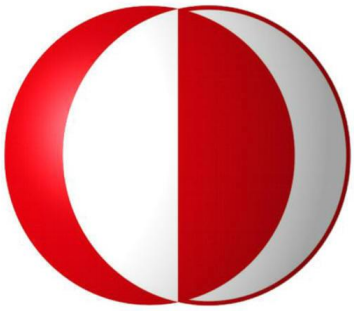
Middle East Technical University

Department of Computer Engineering



Music Recommender System

Software Design Description v1.1

MemoryLeak

Bahtiyar Onur Geyik, 1745975

Emre Ercan, 1745918

Gökçe Aksu, 1745694

Ege Bozkurt, 1745819

PREFACE

This document provides a system design description of the Recommender System. It 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 scope, purpose and the audience of the project.

The following sections include the conceptual model for software design descriptions, concepts about the design description information, design views, design viewpoints, design elements, and all other design related issues.

Last section specifies the design viewpoints for the whole system. Design viewpoints consist of context viewpoint, composition viewpoint, logical viewpoint, interface viewpoint and interaction viewpoint.

Contents

List of Figures

1. Overview

1.1 Scope

This Software Design Description describes the detailed structure of components of the Recommender System. It shows how the software system will be structured to satisfy the requirements. The precise implementation details for satisfying the requirements specified in the Software Requirements Specification are provided in this document. This document is prepared in compliance with the standard which represents a tailoring of the IEEE Std 1016 Recommended Practice for Software Design Descriptions. Therefore it will also comply with IEEE Std 1016.

1.2 Purpose

This document contains the complete design description of Recommender System and the design description defined in this document has multiple purposes. These are to identify required system resources, to describe the algorithms, data and functional structure, to assist in producing test cases and to aid in the maintenance activities. This documents includes the architectural features of the system down through details of what operations each code module will perform and the database layout. It also shows how the use cases detailed in the SRS will be implemented in the system using this design.

1.3 Intended Audience

Intended audience of the Software Design Description is not clients and users. It is the team working on the project, supervisor and reviewers. This document is prepared for knowledgeable software professionals and designers.

1.4. References

- IEEE. IEEE Std 1016TM-2009. IEEE Standard for Information Technology—Systems Design—Software Design . Software & Systems Engineering Standards Committee of the IEEE Computer Society, 2009.
- Sommerville, I. Software Engineering Ninth Edition. Pearson Education, Inc., publishing as Addison-Wiley. 2011.
- IEEE Std 12207TM-2008 Systems and software engineering—Software life cycle processes

2. Definitions

Term	Definition
SDD	Software Design Description
MVP	Minimum Viable Product
UML	Unified Modeling Language
Music Recommender	An information filtering system that seeks to predict the preference that user would give to a song for ARGEDOR Music Application.
JSON	JavaScript Object Notation.
ARGEDOR	Stakeholder
DCG	Discounted Cumulative Gain
SRS	Software Requirement Specification
MongoDB	MongoDB is an open-source document database, which is a NoSQL database.
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.
Java	Java is a object oriented computer

	programming language.
Hadoop	The Apache Hadoop software library is a framework that allows for the distributed processing of large data sets across clusters of computers using simple programming models.
JavaScript	JavaScript (JS) is an interpreted computer programming language.
NoSQL	A NoSQL database provides a mechanism for storage and retrieval of data that employs less constrained consistency models than traditional relational databases.

3. Conceptual Model for Software Design Descriptions

The conceptual model includes basic terms and concepts of SDD, the context in which SDDs are prepared and used, the stakeholders who use them, and how they are used.

3.1 Software Design in Context

This project will be designed as object-oriented design fashion. The recommendation system needs a complex mathematical model to generate accurate recommendation to the users. Since building a complex mathematical model is an evolving process and has many iterations, object oriented-design will ease managing these changes and iterations. Also, since this system will be a part of much larger system, it must be portable to this larger system. Moreover, the larger

system is a website that has the potential of increasing its number of users, user traffic and number of songs. Therefore, the recommendation system needs to be scale up with the larger system in the correct order.

3.2 Software Design Descriptions within the Life Cycle

For better understanding of this document, basic knowledge of databases and programming is required.

3.2.1 Influences on SDD Preparation

The key software life cycle product that drives a software design is typically the software requirements specification (SRS). The product perspective, functional and nonfunctional requirements and the interface requirements in the SRS specify the design of the project.

3.2.2 Influences on Software Life Cycle Products

In this project, the database that stores the user action logs is already designed in the larger system. Therefore, the mathematical model which will generate the users' ratings on items and the database which will store these ratings will be designed first. Then, the map-reduce algorithms should be implemented to process user action logs which is a large dataset. After that a basic recommendation algorithm should be designed. Thereafter, the UI will be designed and the MVP (Minimum Viable Product) will be ready for a pre-demo. Then, the mathematical model and the recommendation algorithm should be improved to generate faster and more accurate recommendations.

As it is explained in the website of Creighton University there are three common metrics, which are recall, precision and DCG (Discounted Cumulative Gain), to assess the quality of the

recommendation method. Recall is the ratio of the number of relevant records retrieved to the total number of relevant records in the database. It is usually expressed as a percentage. Precision is the ratio of the number of relevant records retrieved to the total number of irrelevant and relevant records retrieved. It is usually expressed as a percentage too. DCG measures the usefulness, or gain, of a document based on its position in the result list. The gain is accumulated from the top of the result list to the bottom with the gain of each result discounted at lower ranks. After each improvement of the mathematical model and the recommendation algorithm, these three metrics will be tested.

3.2.3 Design Verification and Design Role in Validation

Verification and validation test will be done after the preparation of the test cases. The recommendation system will be tested against these prepared test cases. Whether the system fulfill the requirements will be checked by these test cases.

4. Design Description Information Content

4.1 Introduction

Software Design Description of Recommender System identifies how this project will be designed and implemented. The design of Recommender System is a modular object-oriented structure and a practical interface will be designed to present the project.

