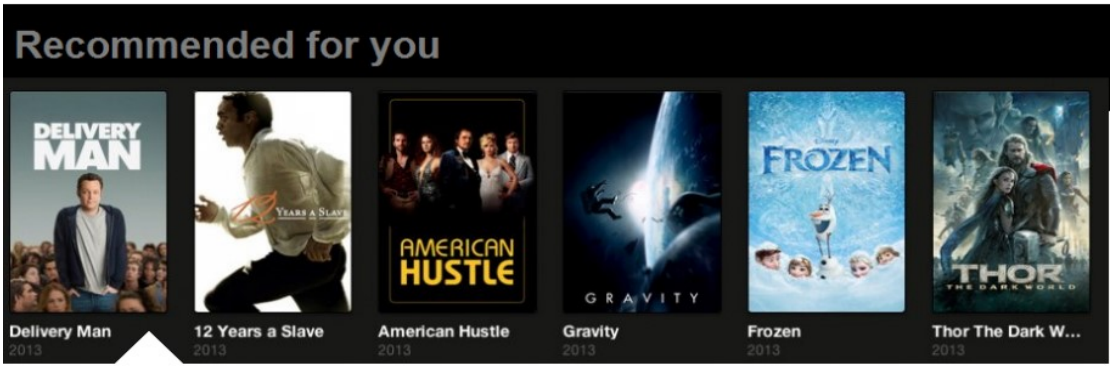


Figure No. 1: Example of Recommendation Engine

Example 2:

Figure No. 2: Example of Recommendation Engine

1.1 General overview of the problem

The growth of the Internet has made it much more difficult to effectively extract valuable information from all the available online information. Users face "The Paradox of Choice" problem, when the user runs into too many items in the catalog; user may end up browsing few top showed products. Eliminating user choices can greatly reduce time, thereby user may not lead into confusion. Huge amount of data requires mechanisms for efficient information filtering.Consumer research suggests that a typical Netflix member loses interest after perhaps60 to 90 seconds of choosing, having reviewed 10 to 20 titles (perhaps 3 in detail)on one or two screens. The user either finds something of interest or the risk of the user abandoning our service increases substantially. The recommender problem is to make sure that on those two screens each member in our diverse pool will find something compelling to view, and will understand why it might be of interest.The growing use of Internet as an information source, has led to the proliferation of technologies to deploy rich web based applications. Among these applications are present the service providers. Some use recommendations techniques to improve the user experience.

1.2 Literature Survey

Sl. No.	Paper Name/Site Name	Author Name	Findings/Methodology/ Research Gap
1	"Algorithms and Methods in Recommender Systems", Available: https://www.snet.tu-berlin.de/fileadmin/fg220/courses/SS11/snet-project/recommender-systems_asanov.pdf	"Daniar Asanov", Berlin Institute of Technology Berlin, Germany	The various algorithms and methods used by recommendation system described.
2	"Movie Recommendations from User Ratings", Available: cs229.stanford.edu/proj2013/Bystrom-MovieRecommendationsFromUserRatings.pdf	"Hans Byström", Stanford University bystrom@stanford.edu	K-means clustering method used.
3	"Movie Recommendations using Hybrid Recommendation Systems", "International Journal on Recent and Innovation Trends in Computing and Communication", ISSN: 2321-8169, Volume: 4 Issue: 12, Available: http://www.ijritcc.org/download/browse/Volume_4_Issues/December_16_Volume_4_Issue_12/1483246580_31-12-2016.pdf	"Khyati Aggarwal",Department of Information Technology Sardar Patel Institute of technology, Mumbai, India khyatiaggarwal@gmail.com	Hybrid Recommendation System, combination of content and collaborative filtering used.

4	"An Improved Collaborative Movie Recommendation System using Computational Intelligence", Available: ksiresearchorg.ipage.com/seke/dms14paper/paper18.pdf	"Zan Wang", Department of Software Engineering, School of Computer Software, Tianjin University Tianjin, 300072, P.R. China wangzan@tju.edu.cn	Collaborative filtering method used for recommendation.
5	"Amazon.com recommendations: Item to item collaborative filtering ",IEEE (Institute of Electrical and Electronics Engineers) Internet Computing, vol.7, no.1, pp. 76–80, 2003., B. Smith and J. York, Available: https://www.cs.umd.edu/~samir/498/Amazon-Recommendations.pdf	"G. Linden", "B. Smith" and "J. York"	Collaborative filtering via item to item relation achieved.
6	"Matrix factorization techniques for recommender systems", Available: https://datajobs.com/data-science-repo/Recommender-Systems-%5BNetflix%5D.pdf	"Yehuda Koren", Yahoo Research, "Robert Bell" and "Chris Volinsky", AT&T Labs-Research	Matrix factorization technique for recommending items.

Table No. 1: Literature Survey

1.3 Problem Definition

Obtaining recommendations from trusted sources is a critical component of the natural process of human decision making. With burgeoning consumerism buoyed by the emergence of the web, buyers are being presented with an increasing range of choices while sellers are being faced with the challenge of personalizing their advertising efforts. In parallel, it has become common for enterprises to collect large volumes of transactional data that allows for deeper analysis of how a customer base interacts with the space of product offerings. Recommender Systems have evolved to fulfil the natural dual need of buyers and sellers by automating the generation of recommendations based on data analysis. Initial formulation for recommender systems were based on straight forward correlation statistics and predictive modelling, not engaging the wider range of practices in statistics and machine learning literature.

1.4 Analysis of the project and Software Requirement and Specification (SRS)

The SRS document usually contains all the user requirements in an informal form. Among all the documents produced during application, writing the SRS document is probably the toughest. One reason behind this difficulty is that the SRS document is expected to cater to the needs of a wide variety of audience. Different people need the SRS document for different purpose.

1.4.1 Functional Requirement:

R1: Register user

Description: Reading user details, validating then storing in database.

Input: “username”, “firstName”, “lastName”, “password”

Output: User prompted to enter details.

R2: User login

Description: Reading user details, authenticating user then redirect to home page if valid user else prompt user to retry.

Input: “username”, “password”

Output: User prompted to enter details.

R3: Rate movies

Description: User can rate movies according to his/her preferences, taste, and interest.

Input: “userId”, “movieId”, “ratings”, “timestamp”

Output: User prompted to enter movie ratings.

Processing: After rating different movies when user clicks save button, new entry will be created in the rating table if the movie is rated for the first time. Else previous rating will be updated with new rating for respective movies.

R4: Top rated movies

Description: User can view top rated movies with respect to average ratings.

Input: “ratings”, “movieId”, “movieName”, “genre”

Output: User prompted to select a particular genre to view top rated movies of respective genre.

Processing: Genre wise all top rated movies are grouped. On click of any genre, top rated movies of that particular genre will be shown.

R5: Generate Recommendations

Description: Executing alternating least square (ALS) algorithm.

Input: “ratings”, “userId”, “movieId”, “movieName”, “training_df”, “validation_df”

Processing: Using training dataframe, models created for different ranks. Then using validation dataframe root mean square error (RMSE) is calculated, i.e., difference between actual value and predicted value. The model with least root mean square error is selected and respective rank is set as best rank.

R6: Recommended Movies

Description: User can view recommended movies with respect to prediction.

Input: “userId”, “genre”

Output: User prompted to select a particular genre to view recommended movies of respective genre.

Processing: Genre wise all the recommended movies are grouped. On click of any genre, recommended movies of that particular genre will be shown.

R7: User History

Description: User can view his/her ratings history. Which movies have been rated, how much rating has been given and when rating is done can be checked.

Input: “userId”, “ratings”

Output: User can view movies rated by him/her along with timestamp.

R8: Analysis Data

Description: User can view charts showing graphical representation of various analysis.

Input: “ratings”, “movieId”, “movieName”

Output: Analysis data represented using pie chart, bar graph, and line graph.

1.4.2 Non- Functional Requirement:

i. The application should be simple and fast.

ii. The GUI should be proper and user friendly.

iii. Response from the server should be fast.

iv. Security should be high such that no intruder can tamper the data.

v. Reliability: The application should generate all updated information in correct order.

1.4.3 Software Requirement:

i. Operating Environment : Windows

ii. Platform: IntelliJ IDEA

iii. Apache server

iv. Back End: JAVA , MySQL, Python, Shiro, Spark, HDFS.

v. Front End: JSP, JQuery, CSS

1.4.4 Minimum Hardware Requirement:

i. Intel Pentium Core i3

ii. 2 GB of RAM

iii. Minimum hard drive space required : 50 GB

1.5 Solution Strategy

1.5.1 Methodology

Collaborative filtering is one of the techniques used for dealing with "The Paradox of Choice" problem. Collaborative filtering technique is based on the collection and analyzing of data (i.e. user's rating, feedback, user's last purchase history etc.).

Basic Idea of Collaborative filtering:

For a given customer A, Find a set of customers who have rated similar items in the past. Aggregate the items bought by these similar set of customers. Eliminate the items that the user has already rated. Present the rest of the items to the user.

An example of collaborative filtering - At first, people rate different items (like videos, images, games etc). After that, the system is making predictions about user's rating for an item, which the user hasn't rated yet.

