

Middle East Technical University
Department of Computer Engineering



RECOMMENDER SYSTEM

Software Design Description Document

V1.1

Dcengo Unchained



DuyguKabakcı – 1746064

Işinsu Katırcıoğlu – 1819432

Sıla Kaya - 1746122

Mehmet KorayKocakaya – 1746189

November 24, 2013

Change History

Date	Revision	Comment
24.11.2013	1.0	Created
29.12.2013	1.1	5.2. Context ViewPoint is updated 5.3. Composition ViewPoint is updated 5.4.1.2. Class Design Concern is updated 5.4.2.2. Class Design Elements is updated 5.5. Interaction ViewPoint is updated 5.6. Interface ViewPoint is updated 5.7. State Dynamics ViewPoint is updated 6. Planning is updated

Contents

1. Overview	5
1.1. Scope	5
1.2. Purpose	5
1.3. Intended Audience	5
1.4. References	5
2. Definitions	6
3. Conceptual Model for Software Design Descriptions	6
3.1. Software Design In Context	6
3.2. Software Design Descriptions Within Life Cycle	7
3.2.1. Influences on SDD Preparation	7
3.2.2. Influences on Software Life Cycle Products	7
3.2.3. Design Verification and Design Role in Validation	8
4. Design Description Information Content	8
4.1. Introduction	8
4.2. SDD Identification	8
4.3. Design Stakeholders and Their Concerns	8
4.4. Design Views	9
4.5. Design Viewpoints	9
4.6. Design Elements	9
4.7. Design Overlays	10
4.8. Design Rationale	10
4.9. Design Languages	11
5. Design Viewpoints	11
5.1. Introduction	11
5.2. Context Viewpoint	11
5.2.1 Design Concerns	11
5.2.2. Design Elements	13
5.3. Composition Viewpoint	14
5.3.1. Design Concerns	14
5.3.2. Design Elements	17
5.4. Logical Viewpoint	18
5.4.1. Design Concerns	18

5.4.1.1. Data Design Concern	18
5.4.1.2. Class Design Concern	19
5.4.2. Design Elements.....	20
5.4.2.1. Data Design Elements	20
5.4.2.2. Class Design Elements	22
5.5. Interaction ViewPoint	23
5.5.1. Design Concerns	23
5.5.2. Design Elements.....	25
5.6. Interface Viewpoint	26
5.6.1. Design Concerns	26
5.6.2. Design Elements.....	27
5.7. State Dynamics Viewpoint.....	30
5.7.1. Design Concerns	30
.....	32
5.7.2. DesignElements	33
6. Planning	34
7. Conclusion	36

Preface

This document includes the Software Design Description (SDD) of the Recommender System. This document is prepared according to the “IEEE Standard for Information Technology – Systems Design – Software Design Descriptions – IEEE Std 1016 – 2009”. Main purpose of this documentation is to provide a complete description of all the system design and views of the project.

The Recommender System aims to enhance the existing recommendation tools served to music streaming and downloading tools. In this project, the challenge is to overcome about data design, modules and design viewpoints. The target audience, who wants to make use of this system, can find all related design descriptions information in this document. It assists the software developer team, the stakeholders and the end users.

The first section of this document includes the conceptual model for SDD. The role and influences of this document are specified. The following sections include design description information content and design viewpoints of the system. Design viewpoints are indicated along with their related diagrams. They, as a whole, provide the details of the system modeling and architectural design concerns.

1. Overview

1.1. Scope

This document gives the design description for the Recommender System. A set of design elements and viewpoints will be presented in order to specify the design and development process of the software product. Each design concern is expressed in the context of one design view and each design view is given with the corresponding design elements. The document provides a clear understanding of the basic structure of the project and how the implementation process will be carried out.

1.2. Purpose

The software design document provides the system modeling and architectural design for the Recommender System. The purpose is to give a general description of the design elements, their interactions, the system's components and behaviour prior to the code development phase. This document verifies whether the design issues conform to the requirements of the Recommender System SRS Document. SDD serves as the primary reference for the code development.

1.3. Intended Audience

As stated in section “1.2. Purpose” of Recommender System SRS Document, the intended audience are project managers, developers, testers and end users. The design document gives detailed explanation of data, architecture (design viewpoints) and procedural design for project managers, developers and testers. The end users can also be guided by system boundaries and services description in this document.

1.4. References

- [1] IEEE. (2009). IEEE Standard for Information Technology-Systems Design-Software Design Descriptions. IEEE Computer Society.
- [2] “Recommender System” Software Specification Requirements (SRS) Document. (2013).
- [3] StarUML 5.0 User Guide. (2005). Retrieved from [http://staruml.sourceforge.net/docs/user-guide\(en\)/toc.html](http://staruml.sourceforge.net/docs/user-guide(en)/toc.html)

[4] Neo4j. (2010). Modeling Categories in a Graph Database. Retrieved from <http://blog.neo4j.org/2010/03/modeling-categories-in-graph-database.html>

2. Definitions

Term	Definition
API	Application Programming Interface
Cypher	A graph query language
Eclipse	Multi-language integrated development environment
IEEE	Institute of Electrical and Electronics Engineers
Neo4j	Open-source graph database implemented in Java
SDD	Software Design Description
SRS	Software Requirements Specification
UML	Unified Modeling Language
Weka	A popular suite of machine learning software written in Java, developed at the University of Waikato, New Zealand

3. Conceptual Model for Software Design Descriptions

This section includes basic project terms, concepts and context of SDD in which the documentation is prepared. The conceptual model aims to give a better understanding of the project terminology, software life cycle and frameworks that the project resides on.

3.1. Software Design In Context

The software product will be designed in an object-oriented and modular fashion. This project will be implemented with Java using Eclipse as IDE. The software product is planned to be a RESTful web service. The portability is an important issue since the system will be integrated to a separate stand-alone web application. But the opportunity of using a web service

makes the product platform independent. However, the entire project is context dependent. Since the project mainly relies on the user data type, the product can not be integrated to different contexts such as movie data. Big data and performance are two important issues in recommendation field. The project must be able to support large number of end user data. This case contradicts the performance issue most of the time. Therefore, it is needed to prioritise one of these issues.

Big data induce the use of graph database which is an important context in terms of designing and storing data. In this project, graph database will be generated by Neo4j. The Cypher language, which is a query language for Neo4j graph database, will be used. In order to implement the recommendation algorithm and mapreduce algorithm, Weka, a machine learning library implemented in Java, is chosen. It works together with Neo4j graph database.

3.2. Software Design Descriptions Within Life Cycle

3.2.1. Influences on SDD Preparation

The primary influence on software design process is the Recommender System SRS Document. The product perspective, functional and nonfunctional requirements determine the outline of the architectural design. The additional specifications of the stakeholders related to the frameworks or data affect the SDD preparation period.

3.2.2. Influences on Software Life Cycle Products

The agile method is adopted for the software process model and the software product will reach to a final stage after a series of iterations. The first part of the software cycle is to construct a simple recommendation model by using the existing open-source implementations. The next part is to customize the algorithms according to the partial data and collaborative approach. After the first iteration a prototype will be presented to the stakeholders. The iterations and additional/changing requirements of the stakeholders have influence on software life cycle products. After the first demo, the model will be scaled to big data which is the main concern in this project.

3.2.3. Design Verification and Design Role in Validation

Software design description is the primary reference for the verification and validation of whether the software product designed fulfills the specified requirements in Recommender System SRS Document. The requirements for each specific intended use of the software product are modeled in the design view parts of the document. The verification and validation of the design view models are carried out based on this document. SDD influences test plans and test cases in further stages. The testing process will be handled after the code development.

4. Design Description Information Content

4.1. Introduction

The SDD is prepared to identify the architectural design of the Recommender System. This document specifies how the Recommender System will be designed and implemented according to some chief design views.

4.2. SDD Identification

The system model identified in this document will be used for each product iteration during the development and implementation periods. After the initial iterations, the prototype will be demonstrated on January 27, 2013. The software product will be released by the end of May, 2014. Tentative date for the date of issue is May 22, 2014. Dcengo Unchained team will be responsible for issuing the Recommender System in association with the issuing organization ARGEDOR. All rights of the end product are reserved. Due to the exclusive rights property, only the copyright holders, Dcengo Unchained and ARGEDOR will be free to modify and distribute the product. However, due to the copyright of the user data received from ARGEDOR, user logs and track information can not be distributed.

Scope, references, context and summary can be found in section “1. Overview”. Glossary and necessary terminology can be found in section “2. Definitions”.

