SOFTWARE DESIGN DESCRIPTION
OF
MUSIC RECOMMENDATION SYSTEM

CENG HISTORY X

HACER NİHAL TARKAN

AYŞE AYBÜKE TAŞDİREK

ASENA OK

BİRANT ALTINEL

PREFACE

This document contains the system design information about Music Recommendation System. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE Std 1016 – 2009”.

This Software Design Documentation provides a complete description of all the system design and views of the project.

The first section of this document includes Project Identification, Audience Identification and requirements, Composition of the developers’ team. The following sections include document purpose and design viewpoints of the system.

TABLE OF CONTENTS

1. Overview

This document is composed of five sections to state the detailed design of the project. The first section is an overview to document. The second section gives some definitions of the system. In the third part conceptual model for software design descriptions will be given. Fourth part includes the design description information content. Finally, in the fifth part design viewpoints will be explained.

1.1 Scope

This document contains complete description of the design of the Music Recommendation System. It consists of class diagrams, association of the classes, sequence diagram to show work flow of the some critical parts of the system, deployment diagram to show physical parts of the system. The design views incorporated are explained in depth and justified for use within this document.

1.2 Purpose

This software design document aims to provide information about the design details of Recommendation System Project. To show the different parts of the system from different viewpoints, component diagram, use case diagram , class diagram, deployment diagram are used. This document will show how the system is composed from. While showing bottom level of elements of the system, document will show their construction purpose, and different kind of static relationship that exist among them. In addition, interaction of the different types of users will be included by describing the properties of the related user interface and showing representative screenshots of the interface.
The document presents a number of different design views to depict different aspects of the system. It is intended to capture and convey the significant design decisions which have been made on the system.

1.3 Intended audience

This document is intended for both the stakeholders and the developers who build the system.

1.4 References

- IEEE. IEEE Std 1016-2009 IEEE Standard for Information Technology – System Design – Software Design Descriptions. IEEE Computer Society, 2009.

2. Definitions

SDD	Software Design Document
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.
Windows	Windows is a series of graphical interface operating systems developed, marketed, and sold by Microsoft.
Apache Mahout	Apache Mahout is a project of the Apache Software Foundation to produce free implementations of distributed or otherwise scalable machine learning algorithms focused primarily in the areas of collaborative filtering, clustering and classification.
Eclipse	A multi-language Integrated development environment (IDE).
GUI	Graphical User Interface
SRS	Software Requirements Specification

3. Conceptual Model for Software Design Descriptions

To understand this document better as a developer, basic knowledge of database management is needed.

3.1 Software Design in Context

This project will be designed using graph database features and object oriented approach. In this structure, the system can work more efficient with respect to recommendation relevancy. According to creation of data, the system can be used for different items different from music. Adaptability is very important for the project.

3.2 Software Design Descriptions within Life Cycle

3.2.1 Influences on SDD Preparation

The key software life cycle product that drives a software design is typically the software requirements specification. The requirements in the SRS (product perspective, functional and non-functional requirements and interface requirements) and also the demands of the stakeholders will specify the design of the project and design constraints to be considered or observed.

3.2.2 Influences on Software Life Cycle Products

The SDD affects several major software life cycle work products' contents. Design decisions, or design constraints observed during the preparation of the SDD, can cause some changes in requirements. One method of recording the relationship between requirements and design is to maintain traceability between these two. SDD can also affect test plans of the project. The content of the SDD can be taken into consideration by the development of test cases and test procedure.

3.2.3 Design Verification and Design Role In Validation

Verification and validation will be tested after preparation of the test cases. All system parts will be tested against these cases. It will be checked for whether the requirements are fulfilled or not.

4. Design Description Information Content

4.1 Introduction

Software Design Description of Music Recommendation System identifies how this system will be designed and implemented. Music Recommendation System will be designed in graph database features and object oriented approach. Java programming language will be used in object oriented approach. Neo4j graph database will be used to represent and store data. Also a qualified interface will be designed. Graph database is known for its time efficiency. To perform algorithm to make recommendation, we will study on objects' attributes. Therefore, that's the reason we use graph database and object oriented approach.

4.2 SDD Identification

After testing for the verification and validation, Music Recommendation System can be used for its original data that comes from ArgeDor. This system can be used for music recommendation using different data sources or some other suitable kind of recommendation in a different context in the future.

4.3 Design Stakeholders and Their Concerns

The music recommender system's stakeholders are the companies that serve music to listeners through an online system. Stakeholder's possible concerns are accuracy and efficiency in matters of memory and time. Put differently, the recommender engine perceive user's inputs from it's stakeholder and return multiple outputs that should be useful for the user. Moreover, because of huge amount of input data, the recommender is expected to manage this data properly and produce recommendations swiftly.

