SOFTWARE REQUIREMENT SPECIFICATION OF MUSIC RECOMMENDATION SYSTEM

CENG HISTORY X

HACER NİHAL TARKAN
AYŞE AYBÜKE TAŞDİREK
ASENA OK
BİRANT ALTINEL

TABLE OF CONTENTS

1. Introduction

This software requirement specification (SRS) report expresses complete description about Recommendation System Project sponsored by AGMLab. This document includes all the functions and specifications with their explanations to solve related problems as a project of Middle East Technical University Computer Engineering Department.

1.1 Problem Definition

We are working on some common problems the recommender systems face and trying to improve the current solutions on these problems to increase the accuracy of the recommendations. The main issues would be both huge, dense data sizes and sparse data sizes. There also may be issues about similarity, calculations, and data integration. To make the picture more clear, if we are to specify data integration process as an example, to integrate data from different sources requires combining heterogeneous data sources under a single query interface which raise an important issue.

The problem about huge and dense data is that it may lead to the need of more complex calculations with inefficient speed even though the recommendation accuracy increases. Therefore we need to work heavily on different paradigms, algorithms, and databases to create an efficient solution even for very large data sets. On the other hand, density of data is an advantage in terms of accuracy of the system. As stated above, if density can be run efficiently on the side of development, it would be improve the quality of the system.

Sparsity of data is also a major issue, because no matter how well the algorithms are, if there is no data to process, this system will be of no use. Therefore we need to create solutions to also deal with these situations. Cold start may cause sparsity. If there is not enough data for a new user in the system, it will affect the recommender negatively.

1.2 Purpose

The purpose of this document is to present a detailed description of the Recommender System that we will design and implement. It will explain the purpose and features of the system, the interfaces of the system, what the system will do, the constraints under which it must operate This document is intended for both the stakeholders and the developers of the system.

The corresponding environments that this project will be integrated to are supposed to and probably will have a large number and variety of users in which many of them need such system to find whatever it is they need. The users may or may not be aware of their need for a recommendation feature on the software or website they are using, but such features can increase efficiency and save time to users, while they are looking in a place where there is a large number and variety of content which causes them to waste a lot of time to find what they need. From another point of view, there might be users that don't exactly know what they are looking for and such situations can also make a useful solution out of this project.

1.3 Scope

The goal is to design a recommender system on music domain. The system will be composed of server side components and client side components. The server-side component will manage the database operations and algorithms that produce recommendation results. The client-side components will be graphical interfaces that are integrated into corresponding larger systems.

Generally, a music recommender system consists of three key components: users, items and user-item matching algorithms. So, the background of our product is becoming more important than foreground component. At the background, the system will gather data from both users and items. To be more specific and precise, those data can be classified as follows:

- User Modeling [1] : Suggest that user data can be categorized into 2 domains: User Profile Data and User Listening Experience Data (shown in Table 1 and Table 2).[2]:

Data Type	**Example**
Demographic	Age, marital status, gender etc.
Geographic	Location, city, country etc
Psychographic	Stable: interests, lifestyle, personality etc. Fluid: mood, attitude, opinions etc.

Table 1.3.1: User Profile Data Classification

Type	Percentage	Features
Savants	7	Everything in life seems to be tied up with music. Their musical knowledge is very extensive.
Enthusiasts	21	Music is a key part of life but is also balanced by other interests.
Casuals	32	Music plays a welcome role, but other things are far more important.
Indiff erents	40	They would not lose much sleep if music ceased to exist, they are a predominant type of listeners of the whole population.

Table 1.3.2: User Listening Experience Data Categorization

- Item Modeling [1] : The second component of recommender systems is music item. The music metadata classified into three categories as in table below:

Item Type	Example
Editorial Metadata (EM)	the cover name, composer, title, or genre etc.
Cultural Metadata (CM)	Similarity between music items
Acoustic Metadata (AM)	Beat, tempo, pitch, instrument, mood etc.

Table 1.3.3: Music Item Metadata

By using both user and item related data and relationship between them, our software will produce meaningful music recommendations to users.