4.3. Design Stakeholders and Their Concerns

Stakeholders include Dcengo Unchained team developers, R&D developers of ARGEDOR, testers and end users. Stakeholder’s main concerns are accuracy and performance for big user data. The end product is expected to enhance the existing systems at least in one of

these concerns. This project is based on enhancing the accuracy of recommendations. Further requirements are specified in the SRS document of the Recommender System.

4.4. Design Views

This Project will be implemented as web service application with simple web service interface. Therefore, stakeholders can integrate to their web application or they can easily use with web service interface.

In this document contextual, composition, interface, logical, interaction and state dynamics view will be explained in next sections. Detailed description and diagrams about these views will clarify them. Each view is given with its corresponding viewpoint.

4.5. Design Viewpoints

Software design description identifies context, composition, interface, logical, interaction and state dynamics viewpoints. Context viewpoint specifies the system boundaries and actors interacting with the system. Composition viewpoint identifies the system modules, components, frameworks and system repositories. Interface viewpoint explains the interaction between different software interfaces. Logical viewpoint includes the detailed description of data design and class diagrams. Interaction viewpoint gives the sequence of events in the system and the state dynamics viewpoint models the system as a state machine and shows the state transitions and conditions on these transitions.

4.6. Design Elements

All the design elements of this software design description are described in the following sections. Design elements, their attributes, relationships and constraints are explained together with the viewpoints which they belong to.

4.7. Design Overlays

The interface viewpoint includes the user interface. This user interface should not be confused with the web application user interface. The UI component in this project references to the interface of the web service. Its main purpose is receiving inputs and displaying results to the external system.

4.8. Design Rationale

The project is planned as a web service. This allows the operations to be software independent and makes the product integrable to stand-alone projects. The project is designed to be able to communicate with different frameworks through separate interfaces.

The Recommender System is completely a data-oriented system. Thus, the motivation behind the design choices is the size and format of the user data. The format of the data reinforces the use of separate classes for each item in the dataset; user, track, album and artist. Because each of them has its unique attributes and relations. This modular way can be seen in the logical viewpoint.

Big data is the challenging part of this project. The algorithm and software frameworks must support big data concept. That is why Neo4j graph database is preferred over other data storing and processing frameworks. Data storage in graphs allows the items (users, tracks, albums or artists) to be represented as nodes and their relationships can be designed as edges between these nodes. In this representation, the similarity between two users or tracks is marked with weights on the edges and this is completely compatible with graph concept. The similarity is found by graph traversal which is a key function in this project. Neo4j comes with a callback based traversal API which allows to specify traversal rules. Since the data size will increase as new logs are received, insertion of new nodes to a graph database is easier in this system mode. Neo4j also supports distributed file systems, therefore the expanding data can be processed in parallel on several different machines. In the component viewpoint, the deployment on several machines can be seen.

The overall system architecture is based on the idea that the database component works in association with a machine learning library component. Weka is chosen as the machine learning library. It provides predictive models and evaluates the quality of the chosen model. It can be

used as a recommendation engine and combined with the Neo4j graph database. These ready-to-use implementations speed up the agile process. Therefore, in the interface viewpoint, the system is in connection with these software libraries.

4.9. Design Languages

Unified Modeling Language (UML) and ER diagram are used for the design viewpoint specification.

5. Design Viewpoints

5.1. Introduction

Design viewpoints determine the conventions for a system including the architectural models, languages and notations. They are used in the realization part of the design descriptions and constraints of the Recommender System. Several design viewpoints are defined in the following subsections. UML is used as the design language.

5.2. Context Viewpoint

This section provides the system boundaries and the actor interactions of the Recommender System. The relationship between the external systems and the Recommender system are displayed within the system environment. The roles of actors and stakeholders are illustrated through UML Use Case diagram.

5.2.1 Design Concerns

Our project will evolve in two ways. One of them is developing a web service which stakeholders integrate into their web applications. The other one is making a web service application. This application has a simple user interface for uploading the dataset or getting recommendation. Interagent gets recommendation via using the web service application or integrating the web service into their web application. The latter one means giving recommendations to end users.

The Recommender System is planned to be a RESTful web service that generates reliable track recommendations when integrated into an existing music downloading and streaming web applications. The system processes the ready-to-use data including user logs, track and album

information. User logs give detailed records relating users to the tracks played at certain times. The system is designed to process the user-track logs of ArgedorMüzik web application. The system is provided with this information, therefore there is no direct interaction with ArgedorMüzik application. The user profile, user feedback, music lists, user comments and the music downloading and streaming services are out of the scope of this software product. It works in association with ArgedorMüzik in terms of data flow. However, the process is restricted to combining the user data and suggesting a track to a specific user. The result is displayed by means of the external system. The end product will be a web service and it can be integrated to other related external systems. But the entire work is context dependent. Therefore, the software product only renders service for music applications.

Moreover, the Recommender System is aiming to make a web service application which includes a simple user interface for getting recommendation even if it is not integrated to a web application. The interagent can upload/update datasets and get recommendation via this simple application.

Figure 5.1 illustrates the configuration of the Recommender system among external systems and Figure 5.2 shows the use case diagram for this system.



Figure 5.1: System Context Diagram

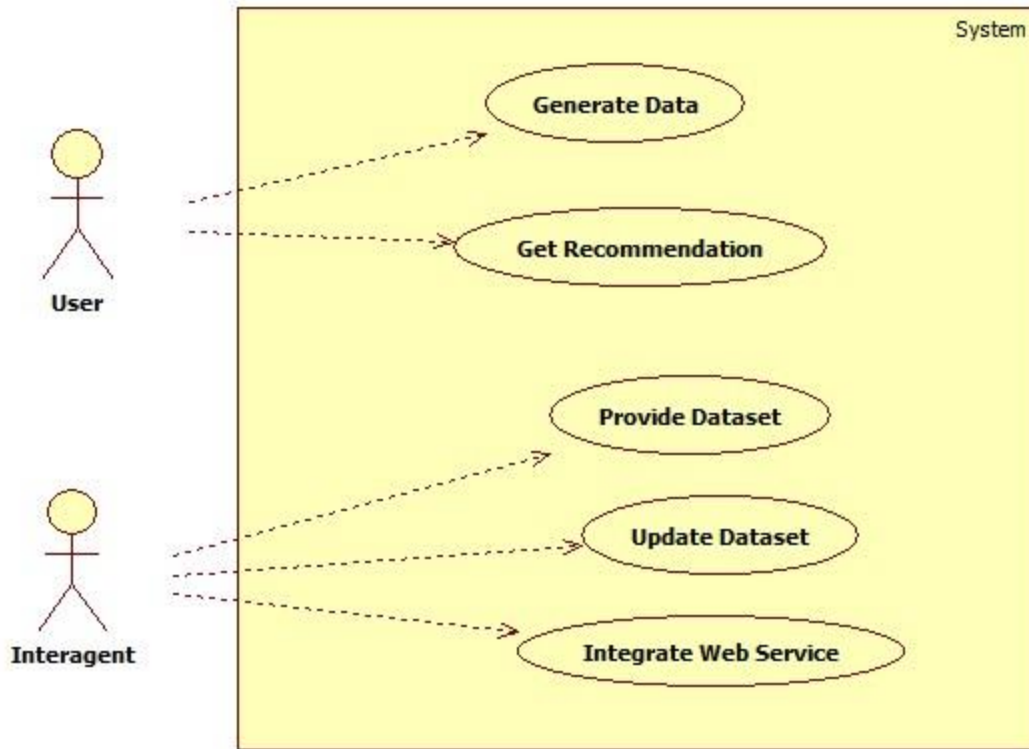


Figure 5.2: Use Case Diagram

5.2.2. Design Elements

One of the major design entities is the group of stakeholders. The Recommender System is a software project conducted in association with ARGEDOR company and Prof. Dr. İsmail HakkıToroslu. The stakeholders are the recommender systems project management team and R&D team of ARGEDOR. Another group of design entities is the actors of the Recommender System which are end users and interagents. The end users of the Recommender System can be anyone who downloads or plays a track through the music downloading and streaming application. The system is not an online user interactive system. Therefore the user gets the output of the system through the interagent application. This is an expected outcome of a web service. However, the user actions are important because their listening, searching and rating activities generate the user logs. The user logs are delivered by ARGEDOR. Apart from the two

functionalities of users, recommender system gives recommendation to the external systems. The interagent stands for the R&D team of ARGEDOR. They provide and update dataset for the integrated web service.

These functionalities can be seen in Figure 5.2. In Figure 5.1, the external system inputs to the Recommender System which outputs back to the external system. The result is then presented to the users.

5.3. Composition Viewpoint

This section provides information about Recommender System components and their connections with each other. The correlation between large system and the Recommender System as a web service is also shown within the overall system environment.