4.2 SDD Identification

After testing for the verification and validation, at May, 2014, recommender system will be released. It will be used with Argedor Music application to make recommendations to its users. Also, it can be used for any website or application that provides a service that allows its users to

listen songs. To import this system to an application or website, activity logs that similar to Argedor Music application needs to be generated. This process, which is unrelated to the design of this system, is another system that will be developed by the team for marketing purposes.

4.3 Design Stakeholders and Their Concerns

The recommendation system stakeholders are online music streaming service companies. Stakeholders' possible concerns are accuracy, time-efficiency and scalability of the system. Therefore, the recommendations will be generated according to the users' tastes to be useful. Also, the recommendations will be generated fast, since the users are using the system in real-time. Moreover, the system will be able to handle large number of users and logs of those users.

4.4 Design Views

UML is used for representing diagrams of views. The information about UML can be reached from this site: http://tutorialspoint.com/uml/

4.5 Design Viewpoints

Contextual viewpoint shows what is expected from the user actor in the system. It includes use cases, a general diagram and explanation of each use case. Input-output relations will be explained in context viewpoint-design elements. Composition viewpoint describes the way the design subject is structured into constituent parts and establishes the roles of those parts. We have component and deployment diagrams in this viewpoint. Logical viewpoint shows the class structure, interaction between each other and how they are designed and implemented. Data description, ER diagram and file indexing structures are included in logical viewpoint.

Interface viewpoint is needed for the test cases. It includes the details of the user interfaces, the external and internal interfaces with hardware and software libraries. It provides information to designers, programmers and testers the means to know how to correctly use the services provided by the design subject.

4.6 Design Elements

Design elements and their attributes, relationships and constraints of this SDD are described in the following sections with the viewpoints to which they belong.

4.7 Design Rationale

Design choices are made according to the significant features, which are accuracy of the recommendations, time-efficiency and scalability of the system. Moreover, the system is designed and the functions in the system software will be written and commented in detail to increase the understandability so that the improvements involving those features will be done easily.

This project will be implemented with Java Programming Language. Since most of the technologies that will be used to implement this system are Java based technologies, Java is selected as the main programming language. Background of the team is also another reason why Java is selected. Since the size of the whole data will be very large, using a relational database will cause performance problems. Because of scalability and performance concerns, a NoSQL must be used in implementation. MongoDB is selected and will be used to hold activity logs and rating data. Mahout, which is a machine learning library that is used in HADOOP, will

be used to ease generating recommendations. Javascript will be used to implement the user interface.

4.8 Design Languages

Unified Modeling Language (UML) is selected as a part of design viewpoint specification.

5. Design Viewpoints

5.1 Introduction

Several design viewpoints in terms of design concerns are defined in following subsections. UML is used as a design language. Main design viewpoints examined in this section are context viewpoint, composition viewpoint, logical viewpoint, interface viewpoint and interaction viewpoint.

5.2 Context Viewpoint

Recommender system software context viewpoint shows the functions provided by the design. There are four major functionalities of the system which are login, logout, view history and view recommendations. The context is defined by the elements that interact with the software like users.

5.2.1 Design Elements

The user has the following use cases:

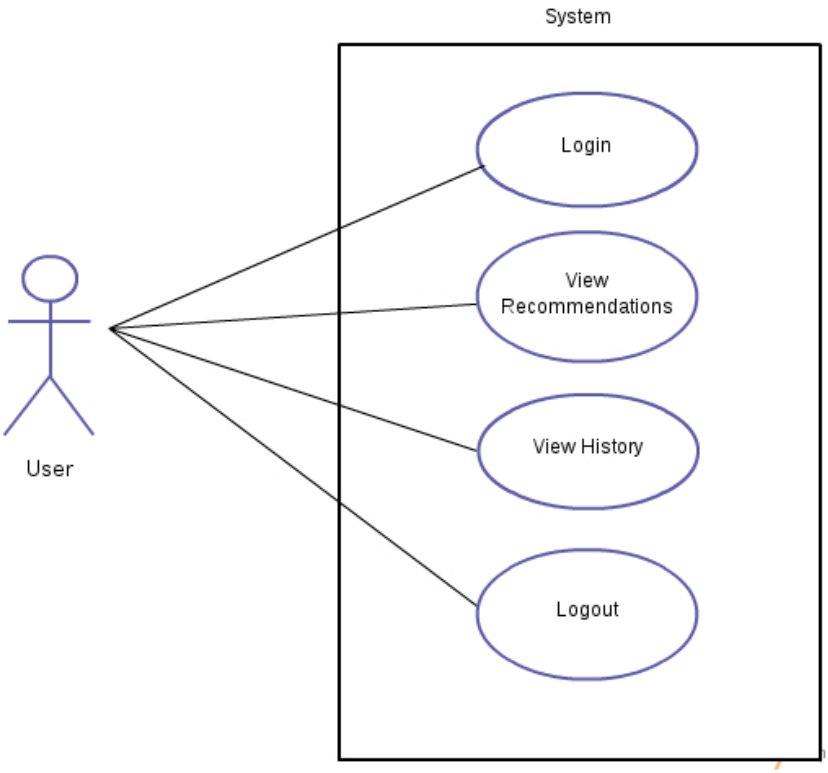


Figure1. Use Case Diagram

Descriptions and step by step definitions of use cases are explained in detail in SRS document.