As we can see from the development of music recommender systems over the past years, the results are becoming more personalized. Using only music itself and user ratings are no longer sufficient. In recent years, great amounts of work have been done in music perception, psychology and music’s effects on human behavior. Undoubtedly, music always has been an important role in people’s life and people have greater access to it. All of these highlight that music recommender is an effective tool for saving our time to find music for our personal interests.

1.4 User and Literature Survey

After that e-commerce gained popularity of selling products and the development of new generation mobile phones and innovative inventions like tablets, shopping become easier than before. Therefore, new kind of advertising techniques came up. Recommendation systems are one of the most popular techniques nowadays. They are useful for both the company and the user, because they increasing product sales and reducing the time spend on shopping. However, they have some problems because of huge data. Main problems are being unable to get really relevant results and being unable to get results in reasonable time.

Up to now, there are a lot of methods developed to resolve the problems stated above such as collaborative filtering, content-based filtering. However, they have some weaknesses. For example, collaborative filtering has the problem called cold start and this means that recommendation system cannot produce any suggestion or recommendations. This problem occurs when items are provided in the system, but there are few customers and few or no rankings. And, the other example for content-based filtering, if the content is in lack of enough information to distinguish the items precisely, the recommendation cannot be precisely done. On the other hand, our system should find the most accurate recommendations.

Our potential users that we desire to help their problems are companies that use internet utilities to sell their products. And by proposing our product to these companies, we will reach to users of the websites that companies use. Therefore we will reach to both customers and companies with our system.

1.5 Definitions, acronyms, and abbreviations

SRS	Software Requirement Specification
Neo4j	Neo4j is a robust transactional property graph database.
Java	Java is a computer programming language that is concurrent, class-based, object-oriented, and specifically designed to have as few implementation dependencies as possible.

Cypher	Cypher is a Graph Query Language.
Neoclipse	Neoclipse is a standalone workbench application to interact with Neo4j (database directory or server).
Windows	Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.
Netbeans	NetBeans is an integrated development environment (IDE) for developing primarily with Java, but also with other languages, in particular PHP, C/C++, and HTML5.
GUI	Graphical User Interface
Eclipse	A multi-language Integrated development environment (IDE)

Table 1.5.1: Definitions, acronyms, and abbreviations

1.6 References

[1] Song Y., Dixon S., Pearce M. "A Survey of Music Recommendation Systems and Future Perspectives", Page 397, 2012 London

[2] Jean J., "Aucouturier and Francois Pachet. Music Similarity Measures: What is the Use. In Proceedings of the ISMIR", Pages 157–163, 2002.

[3]Mortensen M., "Design and Evaluation of a Recommender System", Pages 26-28, 2007, Norway

[4] Javega M., "Content-based Music Recommender System", Page 35, 2005

[5]Whitman B., "How Music Recommendation Works and Doesn't Work", Retrieved from http://notes.variogr.am/post/37675885491/how-music-recommendation-works-and-doesnt-work

[6]Alexander C., "Risk Based Software Development", 2005, Retrieved from http://www.reliablesoftware.com/weblog/blogger.html

1.7 Overview

Following section of this document will focus on describing the system in terms of product perspective, product functions, user characteristics, assumptions and dependencies. In the third section, we will address specific requirements of the system, which will enclose external interface requirements, functional requirements of the system, performance requirements, and other requirements.

2. Overall description

In this part, background information about specific requirements of the system will be provided briefly. General issues that affect the product and outline of the functional requirements will be mentioned, too. In short, this section will mainly give information about product perspective, product functions, constraints, assumptions and dependencies.

2.1 Product perspective

Recommendation system mainly depends on the content of the database which has data that comes from users of the website integrated into the static data the website holds(Music Data). Since the website will have a variety of different users, there will be functionality differences between users with respect to item data. The recommendation systems used by companies have different efficiency levels depending on the context of the solution. Our recommendation system should work efficiently according to music data. So, there exists a user interface that is suitable for music recommendations and this interface will be an e-commerce website. Our system will be working in background of this e-commerce website. Once the recommendation system finds an accurate result, it will be shown on the interface to the user.

In terms of hardware, recommendation system will have a database server which holds the music data with user logs along with another server which is responsible of creating the connection to the database and generating user specific recommendations.