5.3.1. Design Concerns

The aim of this viewpoint is providing information to stakeholders and programmers for planning and controlling the system. This kind of subsystem level illustration can be used for assembling components, cost estimation and schedules in terms of development effort.

System components such as libraries, packages, files and their interconnections are illustrated in UML Component diagram. Recommender System is integrated to larger web based music application as a web service. Larger system is a music streaming and downloading application which collects and provides user logs to recommender web service and its graph database. Graph Traversal Algorithms are used for searching through graph database by Recommender component. Machine learning algorithms component uses Weka library which is a part of recommendation generation. After getting recommendation from web service's Recommender component, recommendations can be shown via the user interface of web service. After the integration of music web application and Recommender System, recommendations can also be shown via larger system's user interface.

Users can use downloading and streaming music application via PC which can be compatible with Windows/Unix operating systems. Server interacts with REST API of web service which connects to computer cluster. Our recommender web service deals with big data but the earlier prototype do not use big data. Therefore, only single master device is enough for communicating with the database. Figure 5.3 illustrates the subcomponents of the Recommender

System and interconnections between components. Figure 5.4 and 5.5 shows the hardware components used to deploy software components for this system.

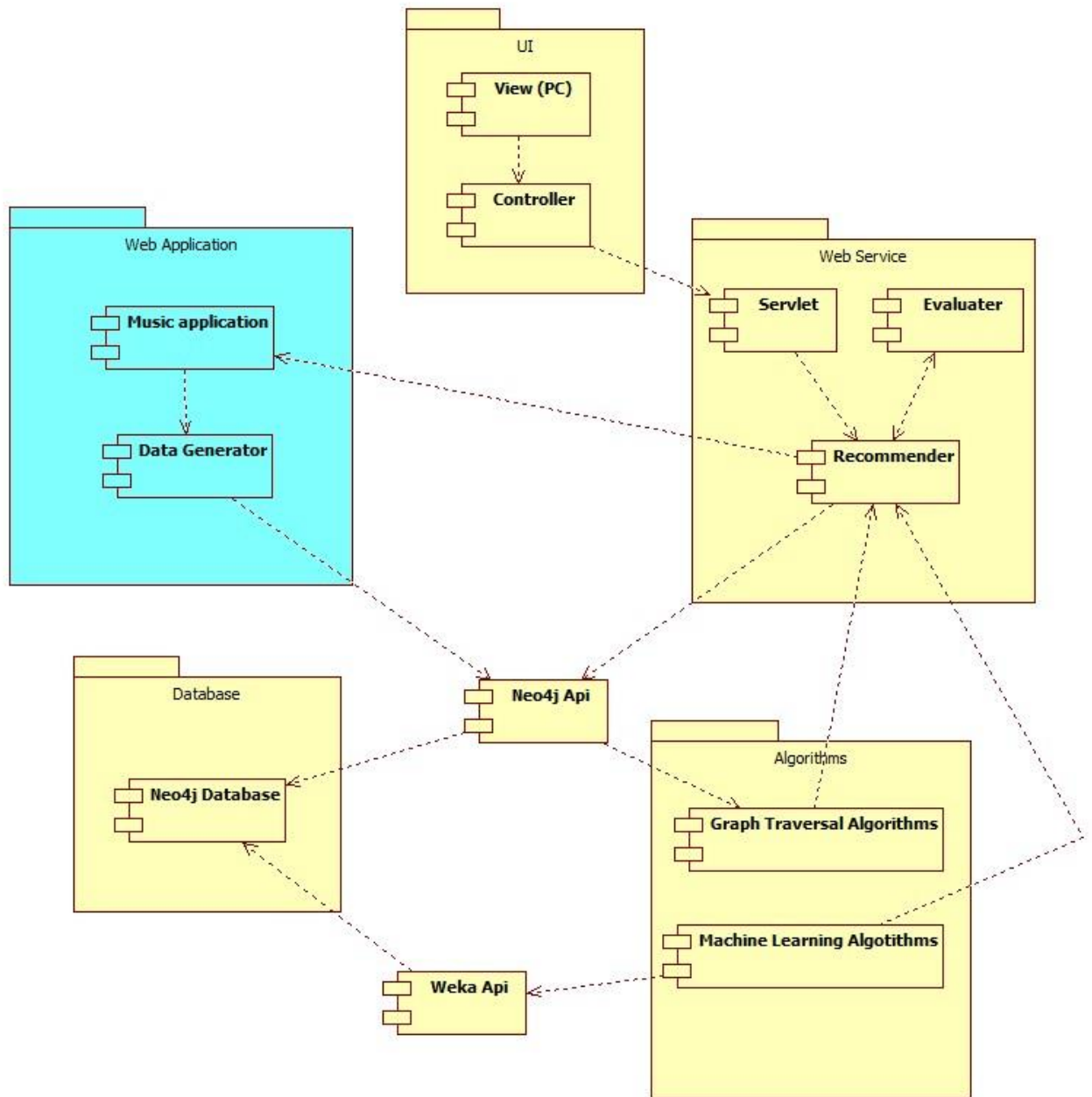


Figure 5.3: System Component and Package Diagram

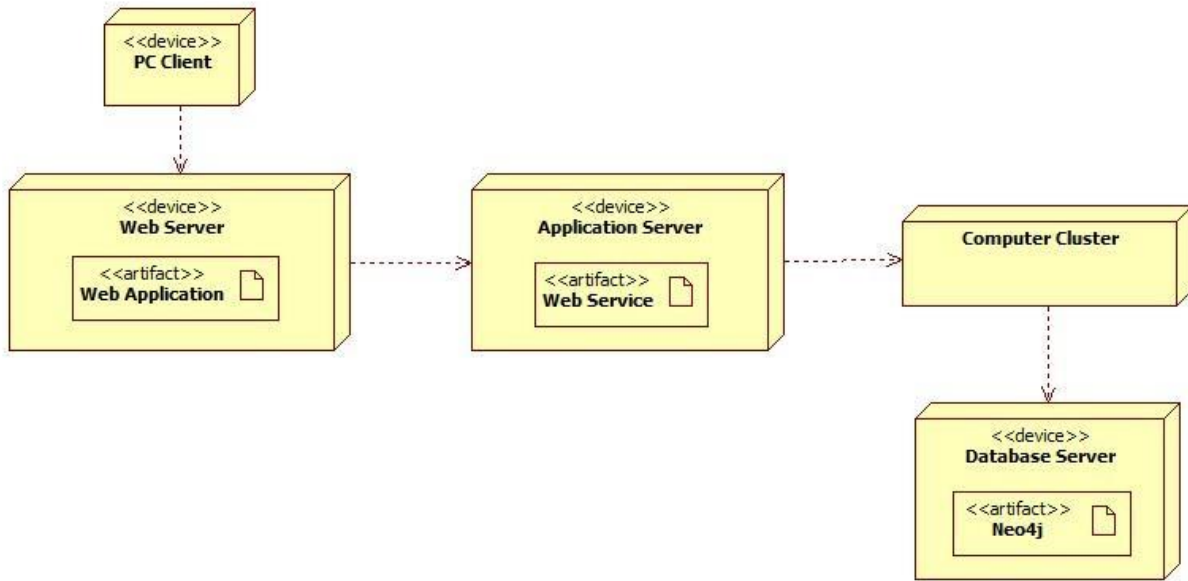


Figure 5.4: System Deployment Diagram

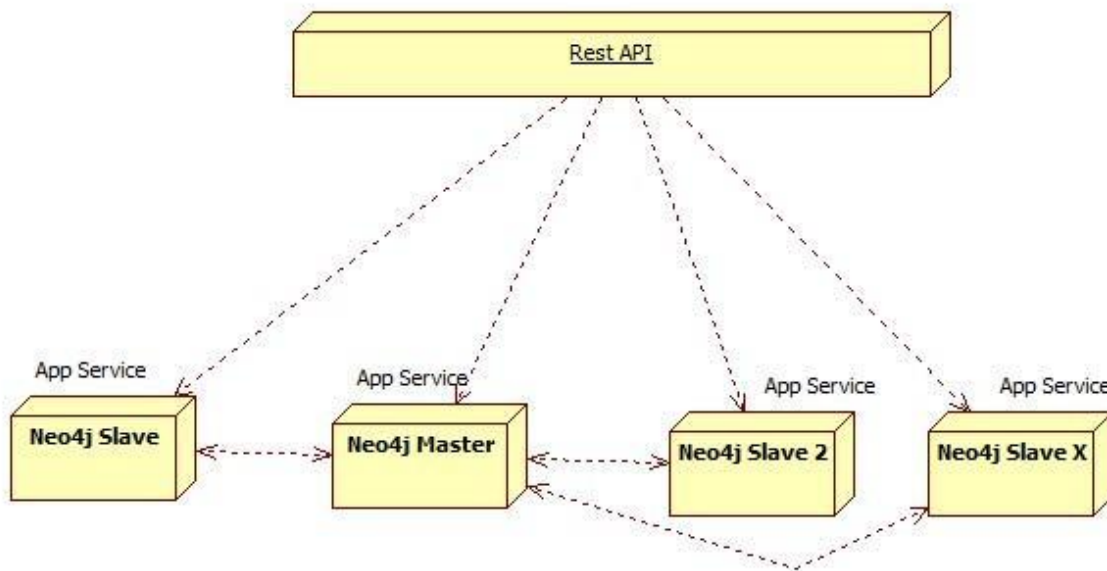


Figure 5.5: Distributed Systems Deployment Diagram

5.3.2. Design Elements

Web application

Functional attribute: Web application provides users with music downloading and streaming. Moreover, it generates data that is used by the web service. Web Application already exists. So, application functions are not our concern except the data flow to our recommender web service.

