## CMPE 282 CLOUD SERVICES

## Fall 2014



**Project Report**

***Project: Music Recommendation System***

Submitted By:

Avi Jain<SJSU ID: 009425806>

Eshan Haval <SJSU ID: 009390875>

Jasleen Kaur<SJSU ID: 009430304>

Raghav Munjal<SJSU ID: 008951605>

Sagar Ruchandani<SJSU ID: 009428731>

Shubham Rajvanshi<SJSU ID: 009428744>

Submitted To:

Prof. Simon Shim

**Contents**

1. INTRODUCTION

1.1 GOALS

1.2 OBJECTIVES

2. REQUIREMENTS

2.1. FUNCTIONAL REQUIREMENTS

2.2 NON FUNCTIONAL REQUIREMENTS

3. API DESIGN

4. Architecture and Functions

5.1. ARCHITECTURE

5.2 FUNCTIONS

5.Analysis Algorithm

6.Workflow

7 .IMPLEMENTATION

7.1. ENVIRONMENT

7.2. TOOLS AND TECHNOLOGIES

7.3. ScreenShots

8. INDIVIDUAL CONTRIBUTION

9. FUTURE WORK

10. CONCLUSION

11. REFERENCES

**Recommendation System:**

User modeling, adaptation, and personalization techniques have hit the mainstream. The explosion of social network websites, on-line user-generated content platforms, and the tremendous growth in computational power of mobile devices are generating incredibly large amounts of user data, and an increasing desire of users to "personalize" (their desktop, e-mail, news site, phone).

The potential value of personalization has become clear both as a commodity for the benefit or enjoyment of end-users, and as an enabler of new or better services –a strategic opportunity to enhance and expand businesses.

An exciting characteristic of recommender systems is that they draw the interest of industry and businesses while posing very interesting research and scientific challenges.

In spite of significant progress in the research community, and industry efforts to bring the benefits of new techniques to end-users, there are still important gaps that make personalization and adaptation difficult for users. Research activities still often focus on narrow problems, such as incremental accuracy improvements of current techniques, sometimes with ideal hypotheses, or tend to overspecialize on a few applicative problems (typically TV or movie recommenders –sometimes simply because of the availability of data). This restrains de facto the range of other applications where personalization technologies might be useful as well.

Thus, we may have reached a good point to take a step back to seek perspective in the research done in recommender systems. Here we are creating the MUSIC recommendation system.

**Music Recommendation System:**

# 1. Introduction:

In a recommender system, users provide ratings for items like movies or songs, and the system computes personalized recommendations for each user. Web-based recommender systems are widely used to deliver both personalized content and personalized advertising at popular internet portals like Yahoo!, and popular online retailers like Aamazon or Blockbuster. Most users rate a small fraction of the available items in a recommender system, leading to massive amounts of missing data. In this Yahoo! Music dataset, we explored how users choose the items they rate in a web portal by conducting a novel study of user rating behaviour in Yahoo! **Music’s Web Scope C15** recommender system. The results of this study show a strong link between user preferences and the songs a user chooses to rate. We show that the presence of this link invalidates the assumptions of previous rating prediction methods. We introduce new prediction methods that explicitly take into account a simple model of the relationship between a user’s underlying preferences and the choice to rate a given song, leading to significant improvements in predictive accuracy.

This system is a discovery engine that provides on spot, relevant, music recommendations, based on one's existing preferences and yahoo dataset for previous music reviews. The purpose of these recommendations is discovery and taste exploration. Sometimes, less known items are recommended instead of more similar, yet much more popular ones, in order to increase the chances of discovering something new. In this work we describe a detailed analysis of a sparse, large scale dataset, specifically designed to push the envelope of recommender system models. The Yahoo! Music dataset consists of more than a million users, 600 thousand musical items and more than 250 million ratings, collected over a decade. It is characterized by three unique features: First, rated items are multi-typed, including tracks, albums, artists and genres. We further present a matrix factorization model exploiting the special characteristics of this dataset. In particular, the model incorporates a rich bias model with terms that capture information from the taxonomy of items and different temporal dynamics of music ratings

Our enterprise idea is to utilize this rich collection of unstructured data, build a recommendation system using some analysis that would recommend the user of this system various music recommendations based on their personal data input. The recommendation system would give knowledge about the factors to be considered like previous reviews and latest rating and recommend based on that data.

## 1.1 Goals

* Design and implement a cloud service (using REST) that contains data analysis (Using hadoop, Mahout)
* To create a marketable could service.
* Use configuration management tool such as OpsWork in order to manage infrastructure (AWS).
* Include provisioning and configuration to orchestration and reporting.
* Use RDS and EC2 when implementing the project at AWS.

## 1.2 Objectives

* Understanding the need of analyzing large pool of data collected from various resources.
* Apply the analysis algorithms and do analysis mapping to reorganize data and come up with recommendation for user depending upon his/her music data set.
* Using RDS database to store the data and deploy the application using EC2 instance and proactively manage change and implement auto-scaling from 1 servers to multiple (>2) servers in the cloud.