In terms of software, our recommendation system will run on the web browsers or specific applications on personal computers, smart phones etc. That is, it will run on any device with an internet connection. The system will work both on Windows and Unix operating systems. Moreover, it will be implemented making use of database management tools such as Neo4j.

This brief information of interfaces is explained in more detailed below.

2.1.1 System interfaces

First of all, the application needs to have music data to make recommendation to user and user data to evaluate. User data is necessary to make a really relevant recommendation. The system will use this information to make inference about recommendation by getting user's actions. Rating value of a music data is a derived value obtained by an algorithm.

2.1.2 User interfaces

There will be one type of user. Therefore there are no differences between users in terms of the basic functionality, visualization and interface.

The user interface mostly depends on the website designers. The website that our system is working on the background can offer different options to its users. For example, a website can offer the right to choose background image to each of its users. However the architecture will probably be the same for all users. On the other hand, the interface will differ in some number of steps such as user login, some top hundred lists, search options etc. But the first step should be user login which will activate the recommendation system. Then the recommendations will be specified according to the algorithm that depends on user actions. These actions are determined by some parameters such as the time that the user listened the music, the types of songs they prefer, their similarities with other users etc. Some of these actions are listened, downloaded, listened before. If music is downloaded before, then other musics' rating value in the album will increase.

Up to now, we mentioned about how the recommendation will be done, but making recommendation more recognizable is also an important point. After getting a recommendation to user, the system should display it on the websites' graphical interface. However, if the recommendations cannot get user's attention, all the work done up to now might come to nothing. Therefore, they should be displayed on the user interface spectacularly. The basic graphical user interface is given below.

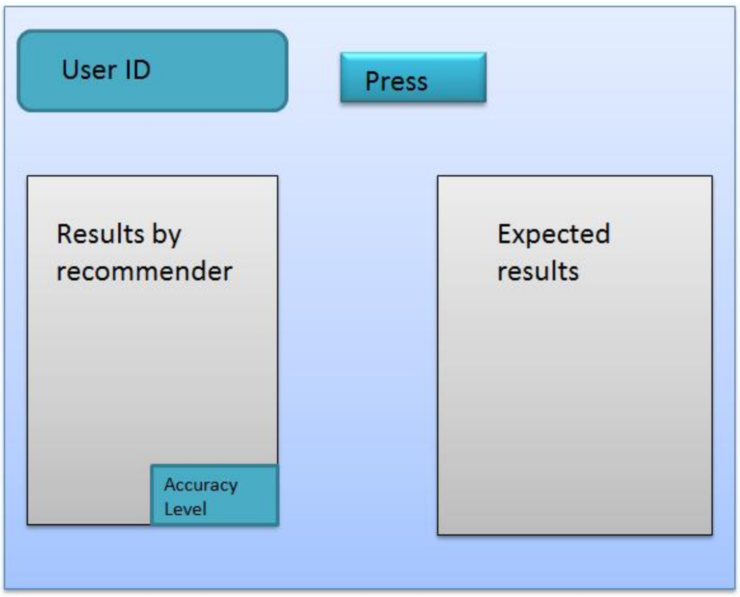

Figure 2.1.2: Basic GUI

2.1.3 Hardware interfaces

The recommendation system can work on any internet connected device. These devices should have some limit requirements to make the application run effectively. We expect that the processor speed and internet speed are high enough to run the website / application.

2.1.4 Software interfaces

First of all the system will work on any platform. Internet connection is a must to reach the system. Moreover, most of the application will be coded by Java. Java APIs of database management tools such as Neoclipse, which is a standalone workbench application to interact with Neo4j. Moreover, the query language used in Neo4j api will be used with Java. Some tools and software are listed below.

Name	**Version number**	**Source**
Neo4j	2.0.0-M06	http://www.neo4j.org/
Netbeans	7.4	https://netbeans.org/
Windows	7/8	Microsoft Company
Eclipse	4.3.1	http://www.eclipse.org/
Java	SE 7	Oracle Corporation

Table 5: Software for our system

As the database Neo4j is used, we will connect the database with our Java code which will be developed on the Eclipse IDE, we will use Java Servlet. While developing the project we will mainly be working on Windows operating systems 7&8 versions.