Subordinates attribute: Web application is composed of Music application and data generator. Music application allows users to listen and download tracks. Data generator provides user logs (comes from music application) to Recommender System.

Web Service

Functional attribute: Web service is used to serve requested recommendation by web application. It communicates the external world such as UI component or web application module.

Subordinates attribute: Web service is composed of evaluator and recommender component. Recommender component traverses data via graph traversal algorithms to make recommendations using machine learning algorithms. Evaluator checks the quality of recommendations in offline mode. Evaluator uses approximately 10% of data set as the test cases to control whether a recommendation is suitable or not.

Algorithms

Functional attribute: This component is used by recommender for traversing data and applying machine learning recommendation algorithms.

Subordinates attribute: Algorithms package is composed of graph traversal algorithms component and machine learning algorithms component. Graph traversal algorithms make transactions on Neo4j graph database via Neo4j api. Machine learning algorithms help recommender to generate recommendations. The implementations of the algorithms comes from Weka library.

Database

Functional attribute: Neo4j graph database stores data about user transactions on tracks.

Subordinates attribute: Neo4j api connects database to other components.

UI

Functional attribute: User interface of recommender web service controls and displays recommendations coming from web service.

Subordinates attribute: UI is composed of view and controller components. Controller checks whether recommendations exist or not. View component displays recommendations.

5.4. Logical Viewpoint

5.4.1. Design Concerns

5.4.1.1. Data Design Concern

Recommender System is a highly data dependable system. The size and the organization of the data designate the project stages. Figure 5.6 illustrates the ER diagram of the data entities, relationships and their attributes. It should be noted that the data entities are not organized according to the relational data model. However, Neo4j graph database community declares that they use the same ER diagram notation as in the case of relational database models.

The entities, relationships and their attributes conform to the content of the dataset delivered by ARGEDOR. Song, album, performer and user are the data entities whose unique attributes are provided with the dataset. The necessary correlations connecting a song to an album and a performer are also included in the ARGEDOR dataset. There are also user logs that match users to the songs, their performers and albums along with additional information on the listening action.

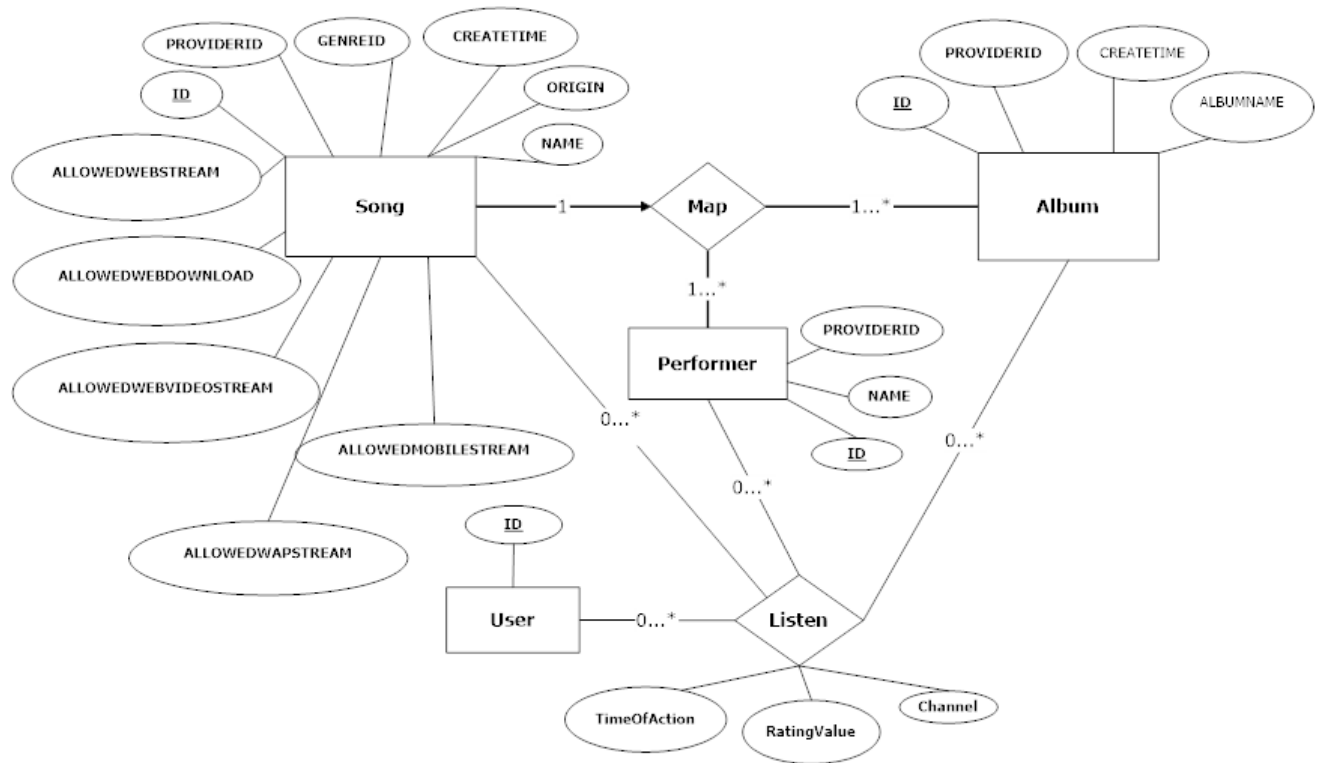


Figure 5.6: ER Diagram

5.4.1.2. Class Design Concern

This section provides the organization of system classes. They are designed according to the previous section which covers the data organization. System classes also include some specific classes governing database actions and similarity detection methods. They, as a whole, operate to return an accurate recommendation instance to the external system. It should be noted that class design concerns the implementation and development part of the project. During the code development, there can be changes in the class design. They will be provided as an update.

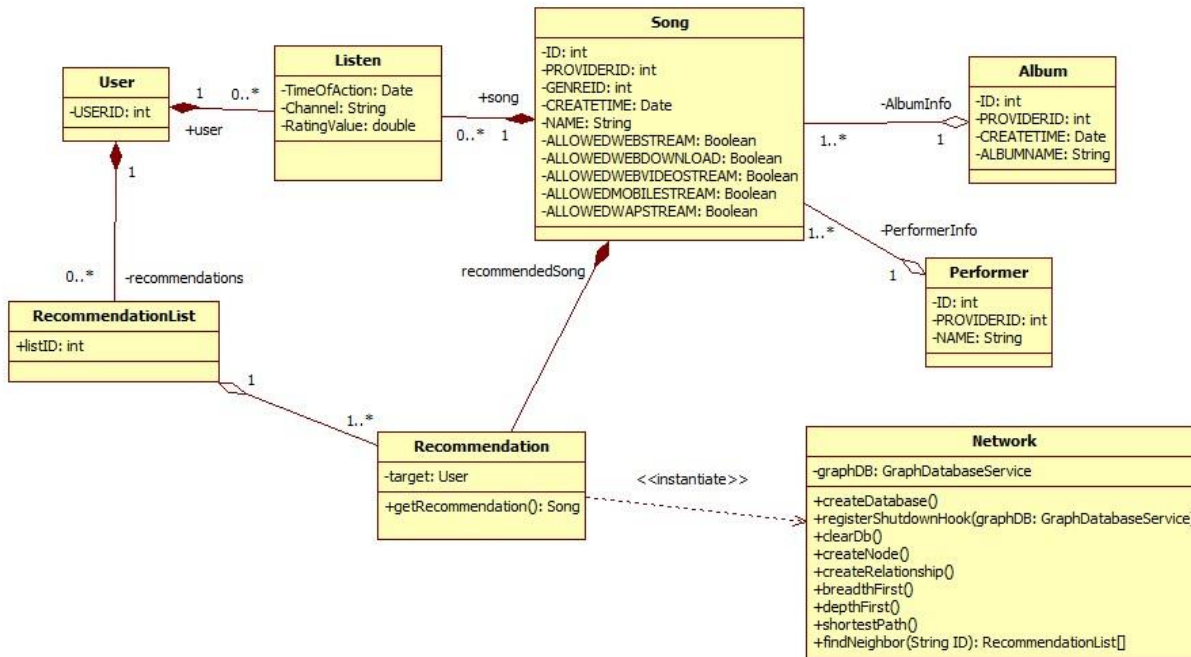


Figure 5.7: Class Diagram

5.4.2. Design Elements

5.4.2.1. Data Design Elements

ER diagram in Figure 5.6 shows that there are four data entities; song, performer, album and user.

