Guru Nanak Institutions Technical Campus

Computer Science and Engineering

Movie Recommendation System based on User Preferences

Software Requirements Specification Document

V1

Batch: A8

Anagondi Shiva Prasad – 19WJ1A0519

Badempet Pavan - 19WJ1A0531

Cheerala Prashanth – 19WJ1A0561

**October 20, 2022**

***CHANGE HISTORY***

|  |  |  |
| --- | --- | --- |
| **Date** | **Revision** | **Comment** |
| 20.10.2022 | 1.0 | Created |

***PREFACE***

This document contains the Software Requirements Specification (SRS) of a

Recommender System. This document is prepared according to the “IEEE Recommended Practice for Software Requirements Specifications – IEEE Std 830 – 1998”. Main purpose of this documentation is to give detailed information about the software requirements and functionalities of the Recommender System.

The Recommender System aims to enhance the existing recommendation tools served to Movie Recommendation Systems. In this project, the challenge is to overcome the major constraints and carry out the basic functionalities of such systems. The target audience who wants to make use of this system, can find all related requirements information in this document.

It assists the software developer ttea and the end users.

The first section gives the definition, purpose and scope of the Recommender System. The following sections include detailed description and requirements of the project. This specification is the primary document upon which all the design, implementation, and test/validation plan will be based.

[TABLE OF CONTENT 1. INTRODUCTION 5](#_Toc31214)

[*1.1. PROBLEM DEFINITION* 5](#_Toc31215)

[*1.2. PURPOSE* 5](#_Toc31216)

[*1.3. SCOPE* 6](#_Toc31217)

[*1.4. DEFINITIONS, ACRONYMS AND ABBREVIATIONS* 6](#_Toc31218)

[*1.5.* *REFERENCES* 7](#_Toc31219)

[*1.6.*  *OVERVIEW* 8](#_Toc31220)

[**2.** **OVERALL DESCRIPTION** **8**](#_Toc31221)

[*2.1.*  *PRODUCT PERSPECTIVE* 8](#_Toc31222)

[*2.1.1. SYSTEM INTERFACES* 9](#_Toc31223)

[*2.1.2. USER INTERFACES* 9](#_Toc31224)

[*2.1.3. HARDWARE INTERFACES* 12](#_Toc31225)

[*2.1.4. SOFTWARE INTERFACES* 12](#_Toc31226)

[*2.1.5. COMMUNICATION INTERFACES* 14](#_Toc31227)

[*2.1.6. MEMORY* 14](#_Toc31228)

[*2.1.7 OPERATIONS* 14](#_Toc31229)

[*2.1.8. SITE ADAPTATION REQUIREMENTS* 14](#_Toc31230)

[*2.2. PRODUCT FUNCTIONS* 14](#_Toc31231)

[*2.3. CONSTRAINTS* 15](#_Toc31232)

[*2.4. ASSUMPTIONS AND DEPENDENCIES* 15](#_Toc31233)

[**3. SPECIFIC REQUIREMENTS** **16**](#_Toc31234)

[*3.1. INTERFACE REQUIREMENTS* 16](#_Toc31235)

[*3.2. FUNCTIONAL REQUIREMENTS* 16](#_Toc31236)

***1. INTRODUCTION***

## 1.1. PROBLEM DEFINITION

Recommendation Systems are software tools or knowledge discovery techniques that provide suggestions for items to a user. Items can be music, books, movies, people or social groups. The aim of these systems is to recommend items that a user is likely to be interested in and learn more about user preferences. This project is based on improving the existing recommendation systems and generating more accurate recommendations for Movie suggestions and provide all the necessary information regarding the suggested movies

## 1.2. PURPOSE

This document provides a detailed description of Software Requirements Specification

(SRS) for Recommendation System. It is prepared according to “IEEE Recommended Statements for Software Requirements Specification - IEEE Standard 830 – 1998”.

The Software Requirements Specification (SRS) document is intended to provide the requirements of the Movie Recommendation System based on User Preferences project. The document includes the project perspective, data model and constraints of the overall system.

The intended audiences of the document are project managers, developers, testers and end users.

**·** Project managers review the document and determine whether the planned system fulfills the requirements. They notify the developers to fill up missing parts.

· Developers provide consistency by using the documentation.

· Testers use the documentation for verification and validation.

. End users make use of this document to learn about the scope of the system and its capabilities.