2.2 Product functions

Use case diagram of the recommendation system and the other subsystems are revealed in Figure 2.2.1. Steps are gathered in distinct entities, the functions of which are stated in further subsections.

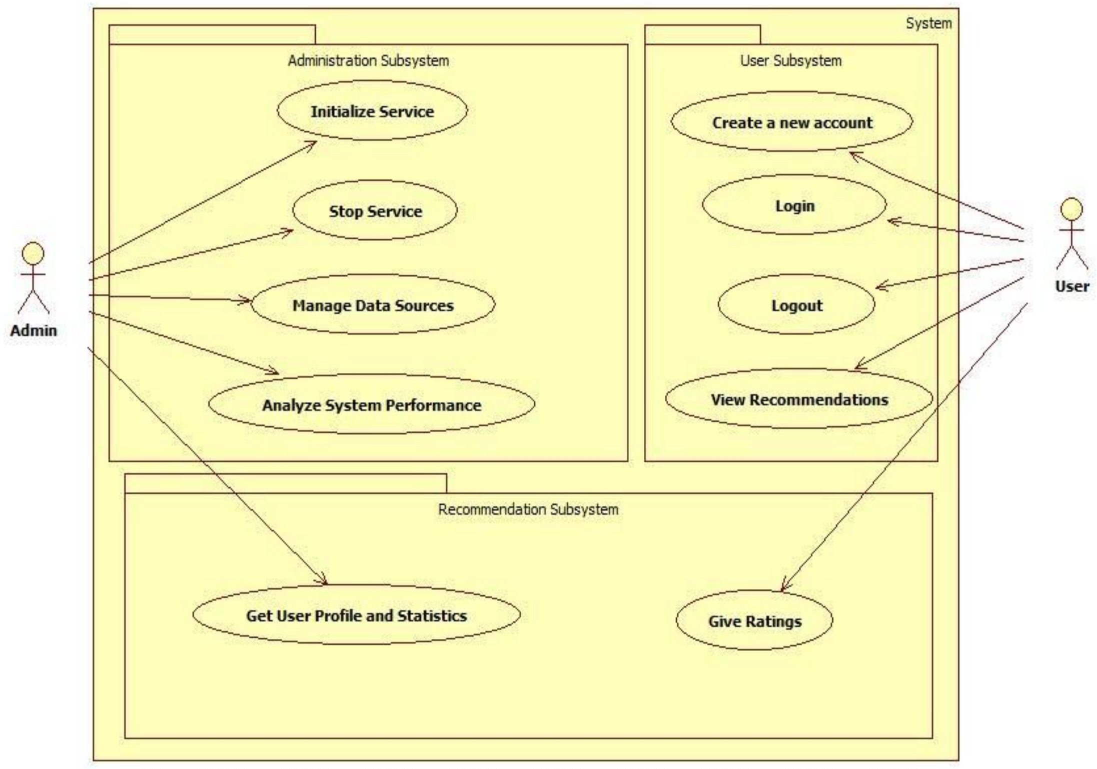

Figure 2.2.1: Use Case Diagram

2.2.1 Initialize Service

This functionality allows the administrator of the system to initialize the recommendation service.

2.2.2 Stop Service

This functionality allows the administrator of the system to stop the recommendation service.

2.2.3 Manage Data Sources

The administrator of the service will be able to manage the data sources, by modifying the existing database. The admin will be able to add new data into the system, as well as alter the existing data.

2.2.4 Analyze System Performance

The administrator will analyze the system performance which is defined as accuracy of recommendations. The analysis will be based on the statistics of recommendations shown to the user, and the information whether the user listened any of the recommendations and what rating they provided.

2.2.5 Get User Profile and Statistics

The administrator will be able to use this functionality to retrieve the information regarding to users, along with the recommendation history of this user.

2.2.6 Create a new Account

When the user creates a new account on the website/application that the recommendation system runs on, also implicitly an account for the user inside the recommendation system will be created to store information about the user profile and statistics to provide better recommendations.

2.2.7 Login

When the user logs in to the website; Music Recommendation System will be informed and a recommendation session for the user will start and generate recommendations.

2.2.8 Logout

When the user logs out from the website; Music Recommendation System will be informed and the associated recommendation session for the user will be closed.