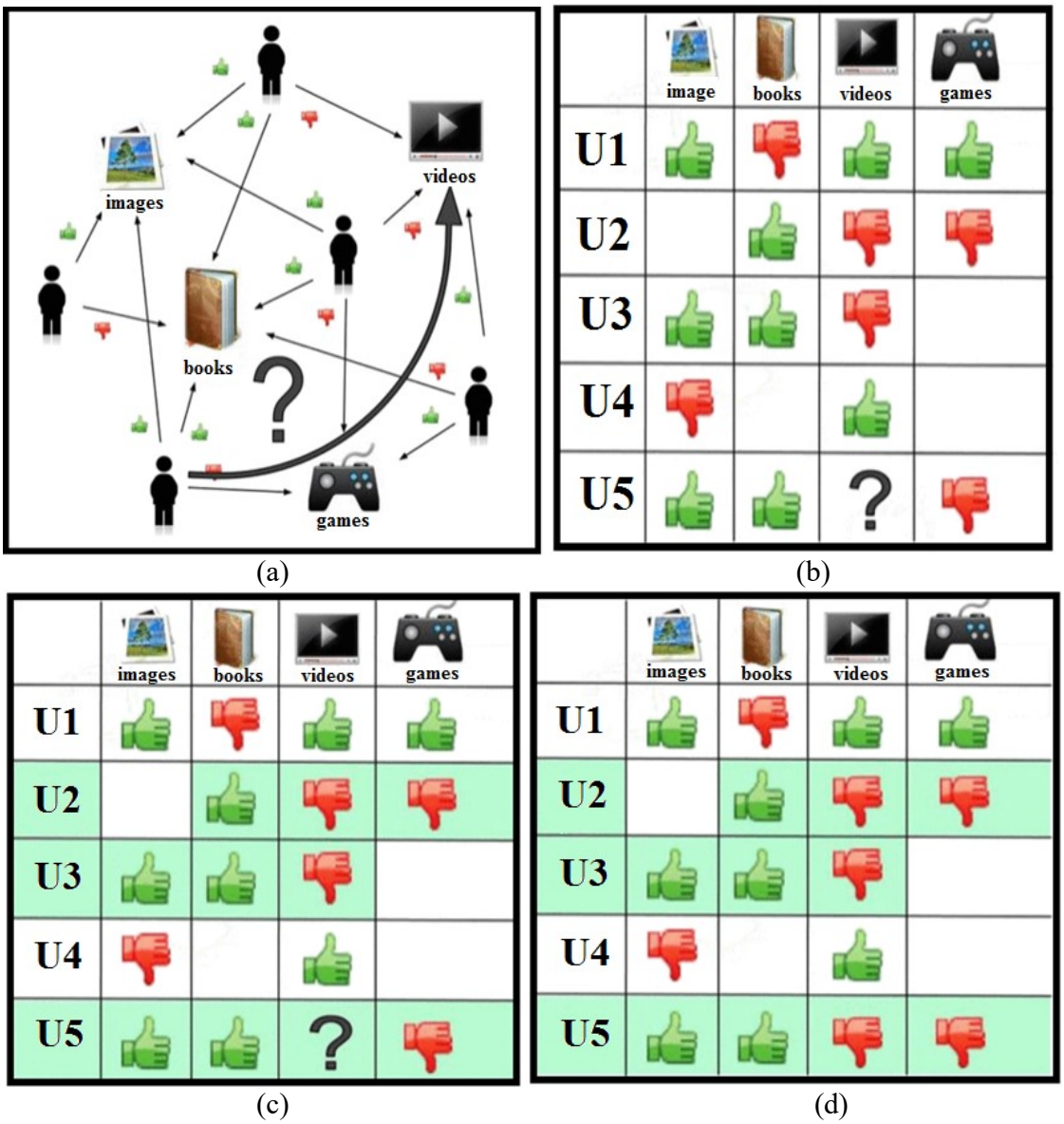


Figure No. 3: Collaborative Filtering Steps (a) Users rate different items (b)Read user ratings for different items (c) Aggregate similar users based on ratings(d) Prediction

- **Basic approach to Recommendation -**

Data Collection -

Let us assume that a user of Amazon website is browsing books and reading the details. Each time the reader clicks on a link, an event may be registered.

Ratings -

Ratings are important in the sense that they tell you what a user feels about a product.

Filtering -

Filtering means grouping products based on ratings and other user data.

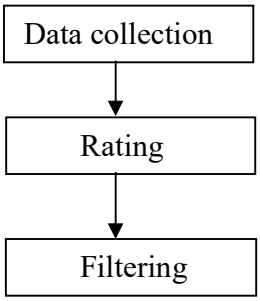


Figure No. 4: Basic block diagram of recommendation engine

1.5.2 Processing

Spark MLlib implements a collaborative filtering algorithm called Alternating Least Squares (ALS).ALS approximates the sparse user item rating matrix of dimension K as the product of two dense matrices--User and Item factor matrices of size U×K and I×K (see picture below).

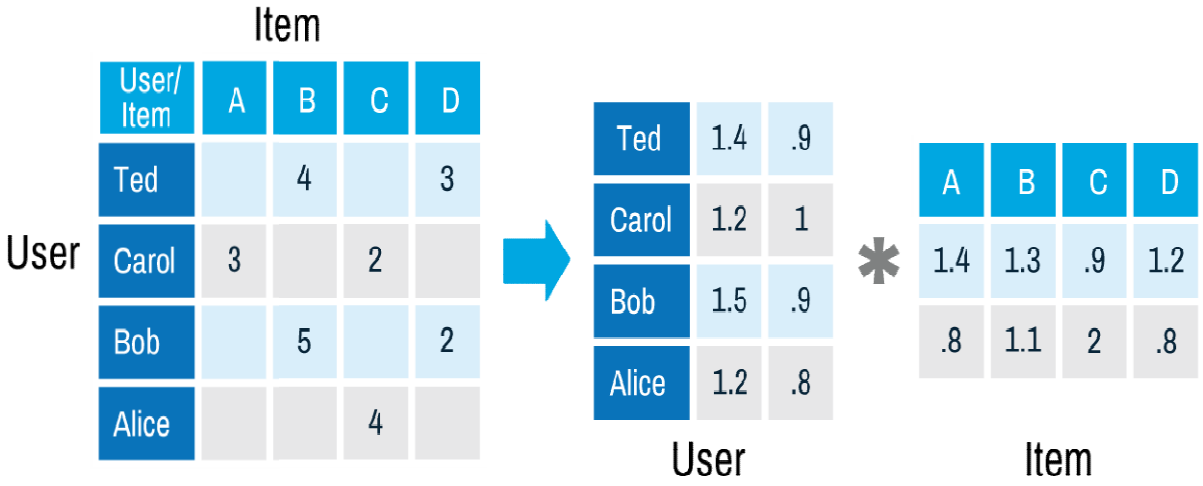


Figure No. 5: Matrix factorization

One matrix tries to describe the latent or hidden features of each user, and one tries to describe latent properties of each movie. In this algorithm, first the user matrix will be fixed and solves for item matrix and secondly the item matrix will be fixed and solves for user matrix. For different ranks ROOT MEAN SQUARE ERROR (RMSE) value is calculated (i.e matrix multiplication of user and item matrices – original given matrix values). The least error value will be the predicted value or recommended value.

1.5.3 Spark

Apache Spark is an open-source cluster-computing framework, originally developed at the University of California, Berkeley's AMPLab, the Spark codebase was later donated to the Apache Software Foundation, which has maintained it since. Spark provides an interface for programming entire clusters with implicit data parallelism and fault-tolerance.

Apache Spark continues to gain momentum in today's big data analytics landscape. Although a relatively newer entry to the realm, Apache Spark has earned immense popularity among enterprises and data Analysts within a short period. Apache Spark is

one of the most active open source big data projects. The reason behind is its versatility and diversity of use.

Some of the key features that make Spark a strong big data engine are:

- Equipped with MLlib library for machine learning algorithms
- Good for Java and Scala developers as Spark imitates Scala's collection API and functional style
- Single library can perform SQL, graph analytics and streaming.

Spark is admired for many reasons by developers and analysts to quickly query, analyze and transform data at scale. In simple words, you can call Spark a competent alternative to Hadoop, with its characteristics, strengths and limitations. Spark runs in-memory to process data with speed and sophistication than the other complement approaches like

Hadoop or MapReduce. It can handle several terabytes of data at one time and perform efficient processing.

One of the excellent benefits of using Spark is that it is often used in Hadoop's data storage model, i.e. HDFS and can well integrate with other big data frameworks like HBase, MongoDB, Cassandra. It is one of the best big data choices to learn and apply machine learning algorithms in real-time. It has the ability to run repeated queries on large databases and potentially deal with them.

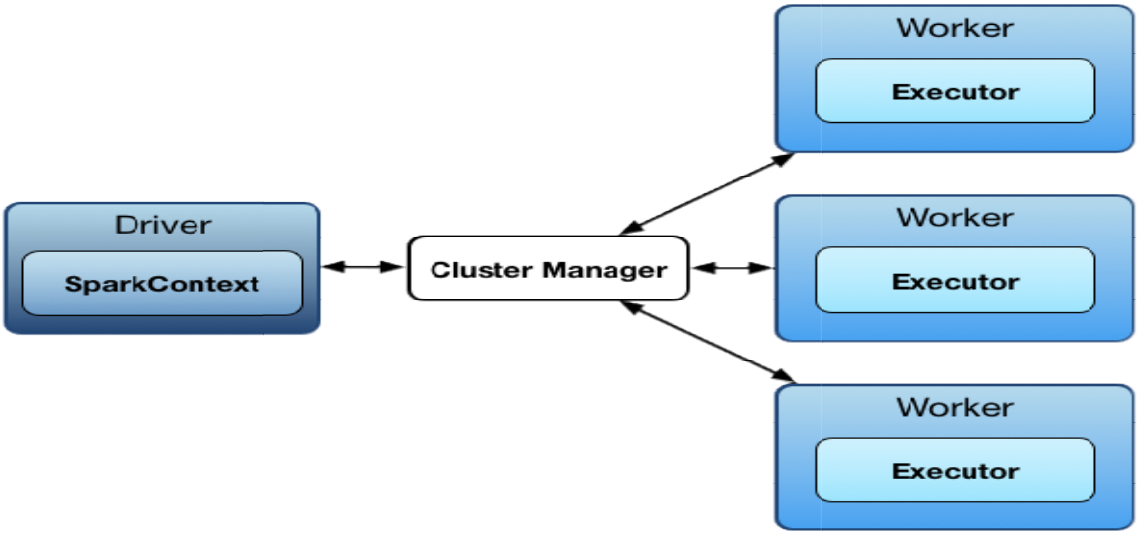


Figure No. 6: Block diagram for spark architecture

There are several useful things to note about this architecture:

1) Each application gets its own executor processes, which stay up for the duration of the whole application and run tasks in multiple threads. This has the benefit of isolating applications from each other, on both the scheduling side (each driver schedules its own tasks) and executor side (tasks from different applications run in different JVMs). However, it also means that data cannot be shared across different Spark applications without writing it to an external storage system.
2) Spark is agnostic to the underlying cluster manager. As long as it can acquire executor processes, and these communicate with each other, it is relatively easy to run it even on a cluster manager that also supports other applications (e.g. Mesos/YARN).
3) The driver program must listen for and accept incoming connections from its executors throughout its lifetime. As such, the driver program must be network addressable from the worker nodes.

4) Because the driver schedules tasks on the cluster, it should be run close to the worker nodes, preferably on the same local area network. If you'd like to send requests to the cluster remotely, it's better to open an RPC (Remote processing call) to the driver and have it submit operations from nearby than to run a driver far away from the worker nodes.

1.5.4 Data transfer

HDFS is a Java-based file system that provides scalable and reliable data storage, and it was designed to span large clusters of commodity servers. Spark does not replace Hadoop's HDFS. It just replaces Hadoop's distributed processing. HDFS and can well integrate with other big data frameworks like HBase, MongoDB, Cassandra. HDFS was used in the project to transfer the CSV (Comma Separated Values) file that contains the dataset which is required by the worker PC's for processing the data.

1.6 Organization of the report

CHAPTER-1 provides an introduction to the problem using minimal language so as to enable an individual to understand and grasp the issue which is being dealt with and gives the details of research and survey done, followed by a formal problem definition and analysis.

CHAPTER-2 describes the design strategy for the solution. It includes the flowcharts and architecture diagram for the different components used for the application.

CHAPTER-3 describes the implementation details through pseudo code and screenshots.

CHAPTER-4 outlines the test strategy and overall test approach for the project. This test strategy, testing objectives, resources (manpower, software and hardware) required for testing, test schedule, test estimation and test deliverables.

CHAPTER-5 includes results and discussion topics which help to know the outcome of the recommendation engine developed.

CHAPTER-6 provides the summary and conclusion of the work. It enlist the achievements achieved in making the project, major difficulties encountered, limitations and foreseeable modifications in the project, during the course of the project.

CHAPTER-7 gives the list of references used in the course of the project.