# 2. Requirements

## 2.1 Functional Requirements

* Collect large pool raw music data of Yahoo Labs Web Scope C15 into HDFS on EC2.
* Process this raw data in Mahout and input the meaningful data to HDFS on EC2.
* With the help of Groovy script, submit this meaningful data to RDS on another EC2 instance.
* Also maintain transactional data on RDS.
* Using this data, deploy the application on AWS EC2, which would be directly accessible by the user.
* Recommend the music based on his profile

## 2.2 Non-Functional Requirements

* Build a working prototype system.
* The system should be scalable for future requirements.
* System should handle heavy workloads.

# 3. API Design

* Application and AWS RDS

REST web service is integrated with RDS instance of MySQL on AWS. This API provides basic functionalities like user Register, login, Profile Management, search, review recommendations and logout.

* HDFS to Mahout

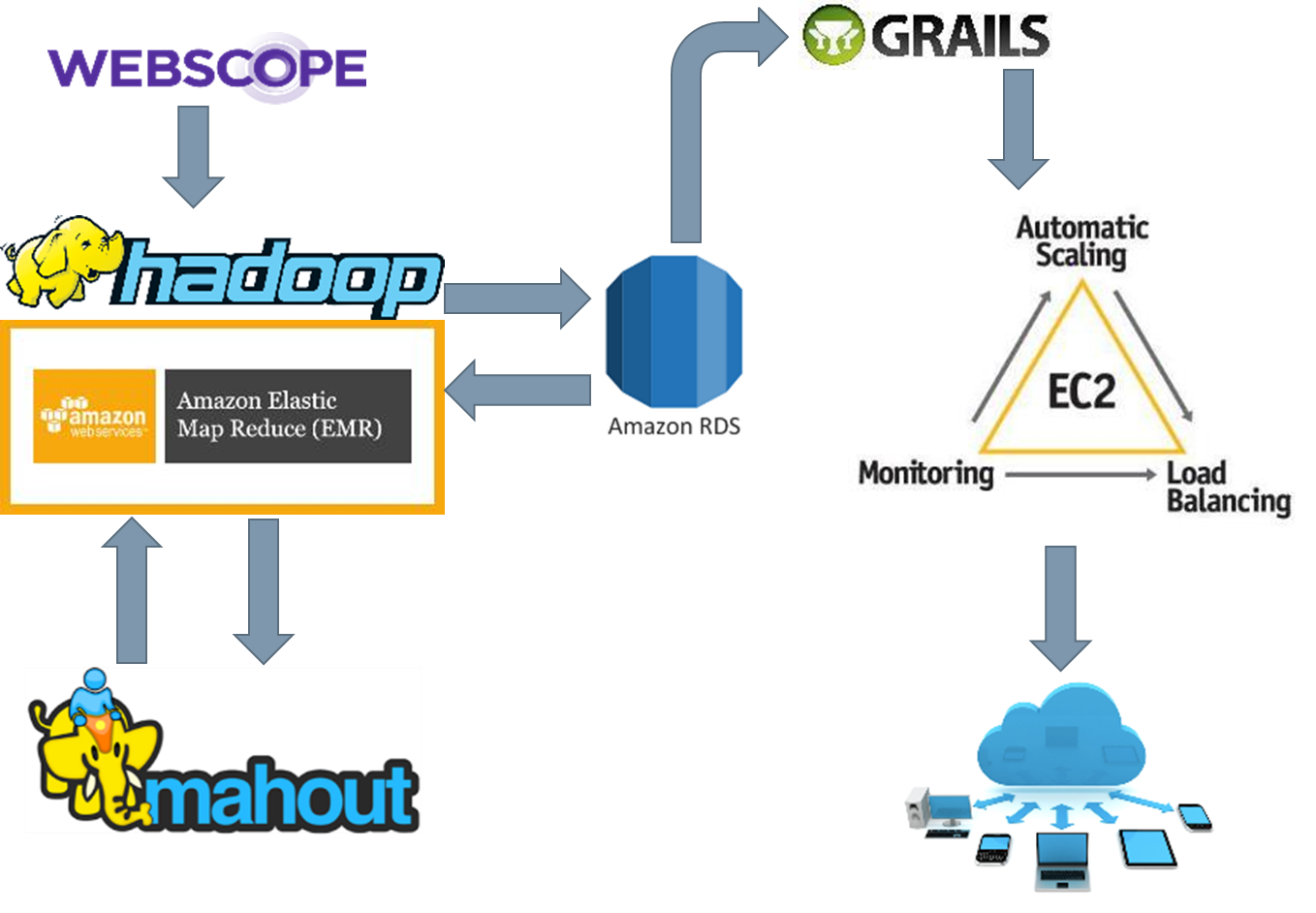
Data set is stored in HDFS initially. Mahout provides a suite of machine learning libraries designed to be scalable and robust Data will be processed from Mahout to Apache Hadoop. Map reduce functionality is being executed with in Hadoop and the required analysis is obtained from using HDFS.

* Groovy to RDS

Upon successful data analysis and results are stored back in HDFS from Mahout, an efficient groovy script collects these results and pushes it onto RDS instance on AWS. This is the main functionality of this API that provides and essential module in system.