2.2.9 View Recommendations

Whenever new recommendations are generated, they will be shown to the user through the recommendation GUI which is embedded into the GUI of the main system. The user will be able to view recommendations, click on them to listen the songs.

2.2.10 Give Ratings

The user will give ratings to the songs that are generated as recommendations on the GUI of our system. These ratings will be stored to provide better recommendations in the future.

2.3 Constraints

- Since we need user profile data while developing the product, to find real time and sufficient data can be a problem for developer because of regulatory policies.
- Millions of data will be needed to test the software. At this stage developers will need huge amount of disk space and clusters.
- We will not design any specific interfaces for the product. It will be suitable for the application that people listen music from. So, developers have to consider common components of these applications so that the software can be integrated to any of this application easily.
- The application gathers real time user profile information from user accounts. Therefore, it must be reliable and keep those data in safe. Moreover, the system will produce new data about users depending on their behavior on the web. Security of this resulting data must be provided by the software also.

- Most important concern of the system is producing accurate recommendations. To provide expected accuracy and to handle with sparse and huge data at the same time is critical.

2.4 Assumptions and dependencies

As stated in the previous section which is constraints, there are several requirements like sufficient music data, user data, servers etc. To accomplish activating recommendation system, these requirements should be provided. Moreover, the system will also need a strong and stable internet connection to keep all the servers intact and working. Losing connection is an important problem, because our system will work on an online platform and needs a connection all the time. If the internet connection cannot be supplied as requested, the whole sales policy will fail and our system will be useless.

Therefore, we assume that ArgeDor will provide us with sufficient and correct data of music and users. Also we assume that there will always be a working internet connection between separate parts of the system.

3. Specific requirements

This section will describe the software requirements in detail as subsections which are interface requirements, functional and non-functional requirements.

3.1 Interface Requirements

Since there is just one type of user, the application will have only one type of GUI which is embedded in the main e-commerce website. Basically the user interface will direct the user through steps and will display the recommendations found by our algorithm. The steps of the user interface are the following;

Supplier Subsystem

This user interface is only for supplier, which is company. Supplier can do initialize and stop service, analyze system performance, and manage data sources (by enlarging data or by altering data).

User Subsystem

This user interface is for customers that are the people who have an account and are logged in to the website. A user can view recommendations on the GUI of the website, listen to the recommended songs.

Recommendation Subsystem

This user interface is the one which we all are interested in because our algorithm will work in the background of it. This interface will be used by both customers and suppliers. Customer cannot do many operations, but their ratings are very important to create a relevant recommendation. Users can only use give rating operation. This subsystem has its part in GUI only as a “rating for this recommendation” part. This provides feedback for improvements in our recommendations. On the background of this subsystem, suppliers can get customer profile and statistics.

3.2 Functional Requirements

In this section, we will explain the major functions of the Recommendation System.

3.2.1 Client

The client-side of the system will be an application with a user interface that is integrated into a music listening website or application. This application gathers the information from users, investigates some actions of the users, and provides the connection with the server. This application is the client-side interface of the Music Recommender, so it does not include the functionalities of the host music environment such as playing music etc.

- **Requesting recommendations**

The client-side application must allow user to request recommendations manually, and interact with the server to receive recommendations.

- **Display recommendations**

The application must display the recommendations that are obtained from the server to the user in a proper way by providing a GUI.

3.2.2 Server

The server-side system will hold the entire data in a graph database, and must include all functionality to perform operations on this database, receive requests from the clients, evaluate, create and send recommendations etc.

- **Handle recommendation requests**

  The server application shall obtain and handle requests for recommendations.
- **Store evaluations**

  The server application shall receive and store music evaluations
- **Data updating**

  The server application shall be able to store the newly retrieved data to the database.
- **Generate Recommendations**

  The server application shall be capable of producing recommendations by interpreting the content and evaluations by actual user. Server will have a recommendation class which contains functions of algorithm.

3.3 Non-functional Requirements

The non-functional requirements of the system are explained below as performance requirements and design constraints.

3.3.1 Performance requirements

- **Accuracy**

  Since we will give the priority to the accuracy of the software, the performance of the Music Recommender will be based on its accuracy on recommendations.
- **Speed**

  The system should generate and provide personalized recommendations to the users in a reasonable time.