5.3 Composition Viewpoint

5.3.1 Design Elements

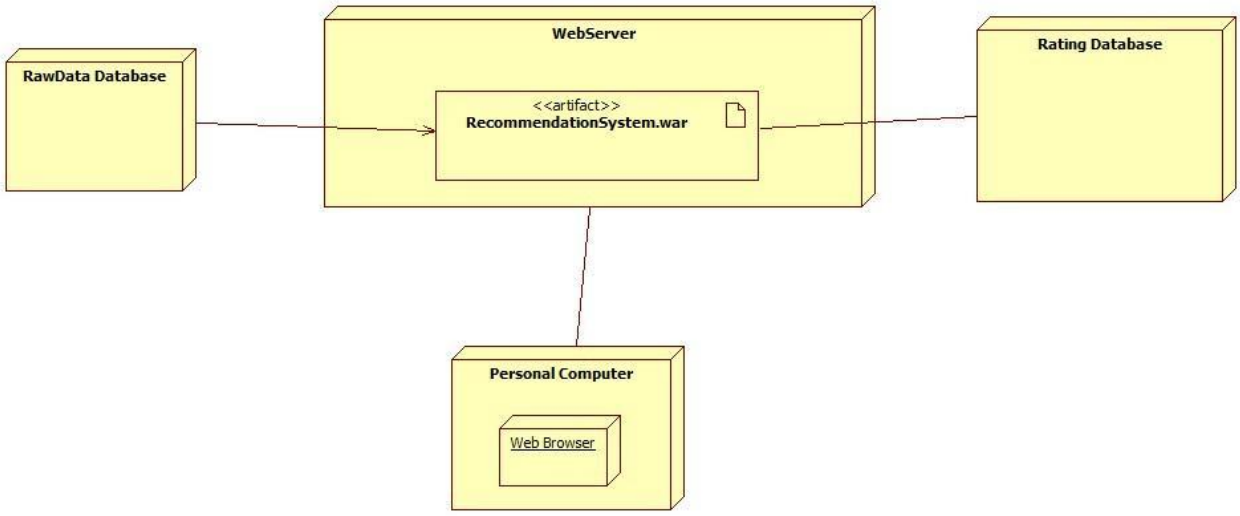


Figure2. Deployment Diagram

For demonstration concerns, this system will be implemented as a java web application. It will need a container server, most probably Apache Tomcat. This application will be a simple .war file. A raw data database which will be a MongoDB will be interacted with this application. This database holds activity logs. A rating database which holds rating information that is necessary for recommendation will also be interacted with the application. Users can access the application via a simple web browser. Actions of the users will be done via HTTP request and response.

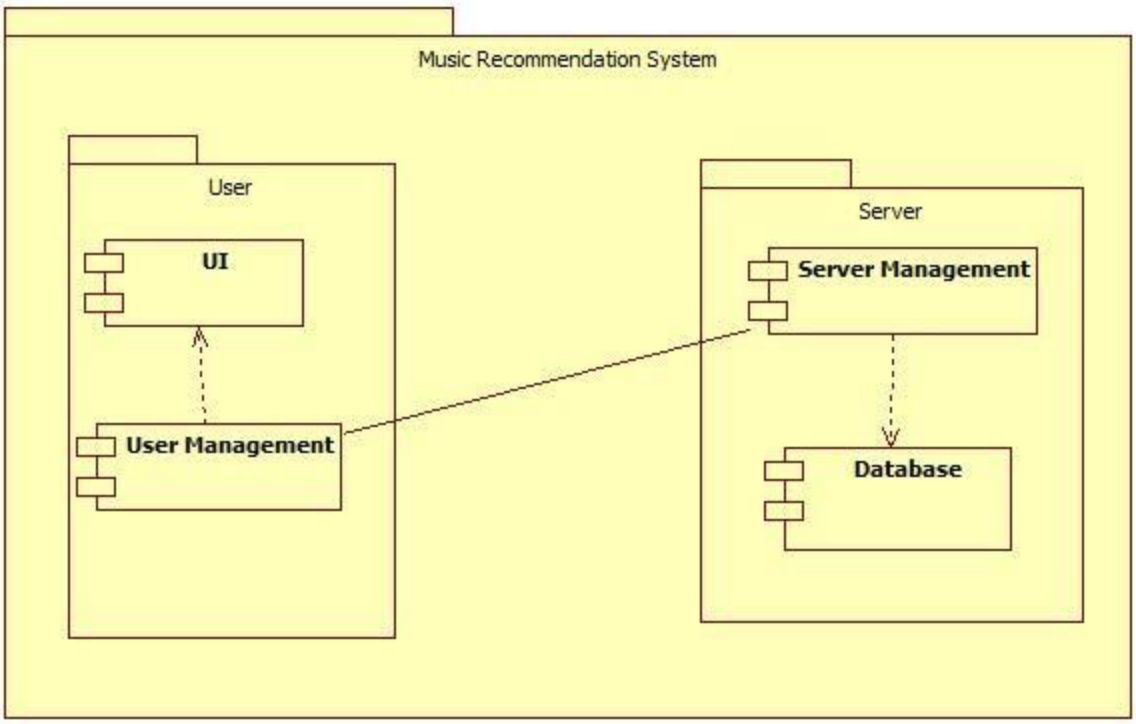


Figure3. Component Diagram

User package of the system will be implemented only for demonstration concerns as explained in detail in the interface viewpoint section. The real system will be a module that can be run on a server; therefore, this module can be imported to a related larger system as mentioned before.