# 4. Architecture and Functions

## 4.1 Architecture



**Architecture Diagram**

**4.2 Functions**

* User SignUp while registration.
* User login upon signing up.
* Maintain profile.
* Search the music.
* Look out for recommendations.
* Data set processing.
* Performing data analysis.

# 5. Analysis Algorithm

Recommendations provided in system are based on two sets of data. One is user data which will bought from music ecommerce and another is data set from Yahoo Labs Web Scope C15. Algorithm used to perform the analysis on these dataset is Log Likely Hood similarity.

Two users are similar when they rate or are associated to many of the same items. However, a certain overlap may or may not be meaningful -- it could be due to chance, or due to the fact that we have similar tastes. For example if two persons have rated 100 items each, and 50 overlap, they are probably similar. But if they each rate 1000 and overlap in only 50, maybe they are not. The log-likelihood metric is just trying to formally quantify how unlikely it is that our overlap is due to chance. The less likely, the more similar they are. So it is comparing two likelihoods, and just looking at their ratio. The numerator likelihood is the null hypothesis: they are not similar and overlap is due to chance. The denominator is the likelihood that it's not at all due to chance -- that the overlap is completely explained is perfectly explained because their tastes are similar and the overlap is exactly what you'd expect given that. When the numerator is relatively small, the null hypothesis is relatively unlikely, so we are similar. The reason the formulation typically then takes -2.0 \* log (likelihood ratio) is by convention, and it makes the result a bit more useful in two ways. One, more similarity will equal a higher log-likelihood, which is perhaps more intuitive than the likelihood ratio which is lower when similarity is higher. But the real reason is that the log-likelihood value then follows a chi-squared distribution and the result can be used to actually figure a probability that the users are similar or not. (We don't use it that way in Mahout though.) And Ted' formulation, which is also right and quite tidy and the one used in the project, is based on Shannon entropy It is, similarly, trying to figure out whether the co-occurrences are "unusually" frequent by asking whether there is any additional information to be gained by looking at user 1 and user 2's preferences separately versus everything at once. If there is, then there is something specially related about user 1 and user 2 and they're similar.

# 6. Workflow

User initially needs to register using Signup module. Upon successful signup, user will be prompted to login. Then user will land in homepage where he can search for music using the search tab also recommendation will be given to user done his previous selections he/she had done. Also in other case when user login for first time will get recommended music based on most popular songs. Data between client and service is transferred using JSON format. Recommendations are being displayed in recommendation tab of user profile page. This flow is from the user point of view.

From admin point of view, initially, data set is stored in HDFS on EC2 instance of Amazon web service. Mahout is deployed on the top of HDFS to perform data analysis. New user will get the recommendation from most recommended music with existing data set on HDFS is performed. When a new user will listen to music and rate it then those ratings will be stored back into HDFS. The data is extracted after data analysis using groovy script and stored onto AWS RDS instance. The set of recommendations that are being stored on RDS after analysis is sent to controller in RESTful web service. Thus the controller displays them on recommendation tab of user profile.

# 7. Implementation

## 7.1 Environment

* This prototype project is having quite big environment that consists of RESTful web services built using groovy, data warehouse consisting of data set.
* Hadoop file system is being used for managing the data set. Mahout is used for scalable and robust data set along with map reduce functionality with in hadoop to come up recommendations.
* Different features of Amazon web services like RDS, EC2, are being utilized in building the complete environment.
* Data has been stored on Amazon EC2 instances for processing. The processed data is stored for quicker access and to increase the scalability along with availability.

## 7.2 Tools and Technologies

* **Restful Web Services**

REST stands for Representational State Transfer which is a stateless web based service that is easy to implement and resource oriented model. Data is transferred either in XML or JavaScript Object Orientation (JSON) or in both formats.

* **MySQL/RDS**

MySQL, one of the most popular in relational database systems needs no introduction. An instance of MySQL has been created in Amazon web services RDS console. All the user related data regarding profile and login information has been stored in this database.

* **Apache Mahout**

Apache Mahout is free machine learning library used to build scalable machine learning tools for use on analyzing large data on a distributed manner. It is framework for classification, clustering and recommendation.