3.3.2 Design constraints

- <u>Hardware Constraints</u>

The system will be integrated with a website. To use recommendation system, user should enter from a personal computer, mobile device with internet connection, tablet etc.

- <u>Software System Attributes</u>
  - ➔ Usability

The software will be embedded in a website. It should be scalable designed to be easily adopted by a system.

➔ Reliability

The system should have accurate results and a fast response to user’s changing habits. It also should work properly for a reasonably long amount of time. The probability of self-induced failure must be low.

➔ Security

User profile information will be used, so data security is one of the most important concerns of the system.

4. Data Model and Description

This section describes information domain for the Music Recommender.

4.1 Data Description

Data objects in the Music Recommender that will be managed/manipulated by the software are described in this section with their attributes using class diagrams.

4.1.1 Data objects

Music recommendation system roughly has 5 types of data objects, namely User, Item , Recommendation, UserManager, EvaluationMeasures. The data is represented as a graph in the Neo4j database. So, we can represent the entities and relationships by using graph property model as shown in the Figure 4.1.1.1 below.

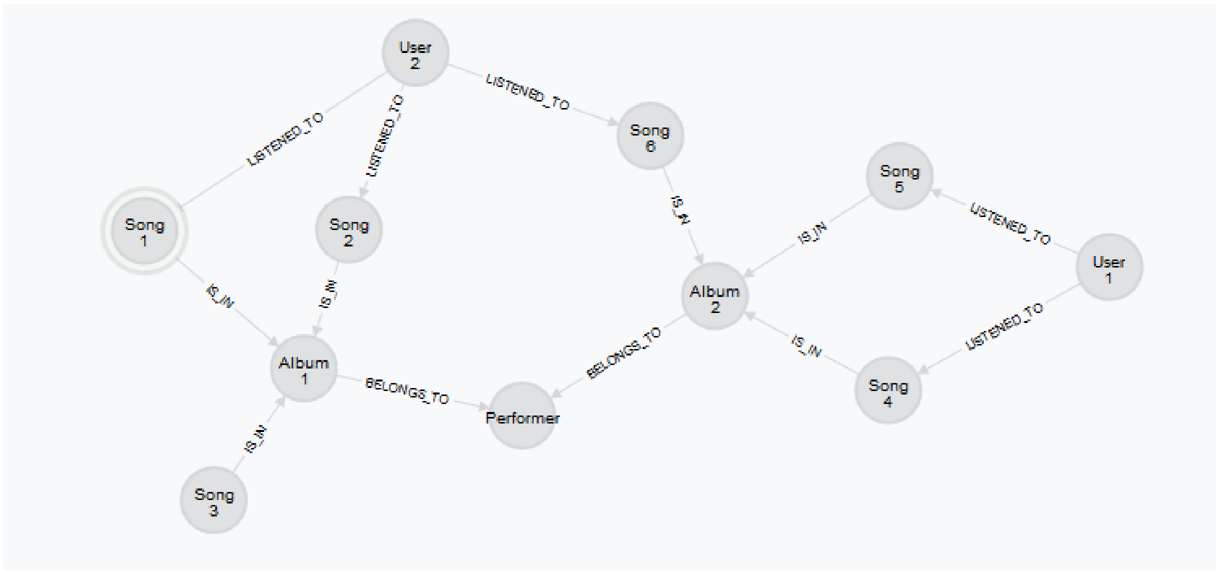

Figure 4.1.1.1: Graph Property Model of Data Objects

4.1.1.1 User

This object will hold the information of a specific user; id, name, login information etc.

User
UserID:int LoginDate: date LoginTime: time
getUserID()

Table 4.1.1.1.1 User Class Diagram

4.1.1.2 Item

This object will hold information about each specific music item; id, name, artist, album, genre etc.

Item
itemID:int providerID:int createTime:time itemName:String itemRating:float

getItemID()
getProviderID()
getCreateTime()
getItemName()
getRating()

Table 4.1.1.2.1 Item Class Diagram

4.1.1.3 Recommendation

This object will hold a specific rating value that a user gave to a item, including the required information of the user and item along with a rating value.