4.4 Design Views

This project will be implemented as modular structure. It is useful for stakeholders, because they can add new features or remove the modules they do not need from the project. Object-oriented approach is also necessary, because we will be dealing with data that have some attributes as objects. Moreover, this project is mainly focusing on database management. Therefore, object-oriented design is necessary. The recommendation engine will be working on the background. When a user enters to his/her account, he/she will not aware of the system working to make a recommendation. Recommendations will be displayed, while the user is performing different actions in his/her account. Then, after some time, if the user does not select any of recommended musics, the engine will generate new recommendations. By this way, user can get more recommendation and system can notice user's favors more accurately. This view shows the flow of staffing that will be made to make the system work. Content of the recommendation system is specified and requirements are given in the SRS document. Team organization and scheduling are also made. In the following sections a logical view of the product is explained and also is supported by diagrams. Class diagrams are given to show the relationships between objects. Dependency views show possible future problems and show the storage of the information. Interface views clearly specify which inputs give what outputs.

4.5 Design Viewpoints

The Context viewpoint shows the expectations from the user by giving reference to an explicit context. That explicit context defined is actually specified by reference to users and stakeholders. The roles of the users and stakeholders are defined clearly. The relationship between the input and the output will be explained in context viewpoint-design section. In the Composition viewpoint the relationships between components of the system are explained. Specifications and assignments of work packages are indicated. This viewpoint can be used for estimating cost, staffing, and schedule for the development effort. Logical viewpoint shows the class structure, interaction between classes, design and implementation of classes, and relationship between classes and interfaces. In Dependency viewpoint, the subsystem of the system and interconnections between these subsystems are defined.

4.6 Design Elements

In this part of the document, technical tools are explained. As database, Neo4j will be used. It has several advantages over other databases. It is cost efficient in terms of time and make it easier to construct relationships between system elements. These make Neo4j useful for recommendation systems. Moreover, to raise accuracy the algorithm needs to reach an items from different relations. For this purpose, using graph database will be more logical for the project.

There will not be any foreseen library in the system. Developers will design and implement background of the recommender from scratch.

In the Design Viewpoint part, detailed explanation about components, subsystems and modules will be provided.

4.7 Design Rationale

By taking into consideration some important features such as performance, maintainability, generalizability of the system, we agreed on how we will design the project. On the development side, it is important to traverse all nodes in the database swiftly and reach the desirable data. Therefore, while designing the database we will use indexing future and ‘Path’ API of Neo4j. We cannot test our system for a music site practically. Therefore, to present the outcomes of the recommender engine and to measure and show accuracy of the project, we are planning to design a basic GUI. On the algorithm side, it is very difficult to contribute literature. Already known algorithms will be sufficient for the project, to implement them properly is a challenge for the developers of the system.

We will write our code in an easily understandable way by adding comments. So, it will be easy to understand which function is doing what. This will also facilitate to modify the code according to stakeholders' demands.

4.8 Design Languages

We chose UML to represent schemes and modeling techniques to develop, analyze, and document the system design. We also used the built-in visualization feature of Neo4j to show the data structure of the graph database.

5. Design Viewpoints

5.1 Introduction

In this part of the document 5 main design viewpoints will be explained in detail:

- Context Viewpoint
- Composition Viewpoint
- Logical Viewpoint
- Dependency Viewpoint
- State Dynamics Viewpoint

All of these viewpoints are explained with supported UML diagrams.

5.2 Context Viewpoint

The purpose of the Context Viewpoint of Music Recommendation System is depicting the services provided by the system. We can look at utilities that the system provides from both side, user and stakeholder.

Provided functionalities to user are mainly divided into two categories. First category is user authentication category. This category is not directly associated with the recommendation. However, it is associated with the time that recommendation system should work. The second category is selecting an item that is music. This functionality provides data that can be used for future recommendations.

Provided functionalities for stakeholders or in other words suppliers are also divided into two categories. The first one is data management use cases that provide modifying data. The second one enables the supplier to modify recommendation algorithm. Context is defined by reference to the actors who will interact with the system. That is users and stakeholders based context is defined.