* **HDFS**

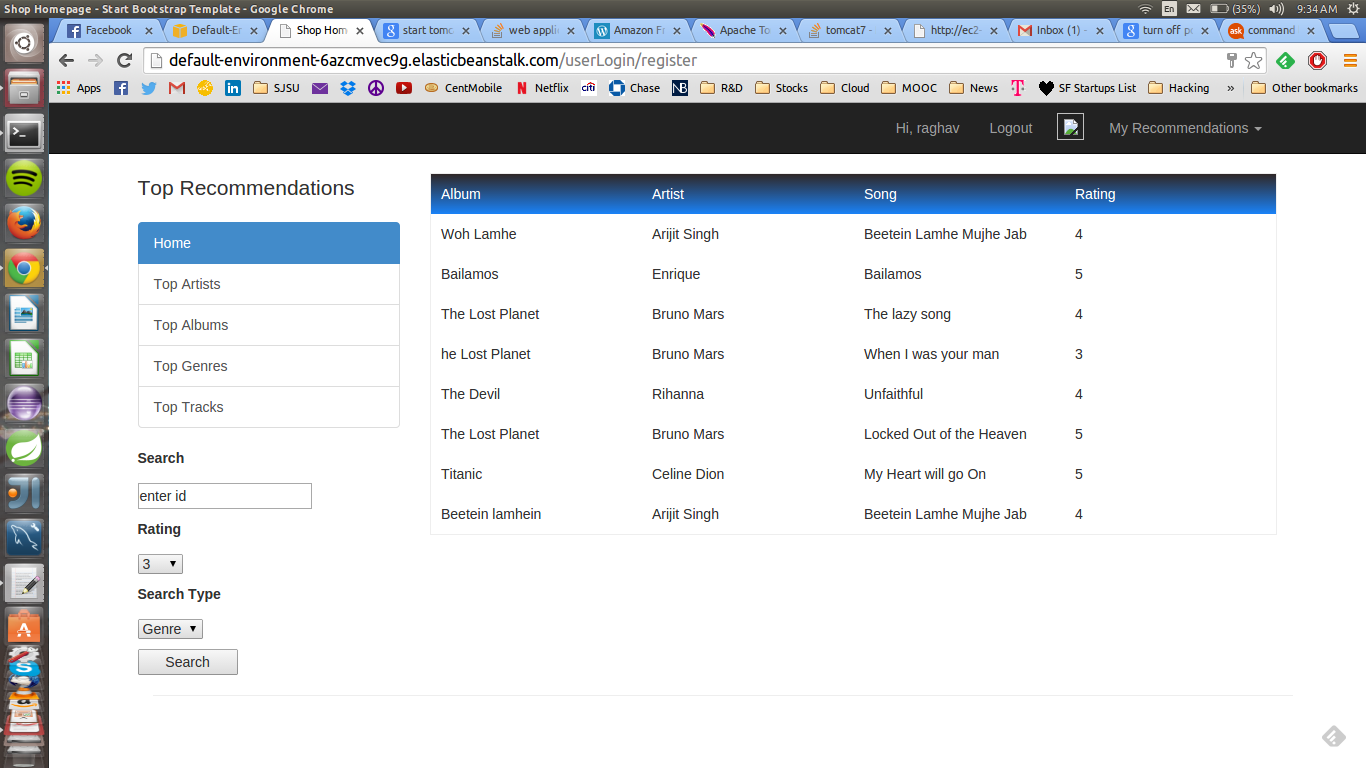
Hadoop Distributed File System is a java- based file system which provides reliable and scalable storage of data, helps in spanning large servers and clusters. It assure about the fault tolerance and computation can be performed on various nodes and clusters parallel.

* **Groovy**

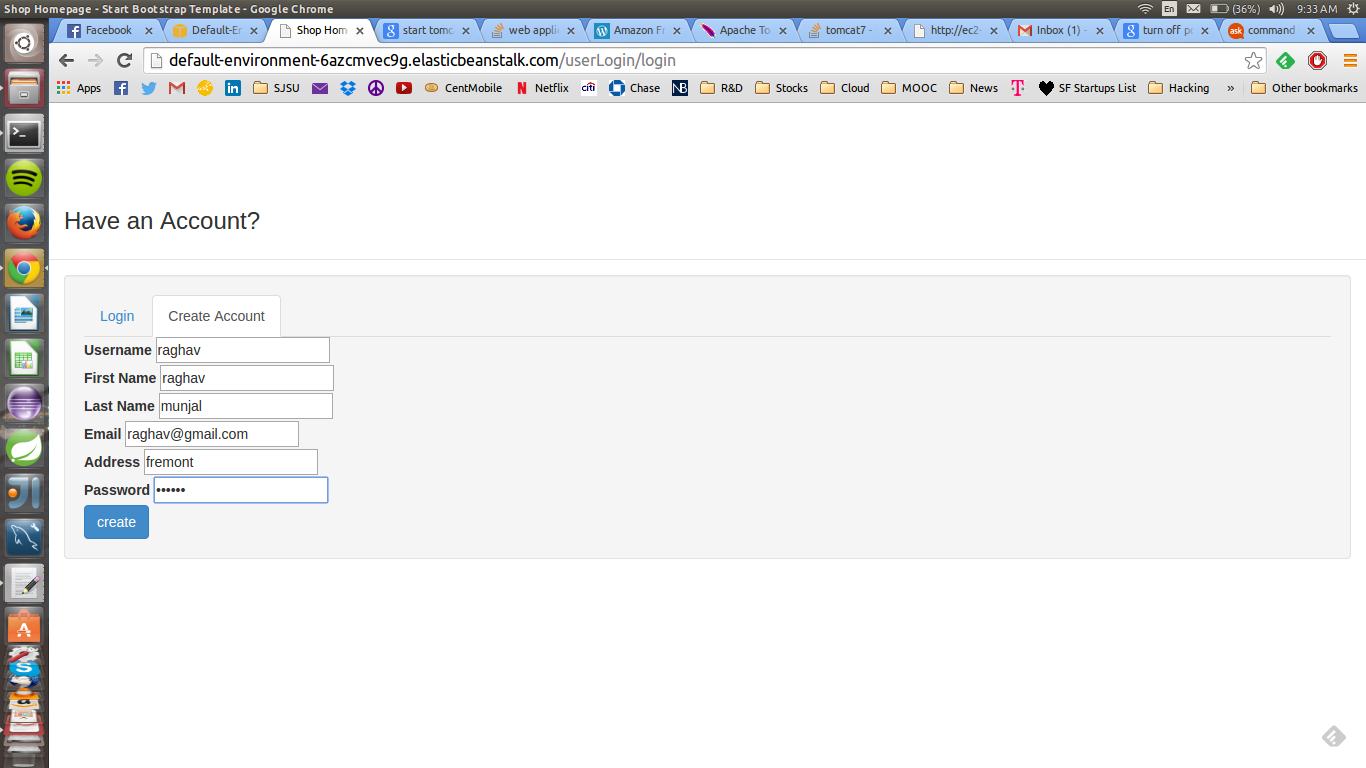
**Groovy** is an object oriented programming language for the Java Platform. It is dynamic language with features similar to those of Python, Ruby, Perl and Smalltalk. It can be used as a scripting language for the Java Platform, is dynamically compiled to Java Virtual Machine JVM byte code and interoperates with other Java code and library Groovy uses a Java-like curly bracket syntax. Most Java code is also syntactically valid in Groovy.