Song entity represents the tracks that can be either played or downloaded through the music downloading and streaming application. This entity has the following attributes:

- **ID:** This attribute gives the unique identifier of a song. It is the indexing attribute (the primary key in relational models) of the song entity.
- **PROVIDERID:** It is the id of the provider of the song.
- **GENREID:** It is the identifier of the group or type of the song.
- **CREATETIME:** This attribute specifies the time that the song instance is created.
- **ORIGIN:** It states whether the song instance is a Turkish song or not.
- **NAME:** It is the name of the song.

- ALLOWEDWEBSTREAM: It indicates whether the song can be played instantly through the web application.
- ALLOWEDWEBDOWNLOAD: It indicates whether the song can be downloaded.
- ALLOWEDWEBVIDEOSTREAM: It indicates whether the video of the song can be played instantly before the entire file is transmitted.
- ALLOWEDMOBILESTREAM: It indicates whether the song can be played instantly through the mobile application.
- ALLOWEDWAPSTREAM: It indicates whether the song can be played instantly via mobile wireless network.

Performer entity represents the author of songs. It has the following attributes:

- ID: This attribute gives the unique identifier of a performer. It is the indexing attribute (the primary key in relational models) of the performer entity.
- PROVIDERID: It is the id of the provider of the performer information.
- NAME: It is the name of the performer.

Album entity represents the collection of songs and it has the following attributes:

- ID: This attribute gives the unique identifier of an album. It is the indexing attribute (the primary key in relational models) of the album entity.
- PROVIDERID: It is the id of the provider of the album information.
- CREATETIME: This attribute specifies the time that the album instance is created.
- ALBUMNAME: It is the name of the album.

User entity represents the end users who make use of the music downloading and streaming application. It has the following attribute:

- ID: This attribute gives the unique identifier of a user. It is the indexing attribute (the primary key in relational models) of the user entity.

Map relationship matches the songs to their corresponding performers and albums. In this relation, each song instance must appear once and only once while the performer and album instances can appear at least one time.

Listen relationship matches the users to the songs played or downloaded. Although the mapping of performer and album information is redundant in this case, the delivered dataset includes this mapping as well. This relationship has three attributes:

- TIMEOFACTION: It indicates the time of playing or downloading a song.
- RATINGVALUE: This attribute is a metric evaluated by ARGEDOR and its content is not shared.
- CHANNEL: It identifies whether the user played the song from his/her own account or from the web browser.

5.4.2.2. Class Design Elements

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.

5.5. Interaction ViewPoint

Interaction viewpoint is provided through UML Sequence diagram, to explain the main functionality of the Recommendation System.

5.5.1. Design Concerns

The aim of this view is showing the flow of application running system. Recommendation System project has one main sequence because its major purpose is making accurate recommendations.

The UML Sequence diagram illustrates the behavior of the system and classes which take part in the sequence and their transition to each other. The relation among classes and function calls while generating the recommended object are shown. Our project depends on user and Argedor Web Service indirectly which means those parts are out of concern. However, stakeholders provide the dataset that combines user logs to the system. Therefore, user and Argedor web application should be incorporated in our sequence diagram. Figure 5.8 illustrates the flow of sequence of Recommender System in integrated web service case.

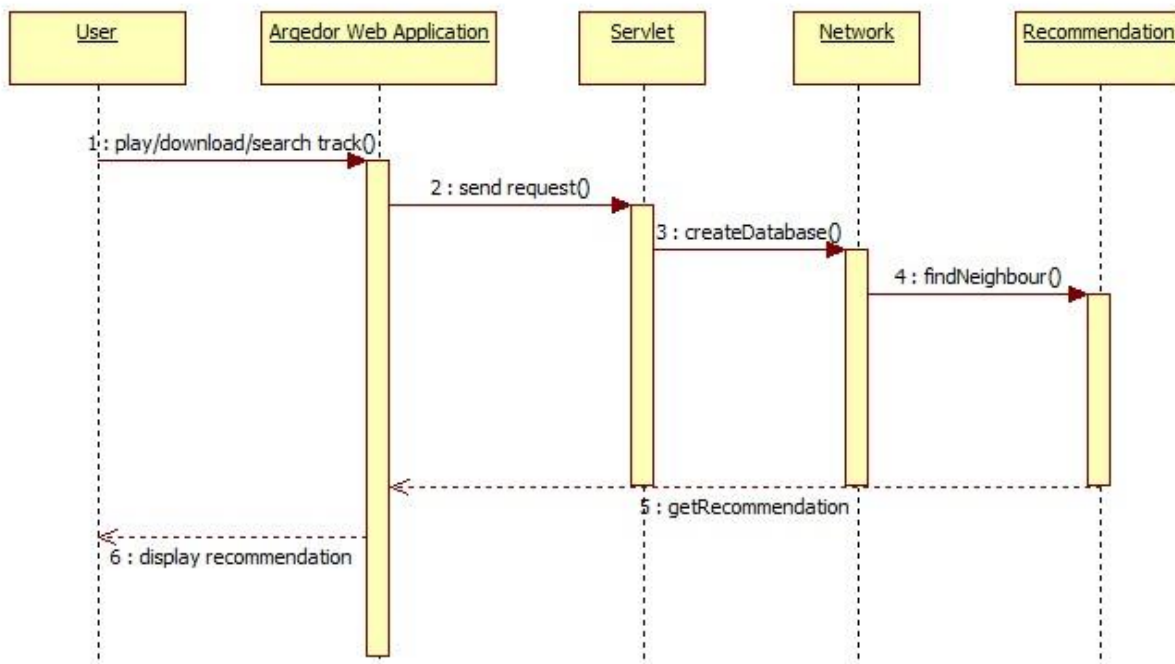


Figure 5.8: Sequence Diagram -1

*Argedor Web Application is not an object of an existing class. It is used to increase the understandability of the diagram.

On the other hand, Argedor web application is used in the real project. As soon as the project shows progress, web service interface will be added for uploading datasets and displaying the recommendations. In this case, Web Service Interface will be shown instead of Argedor Web Application. Figure 5.9 illustrates the flow of sequence of Recommender System when web service interface is used.

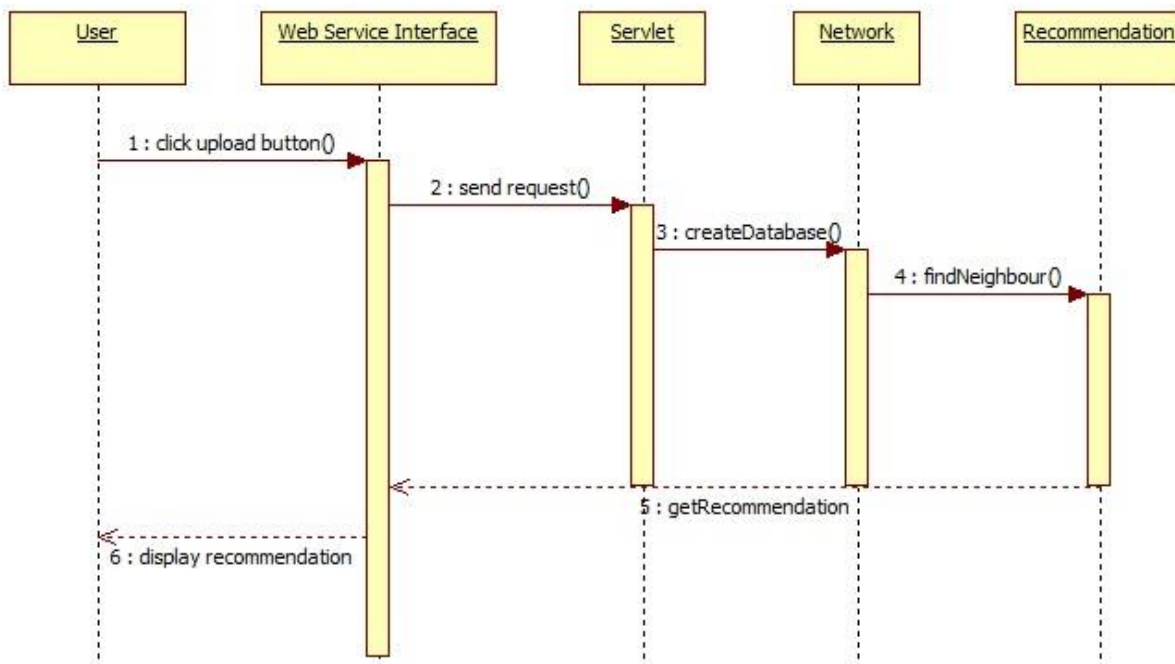


Figure 5.9: Sequence Diagram -2

*Web Service Interface is not an object of an existing class. It is used to increase the understandability of the diagram.