## 1.3. SCOPE

The targeted software product is a Recommendation System. The system makes an extensive use of user data to come up with reasonable track predictions about user preferences and there are two main paradigms in this sense: Content-based and collaborative filtering. As a beginning, the scope of the system is scaled down to content-based filtering. Content based filtering is a technique that is used to recommend movies. Apart from providing recommendations the system also provides posters of the Movies along with Release Date, Budget, Revenue, Popularity, Similarity between selected Movie and an Overview as well. The System uses the concept of vectorization based on common features and uses Cosine Similarity with respect to each other vectors to determine the most similar movies. The system is aimed to have additional solutions for big data and sparse information. The Movies Dataset is taken from Kaggle. Kaggle allows users to collaborate with other users, find and publish datasets, use GPU integrated notebooks, and compete with other data scientists to solve data science challenges. The project limits the improvement of the recommendation system around these challenges and favors accuracy over performance while giving the final suggestions. As a milestone, the system should follow the methods to enhance accuracy, the performance enhancements are not included in the scope. The Recommender System will be a Web Service and the end users will be directly exposed to the suggestions made for them.

## 1.4. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

|  |  |
| --- | --- |
| **Term** | **Description** |
| **API** | Application Programming Interface |
| **Big Data** | A collection of [data sets](http://en.wikipedia.org/wiki/Data_set) so large and complex that it becomes difficult to process using onhand database management tools or traditional data processing applications |
| **Cold Start** | The issue that the system cannot draw any inferences for [users](http://en.wikipedia.org/wiki/User_(computing)) or items about which it has not yet gathered sufficient information |
| **IEEE** | The Institute of Electrical and Electronical  Engineers |
| **OS** | Operating System |
| **UML** | Unified Modeling Language |
| **Web Service** | A software function provided at a network address over the web or the cloud |
| **UI** | User Interface |

## 1.5. REFERENCES

[1.] A Comprehensive Survey on Movie Recommendation Systems; Authors: R. Lavanya, Utkarsh Singh, Vibhor Tyagi; Year: 2021

[2.] A Recommendation Model Based on Content and Social Network; Authors: Hang Xue, Dongmei Zhang; Year: 2019

[3.] Recommendation of Indian Cuisine Recipes based on Ingredients; Authors: Nilesh Nilesh, Madhu Kumari, Pritom Hazarika, Vishal Raman; Year: 2019

[4.] Recommendation system for property search using content based filtering method; Authors: Tessy Badriyah, Sefryan Azvy, Wiratmoko Yuwono, Iwan Syarif; Year: 2018

[5.] The Cosine Similarity Technique for Removing the Redundancy Sample; Authors: Worasak Rueangsirarak, Teeravisit Laohapensaeng, Suppakarn Chansareewittay, Anusorn Yodjaiphet, Year: 2019

## 

## 1.6. OVERVIEW

This document includes six chapters:

# ● Overall description

* Specific requirements
* Data Model and Description
* Behavioral Model and Description
* Planning
* Conclusion

The software requirements begin with the overall description of the project. The product perspective, major functions and constraints are covered briefly. The software and hardware interfaces are stated in the next section as well. The following chapter gives the detailed requirements and functionalities of the software product. The functional and nonfunctional requirements are explained broadly. The design model and design constraints are mentioned extensively. One complete chapter is reserved for the data description and the behavioral model separately. SRS document involves UML diagrams for the project design. The use case, class and state diagrams are depicted in related sections. The documentation is finalized with the estimated schedule and team work for the project.

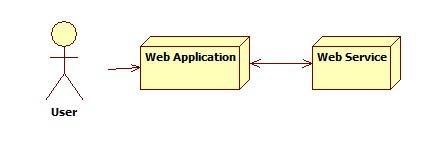
***2. OVERALL DESCRIPTION***

## **2.1. PRODUCT PERSPECTIVE**

Recommendation system is a software tool that provides suggestions for items to a user. Cloud Platform would be used to deploy our system to main web application as a web service. Web services extend the World Wide Web infrastructure to provide the means for software to connect to other software applications. Our web service helps existing larger system to increase in usage of system by accurate suggestions.

TMDB is a Movie Database which allows access to its database via API. This service provides users with information about Movies. It also provides APIs for Developers to work on with the TMDB Movies Database. With the APIs the sservice provides the opportunity of displaying Release Date, Budget, Revenue, Popularity & Overview of the Movies. Moreover, it provides with a very large number of Posters.a All these actions generate dataset to our system for accurate recommendations based on content similarity. The database is the most important part of our system therefore main system's existence and wide usage are important for generating recommendations on our system.