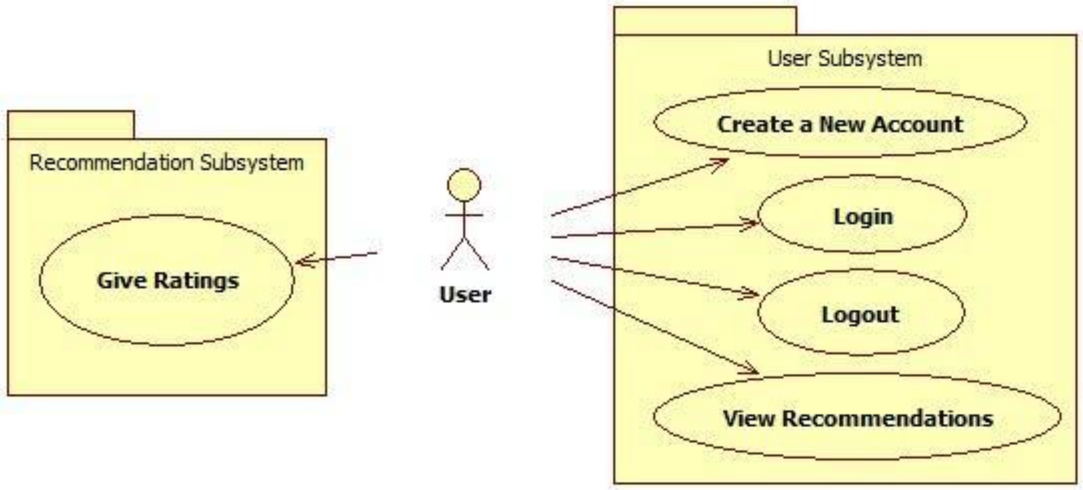

Figure 1: User Use Cases

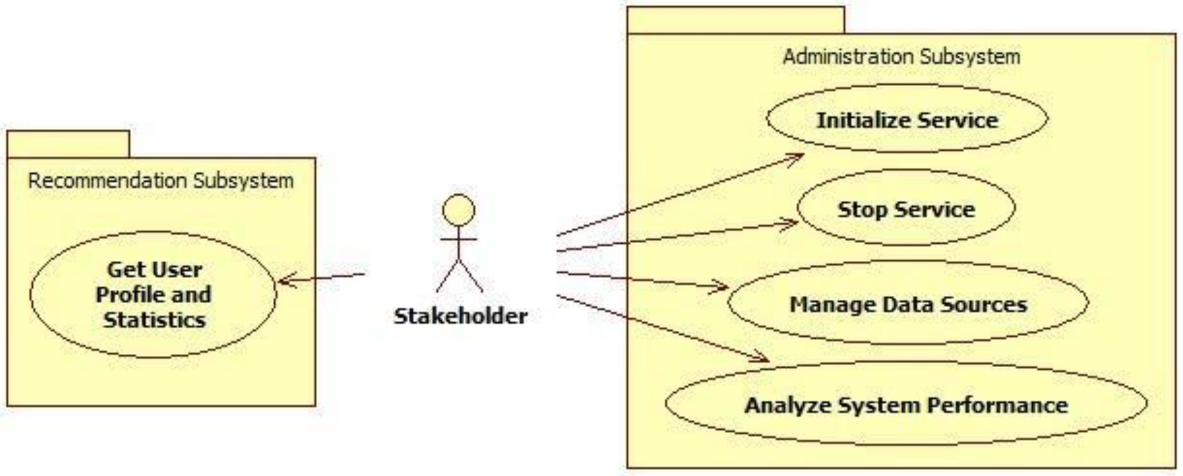

Figure 2: Stakeholder Use Cases

The use cases were explained in SRS document. However, here they are indicated simpler than the ones in SRS document. If necessary to look at the use cases briefly, we can look at the list below.

- **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.

- **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.
- **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.
- **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.
- **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.
- **Initialize service:** This functionality allows the administrator of the system to initialize the recommendation service.
- **Stop service:** This functionality allows the administrator of the system to stop the recommendation service.
- **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.
- **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.
- **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.