5.4 Logical Viewpoint

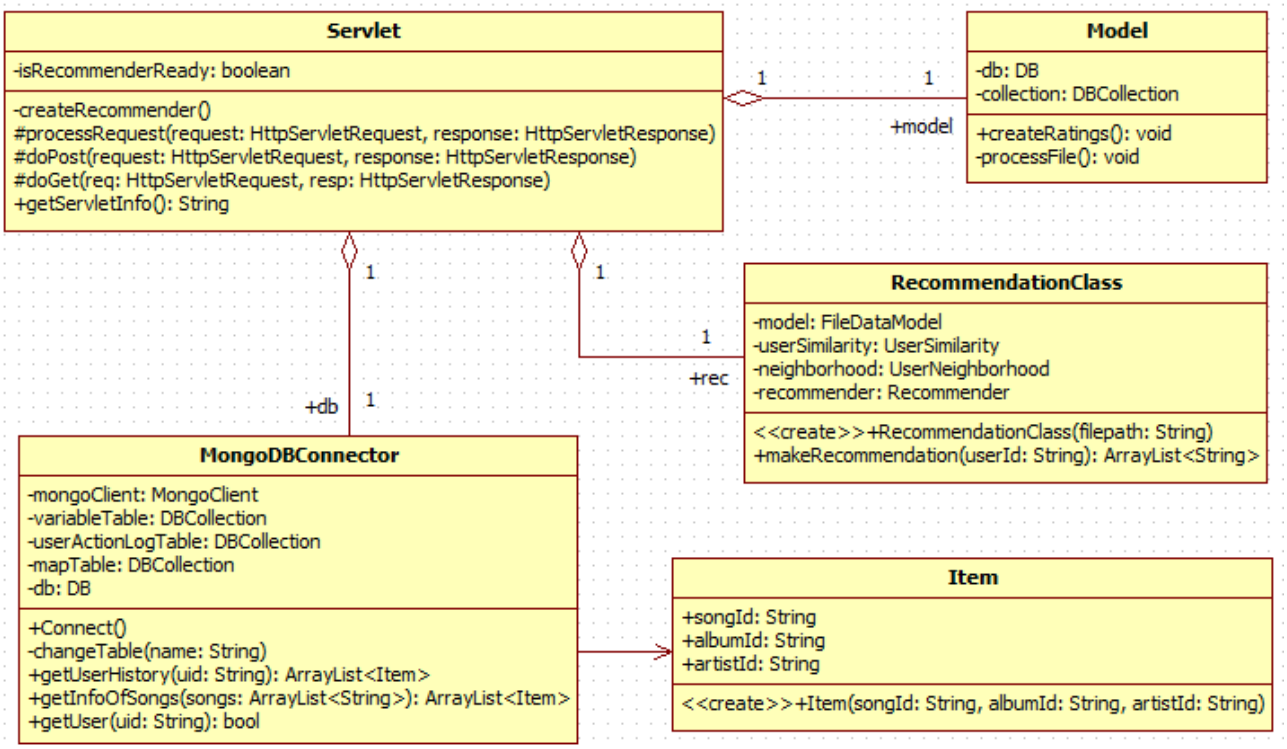


Figure4. Class Diagram

5.4.1 Servlet Class

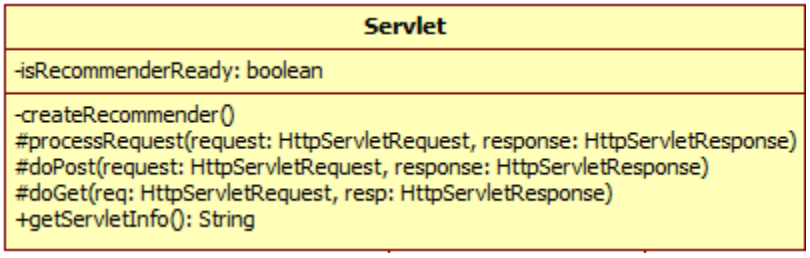


Figure5. Servlet Class

Name	Type / Return Value Type	Visibility	Definition

isRecommenderReady	boolean	private	True if the recommender is ready to make recommendations, false otherwise.
createRecommender()	void	private	Creates the recommender to make recommendations.
processRequest(request: HttpServletRequest, response:HttpServletResponse)	void	protected	Processes the request sent from the client-side
doPost(request: HttpServletRequest, response:HttpServletResponse)	void	protected	Used to postprocess a request
doGet(req: HttpServletRequest, resp:HttpServletResponse)	void	protected	Used to preprocess a request

getServletInfo()	String	public	Returns information about the servlet.

5.4.2 Model Class

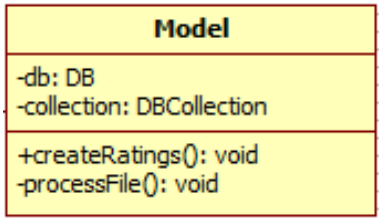


Figure6. Model Class

Name	**Type / Return Value Type**	**Visibility**	**Definition**
db	DB	private	Stores the database table.
collection	DBCollection	private	Stores a database collection.
createRatings()	void	public	Generates ratings for recommender.
processFile()	void	private	Helper function of createRatings.

5.4.3 MongoDBConnector Class

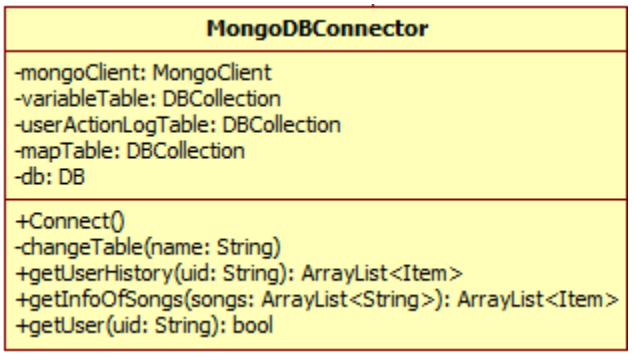


Figure7. MondoDBConnector Class

Name	**Type / Return Value Type**	**Visibility**	**Definition**
mongoClient	MongoClient	private	Database connection object.
variableTable	DBCollection	private	Stores a database collection.
userActionLogTable	DBCollection	private	Stores the database collection whose name is “user_action_log”
mapTable	DBCollection	private	Stores the database collection whose name is “map”

db	DB	private	Stores the database table.
Connect()	void	public	Connects to the database.
changeTable(name: String)	void	private	Makes variableTable store the database collection whose name is given.
getUserHistory(uid: String)	ArrayList<Item>	public	Returns the history of given user.
getInfoOfSongs(songs: ArrayList<String>)	ArrayList<Item>	public	Retrieves album id and artist id of given song ids and returns them in a list.
getUser(uid: String)	Boolean	public	Checks if the user is registered in the system.

5.4.4 Recommendation Class

This is the class which makes the recommendations to the users. It retrieves the rating data from RatingConnections class. Then, it processes the data and generates recommendations to the user using collaborative filtering.