The external interface and the interconnection between the Recommender System and API Web Service can be seen in Figure 1.



**Figure 1:** Block diagram

### 2.1.1. SYSTEM INTERFACES

There is just one system requirement which is Internet Access.

### 2.1.2. USER INTERFACES

Movie Recommendation System has a user-friendly GUI to provide ease of use and effectiveness to the users. Our web application will require a User Interface in order to achieve a GUI. In order achieve this we could use any of the framework available for web application development.

A screenshot of a computer

Description automatically generated with medium confidence

**Figure 2:** Web Application Interface

In Web Application Interface, there are four elements Input Field, Recommend Button, Progress Bar & Tabs. (Figure 2) shows the main user interface displaying a title. Providing a User Input Field for either Searching the List or Typing with Recommendation Search Filters. A Recommend button is provided which could be activated by user upon entering the input a Movie that the user has liked.

Graphical user interface, application

Description automatically generated

**Figure 3:** Web Application Interface on Recommend Action

In (Figure 3) We have the interface upon clicking the Recommend Button after selecting a Movie. This interface consists of Tabs for Ten Similar Movies and 2 Columns in each Movie Tab. The Column. 1 is proposed for Movie Poster while Column 2 Display all the basic necessary information regarding the movie to User.

Graphical user interface, website

Description automatically generated

**Figure 4:** Web Application Interface of All Recommended Movies

### 2.1.3. HARDWARE INTERFACES

Even if we are using Cloud Platform for handling all the underlying hardware requirements for a Project. Otherwise the Project would need.

PROCESSOR : Pentium Processor

RAM : 2GB DDR3 RAM

HARD DISK : 10 GB Free Disk Space

### 2.1.4. SOFTWARE INTERFACES

The software requirements provide a basis for creating the software requirements specification. It is useful in estimating cost, planning team activities, performing tasks and tracking the teams and tracking the team’s progress throughout the development activity.

Operating System : Windows 7/8/10

Platform : Azure

Programming Language : Python

Front End : Streamlit

### 2.1.5. COMMUNICATION INTERFACES

This system is a web application; therefore network connection with TCP/IP protocol is necessary.

### 2.1.6. MEMORY

At the beginning of the Recommendation System project, we will be using our local computers. Therefore minimum 2 GB DDR3 RAM and 5 GB hard disk space will be needed.

### 2.1.7 OPERATIONS

Our web appliation service makes recommendations even if the user hasn’t played any tracks. This is cold start and the recommendation operations related to this phase are not user specific. For gaining better recommendations, user shall search, listen or download tracks. These recommendations are specific for users.

### 2.1.8. SITE ADAPTATION REQUIREMENTS

We are going to study with movies data on this project. However, our local computers restrict us and we cannot deal with big data. Therefore, we need to deploy our project on a web server.

## 2.2. PRODUCT FUNCTIONS

Recommendation Systems are considered as one of effective knowledge management engines that helps us filter out unwanted data and provide targeted data based on the feedbacks from old data and similar data from user’s search. Many Recommendation Systems have been introduced till date following different approaches for the computation like CBF, CF and hybrid models for recommendation. Sentiment Analysis is also used to improve the recommendation efficiency.

## 2.3. CONSTRAINTS

Recommendation systems suffer from three major problems; cold start, huge data and sparsity. Cold start is a potential problem in computer-based information systems which involve a degree of automated data modeling. Specifically, it concerns the issue that the system cannot draw any inferences from items about which it has not yet gathered sufficient information. There are basically two types of this problem. The first one occurs when a new user starts using the system. The system knows little about their preferences and it is necessary to pick some training points for rating so that the system begins learning what the user wants. The second type occurs when a new movie is introduced to the system. Again it is essential to rate this item in order to quickly improve the prediction accuracy about it.

The next problem is huge data. In many of the environments that these systems make recommendations in, there are millions of users and products. It is quite a challenge to produce high quality recommendations and perform many recommendations per second for millions of customers and items. Thus, a large amount of computation power is often necessary to calculate recommendations. Moreover, at the beginning of the project, we are going to study on our local computers. We are not going to manage with huge data because we have limited memory. After we start using server, we are going to practice with big data.

## 2.4. ASSUMPTIONS AND DEPENDENCIES

Our end product is planned to be a Azure Web App. There must be Internet connection.