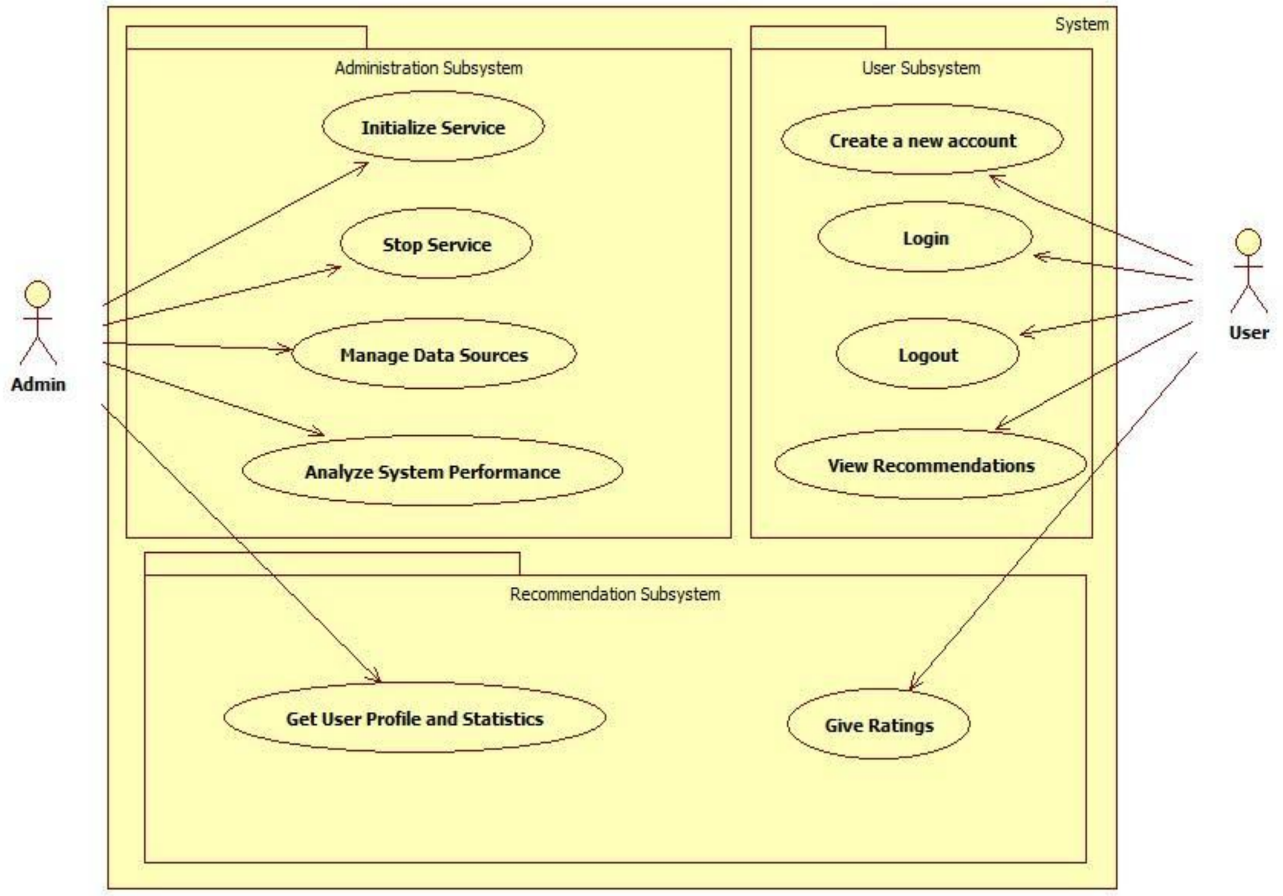

Figure 3: General Use Cases

5.3 Composition Viewpoint

The purpose of the composition viewpoint of Recommendation System is to define the system as a composition of its subsystems. The project is formed by 4 main submodules: Database, Recommender, Evaluation and End User GUI. Detailed explanation about the relations between these modules is given below:

- “Recommender module”, while calculating the accurate recommendation, uses the data stored in “Database module” via Neo4j api.
- Recommender module executes the methods on background and by connecting to the GUI module, via RecommendationRetrieving interface, the resulting recommendations is shown to the user. Thus, “recommendation module” is directly connected with “GUI module” simultaneously.

- Recommendation system has also its own “evaluator module” inside, thus “recommender module” of the project also is connected to evaluator module by EvaluationRecommendation interface. This interface of evaluator module gets its input data from the already given recommendations of the recommender module. Evaluator module specifies the quality of the recommendations of recommender module with respect to decided criteria such as accuracy and speed of the recommendation.
- Finally, since the evaluator module’s outputs affect the resulting recommendation shown on GUI, the “GUI” and the “Evaluator modules” are also connected indirectly.

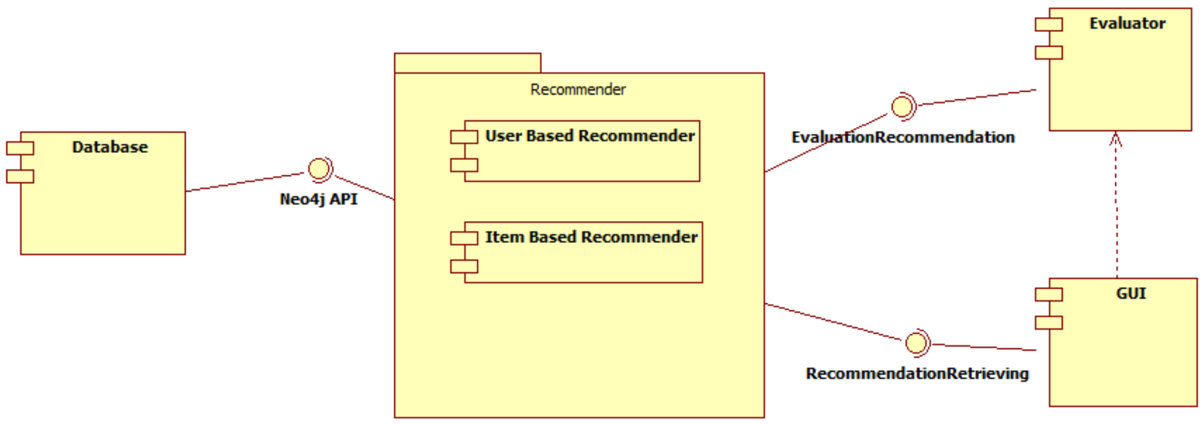

Figure 4.1: Component Diagram

All modules of the system and their internal relations will be explained clearly in the Dependency Viewpoint. Also deployment of these modules is explained in the below diagram:

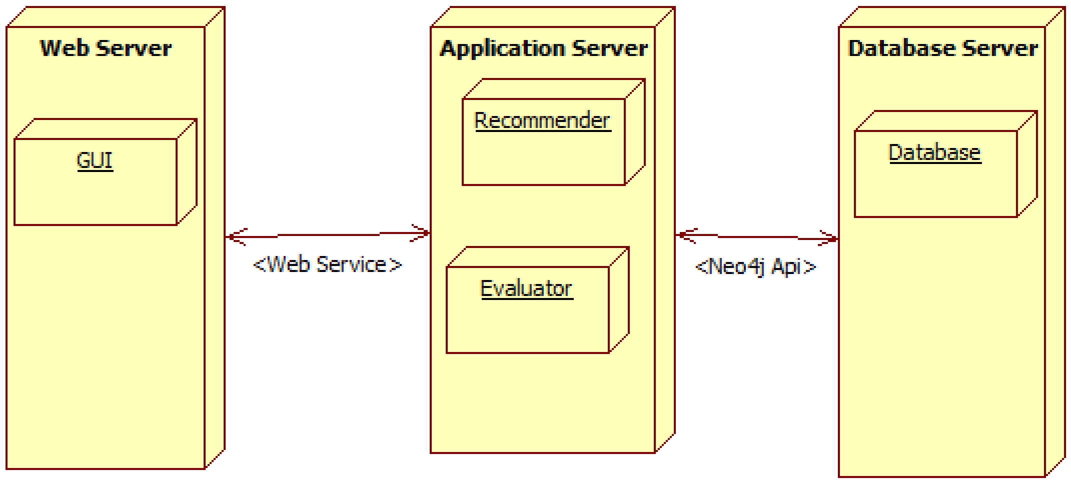

Figure 4.2: Deployment Diagram

5.4 Logical Viewpoint

In this viewpoint, the classes that will be used in the project are explained with their attributes and methods. For each class, there will be a diagram to overview the class and then tables that name, return type, visibility of the class diagram are shown in. Also, definition of each class element is provided. After all classes are explained, the class diagram that shows relationships between the classes are drawn.

This project is based on the data that is provided by ArGeDor. Therefore, in this part of the document, the data that will be used during implementation and test processes is explained also. The recommendation system includes 9 classes; User, Song, Album, Recommender, ItemSimilarity, UserSimilarity,TopItems, RecommendedItems and Performer classes.

5.4.1 User Class

This class is used to keep user’s information.

Diagram

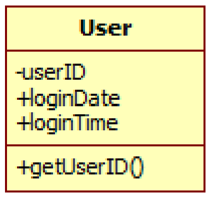

Figure 5: User Class Diagram

Definitions

Name	Type/Return Value Type	Visibility	Definition
userID	int	private	This variable defines unique ID for each user
loginDate	date	public	This variable defines listening date of song
loginTime	time	public	This variable defines listening time of song
getUserID()	userID	public	This function is used to get IDs of user which is a private variable

5.4.2 Song Class

This class is used to keep song information.

Diagram:

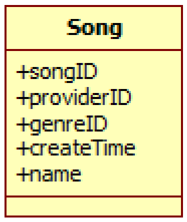

Figure 6: Song Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
songID	int	public	This variable defines unique ID for each user

providerID	int	public	This variable defines unique ID for each provider
genreID	int	public	This variable defines unique ID for each genre
createTime	time	public	This variable defines the time that data is created
name	string	public	This variable defines name of the song

5.4.3 Album Class

This class is used to keep album information.

Diagram:

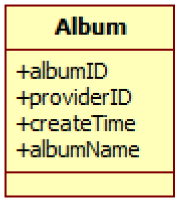

Figure 7: Album Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
albumID	int	public	This variable defines unique ID for album
providerID	int	public	This variable defines unique ID for provider
createTime	time	public	This variable defines album’s creating time

albumName	string	public	This variable defines name of album

5.4.4 Recommender Class

This class is used to implement recommender algorithm. It connect users to items.