DESIGN STRATEGY

2.1 Block Diagram

2.1.1 Block Diagram for Movie Recommendation engine

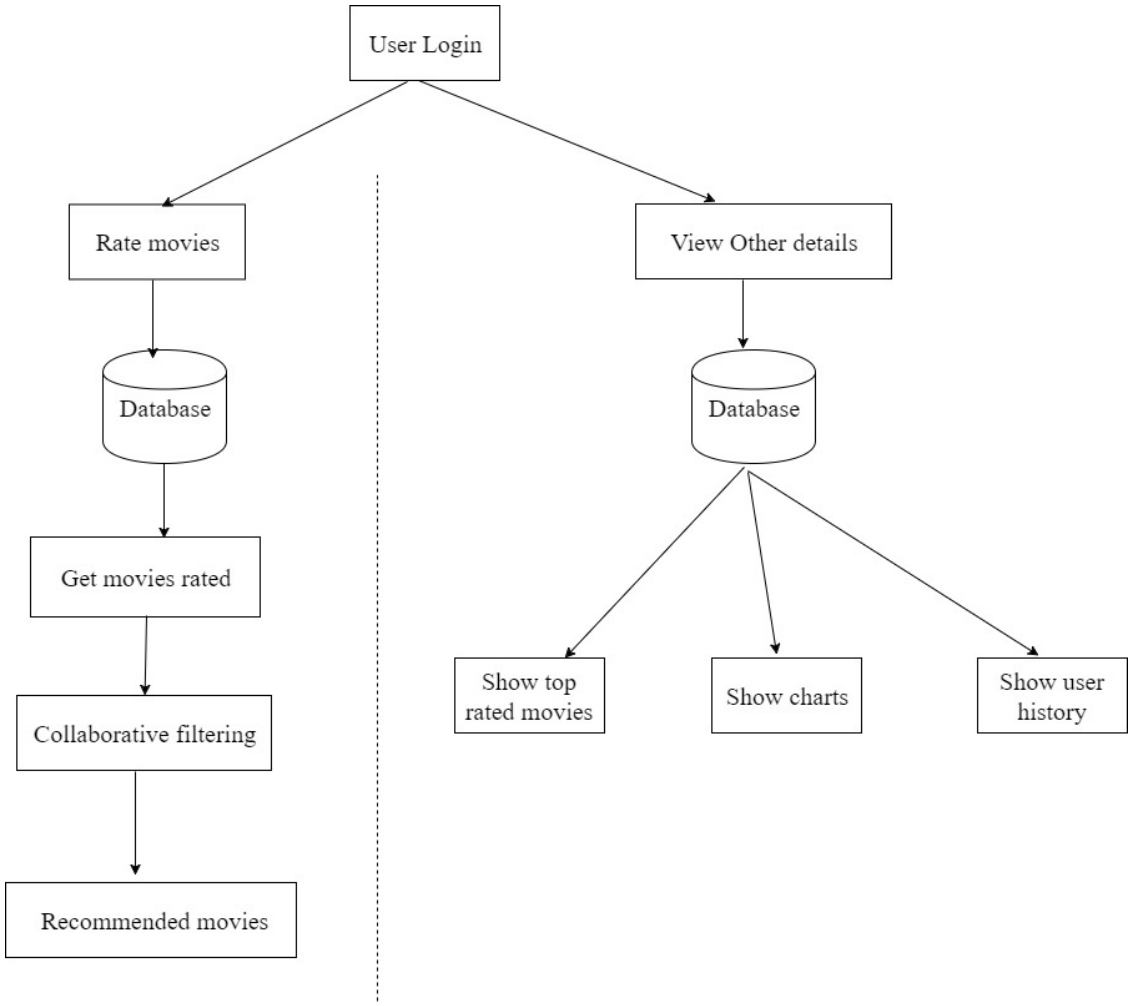
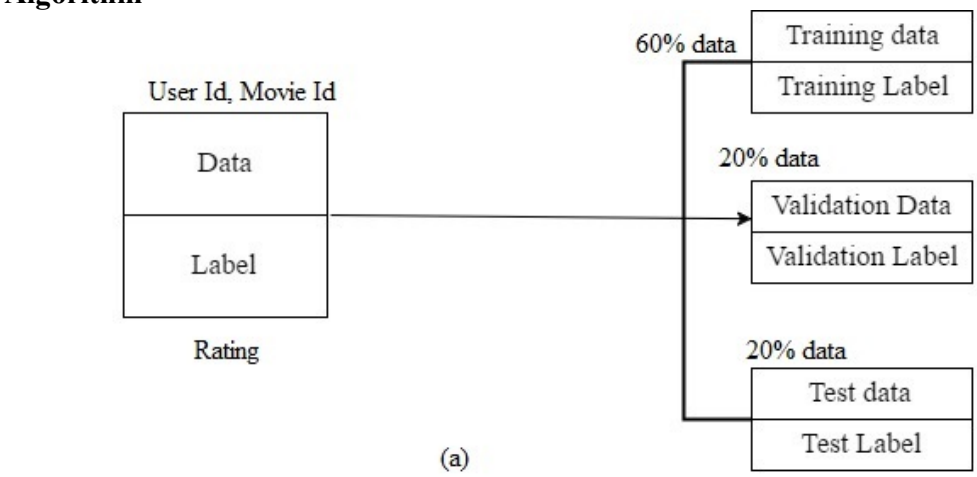
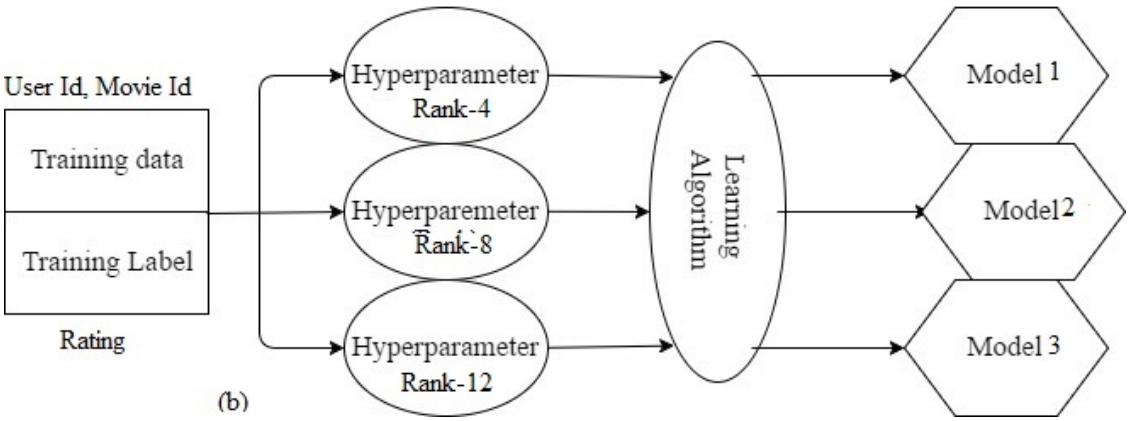


Figure No. 7: Block diagram for recommendation engine

2.1.2 Block Diagram for Machine Learning Algorithm i.e. Alternating Least Square Algorithm



(a)



(b)

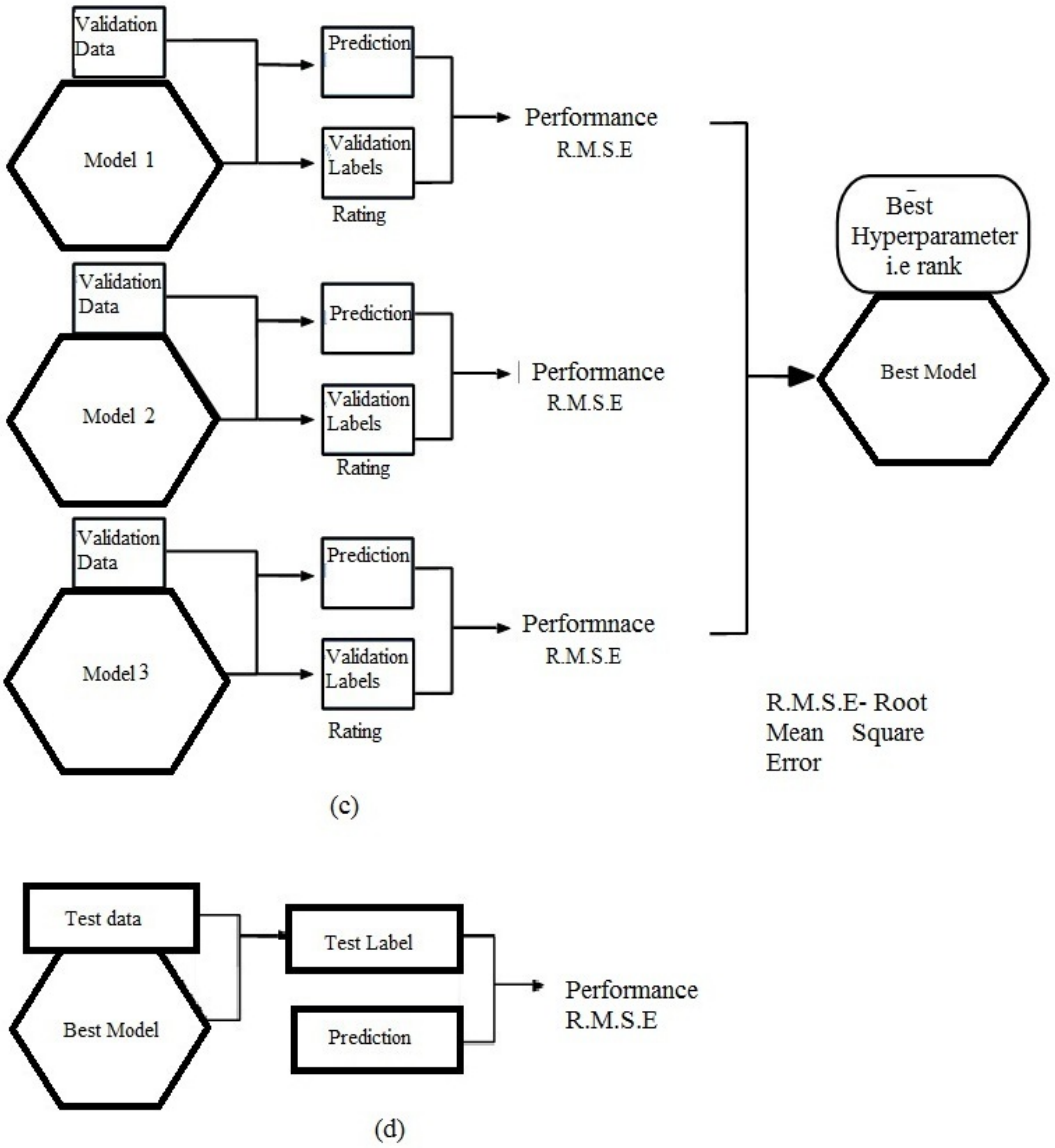


Figure No. 8: Machine Learning Steps (a) Splitting dataset (b) Training Models with different ranks (c) Finding best model (d) Checking best model's performance