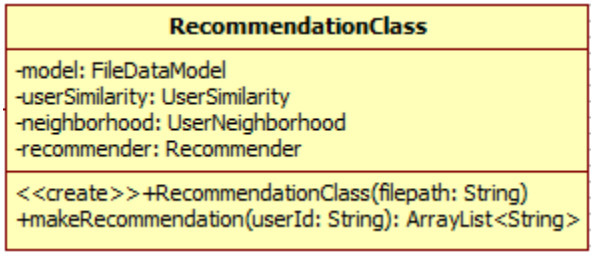


Figure8. Recommendation Class

Name	**Type / Return Value Type**	**Visibility**	**Definition**
model	FileDataModel	private	Data Model that reads and stores the (userId,songId,rating) values
userSimilarity	UserSimilarity	private	Stores the user similarity matrix.
neighborhood	UserNeighborhood	private	User neighborhood object to help recommender generate

			recommendations based on user similarity matrix.
recommender	UserRecommender	private	User based recommender to generate recommendations.
RecommendationClass(filepath: String)	void	public	Constructs the RecommendationClass based on the given rating file.
makeRecommendation(userId: String)	ArrayList<String>	public	Generates the recommendations to the given user and returns the recommended items in an ArrayList.

5.4.5 Item Class

This class is responsible for storing the song id, album id, artist id of an item.

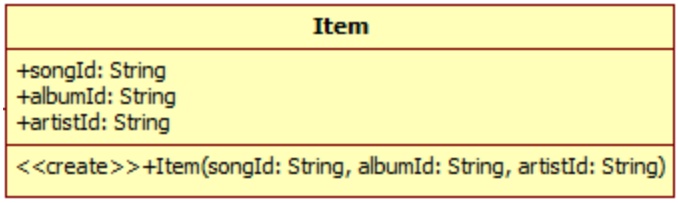


Figure9. Item Class

Name	**Type / Return Value Type**	**Visibility**	**Definition**
songId	String	public	The id of the song.
albumId	String	public	The id of the album which contains the song.
artistId	String	public	The id of the singer who sings the song.
Item(songId: String, albumId: String, artistId: String)	void	public	Constructor for the item

5.5 Interface Viewpoint

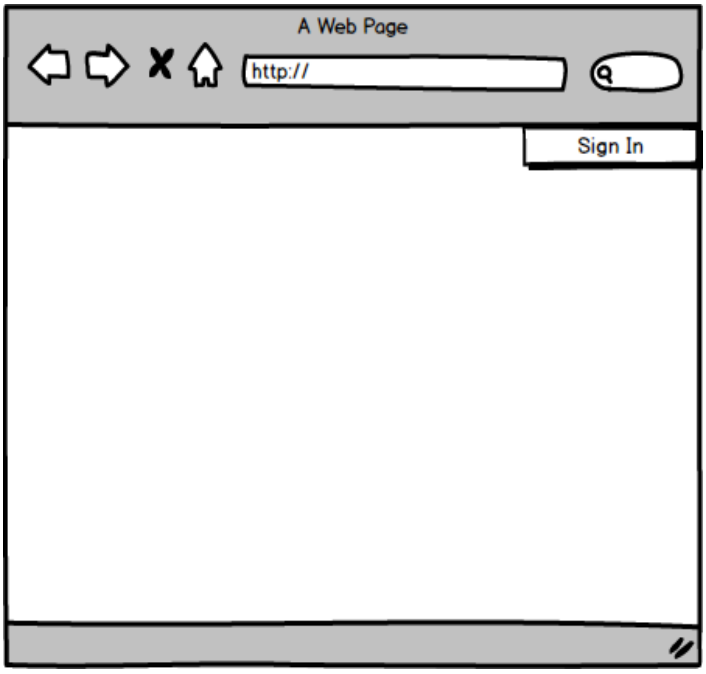
5.5.1 Design Concerns

A user interface is already designed in the larger system to show users generated recommendations. However, for demonstrating this system not only in this specific larger system but also another possible buyers or customers, a simple user interface will be designed and used.

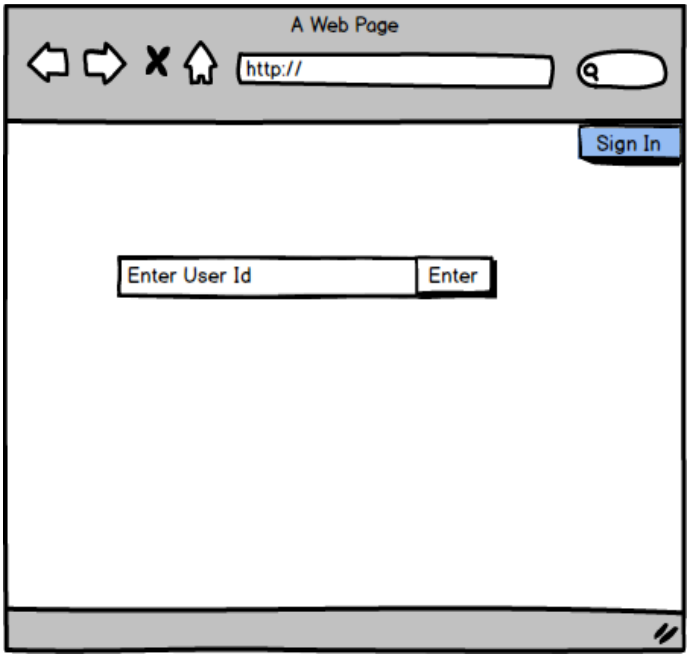
Since it needs to be a web application to be able to demonstrate the system, a simple interface will be designed on top of this very simple application.

Users will enter a user id to reach the main page of their listening history and featured recommendations. Since every user has a listening history, one or more songs, featured recommendations will be shown as a list right under the users' listened song list.