# SPECIFIC REQUIREMENTS

## **INTERFACE REQUIREMENTS**

The user interface of our Recommendation System might use any kind of Web App Creation Framework. We only implement a simple user interface for showing movie recommendations. This user interface gives chance to display recommendations along with information about those movies.

## **FUNCTIONAL REQUIREMENTS**

This section encapsulates the major software functions and data flow among the participants of the system. The Dataset is provided as input to the Model. Eventually training the model on the supplied dataset input.



**Figure 5:** Use Case diagram

## NON-FUNCTIONAL REQUIREMENTS

### Usability

The system is designed with completely automated process hence there is no or less user intervention.

### Reliability

The system is more reliable because of the qualities that are inherited from the chosen platform java. The code built by using java is more reliable.

### Performance

This system is developing in the high-level languages and using the advanced front-end and back-end technologies it will give response to the end user on client system with in very less time.

### Supportability

The system is designed to be the cross platform supportable. The system is supported on a wide range of hardware and any software platform, which is having JVM, built into the system.

### SAFETY

No safety requirements have been identified.

### OTHER REQUIREMENTS – GNU GPL LICENSE

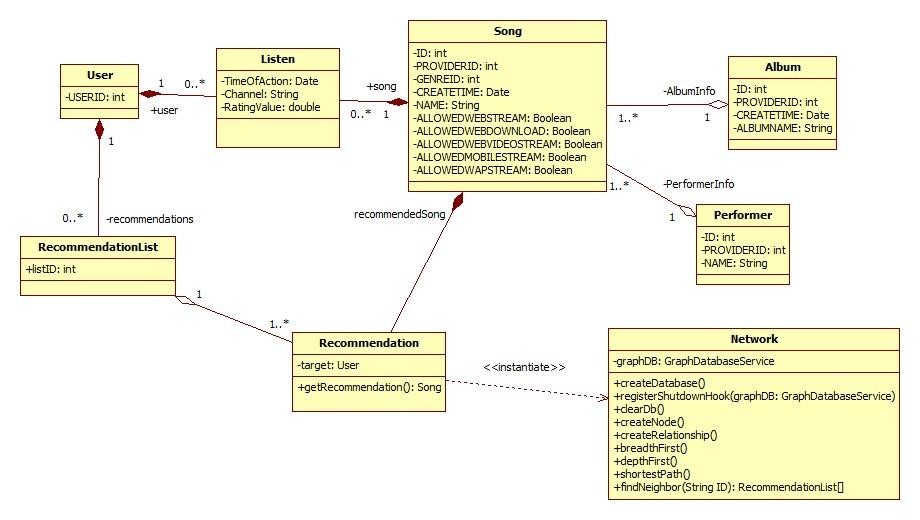
The project will be released under GNU General Public License, which allows the use, change and redistribution of the software freely. The philosophy of this license is applied to this recommendation system to achieve the greatest possible use and development by public.

# DATA MODEL AND DESCRIPTION

## DATA DESCRIPTION

### DATA OBJECTS

This subsection of the document explains system's classes and their relations with each other. Most of the system functionalities are represented in Figure 11.



**Figure 11:** Class Diagram

User: This class represents the user entity and it keeps a unique id.

Song: This class represents the song entity and it has the fields defined in the previous section. It has an aggregation association with Album and Performer classes. It contains *AlbumInfo* and *PerformerInfo* fields. An Album instance or a Performer instance can be mapped to one or more Song instances.

Album: This class represents the album entity and it has the fields defined in the previous section.

Performer: This class represents the performer entity and it has the fields defined in the previous section.

Listen: This class is a group of instances that represent listening action. Each time a new log is processed, an instance of this class is created. Its fields are defined in the previous section. This class has composition association with User and Song classes. It contains *user* and *song* fields.

Network: This class encapsulates database related methods and it keep a

GraphDatabaseService object which is the Neo4j database object. It has some specific methods such as *createDatabase*, *registerShutdownHook*,*clearDb*, *createNode*, *createRelationship* methods to construct/remove a database and add/delete nodes. The similarity construction requires the traversal of the graph database which is done via *depthFirst*, *breadthFirst* and *shortestPath* methods. The similar user and recommended songs are determined via *findNeighbor* method.

Recommendation: This class represents the recommendation object to be returned to the external system. It contains either a cluster of users that are found similar or a cluster of songs. The user to which this recommendation will be given is kept inside *target* field. This class has a composition association with the Song class and contains *recommendedSong* field. It also has an aggregation association with RecommendationList class. Whenever an instance of RecommendationList is deleted, the corresponding instances of Recommendation class still exist. RecommendationList: This class represents the group of recommendations given to a specific user. It has a unique id. This class has a composition association with User class.