Recommendation
predictedRating:float averageRating:float itemReview:float

setPredictedRating()
serAverageRating()
getItemReview()

Table 4.1.1.3.1 Recommendation Class Diagram

4.1.1.4 UserManager

This object handles the user addition and deletion operations.

UserManager
userManagerID: int

getUserManagerID()
addUser()
deleteUser()
deleteUser()

Table 4.1.1.4.1 UserManager Class Diagram

4.1.1.5 EvaluationMeasures

This object handles the operations about evaluating the data.

EvaluationMeasures
precision:float recall:float mse:mse

getPrecision()
getRecall()
getMse()
getMap()

Table 4.1.1.5.1 EvaluationMeasures Class Diagram

4.1.2 Data dictionary

We have given details about description of object attributes at section 4.1.1. Therefore, it is not necessary to mention about them again.

5. Behavioral Model and Description

5.1 Description for software behavior

The recommendation system, also can be named as a recommendation session for a specific user, starts whenever a user logs in to the website. After that, it will start the recommendation generating procedures. It connects to and sends queries to the database server, to retrieve and compute the returned results to generate user specific recommendations. The system presents these results to the end user by using GUI of the system that recommender system is embedded in. Whenever a user selects one of these songs, the user-specific data will be updated and the system will generate new recommendations. Also, if the user does not select any of the recommended songs, the system will automatically request for new recommendations. This loop will be performed until the user logs out from the website, and the user-specific recommendation session closes.

5.2 State Transition Diagrams

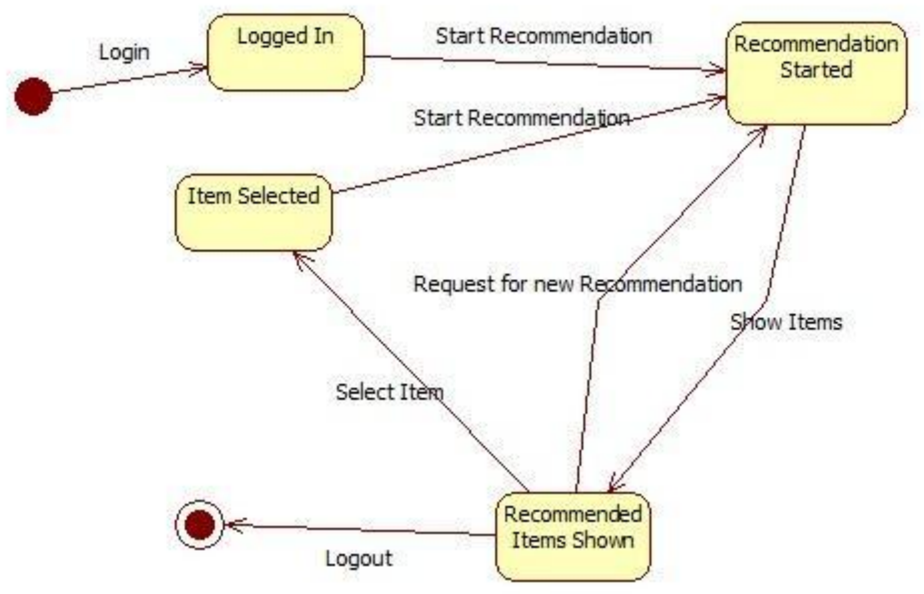

Figure 5.2: State Transition Diagram of System

This diagram shows how the system modes are altered during a session.

6. Planning

In this part of the document, the structure of the team responsible from the project, the basic schedule, and the process model will be presented.

6.1 Team Structure

We plan to divide the workload equally at the technical side. To contact academic staff and AGMLab, to search for project competitions and to write paper, we will do job sharing according to our field of interests. The basic structure of workload of team as follows:

Asena Ok: Algorithms, contact academic staff, research challenges

Aybüke Taşdirek: Algorithms, contact academic staff, research challenges

Birant Altinel: Database, contant AGMLab, academic research

Nihal Tarkan: Database, contant AGMLab, academic research

6.2 Estimation (Basic Schedule)

Task Name	November	December	January	February	March	April	May
Project Planning	2 week						
Learning and Improvement	2 weeks	1 week					
Learning Tools		3 weeks					
Design			3 weeks				
Algorithms			1 week	3 weeks			
Coding &Database Designing				1 week	4 weeks	4 weeks	
Testing							2 weeks

6.3 Process Model

In this project, spiral model, as can be seen in Figure 6.3.1 will be used.[6] The spiral model is a software development process combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts.