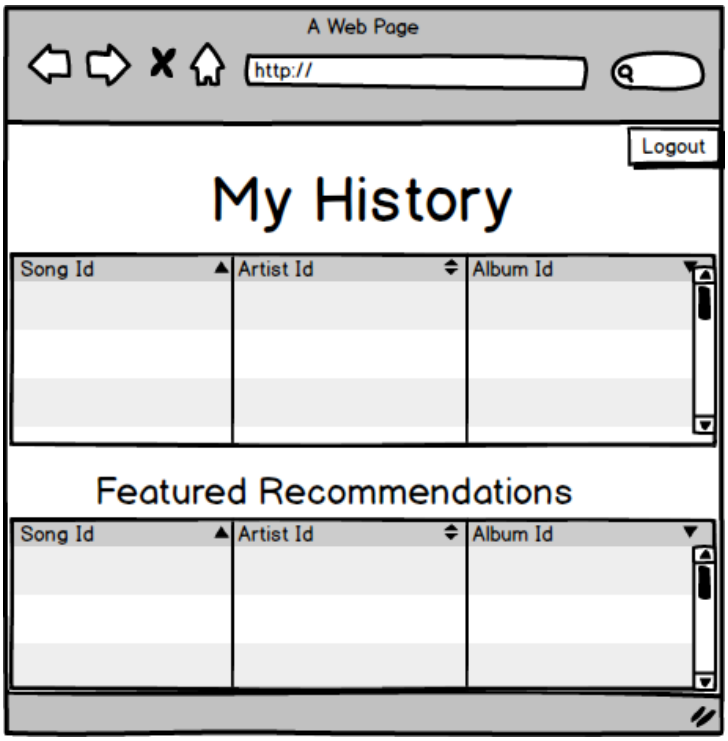
5.5.2 Design Elements



User will first reach this page to sign in.



User needs to enter a user id.



After signing in, the user will reach his listening history and featured recommendations.

5.6 Interaction Viewpoint

Interaction viewpoint is provided through sequence diagrams, to explain the main functionalities of the recommendation system.

5.6.1 Login Sequence Diagram

This sequence refers to login use case of section 5.2.1 of this SDD document.

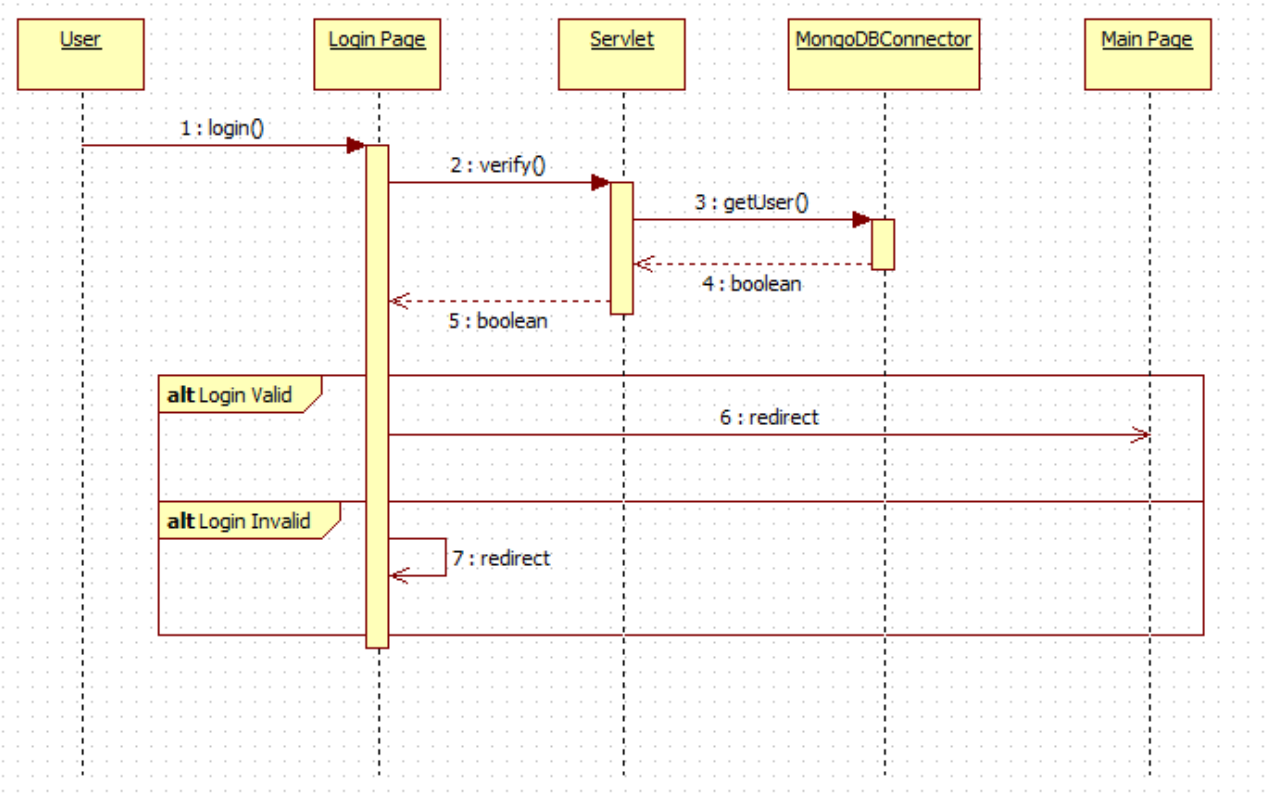


Figure13. Login Sequence Diagram

User refers to the real user of the system. It is not an object of an existing class. It is used only to increase understandability of the diagram.

5.6.2 View History Sequence Diagram

This sequence refers to view history use case of section 5.2.1 of this document.