### DATA DICTIONARY

|  |  |
| --- | --- |
| **Term** | **Definition** |
| |  | | --- | | **userID** | | **User ID are not same with real ones, company change**   |  | | --- | | **them in order to provide security and it is unique** | |
| **songID** | **songID is used for specific track** |
| |  | | --- | | **albumID** | | |  | | --- | | **albumID represents a unique album containing tracks** | |
| **artistID** | **This attribute specify the artist uniquely** |
| |  | | --- | | **timeOfAction** | | **timeOfAction shows the date and time which user download or listen track** |
| **ratingValue** | **This attribute is estimated according to that user download or listen track** |
| |  | | --- | | **channel** | | **Channel means that user listening track from browser,**   |  | | --- | | **him/her own list or another user list** | |

# BEHAVIORAL MODEL AND DESCRIPTION

This section provides the overall behavior of the system. Major events and states are displayed in a state chart diagram.

## DESCRIPTION FOR SOFTWARE BEHAVIOR

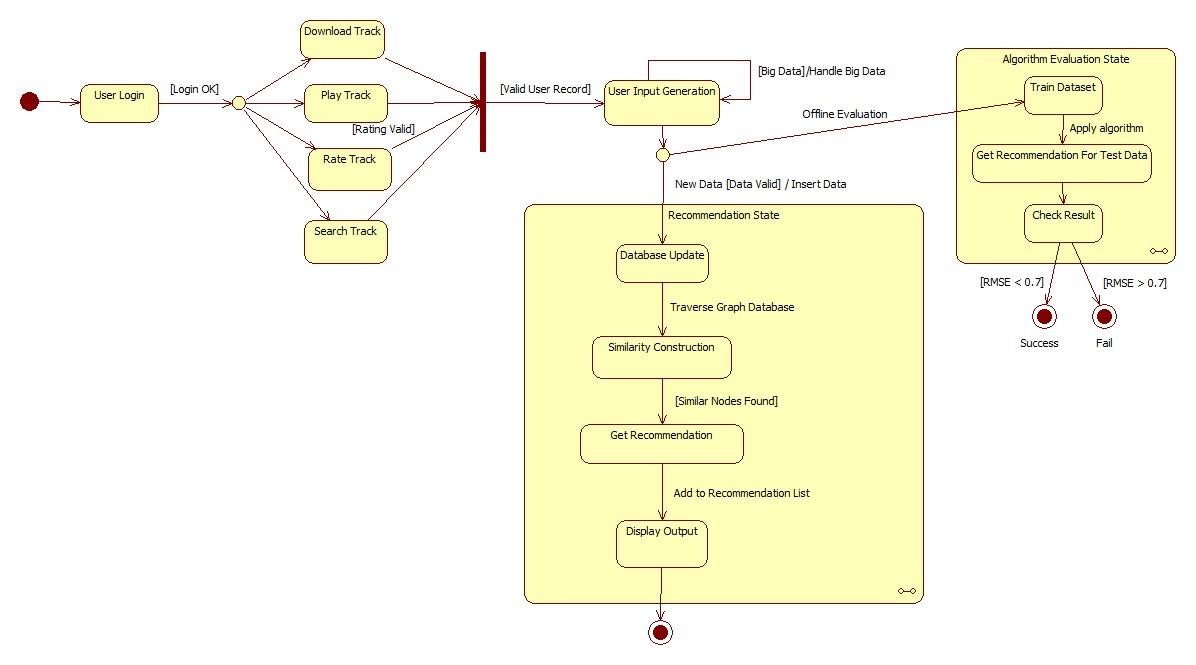
The state chart in Figure 14 illustrates the behavior of the Recommender System within a larger system. The larger system is a stand-alone music application that our software product is going to be integrated to. The recommender system is shown as an independent sub state. The states that remain outside the Recommender state are out of the scope of this project. However, these states and related events are necessary for our software product to proceed. The pre states and post states of the Recommender state are assumed to be completely encountered. The host application takes user input through the downloaded, played, rated and search track information. Records for each user are formed and they are supplied to our Web Service. The behavior of our Web Service mainly resides on database transactions. Since the project team uses graph database, the system traverses the large graph, which is actually the storage and notation method of the big user data, to find out the closest match to a user or to a track. After evaluating how successful this match is, the recommendation is presented to the user.