Diagram:

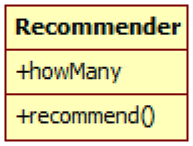

Figure 8: Recommender Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
howMany	int	public	This variable defines desired number of recommendations
recommend()	List<RecommendedItem>	public	This function is used to get list of recommended items.

5.4.5 ItemSimilarity Class

This class is used to compare two items and determine whether they are similar.

Diagram:

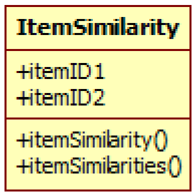

Figure 9: ItemSimilarity Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
itemID1	int	public	This variable defines the unique ID for item
itemID2	int	public	This variable defines the unique ID for item
itemSimilarity()	double	public	This function is used to get similarity between itemD1 and other items
itemSimilarities( )	List<int>	public	This function is used to get IDs of all similar items

5.4.6 UserSimilarity Class

This class is used to compare two users and determine whether they are similar.

Diagram:

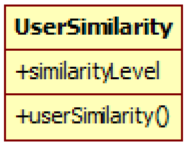

Figure 10: UserSimilarity Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
similarityLevel	int	public	This variable defines similarity level between users
userSimilarity( )	double	public	This function is used to get similarity between two users

5.4.7 TopItems Class

This class is used to specify most listened songs.

Diagram:

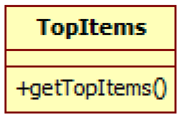

Figure 11: TopItems Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
getTopItems()	List<Song>	public	This function is used to get most popular items among users

5.4.8 RecommendedItem Class

This class is used to determine items that can be listened by a specific user.

Diagram:

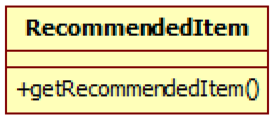

Figure 12: RecommendedItem Class Diagram

Definitions:

Name	Type/Return Value Type	Visibility	Definition
getRecommendedItem()	List<RecommendedItem >	public	This function returns IDs of recommended items

5.4.9 Performer Class

This class is used to keep the performer information.

Diagram:

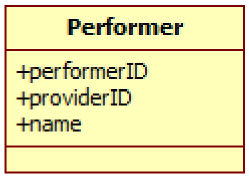

Figure 13: Performer Class Diagram

Definitions:

Name	**Type/Return Value Type**	**Visibility**	**Definition**
performerID	int	public	This variable defines unique ID for each performer
providerID	int	public	This variable defines unique ID for each provider
name	string	public	This variable defines name of the performer

5.4.10 Relationships between Classes

All these relationship are shown on Class Diagram in the Figure 14.