5.5.2. Design Elements

The system starts with user actions because it is the source of collecting data. Our project is based on collecting and processing data. Therefore, when user is playing, downloading or searching track, Argedor Web Service collects all information about user track profile. Then, it sends request to our web service to upload data. Our network class is dealing with the connection with database, uploading data and converting database to graph database. It requests Recommendation class with findNeighbour(userid) method to find most similar objects between tracks or users for accurate recommendation. Finally, Recommendation class generates recommended object, sends it to Servlet which displays recommendation to user via Argedor Web Application. In other case, recommended objects are displayed via Web Service Interface.

5.6. Interface Viewpoint

This part of the project is provided to show user and external interfaces, hardware and software libraries and tools and relationship between them. UML Component diagram is used for this section of the document.

5.6.1. Design Concerns

Interface Viewpoint is used to indicate the relationship between libraries, tools and interfaces. The Recommender System application starts with user action. User interface is provided by company Argedor. When a user action is provided, Argedor generates and classifies data which they send to Recommender System. Programmers get involved in this part of the project. They put this data to the graph database Neo4j. After that, recommendation is generated by using this data set and algorithms applied to the dataset. These algorithms can be graph algorithms and machine learning algorithms (these are provided by WekaApi). Before this Recommender System is integrated into web application, Dcengo Unchained team creates Web Service Interface and this interface is used by testers to test the Recommender System. Web service is integrated into the application by Argedor programmers and application can be served to the users. When a customer listens or downloads a track, this process starts again. For this process, programmers, designers and testers should work collaboratively.

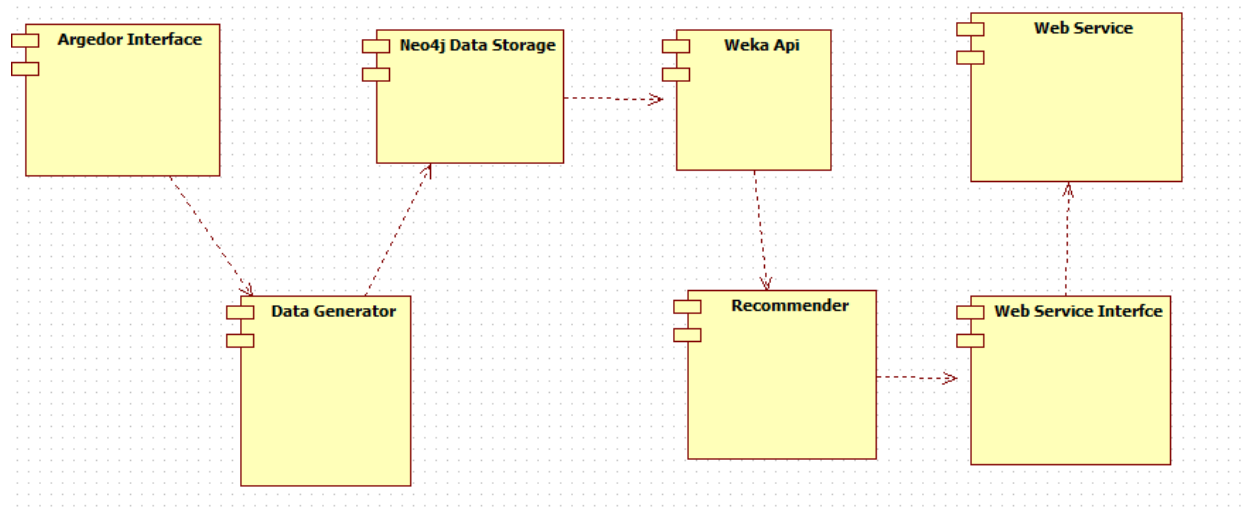


Figure 5.10: Component Diagram

5.6.2. Design Elements

Interface Viewpoint consists of six components. Argedor Interface is the first one. Argedor serves to users by using User Interface.

Argedor takes the dataset from the user and classifies them according to their information (user, song, album, artist, time of action, channel). After that, they change the name of the user, song, album and artist with id for security. In addition to this, they add a new attribute; rating value which is a value obtained by a formula depending on user's action (listened, downloaded etc.).

After this process data is transmitted to graph database Neo4j. Graph database is 1000 times faster for many queries than relational database. It consists of nodes, edges (relationships) and properties. These relationships allow to find similar items easily.

A series of algorithms are applied to data in order to get recommendation. These are provided by Weka API. Weka (Waikato Environment for Knowledge Analysis) is a data mining tool and a collection of machine learning algorithms. Weka has many preprocessing tools (named as filters) and algorithms for clustering, regression, classification and finding association rules.

To get some recommendation, the system gets dataset from database and applies them graph algorithms and machine learning algorithms of Weka API. Before the system is integrated into the web application, an interface will be developed to show the work and to test it. Web Service Interface is as follows:

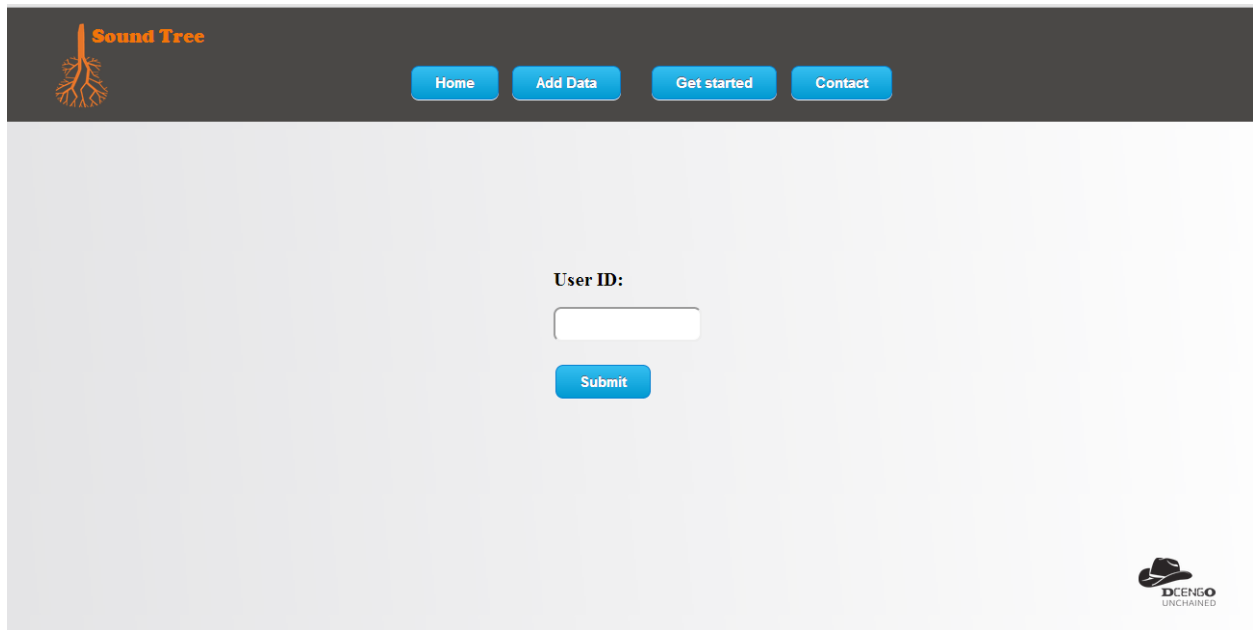



Figure 5.11: Web Service Interface

In Web Service Interface, there are four pages Home, Add Data, Get Started and Contact (Figure 5.11). This page (Get Started) contains a text field and a button. The user of the system who want to get recommendation fill the text field with proper user id. After s/he fill the text field and click the submit button, new page(Figure 5.12) is opened and recommendations are shown in this page. It contains a table with three columns which are Song Name, Performer Name and Album Name. Recommendations are listed in the table according to their song, performer and album names. The row number of the table can change according to how many recommendations are given to the user.