## STATE TRANSITION DIAGRAMS

The state chart diagram in Figure 13 shows all the states, triggers and conditions for each transition and related events. The trigger induces a state change given that the condition is satisfied. The conditions are written inside the square brackets.

As soon as the system is initialized, User Login state is triggered. It is followed by the login action. If the login information is valid, the user is directed to the actions involving tracks. The user can either download a track, play a track, rate a track or search a track. These states are the main sources for user data. The transition from these states is a synchronized action in order to obtain a complete user data. If the user data is valid, User Input Generation state is initiated. This state has a self-transition due to the big data condition. The project relies on the assumption that the project model is consistent with big data. It is indicated by the Handle Big Data event. As long as there are millions of users and tracks in the system, this state is instantiated again. When the dataset is completed, the actual Recommendation sub state is triggered. The arrival of new data triggers the database update on the condition that the new data are valid. The insertion event occurs at this point. New data are inserted into the graph database. New nodes and edges are formed inside the huge graph. The next state is Similarity Construction. The event related to this transition is the traversal of graph database. Instead of writing queries in a relational database, the recommender system requires traversing the graph nodes to find the most similar users. Get Recommendation state is the most crucial part of this set of behaviors. The closest track(s) according to the previous preferences of a user is/are determined in this state. As the last state, the output is displayed and the Recommender State is finalized.

In the Algorithm Evaluation State, the algorithm is tested on the user data in offline mode. Some part of the data is used for training while the rest is used for testing. RMSE (Root mean square error) is used in order to evaluate the accuracy of the system. It is a measure of how accurate the recommended songs are according to the actual logs. The threshold value for RMSE is currently fixed as 0.7. If the error gets higher, the system fails.



# PLANNING

## TEAM STRUCTURE

The four members of the Dcengo Unchained team are Duygu Kabakcı, Işınsu

Katırcıoğlu, Sıla Kaya, Koray Kocakaya. The team structure is as follows:

|  |  |
| --- | --- |
| **Task** | **Members** |
| Graph Traversal Algorithm Component | Duygu Kabakcı, Işınsu Katırcıoğlu |
| Neo4j Database Component | Duygu Kabakcı, Işınsu Katırcıoğlu |
| Neo4j API Component | Duygu Kabakcı, Işınsu Katırcıoğlu |
| Machine Learning Algorithm Component | Sıla Kaya, Mehmet Koray Kocakaya |
| UI Component | Sıla Kaya, Duygu Kabakcı |
| Weka API Component | Sıla Kaya, Mehmet Koray Kocakaya |
| Evaluator Component | Mehmet Koray Kocakaya, Işınsu Katırcıoğlu |
| Recommender Component | Sıla Kaya, Mehmet Koray Kocakaya |

## ESTIMATION

|  |  |
| --- | --- |
| **Estimation Date** | **Task** |
| **04.10.2013** | **Deciding Project** |
| **13.10.2013** | **Project Idea Proposal** |
| **22.10.2013** | **Elevator Pitch** |
| **30.10.2013** | **Software Requirements Specification (SRS)** |
| **30.10.2013 - 07.11.2013** | **Researching Requirement Systems and**  **Tools** |
| **07.11.2013 - 14.11.2013** | **Trying Software Tools** |
| **14.11.2013 - 18.11.2013** | **Documenting Design Report** |
| **18.11.2013** | **Design Report** |
| **18.11.2013 - 04.12.2013** | **Trying Software Tools** |
| **04.12.2013 - 16.12.2013** | **Documenting Updated Reports** |
| **16.12.2013** | **Updated Reports** |
| **16.12.2013 - 20.01.2013** | **Implementing Base Recommendation**  **System** |

## **PROCESS MODEL**

We will apply agile model for our recommendation system so that system can respond quickly to changing requirements without excessive rework. Agile method is based on an iterative approach, each iteration involves planning, requirements analysis, design, implementation, testing. Each iteration takes approximately four weeks. Once we will generate the initial version of recommendation system, then our system will be developed according to accuracy of recommendations, performance results on scaled big data.

# CONCLUSION

This Software Requirement Specification document is prepared to give requirement details of the project “Recommender System”. The detailed functional and nonfunctional requirements, system, user, software and hardware interfaces, data and behavioral model are stated in an extended outline. This document will be helpful at constituting a basis for design and development of the system to be developed.

*This page intentionally left blank*