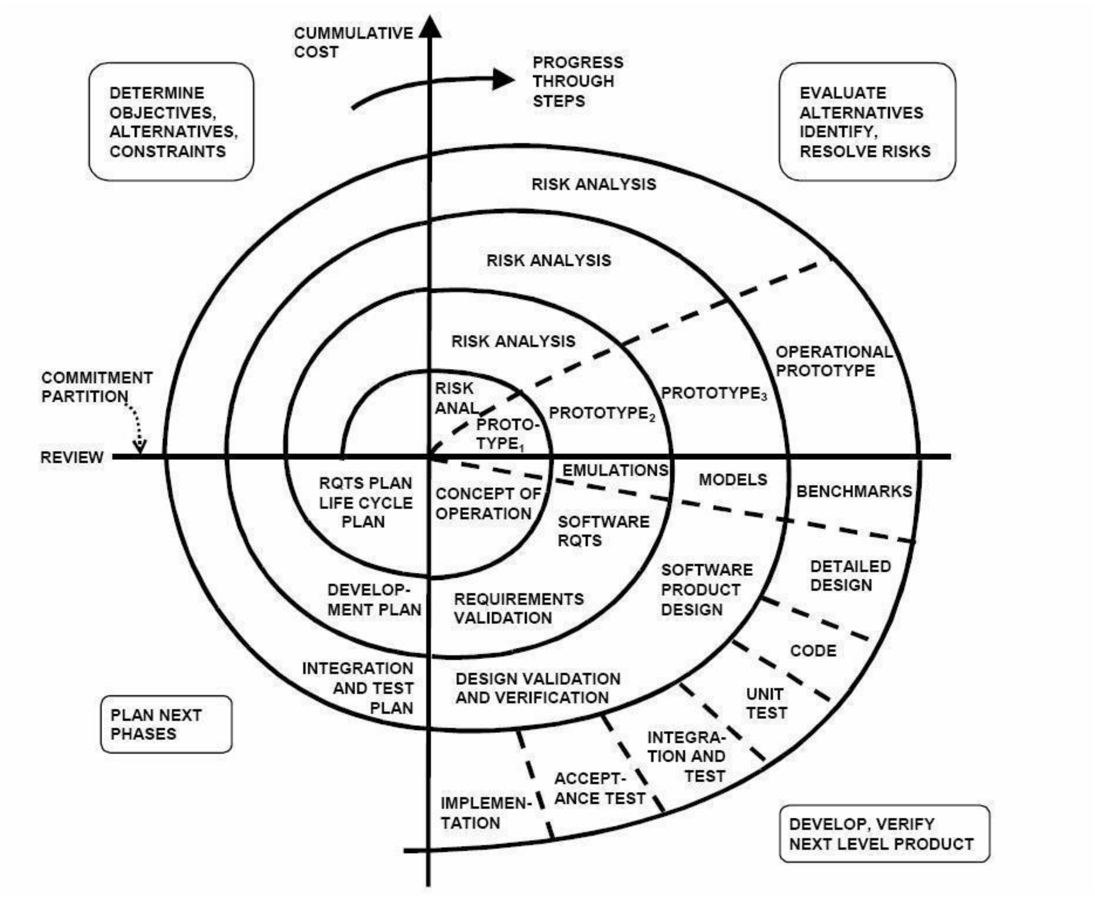

Figure 6.3.1: Spiral Diagram

Retrieved from: http://www.reliablesoftware.com/weblog/blogger.html

7. Conclusion

Software Requirement Specification is an important part of the process of project development. Moreover, it is a prerequisite for creating the following design documentation. In this document, we have provided the information about general product description, data elements that the product deals with, specific requirements like product's interfaces and the functions will be implemented. In addition general behavior of the product has been explained in order to make it easy for the customer to understand to product's usage clearly. This document has been created through the help of various researches and depending on the demands of the customer. However, some little specifications are prone to be changed in the future.