2.2 Activity diagram

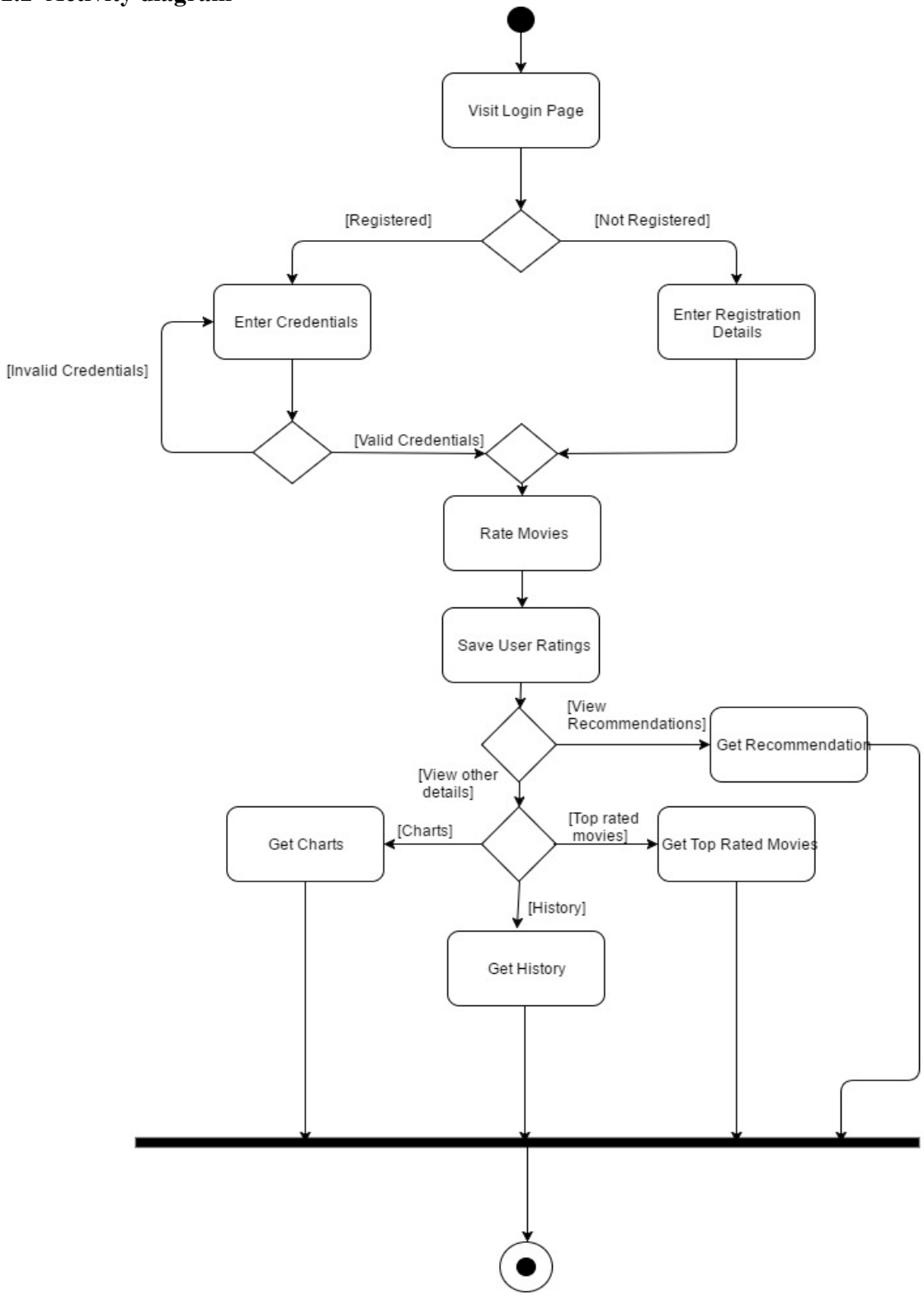


Figure No. 9: Activity diagram for recommendation engine

2.3 Use Case Diagram

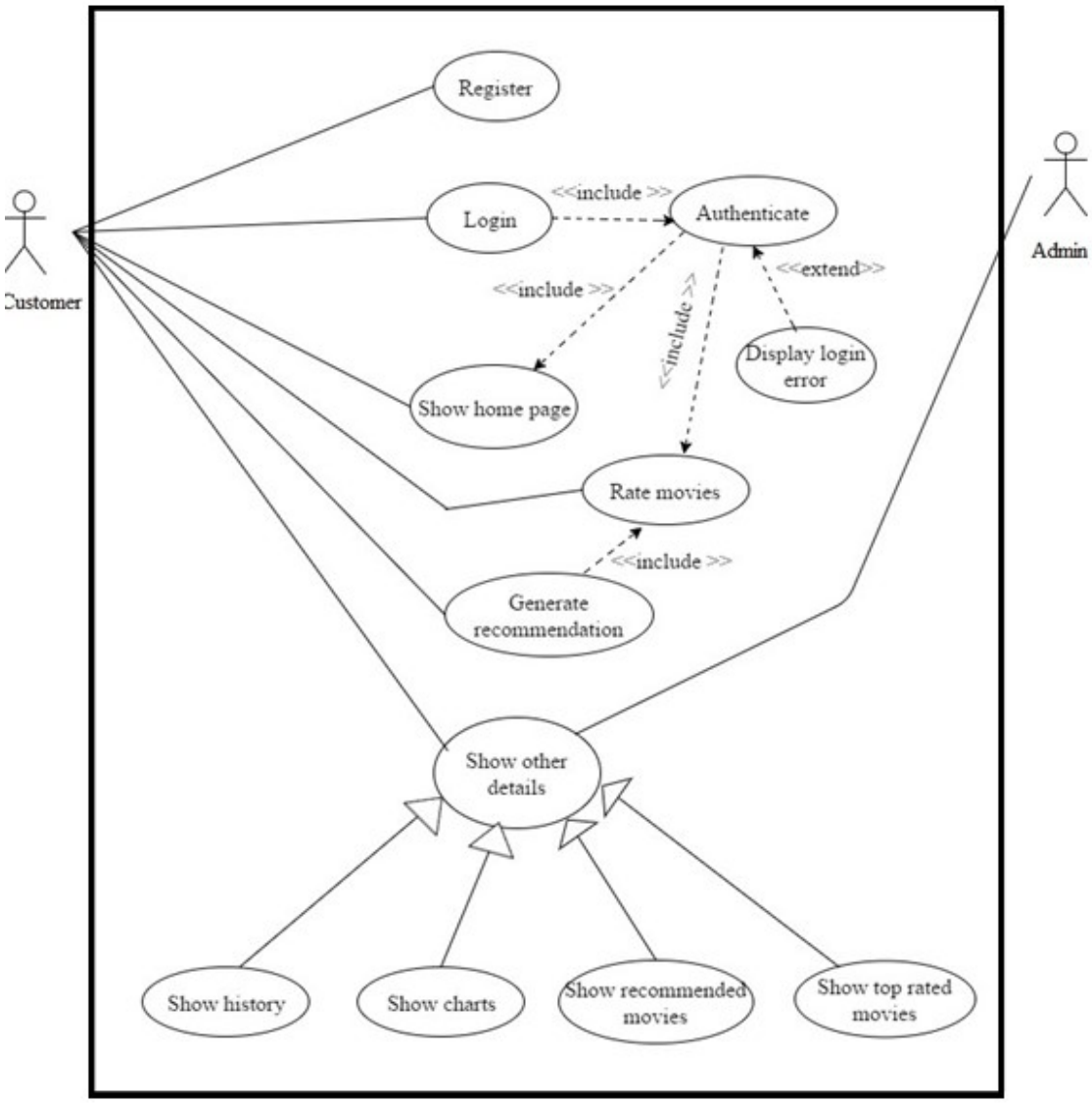


Figure No. 10: Use Case diagram for recommendation system

DETAILED TEST PLAN

- **Optimistic Test Cases**

Serial No	Test Condition	Input Data	Expected Result (Error Message)	Result Obtained
1	At Signup & Login Page, User clicks signup	User Details	Generated User ID	New User ID generated
2	At Signup & Login Page, User clicks login	Username and password	Homepage should open	User Homepage opens
3	At Homepage, User clicks Top Rated Movies-View More	Average Ratings from Database	Top Rated Movies with respect to average ratings	Top Rated Movies with respect to average ratings is shown
4	At Homepage, User clicks Recommended Movies-View More	Ratings	Recommended Movies with respect to prediction	Recommendation as per user taste shown
5	At Rate Movies Page, User rates different movies and clicks save	Rating out of 5	Store user-movie ratings into table	User-movie ratings stored into table
6	At Top Rated Movies Page, User clicks on any particular genre	Genre Name	Top Rated Movies of particular genre should be shown	Top Rated Movies of particular genre displayed
7	At Recommended Movies Page, User clicks on any	Genre Name	Recommended Movies of particular genre should be	Recommended movies of particular genre shown

	particular genre		shown.	
8	At Analysis Page, User clicks on click any of the sector in the pie chart	Movie Name	User id, movie rating, timestamp should be shown	User id, movie rating, timestamp displayed
9	At Analysis Page, User clicks on any bar in the bar graph	Rating	Movie id, movie name, count, genre should be shown	Movie id, movie name, count, genre displayed
10	At Analysis Page, User clicks on any particular genre on the line graph	Genre	All movies with average ratings of particular genre should be shown	All movies with average ratings of particular genre shown

Table No. 2: Optimistic Test Cases

- **Pessimistic Test Cases**

Serial No	**Test Condition**	**Input Data**	**Expected Result (Error Message)**	**Result Obtained**
1	At Signup & Login Page, User clicks signup	No input	Error message "Fields cannot be blank"	Error Message displayed
2	At Signup & Login Page, User clicks login	No input or either username or password	Error message "Fields cannot be blank"	Error Message displayed
3	At Signup & Login Page, User clicks login	Incorrect Username or password	Error message "Username/Password not matched"	Error Message displayed

Table No. 3: Pessimistic Test Cases

IMPLEMENTATION DETAILS

4.1 Load and Cache

1. Begin
2. Create movies DataFrame as movies_df and ratings DataFrame as ratings_df with required schema.
3. Load data to movies_df and ratings_df from CSV files (datasets).
4. Cache both the movies_df and ratings_df in memory.
5. End

4.2 Top Rated Movies

1. Begin
2. Transform ratings_df to another DataFrame, movie_ids_with_avg_ratings_df that adds the average rating to the movieIds.
3. Transform movie_ids_with_avg_ratings_df DataFrame to another DataFrame, movie_names_with_avg_ratings_df that adds the movie name to each row.
4. Sort movie_names_with_avg_ratings_df into descending order.
5. Create a new DataFrame topRatedMovies_df from movie_names_with_avg_ratings_df where count is greater than threshold value.
6. End

4.3 Split Dataset

1. Begin
2. Split up the ratings_df dataset into three pieces using randomSplit.
   2.1 A training set (DataFrame) as training_df, used to train models.
   2.2 A validation set (DataFrame) as validation_df, used to choose the best model.
   2.3 A test set (DataFrame) as test_df, used for our experiments.
3. End

4.4 Alternating Least Square

1. Begin
2. Use rank which is the number of columns in the Users matrix or the number of rows in the Movies matrix, as parameter to control the model creation process.