Home Add Data Get started Contact

Song Name	Performer Name	Album Name
Hadi Ordan	Gökhan Keser feat. S?la	Hadi Ordan
Yoruldu	S?la	Türkçe Pop 2012
Sevi?meden Uyumayal?m	S?la	Power Türk En ?yiler 2009 - Avrupa 2009
Bir O?lumuz Var	Demet Akal?n	Giderli 16
A?iyorum	Demet Akal?n	Giderli 16




Figure 5.12: Web Service Interface

In Add Data page (Figure 5.13), there are five segments. The user of the page should fill these five segments with text file. Song, Performer and Album document include information about them. Map document (relationship between song, performer and album) matches these three items. User Logs Document contains all of the information about played or downloaded songs. These five documents must be formed according to dataset provided by Argedor. When Submit button is pressed, database is updated with information in these text files. This page is created to update data and add new information when new log, song, album, performer and map data is created.

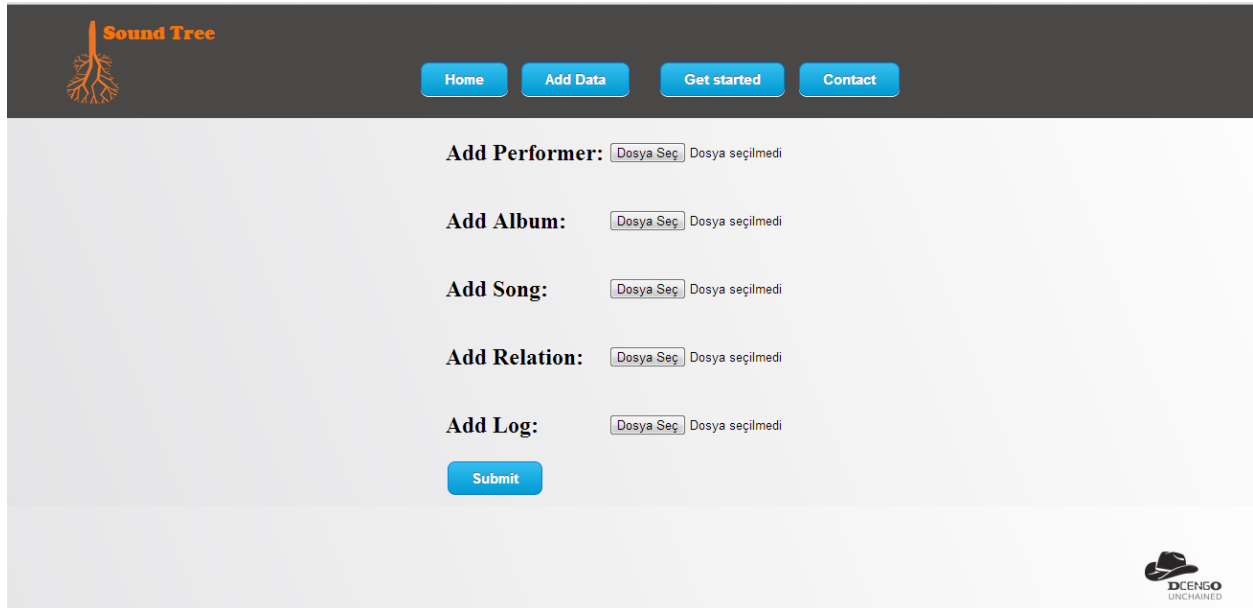


Figure 5.13: Web Service Interface

This interface can be used by testers. After testing, recommender system will be served as a web service to Argedor.

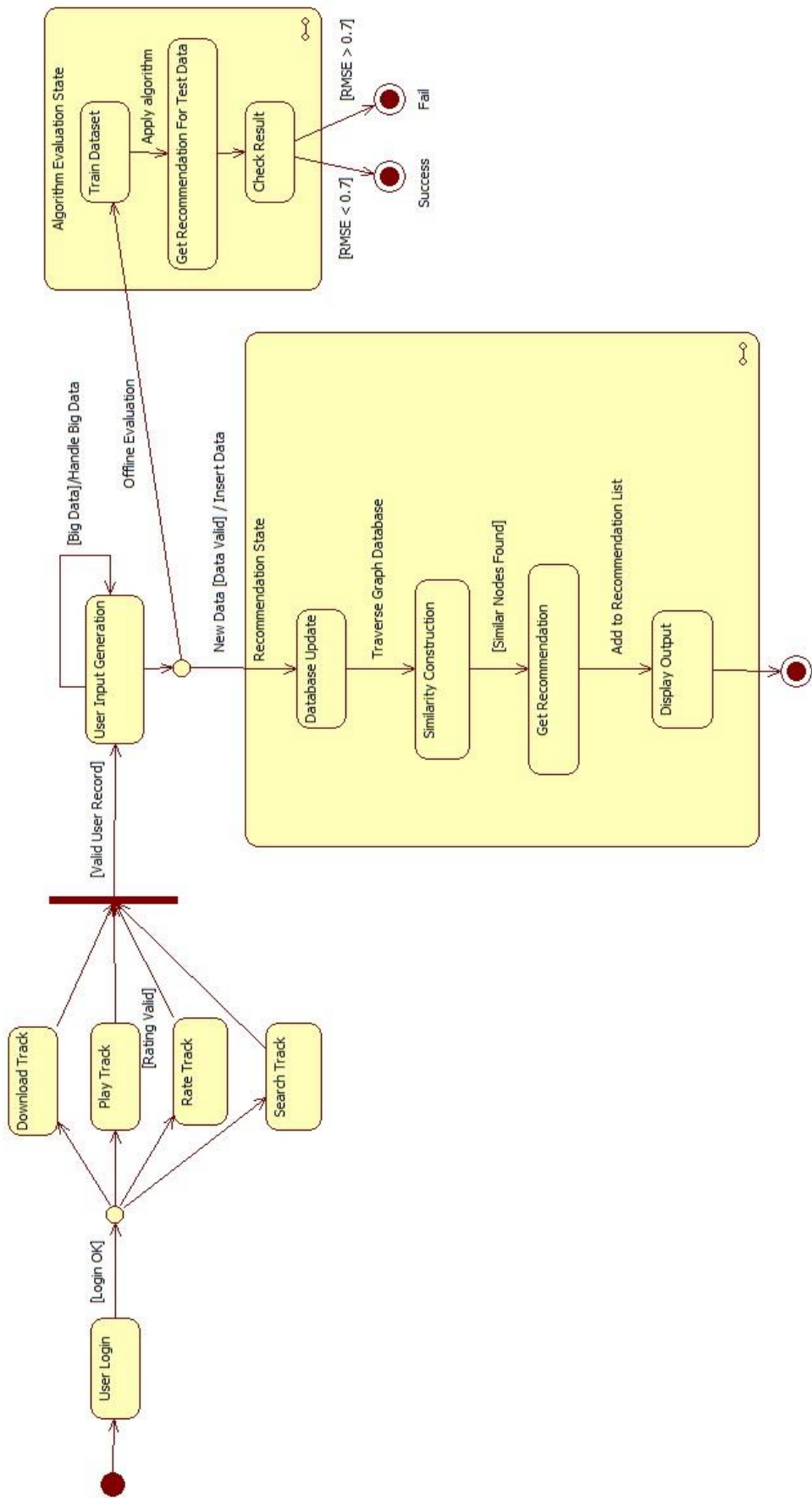
5.7. State Dynamics Viewpoint

This section includes the system behaviour and states of the Recommender System. The states and transitions are illustrated through UML State Machine diagram.

5.7.1. Design Concerns

The statechart in figure illustrates the behaviour 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 and algorithm evaluation systems are shown as independent substates. The states that remain outside the recommendation and algorithm evaluation states 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. After that point, the state machine either proceeds to

the recommender state or to the algorithm evaluation state. In the recommender state, the behavior of the web service mainly resides on database transactions. Since the project team uses graph database, the system traverses the large graph, which is actually a storage and notation method for the big user data, to find out the closest match to a user or to a track. In the 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.



5.7.2. DesignElements

The state chart diagram in figure 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 larger 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, either the actual recommendation sub state or the algorithm evaluation sub state is triggered. In the recommender sub state, 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 and tracks. From that point on, if Similar Nodes Found guard is satisfied, the recommendation is generated and displayed to the external system. The algorithm evaluation sub state is triggered on the condition that offline evaluation is demanded. In the first state of this sub state, some part of the user data is trained to the system's recommendation algorithm. Then the rest of the data is used as the test data to check if the recommendations to these test data comply with the actual matches. If expected results are received then the state machine halts successfully. Otherwise, the algorithm fails.