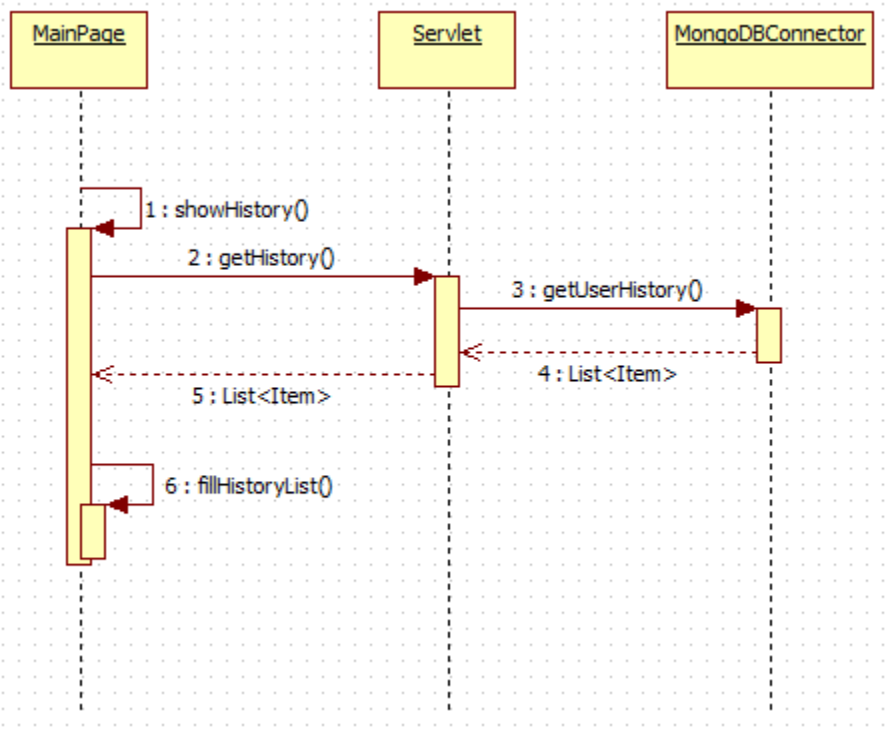
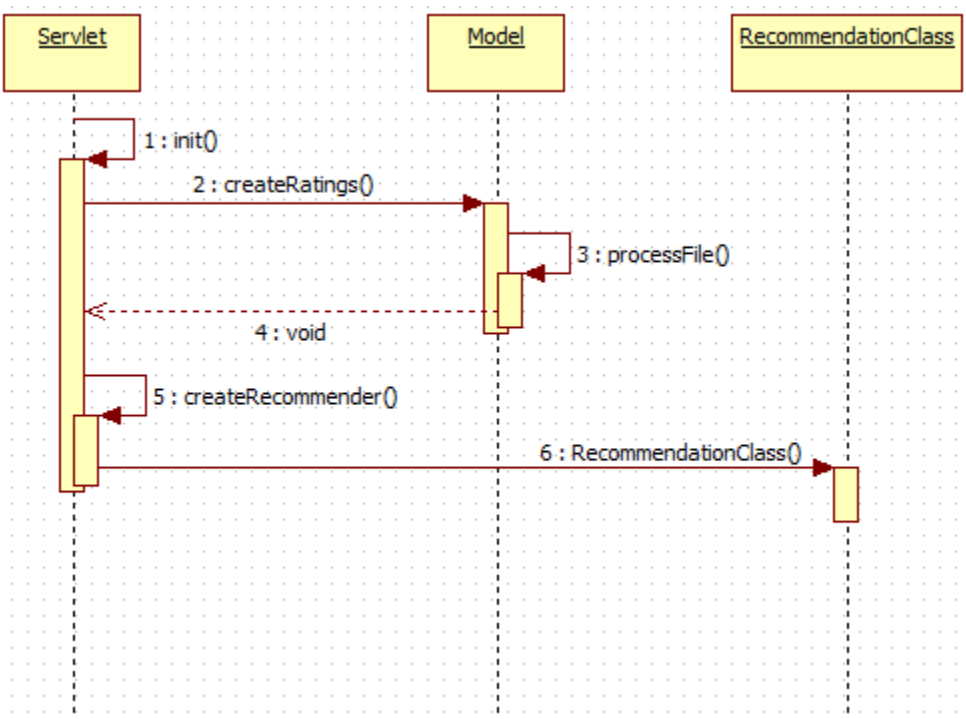


Figure14. View History Sequence Diagram

5.6.3 Generate Recommender Sequence Diagram



5.6.4 View Recommendations Sequence Diagram

This sequence refers to view recommendations use case of section 5.2.1 of this document.