3. Create an RMSE evaluator using the label and predicted columns.
4. We will train models with ranks of 4, 8, and 12 using the training_df dataset.
5. Set the appropriate parameters on the ALS object
   4.1 Set the "User" column to the values in our userId DataFrame column.
   4.2 Set the "Item" column to the values in our movieId DataFrame column.
   4.3 Set the "Rating" column to the values in our rating DataFrame column.
   4.4 Use regularization parameter of 0.1.
6. Repeat Step 6 for each rank
   6.1 Set the rank
   6.2 Create the model with this parameters.
   6.3 Run the model to create a prediction. Predict against the validation_df.
   6.4 Remove NaN values from prediction (due to SPARK-14489), store result in predicted_ratings_df DataFrame.
   6.5 Run the RMSE evaluator, reg_eval, on the predicted_ratings_df DataFrame.
   6.6 Check the error.
   6.7 Keep the model with the lowest error, set the rank as best_rank.
7. End

4.5 Read Movie Ratings

1. Begin
2. Read movie ratings given by user from UI.
3. From UI pass the formatted data to controller through Ajax calls.
4. Controller passes the data to appropriate methods in Java.
5. The method stores the data into ratings database table.
6. Return success/fail code from controller to UI.
7. Print alert message according to the return code.
8. End

4.6 Get Recommendation

1. Begin
2. Read movie ratings for different users using Read Movie Ratings [4.5].

3. Create a new my_unrated_movies_df DataFrame which holds the unrated movies by a user.

4. Load data to my_unrated_movies_df using filter.
5. Using ALS (Alternating Least Square) [4.4], find predicted ratings for the movies user did not manually rate.
6. Store the result of step 5 in predicted_ratings_df DataFrame.
7. Join predicted_ratings_df DataFrame with the movie_names_with_avg_ratings_df DataFrame as predicted_with_counts_df DataFrame
8. Use predicted_with_counts_df DataFrame to obtain the ratings counts for each movie.
9. Filter predicted_with_counts_df DataFrame where ratings count>=50 and prediction>=3.0
10. Add the result of step 9 to a list of prediction and return.
11. End

4.7 Test Model

1. Begin
2. Use the best_rank determined in ALS (Alternating Least Square) [4.4] to create a model
3. Run a prediction, using my_model as created above, on the test dataset (test_df), producing a new predict_df DataFrame.
4. Filter out unwanted NaN values (necessary because of a bug in Spark, SPARK-14489).
5. Use the previously created RMSE evaluator, reg_eval to evaluate the filtered DataFrame.
6. Check RMSE (Root Mean Square Error) for the results predicted by the model versus the values in the test set.
7. End

RESULTS AND DISCUSSIONS

5.1 User SignUp and Login

5.1.1 SignUp

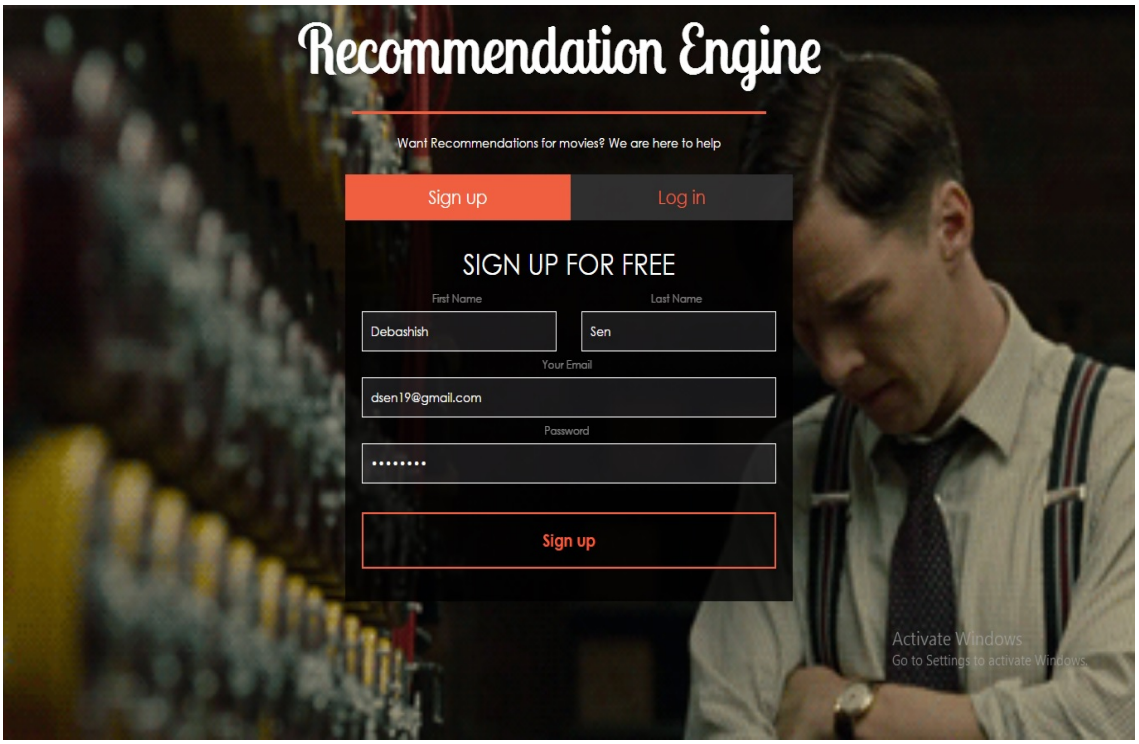


Figure No. 11: SignUp Page

- Read user details.
- Validation done and stored in database.

5.1.2 Login

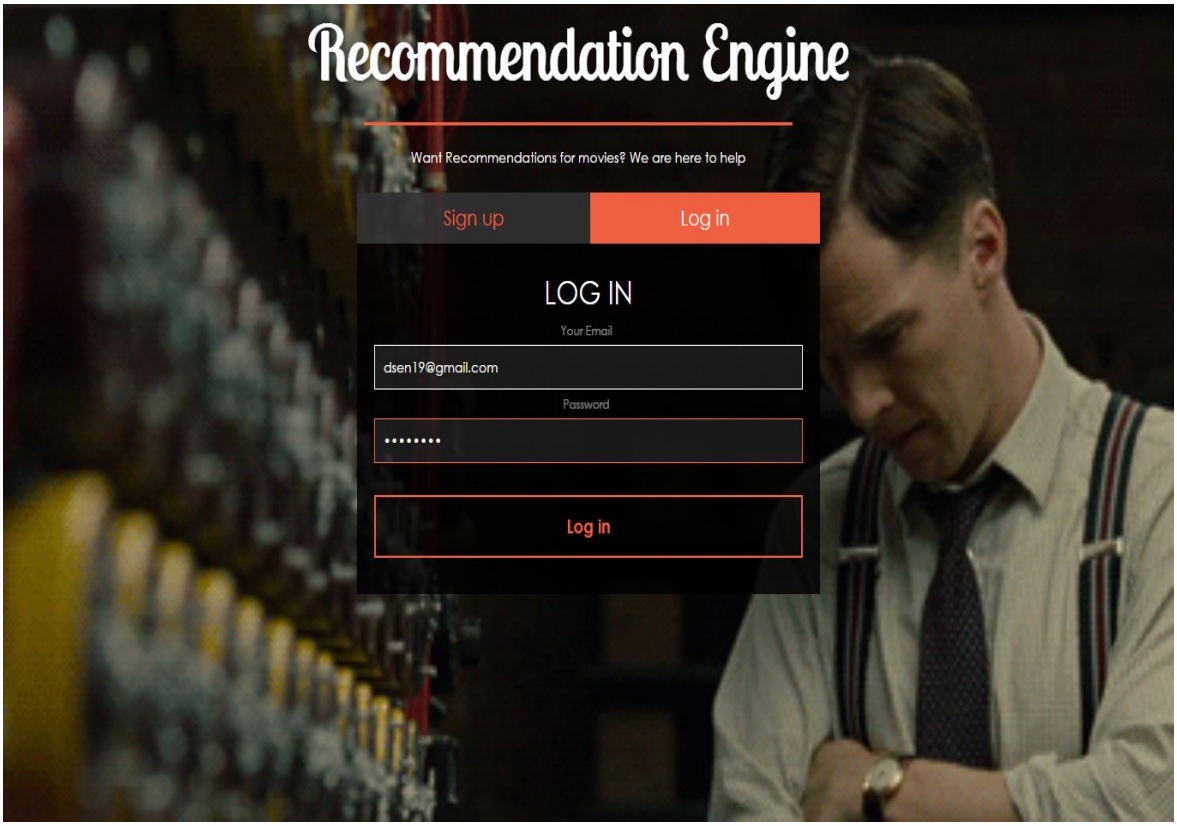


Figure No. 12: Login Page

- Read User Email ID and Password.
- If User Email ID and Password matches redirect to home else prompt user to retry.

5.2 Home

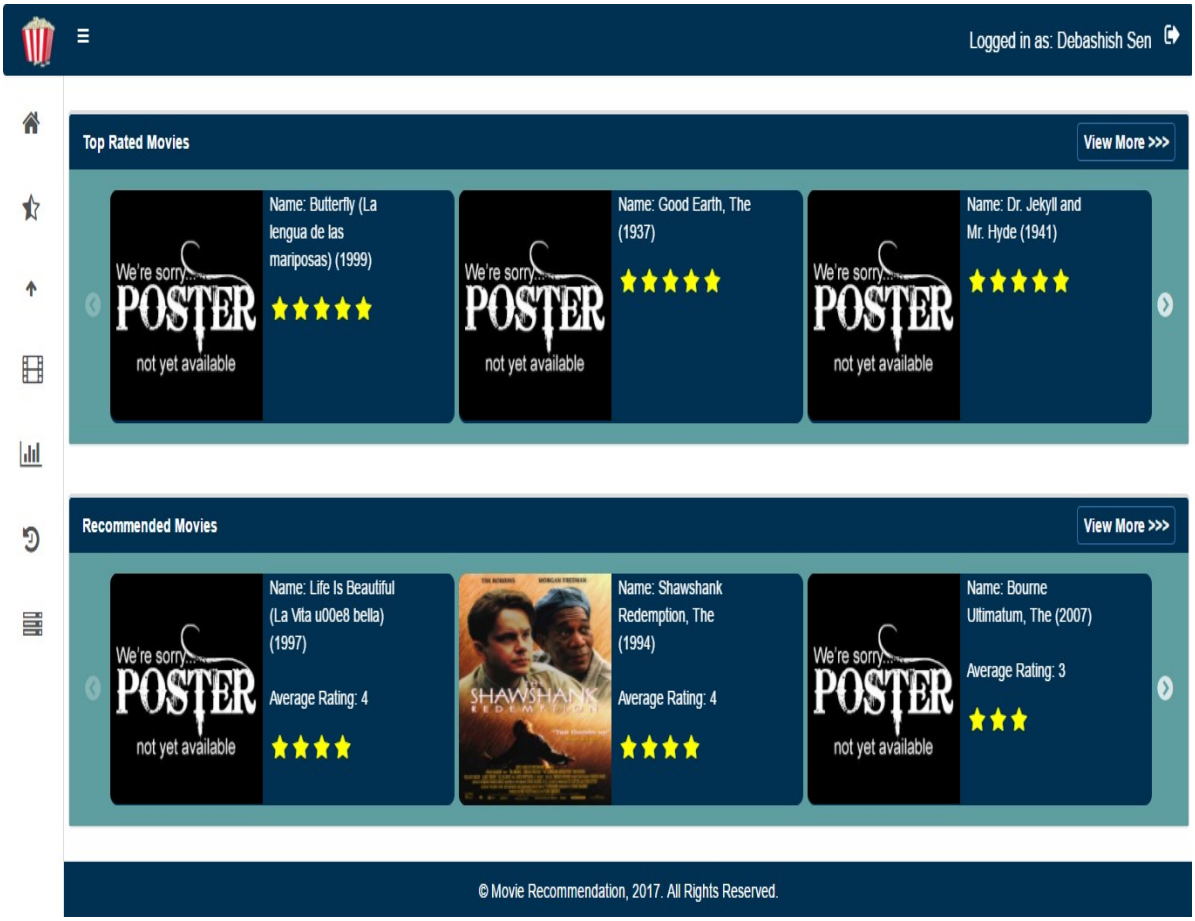


Figure No. 13: Home Page

- Top 10 Movies with respect to average rating is shown.
- Top 10 Recommended Movies shown with respect to prediction.
- By Clicking view more, user can see genre wise grouped movies respectively.