## 7.3 Screenshots

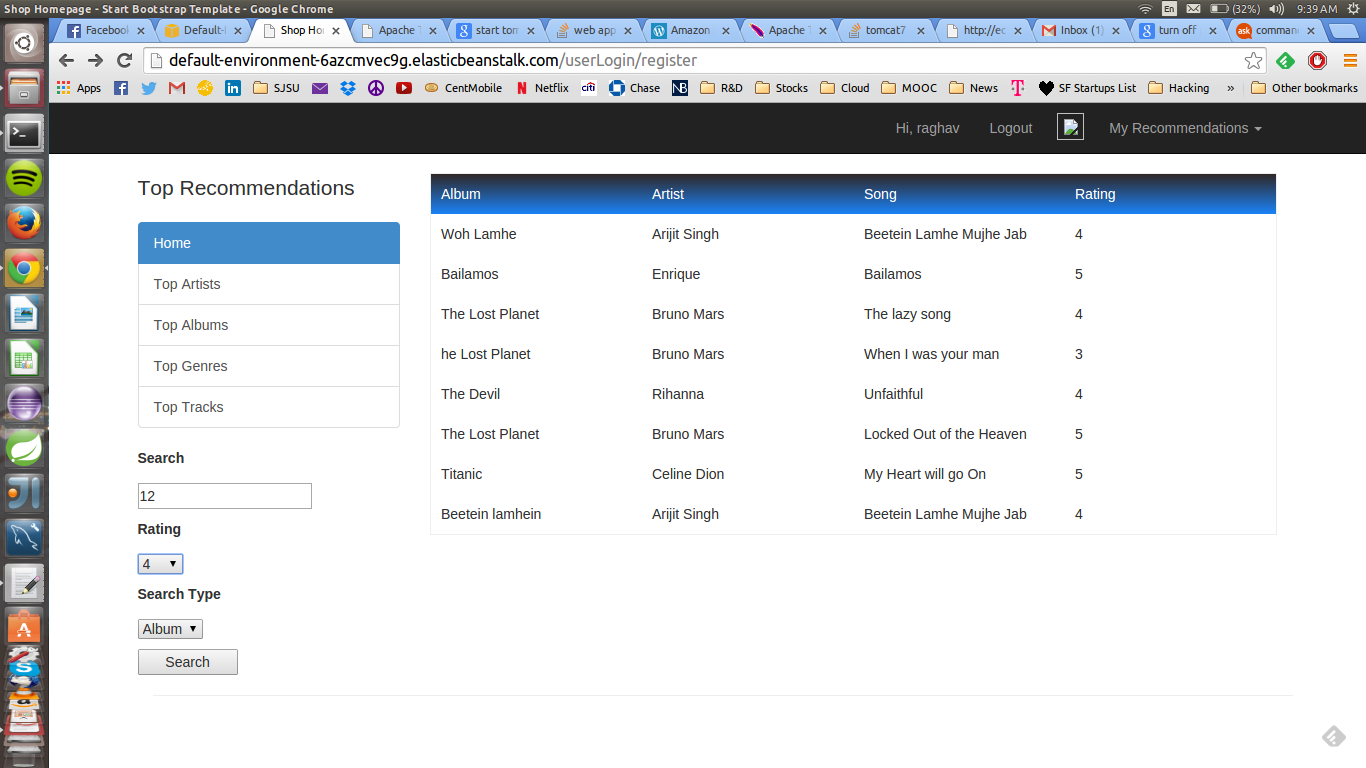
1. Home Page



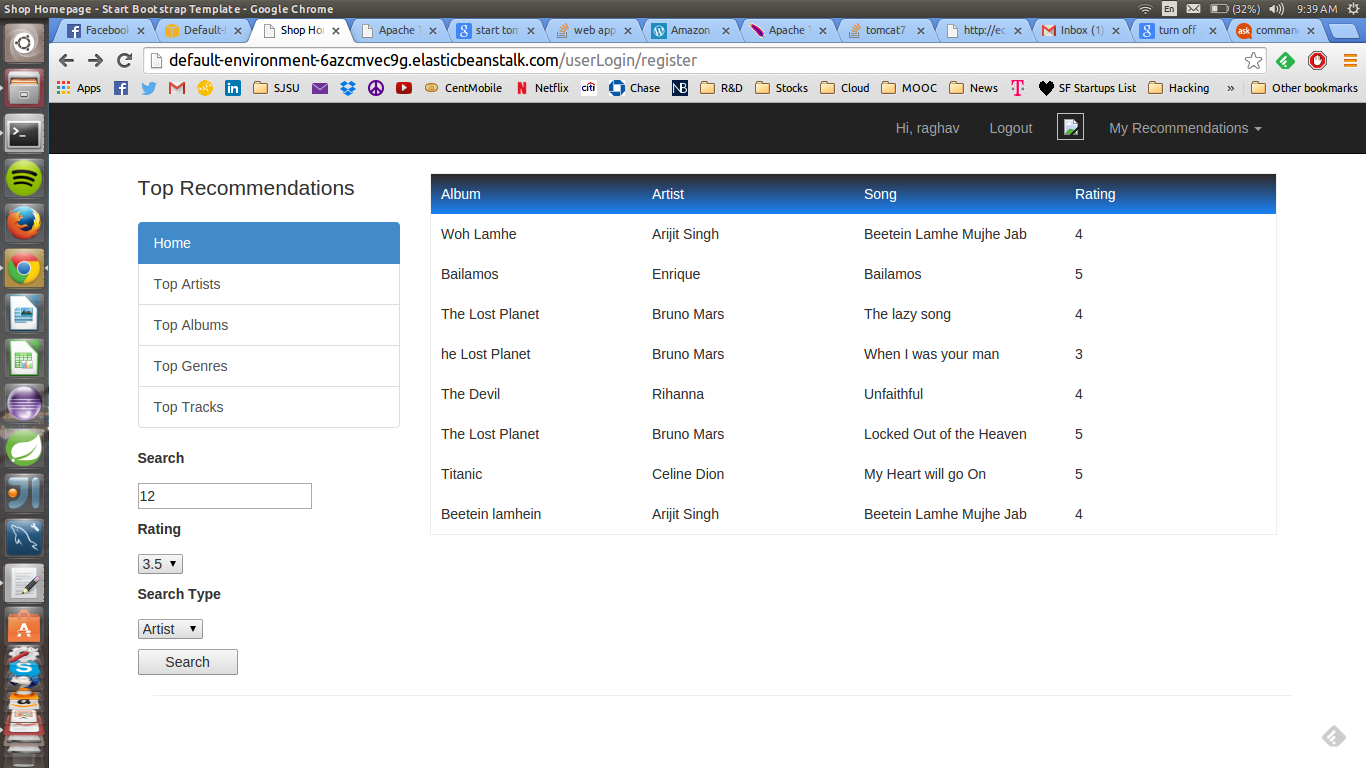
1. New User register



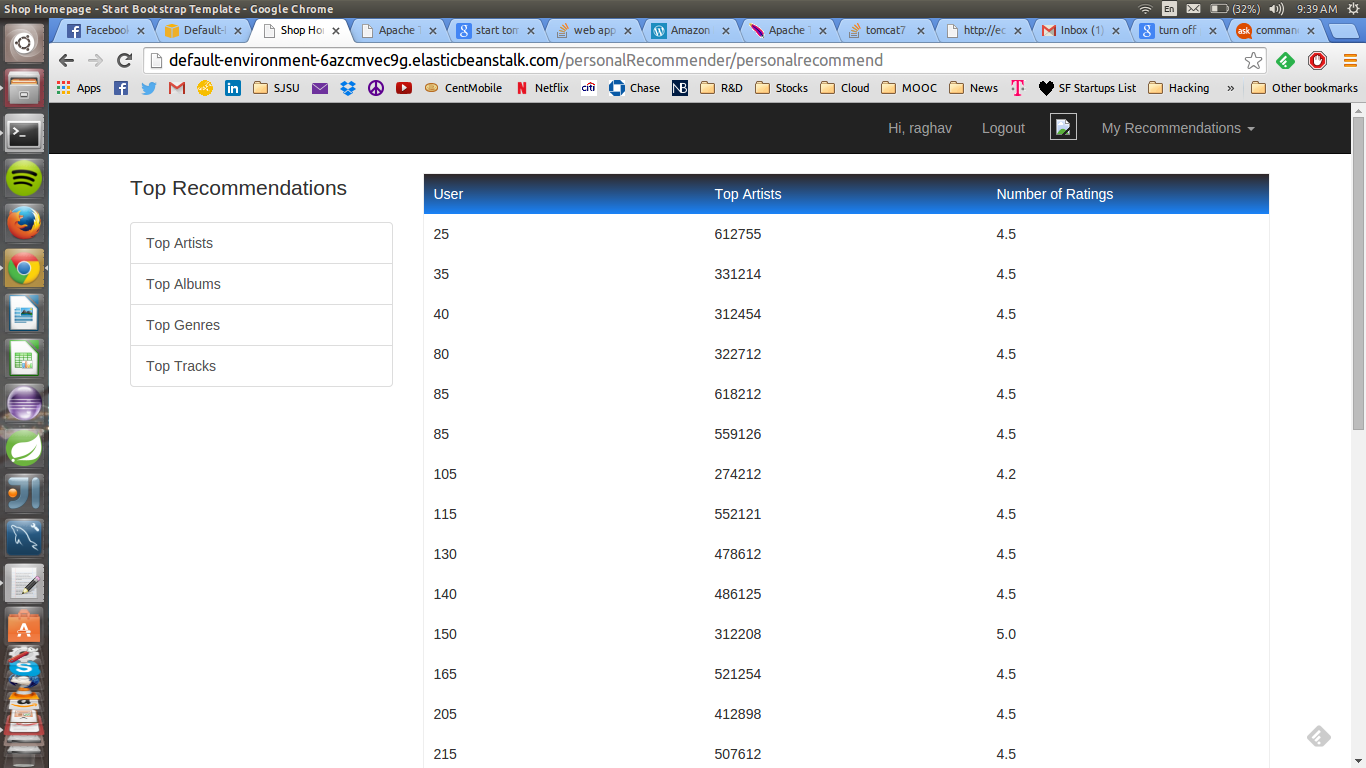
1. Search Album



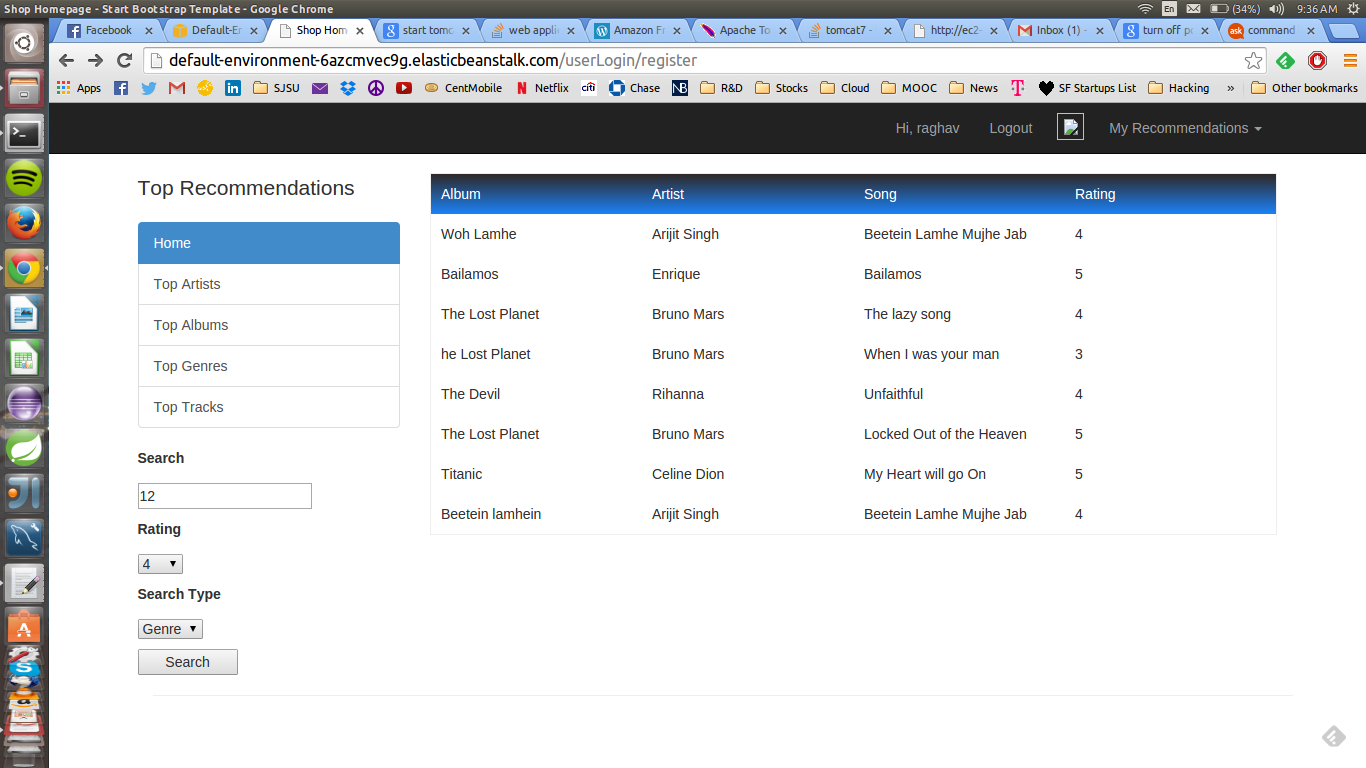
1. Search Artist



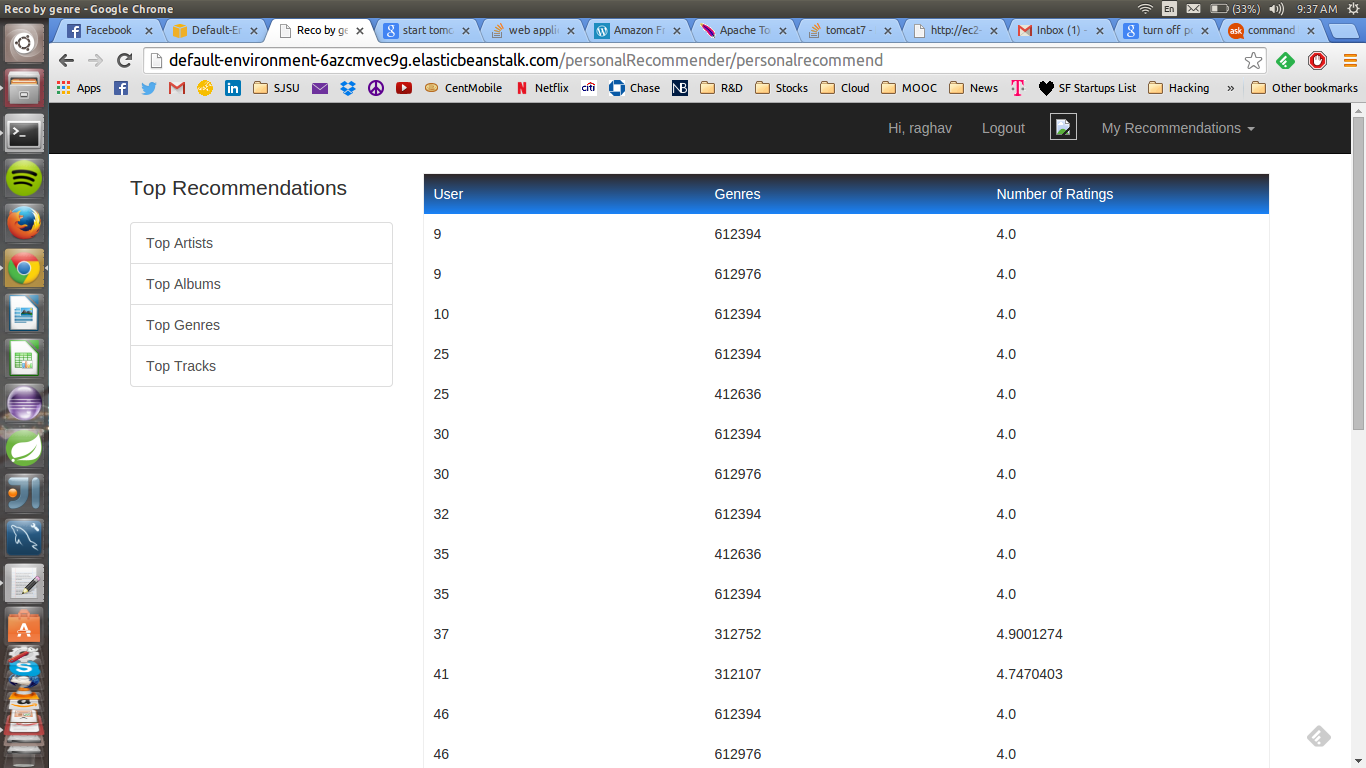
1. Search Artist results



1. Search Genre



1. Search Genre results



# 8. Individual Contribution

|  |  |
| --- | --- |
| **Team Member Name** | **Contribution** |
| Avi Jain | Data Segregation, Data Load |
| Eshan Haval | Web App Development, Mahout |