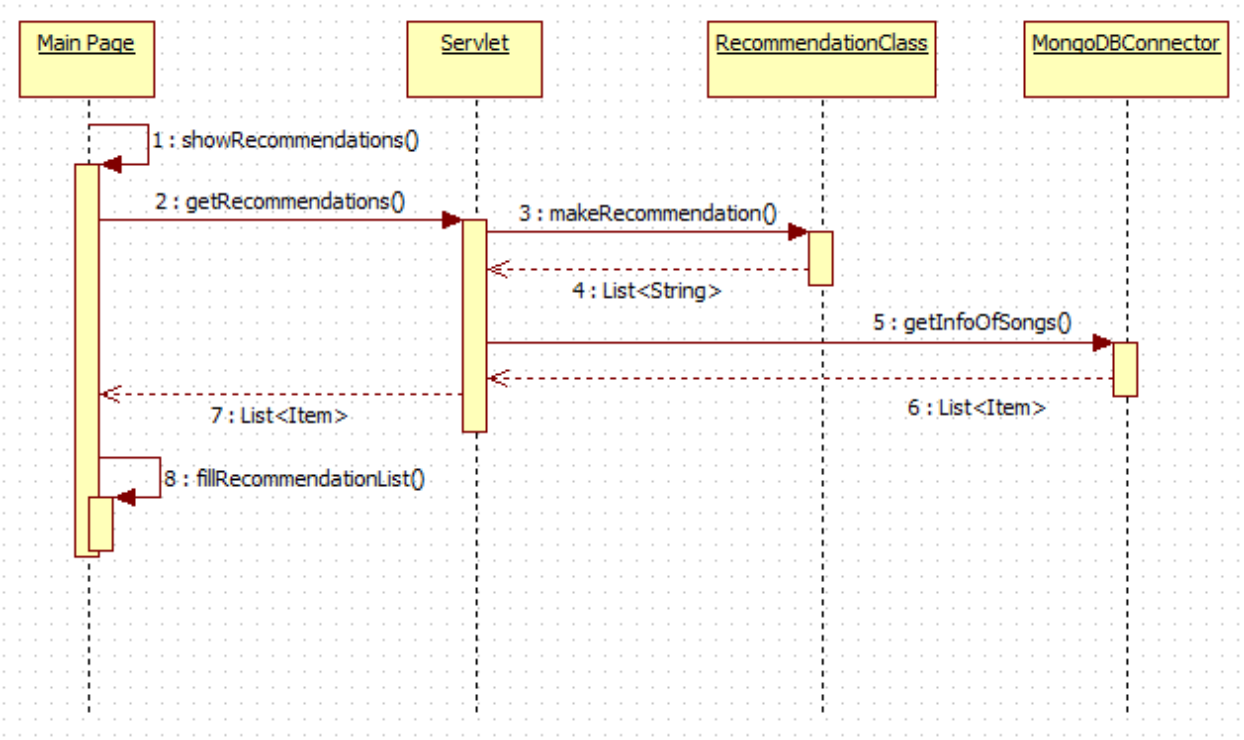


Figure15. View Recommendations Sequence Diagram

5.6.5 Logout Sequence Diagram

This sequence refers to logout use case of section 5.2.1 of this document.

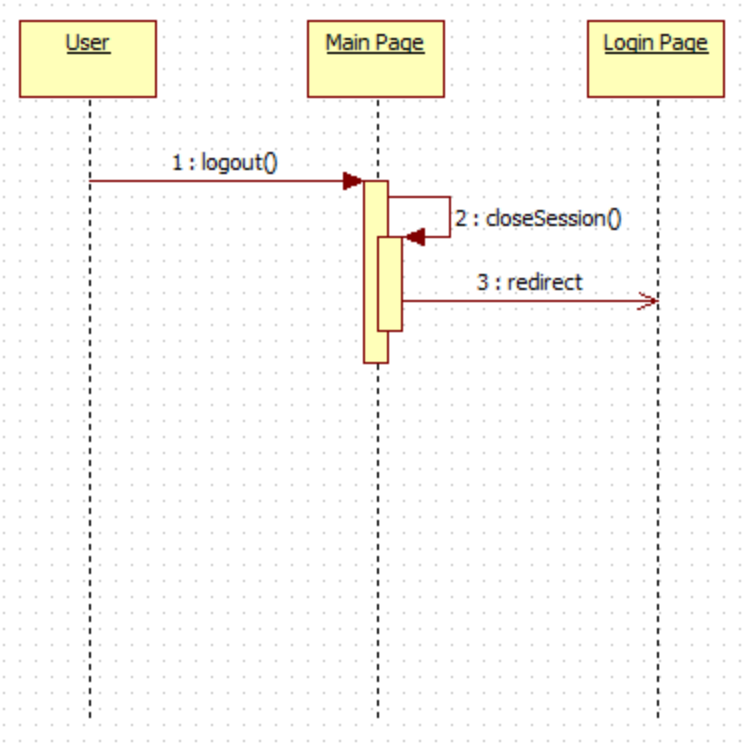


Figure16. Logout Sequence Diagram

User refers to the real user of the system. It is not an object of an existing class. It is used only to increase understandability of the diagram.

-planning section, who is implementing what, gannt chart

-in 5.2.1 update use case, make view rec. more detailed

-component diagram a accuracy component I ekle

-interface viewpoint e external interface bölümü ekle (hardware and software libraries)

6 Planning
6.1 Team Structure

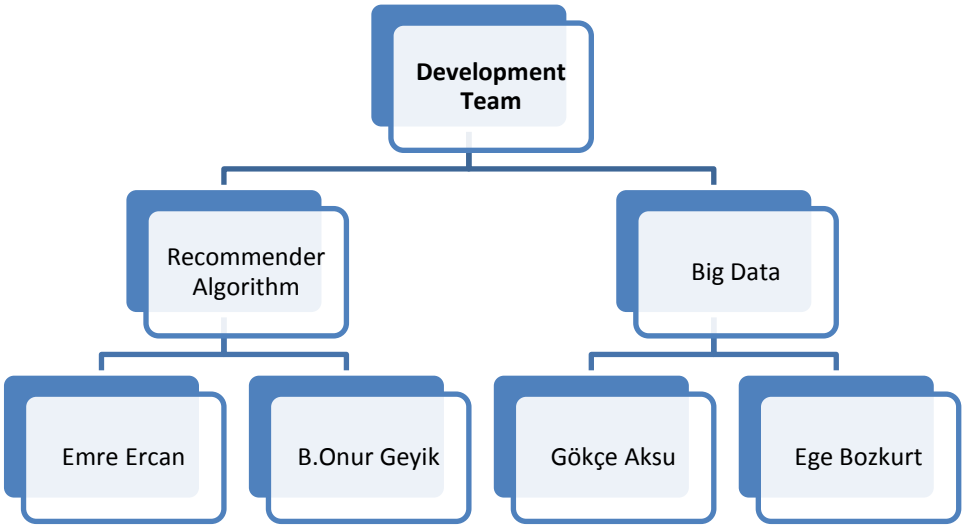


Figure10. Team Structure

6.2 Estimation (Basic Schedule)

Package-1	Research
Package-2	SRS Iteration
Package-3	SDD Iteration
Package-4	Big Data Handling
Package-5	Recommender Algorithm Development
Package-6	Testing

Packages	MONTHS								
	1	2	3	4	5	6	7	8	9
1									
2									
3									
4									
5									
6									