5.3 Rate Movies

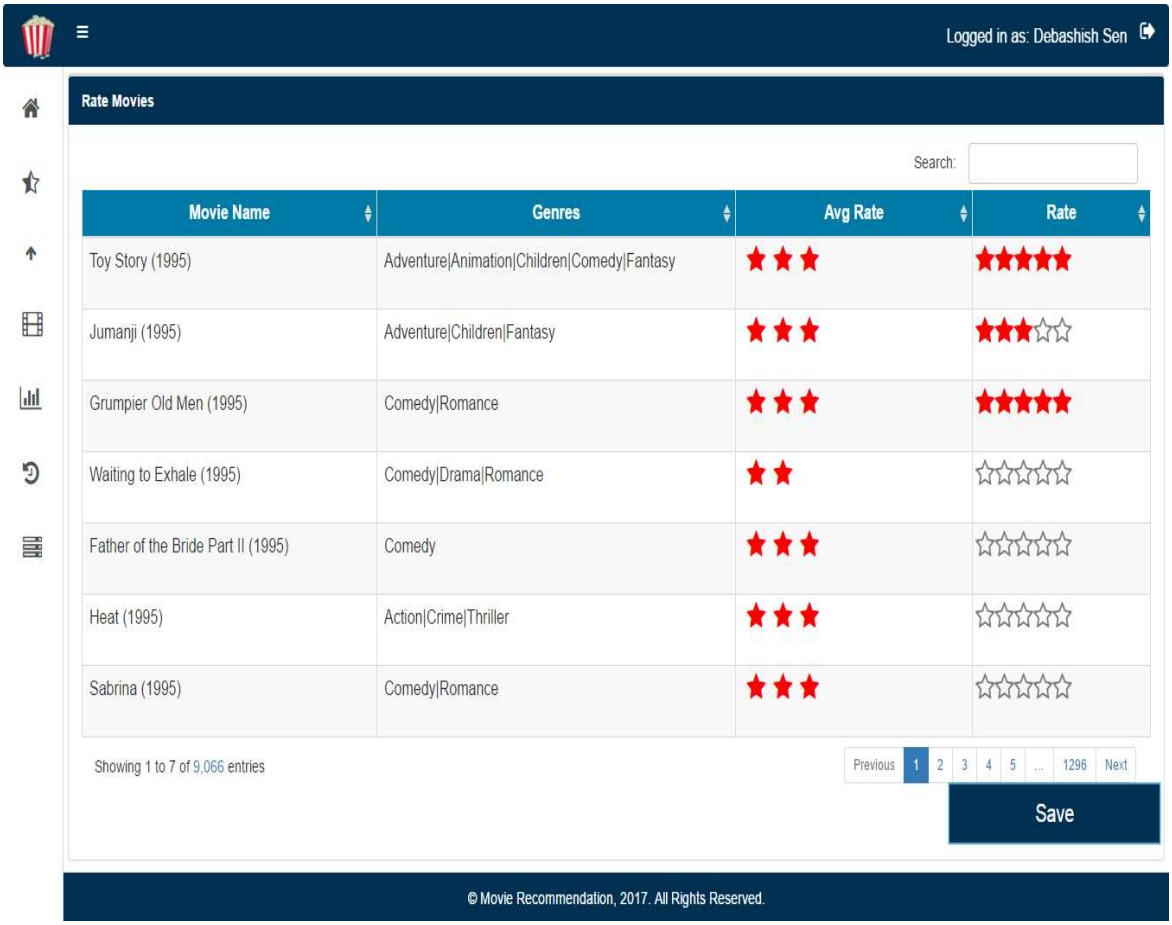


Figure No. 14: Rate Movies Page

- All the movies are listed here.
- User can rate movies according to his/her preferences, taste, and interest.
- By clicking 'Save' button, new entry will be created in the rating table if the movie is rated for the first time. Else previous rating will be updated with new rating for respective movies.

5.4 Top Rated Movies

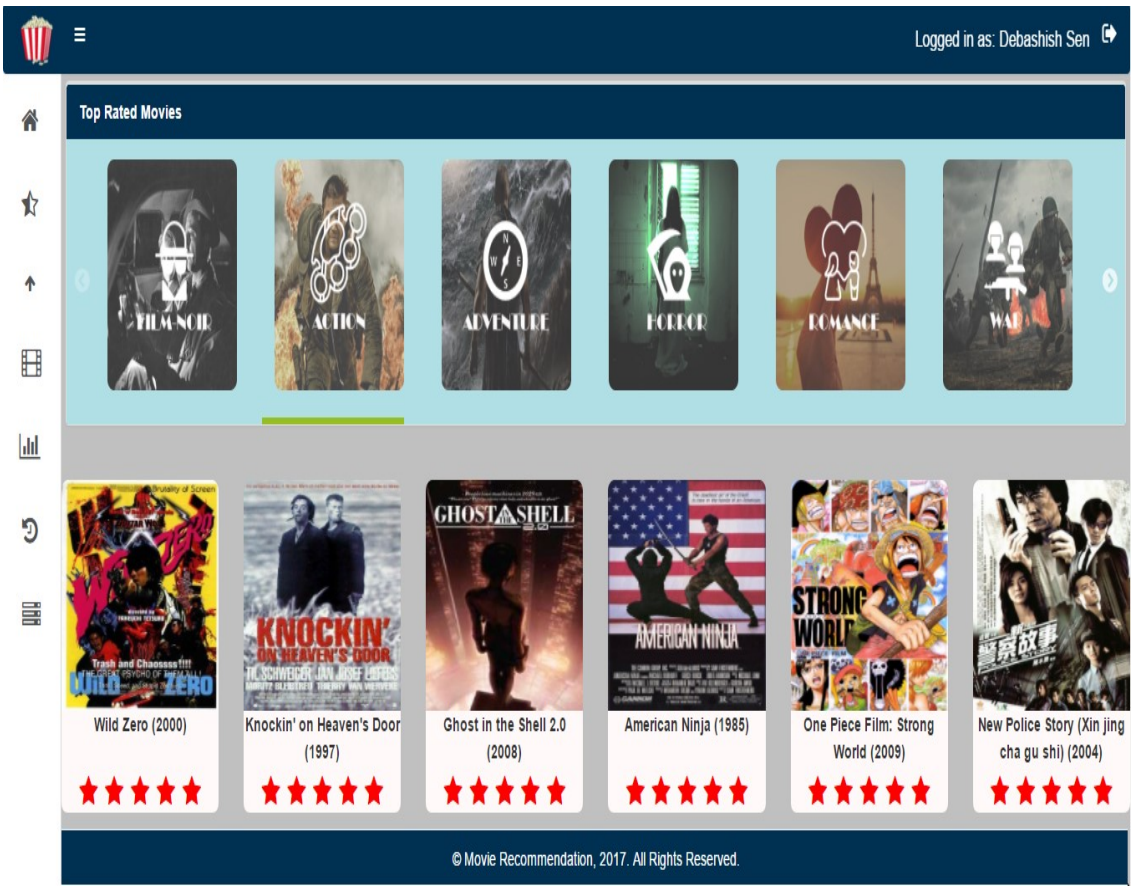


Figure No. 15: Top Rated Movies Page

- Top Rated Movies with respect to average ratings is show here.
- Genre wise all top rated movies are grouped.
- On click of any genre, top rated movies of that particular genre will be shown.

5.5 Recommended Movies

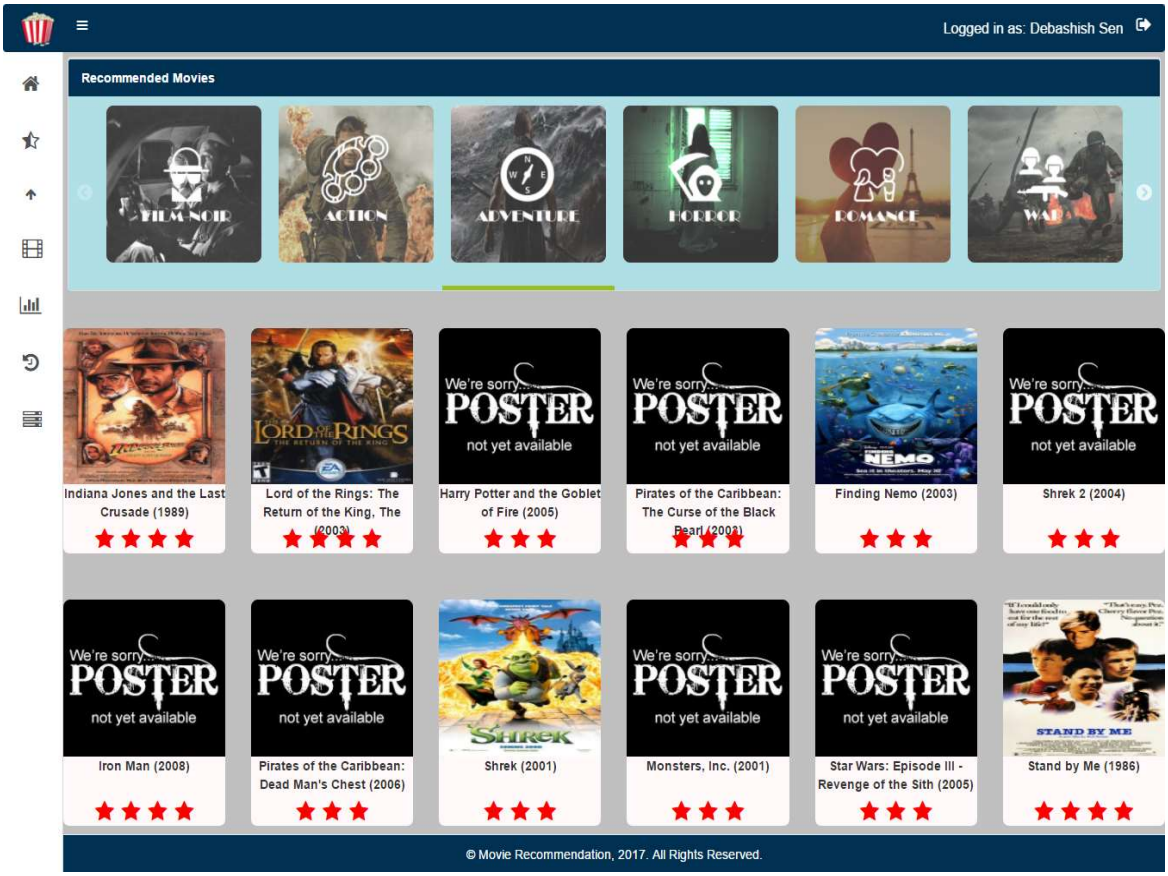


Figure No. 16: Recommended Movies Page

- Recommended Movies with respect to prediction is shown here.
- Genre wise all the recommended movies are grouped.
- On click of any genre, recommended movies of that particular genre will be shown.