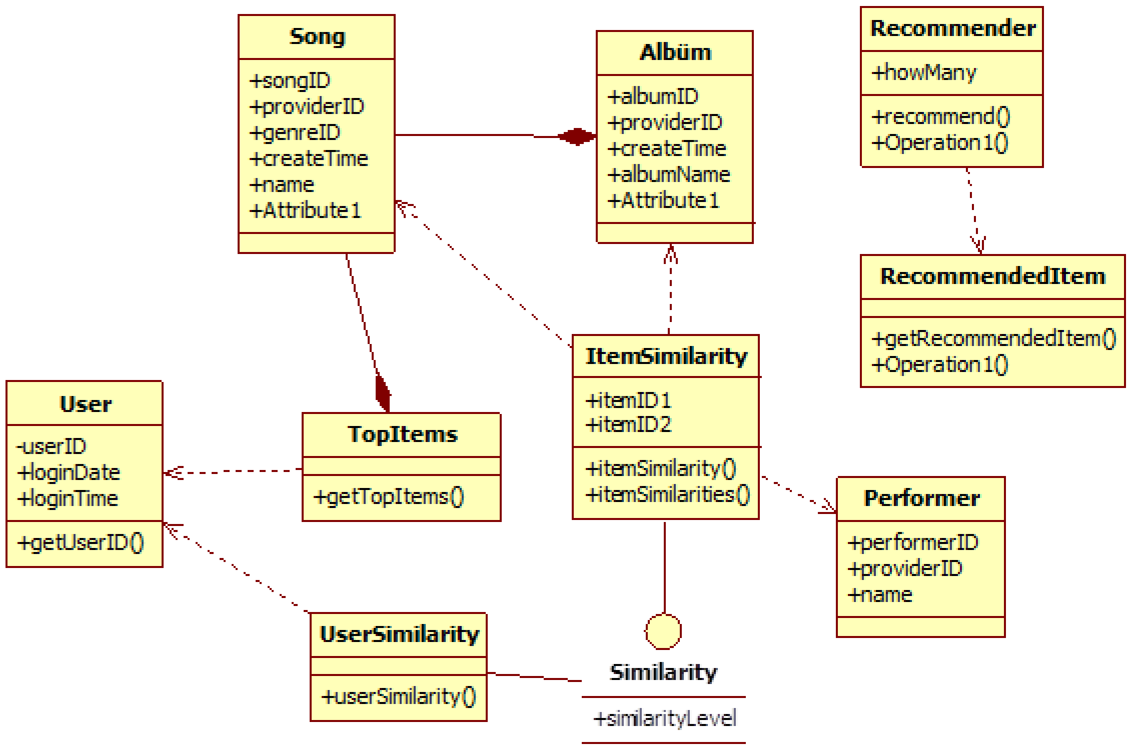

Figure 14: Relationships between classes

5.4.11 Data Structure in Graph Database

The structure of the data is shown here as it is stored in the graph database, including both nodes and relationships between them. The nodes User, Song, Album and Performer represent the corresponding classes that are shown on sections 5.4.1, 5.4.2, 5.4.3 and 5.4.9. They possess all the attributes that are explained in those sections also in the graph database, but the attributes are not shown in the visual.

Diagram:

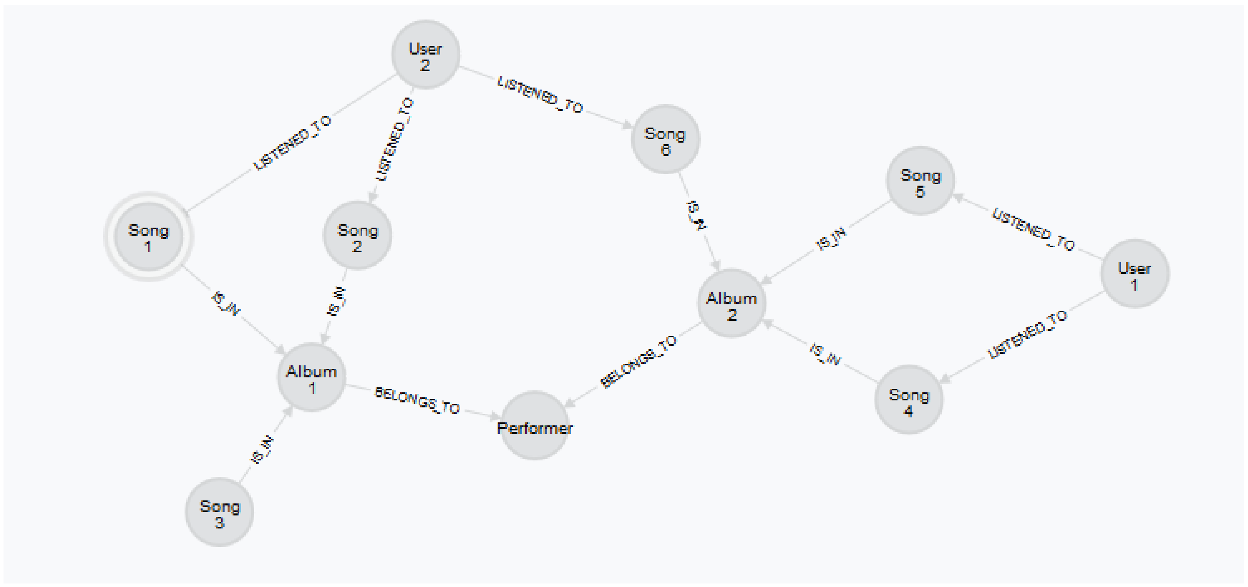

Figure 15: Property Graph Model

Definitions of the Relations:

Name	**Type**	**Definition**
BELONGS_TO	Relation	This relationship defines a relation between an “Album” node and a “Performer” node.
IS_IN	Relation	This relationship defines a relation between a “Song” node and an “Album” node.
LISTENED_TO	Relation	This relationship defines a relation between a “User” node and a “Song” node.

5.5 Dependency Viewpoint

In dependency viewpoint, the modules that will be used in the project are explained with their submodules. For each module, there will be detailed explanation.

5.5.1 Recommendation Module

This module is the main module of the project. It gathers data from Neo4j database using an interface and process the data via recommendation algorithms in it. Considering the data that is given by ArgeDor, only item based and user based recommendations can be performed.

5.5.2 Evaluation Module

As stated before, the concern of the project is to provide accurate recommendations to users. Accuracy will be our base point. Therefore, while evaluating the system, to get accurate recommendations is crucial for the project. How to evaluate the recommendation system in manner of accuracy? This is the question that will be answered in this section.

Data that is provided by ArgeDor will be divided into 2 part. First part, training data, is 10% of total the data. Accuracy of recommendations produced by training data is known. Second part includes 90% of the data. According to the known result of first part of the data, results from the second part will be evaluated.