| Jasleen Kaur | Web App Development, Data Segregation |
| Raghav Munjal | Data Load, Web App Development |
| Sagar Ruchandani | Map Reduce, Data Load, Data Segregation |
| Shubham Rajvanshi | Web App Development, Data Load |

# 9. Future Work

* Music recommendation application can be developed on different environments.
* We can add more number of data sets and performance metrics and get more accurate results.
* We can scale the whole application we can scale it for more number of hosts and for each host using quad core we can maximize number of threads that are collecting data from database.
* We can implement more robust analytics algorithm for this data, thus improving the overall analysis and recommendation for the user.

# 10. Conclusion

There is a vast pool of unstructured data being generated everywhere, it is crucial to utilize this data in a structured way to improve efficiencies across broad sectors in health care field. This vast amount of important data can be used by healthcare industry to improve medical care and operational efficiencies. Our marketable could service Web Application is just a small step to help users find solutions to their health issues and take precautionary actions before its too late.

# 11. References

http://tdunning.blogspot.com.tr/2008/03/surprise-and-coincidence.html

<http://webscope.sandbox.yahoo.com/catalog.php?datatype=c>

http://en.wikipedia.org/wiki/Apache\_Mahout

http://en.wikipedia.org/wiki/Amazon\_Web\_Services

<http://en.wikipedia.org/wiki/Groovy_%28programming_language%29>

http://www.eng.tau.ac.il/~noamk/papers/DKK11.pdf