5.6 Analysis

5.6.1 Top Rated Movies with Count

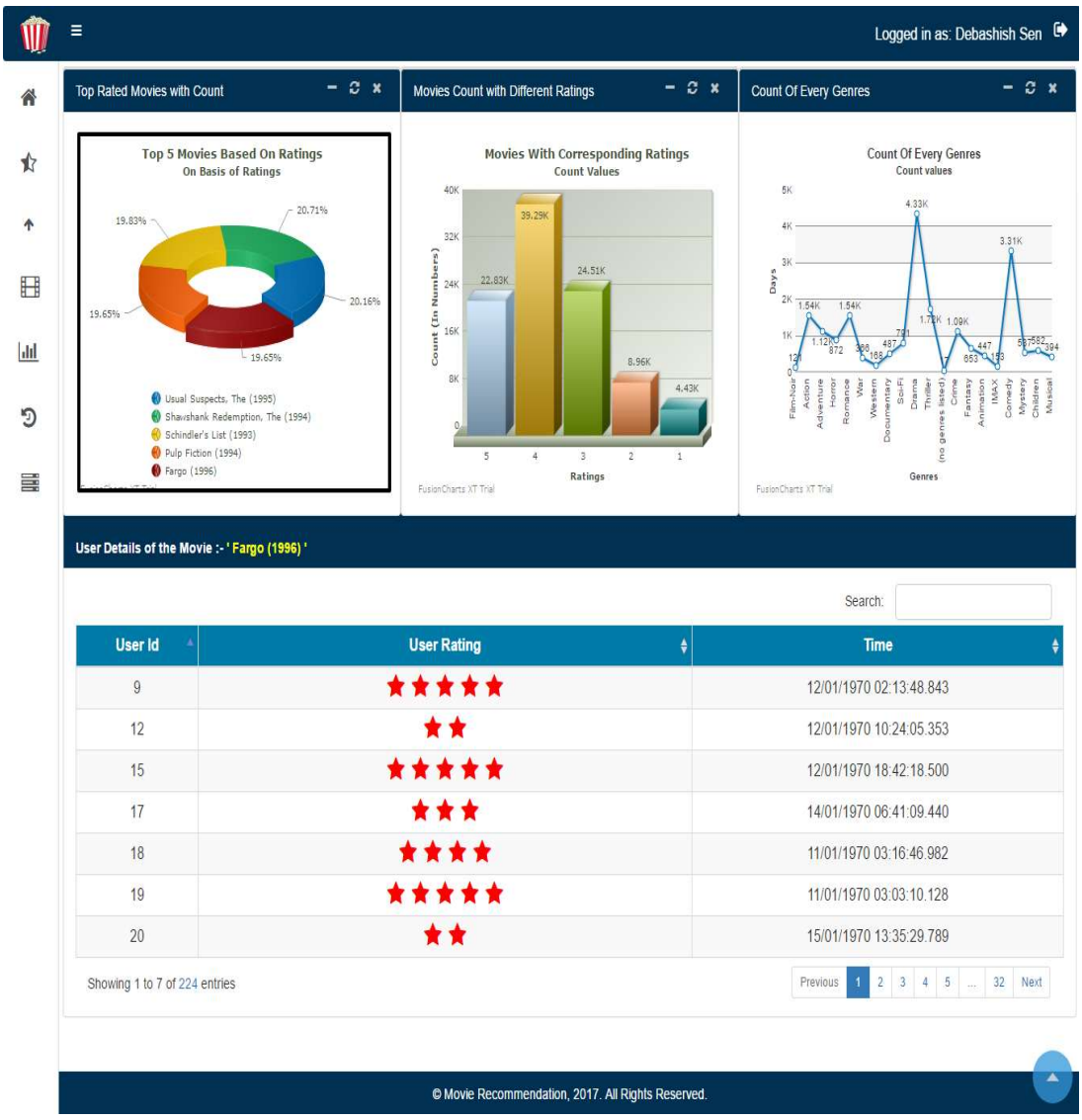


Figure No.17: Analysis Page (Top Rated Movies with Count)

- Top 5 rated movies are show by using a pie chart, where each sector represents a movie.
- User may click any of the sector, to view who has rated, how much rated and when rated.

5.6.2 Movies with corresponding ratings

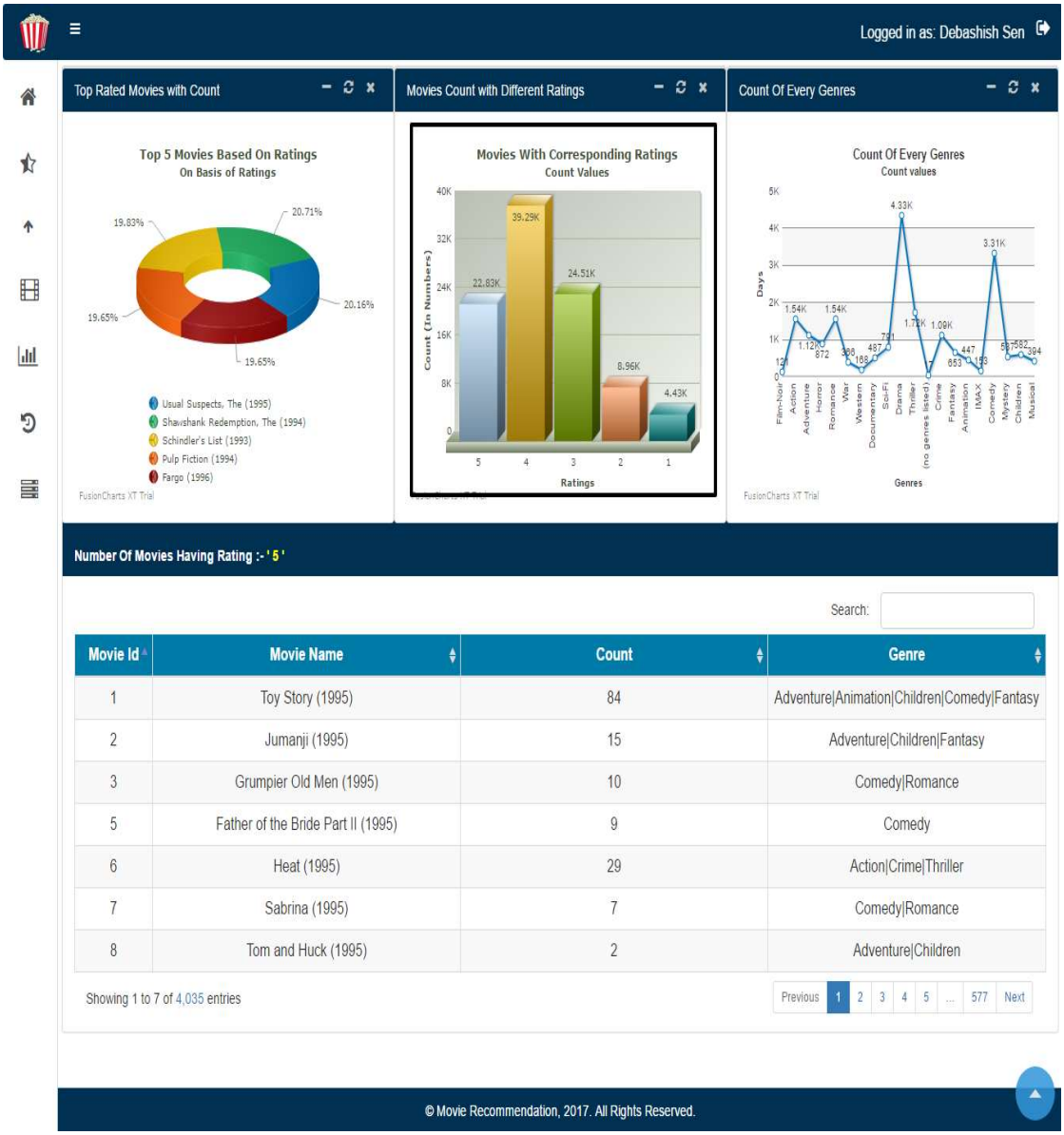


Figure No.18: Analysis Page (Movies with corresponding ratings)

- Using bar graph, movies are grouped into 5 different rating given by users.
- On click of any bar, movie details along with count of how many users have rated will be shown correspondingly.

5.6.3 Count of Movies in each Genre

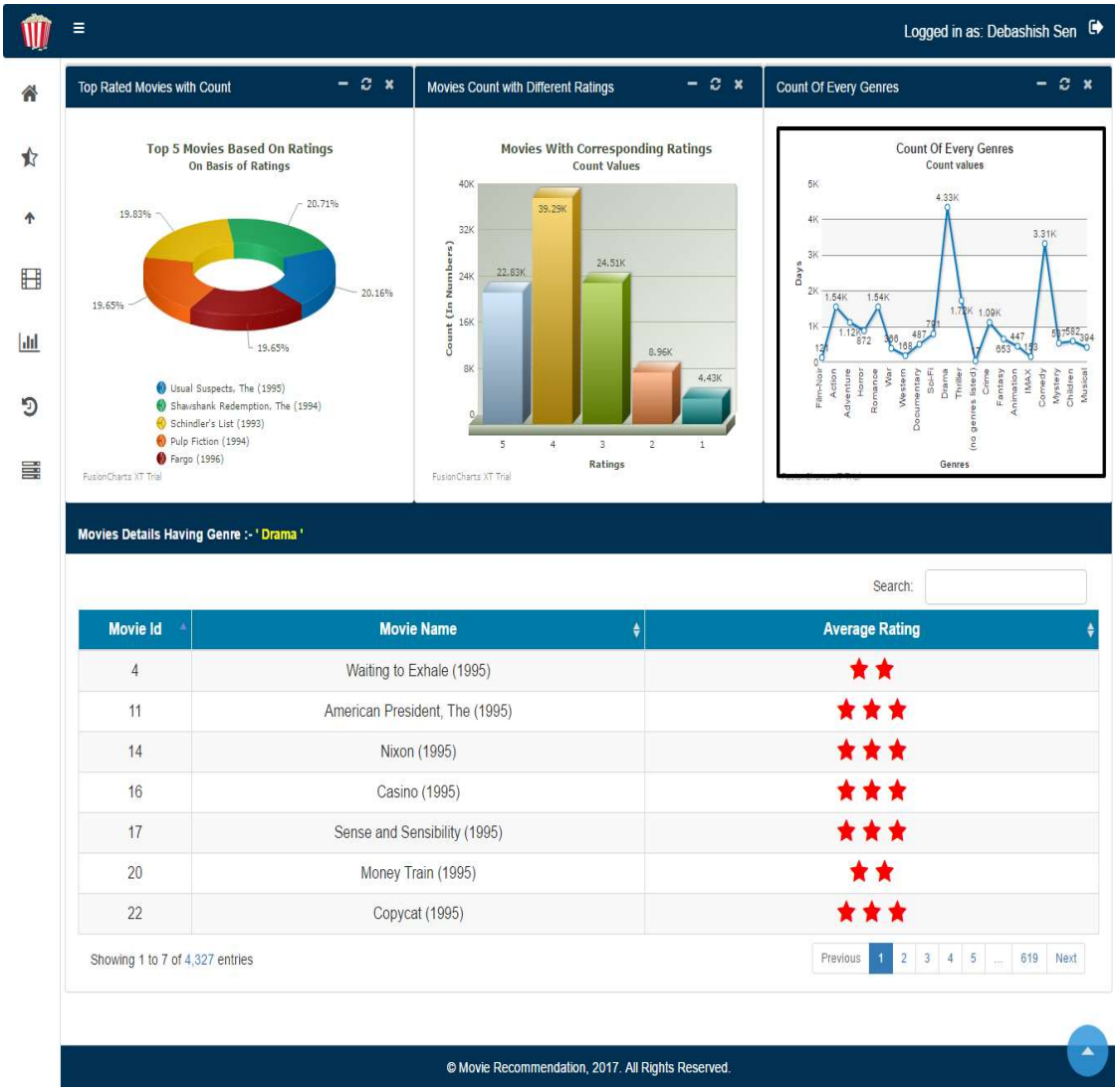


Figure No.19: Analysis Page (Count of Movies in each Genre)

- Using line graph, relationship between genres and ratings is depicted.
- On click, user may view all movies with average ratings of particular genre.