6. Planning

The distribution of tasks among the team members is as follows:

Task	Members
Graph Traversal Algorithm Component	DuyguKabakcı, İřinsu Katırciođlu
Neo4j Database Component	DuyguKabakcı, İřinsu Katırciođlu
Neo4j API Component	DuyguKabakcı, İřinsu Katırciođlu
Machine Learning Algorithm Component	Sıla Kaya, Mehmet KorayKocakaya
UI Component	Sıla Kaya, DuyguKabakcı
Weka API Component	Sıla Kaya, Mehmet KorayKocakaya
Evaluator Component	Mehmet KorayKocakaya, İřinsu Katırciođlu
Recommender Component	Sıla Kaya, Mehmet KorayKocakaya

The Gantt chart below illustrates the long term planning of the project stages. The dark and light blue regions represent team work whereas other colors represent the workload of distinct members or subgroups of the team.

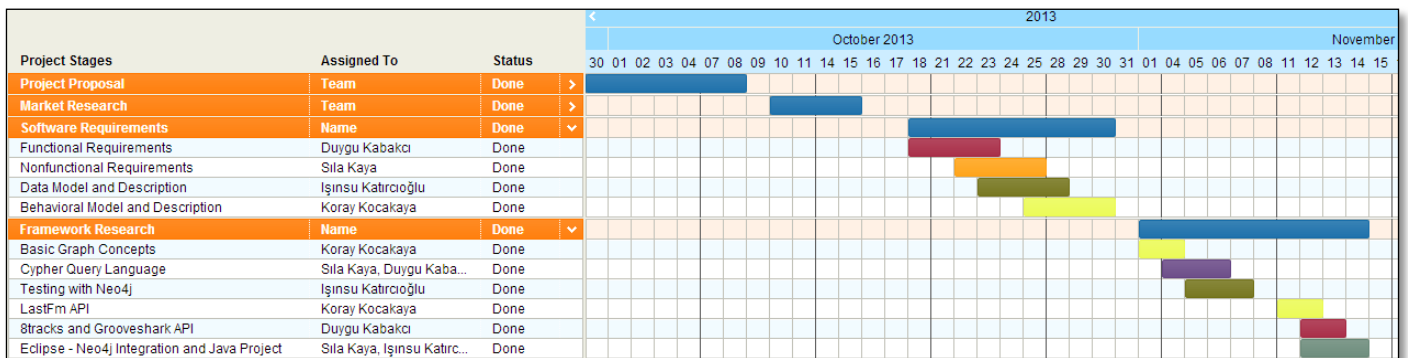


Figure 6.1: Gantt chart - 1

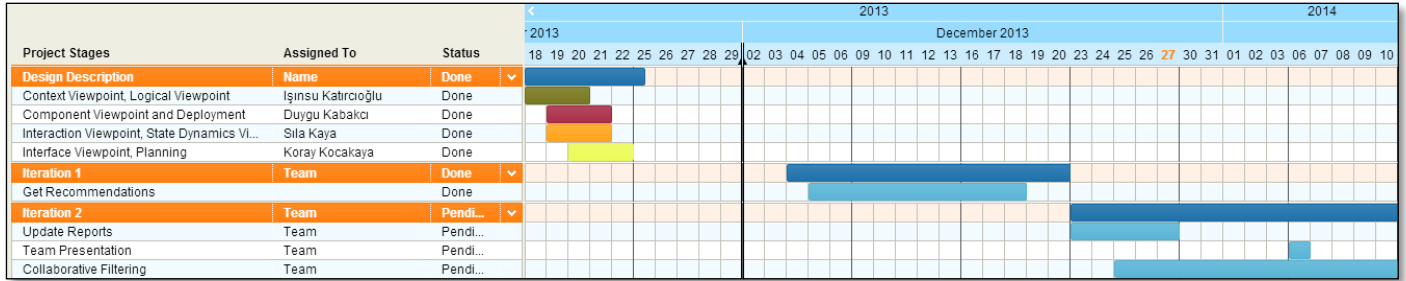


Figure 6.2: Gantt chart - 2

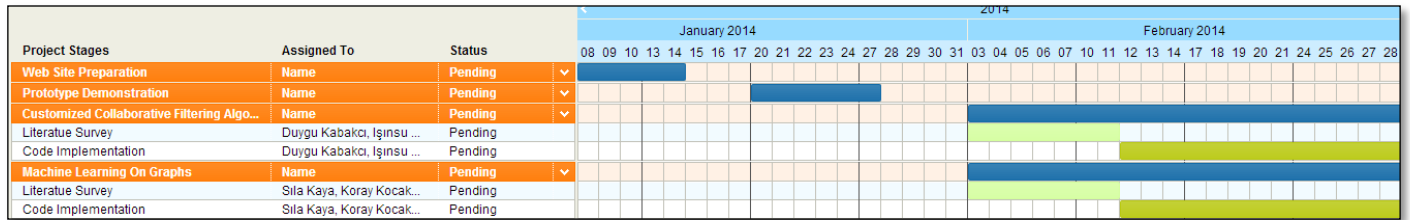


Figure 6.3: Gantt chart - 3

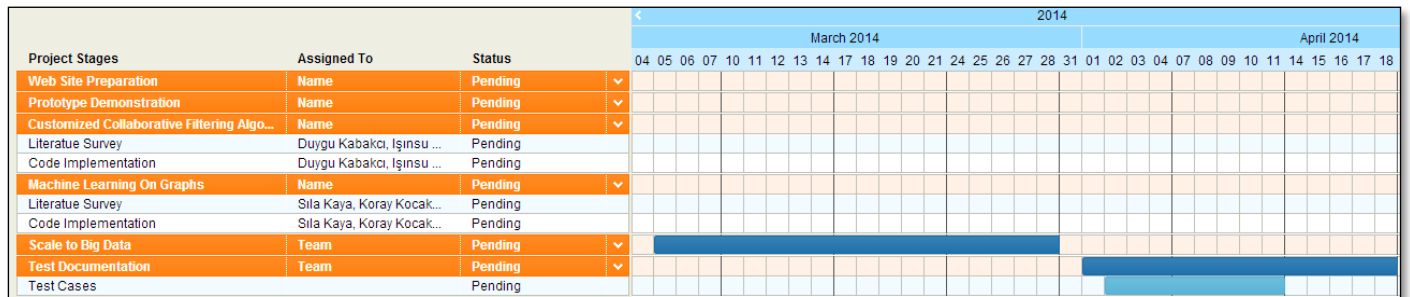


Figure 6.4: Gantt chart - 4

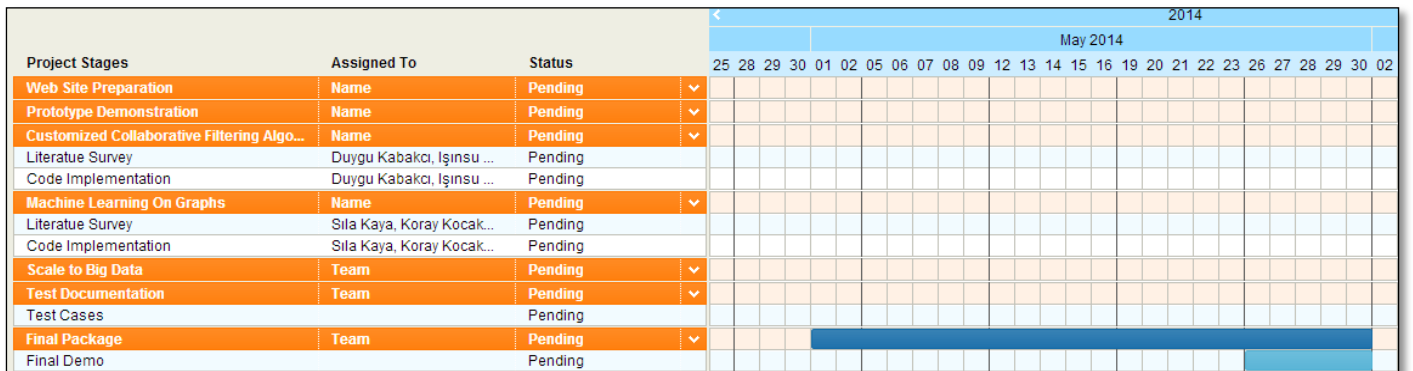


Figure 6.5: Gantt chart - 5

7. Conclusion

This Software Design Description Document includes the system architecture and implementation details of the Recommender System project. In this document, basics of data design, modules and design viewpoints of the system are described respectively. The software tools, frameworks and libraries that will be used while designing and developing the system are also identified. The design issues are specified along with the related design viewpoints. The Gantt chart that covers the project stages of this semester and next semester is provided in order to indicate the major milestones and schema of the project.

This page intentionally left blank