5.5.3 Database Module

The Database Module is the module that all the data of the software will be kept. It will be containing a graph database (neo4j) and is highly interconnected with the Recommendation Module. It will be mainly accessed and used by the Recommendation Module. Recommendation Module will send queries to this module. The queries will be run and generated results will be provided to the Recommendation Module.

5.5.4 User Interface Module (GUI)

The GUI part is not intended to design for end users. The purpose is to show the results of the recommender system to the stakeholder. Also, GUI part will be connected to the evaluation part. Stakeholders will see the expected results and the results produced by the recommendation systems via GUI. To clarify this point, follow the figure below:

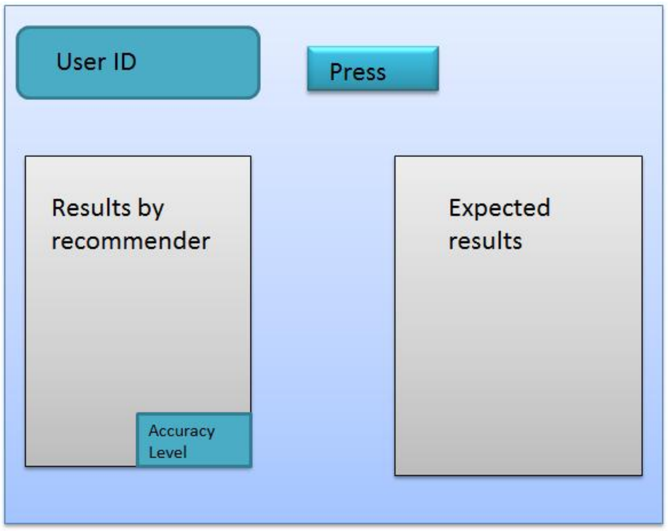

Figure 16: GUI

After enter a valid user ID, press the button. In the ‘Expected results’ section, top 10 songs that estimated by training data will appear. In the ‘Results by recommender’ area, estimation of top 5 songs that produced by recommender will be appear. ‘Accuracy Level’ indicates how much ‘Results by recommender’ close to ‘expected results’. It will be computed at the background.

5.6 State Dynamics Viewpoint

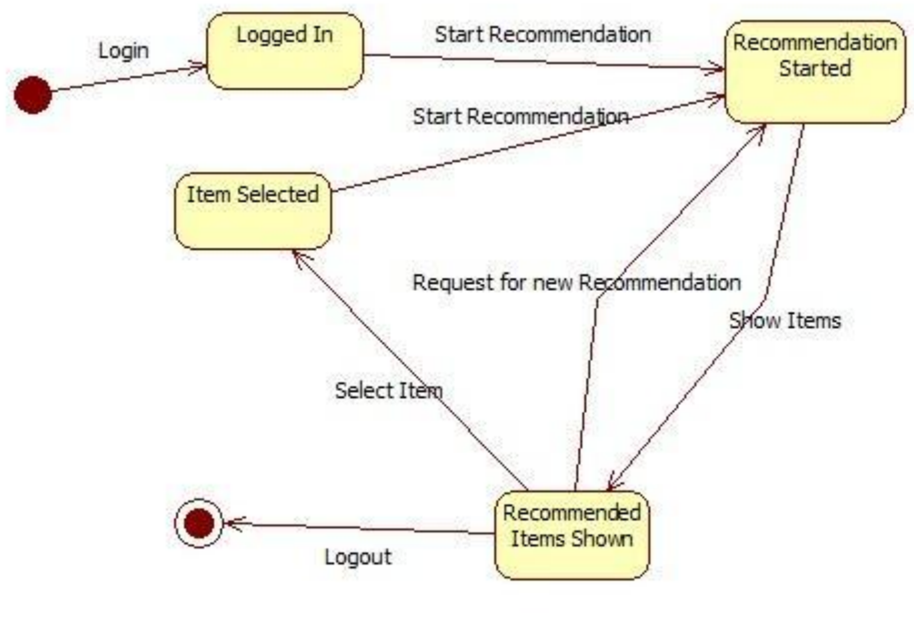

Figure17: State Machine Diagram

This diagram shows how the system works after a user logs in his/her account. Start recommendation is not actually done by the user, but login leads to start recommendation. Then, recommendations are shown on the user interface. User can select one of them. Whole flow of these states stops, when user logs out.

6. Conclusion

Implementation details of Music Recommendation System are provided into some extent in this document. Modules, data structure, design viewpoints are defined clearly. The tools and the stages of the design are also given. The first prototype of the Recommendation System will be developed by referring to this document.

7. Forthcoming Period Planning & Distribution of the Roles

Asena Ok: Recommender, Evaluator Module

Aybüke Taşdirek: Recommender, Evaluator Module

Birant Altinel: Database Module

Nihal Tarkan: Database Module, User Interface Module

Gantt Chart of Ceng History X: (Updated)

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