5.7 History

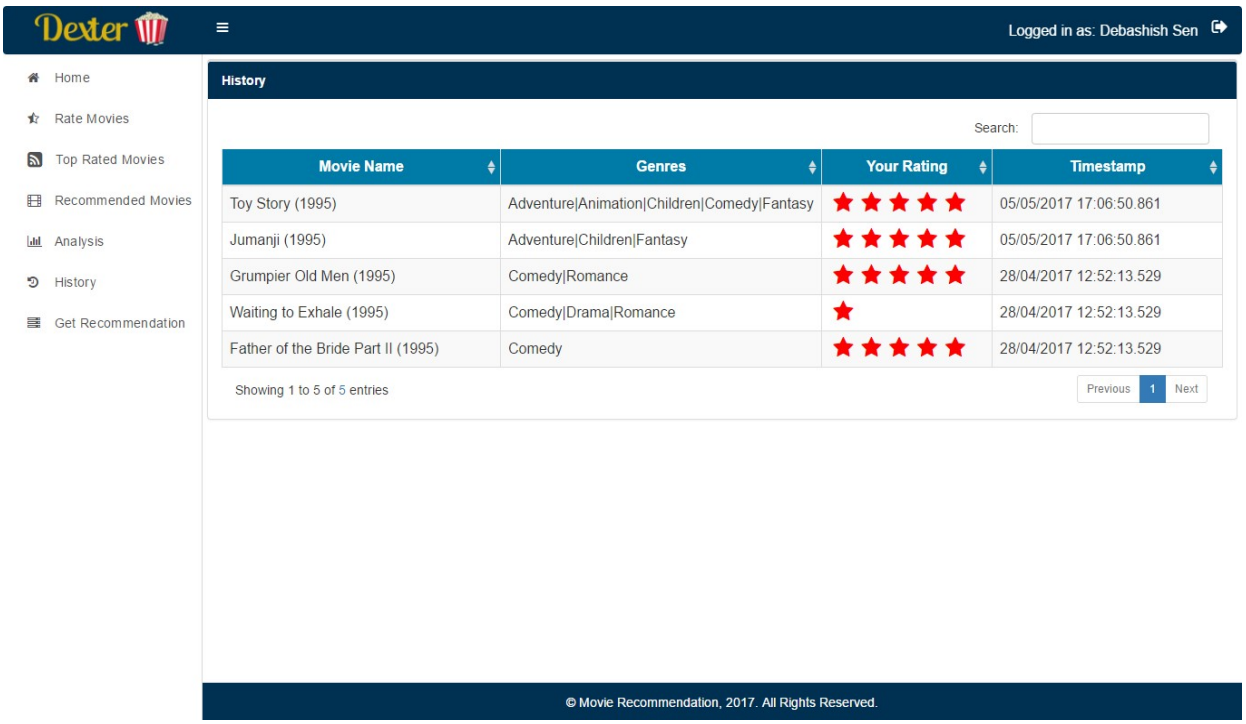


Figure No.20: History Page

- User history is shown here.
- Which movies have been rated, how much rated and when rated can be viewed.

5.8 Get Recommendation

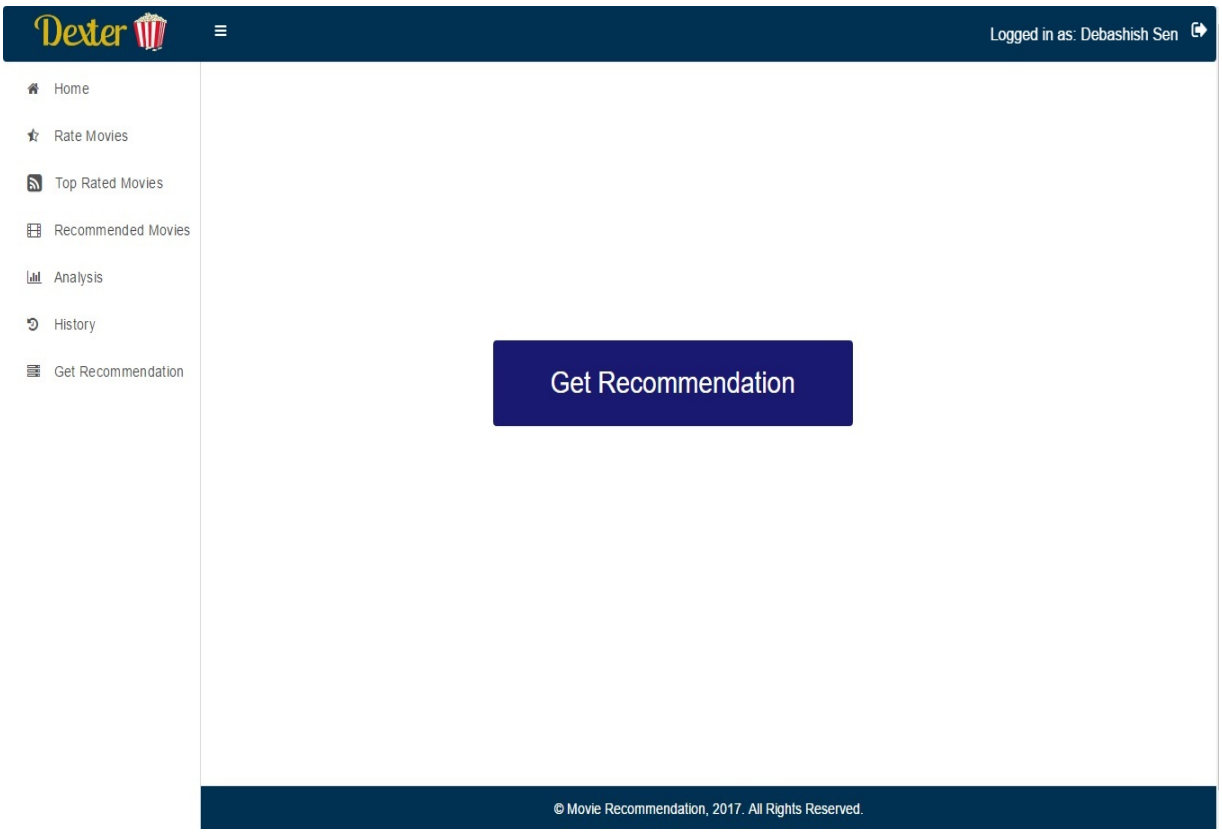


Figure No.21: Get Recommendation Page

- On click of Get Recommendation button, recommendation table gets populated as per users' current preferences.

SUMMARY AND CONCLUSION

6.1 Summary of Achievements

- Building a prototype recommendation engine.
- Using machine learning approach.
- Distributed processing achieved using Spark.
- Quality recommendation generated using minimized Root Mean Square Error.
- Using MySQL indexes, which provided the basis for both rapid random lookups and efficient ordering of access to records.
- Versatile Engine, i.e., may be used for Cross-Domain Recommendation with some changes.

6.2 Main difficulties encountered and how they were tackled

- **Lack of Data:** Recommendation Engine needs a lot of data to effectively make recommendations. It's no coincidence that the companies most identified with having excellent recommendations are those with a lot of consumer user data: Google, Amazon, Netflix, Last.fm. We collected datasets from GroupLens. GroupLens Research provides the data sets collected from MovieLens website for research use.
- **Changing User Preferences:** User preferences and taste may vary with time. Then recommendations also should change accordingly. Therefore we have a provision to re-rate items which a user had previously rated and generate new recommendations as per current preferences.

6.3 Limitation of the project

- **Cold Start Problem**: It concerns the issue that the system cannot draw any inferences for users about which it has not yet gathered sufficient information.
- The "long tail" Issue: To recommend less known / unknown items and avoid the bias of recommending popular content only.

6.4 Future Scope of work

- To use hybrid filtering technique, this is combination of both content filtering technique and collaborative filtering technique.
- To build a mobile application for android users.

GANTT CHART

ACTIVITY	TIME FRAME				
	9th-31st Jan. 2017	1st-28th Feb. 2017	1st-31st March 2017	3rd-28th April 2017	2nd-8th May 2017
FEASIBILITY STUDY					
LITERATURE SURVEY					
DESIGN AND CODING					
IMPLEMENTATION					
TESTING					
DOCUMENTATION					

PROPOSED ACTIVITY
ACTIVITY ACHIEVED
ACTIVITY TO BE IMPLEMENTED

REFERENCES

[1] Gediminas Adomavicius and Alexander Tuzhilin. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. on Knowl. and Data Eng.*, 17(6):734–749, 2005.

[2] Marko Balabanovic and Yoav Shoham. Fab: Content-based, collaborative recommendation. *Communications of the Association for Computing Machinery*, 40(3):66–72, 1997.

[3] C. Basu, H. Hirsh, and W. Cohen. Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98)*, pages 714–720, July 1998.

[4] Daniel Billsus and Michael J. Pazzani. Learning collaborative information filters. In *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, pages 46–54, Madison, WI, 1998. Morgan Kaufmann.

[5] John S. Breese, David Heckerman, and Carl Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, Madison, WI, July 1998.

[6] Robin Burke, Bamshad Mobasher, Runa Bhaumik, and Chad Williams. Segment-based injection attacks against collaborative filtering recommender systems. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 577–580, Washington, DC, USA, 2005. IEEE Computer Society.

[7] M. Claypool, A. Gokhale, and T. Miranda. Combining content-based and collaborative filters in an online newspaper. In *Proceedings of the SIGIR-99 Workshop on Recommender Systems: Algorithms and Evaluation*, 1999.

[8] Jonathan L. Herlocker, Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. Evaluating collaborative filtering recommender systems. *ACM Trans. Inf. Syst.*, 22(1):5–53, 2004.

[9] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the Association of Computing Machinery*, 35(12):61–70, 1992.

[10] Raymond J. Mooney and Loriene Roy. Content-based book recommending using learning for text categorization. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 195–204, San Antonio, TX, June 2000.

[11] Abhay S. Harpale and Yiming Yang. Personalized active learning for collaborative filtering. In *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 91–98, New York, NY, USA, 2008. ACM.