**Software Requirements Specification**

**For**

**CODEGO**

**Prepared by :- AYESHA JAVED**

**AYESHA MALIK**

**MAHNOOR ALTAF**

**KIREN RIASAT**

**<BSCS>**

**<23-09-2019>**

**PREFACE**

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

The Recommender System aims to enhance the existing recommendation tools served to Source code . In this project, the challenge is to overcome the major constraints and carry out the basic functionalities of such systems. The target Develpers who wants to make use of this system, can find all related requirements information in this document. It assists the software developer team, the stakeholders and the end users.

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

**TABLE OF CONTENT**

**1. INTRODUCTION ...........................................................................................**

1.1 PROBLEM DEFINITION .................................................................................................

1.2 PURPOSE .....................................................................................................................................

1.3 SCOPE .........................................................................................................................................

1.4 OVERVIEW .......................................................................................................

**2. OVERALL DESCRIPTION ............................................................................**

2.1. PRODUCT PERSPECTIVE ...............................................................................................................

2.1.1. SYSTEM INTERFACES ...................................................................................................................

2.1.2. USER INTERFACES .......................................................................................................................

2.1.3. HARDWARE INTERFACES...........................................................................................................

2.1.4. SOFTWARE INTERFACES ............................................................................................................

2.1.5. COMMUNICATION INTERFACES .................................................................................................

2.2. PRODUCT FUNCTIONS .....................................................................................................................

2.3. CONSTRAINTS ...................................................................................................................................

2.4. ASSUMPTIONS AND DEPENDENCIES ................................................................................................

**3. SPECIFIC REQUIREMENTS .............................................................................**

3.1. INTERFACE REQUIREMENTS ............................................................................................................

3.2. FUNCTIONAL REQUIREMENTS .........................................................................................................

3.2.1. USER USE CASES ......................................................................................................................

3.2.1.1. USE CASE: GENERATE DATA ................................................................................

3.2.1.2. USE CASE: GET RECOMMENDATION .....................................................................

3.2.2. INTERAGENT..............................................................................................................................

3.2.2.1. USE CASE: PROVIDE DATASET ............................................................................

3.2.2.2. USE CASE: UPDATE DATASET ..............................................................................

3.2.2.3. USE CASE: INTEGRATE WEB SERVICE ..................................................................

3.3. NON-FUNCTIONAL REQUIREMENTS .................................................................................................

3.3.1. PERFORMANCE REQUIREMENTS ...............................................................................................

3.3.2. SECURITY ..................................................................................................................................

3.3.3. USABILITY .................................................................................................................................

3.3.4. PORTABILITY .............................................................................................................................

3.3.5. SAFETY ......................................................................................................................................

3.3.6. OTHER REQUIREMENTS – GNU GPL LICENSE ........................................................................

**4. DATA MODEL AND DESCRIPTION ...............................................................**

4.1. DATA DESCRIPTION .........................................................................................................................

4.1.1. DATA OBJECTS ..........................................................................................................................

4.1.2. DATA DICTIONARY ....................................................................................................................

**5. BEHAVIORAL MODEL AND DESCRIPTION .....................................................**

5.1. DESCRIPTION FOR SOFTWARE BEHAVIOR ........................................................................................

5.2. STATE TRANSITION DIAGRAMS.........................................................................................................

**6. PLANNING.....................................................................................................**

6.1. TEAM STRUCTURE ............................................................................................................................

6.2. ESTIMATION .....................................................................................................................................

6.3. PROCESS MODEL .............................................................................................................................

**7. CONCLUSION ...........................................................................................**

1. **INTRODUCTION**

Contemporary software development processes involve ﬁnding reusable software components from online repositories and integrating them to the source code, both to reduce development time and to ensure that the ﬁnal software project is of high quality. Although several systems have been designed to automate this procedure by recommending components that cover the desired functionality, the reusability of these components is usually not assessed by these systems. In this work, we present CODEGO, a recommendation system for source code components that covers both the functional and the quality aspects of software component reuse. Upon retrieving components, CODEGO provides a ranking that involves not only functional matching to the query, but also a reusability score based on conﬁgurable thresholds of source code metrics. The evaluation of CODEGO indicates that it can be eﬀective for recommending reusable source code.

* 1. **PROBLEM DEFINITION**

Lately, the rise of the open source community and the introduction of online source code repositories have provided numerous exploitation possibilities in the context of software component(source code) reuse. Developers often rely on ﬁnding reusable source code components, both to reduce the time spent to develop them and to ensure that the resulting software is of high quality (in terms of reliability and functionality delivered).Although several systems have been designed to automate this procedure by recommending components on the basis of reusability score metrics that cover the desired functionality which involve ﬁnding reusable software components from online repositories and integrating them to the source code, both to reduce development time and to ensure that the ﬁnal software project is of high quality, but the reusability of these components is usually not assessed by these systems.

* 1. **PURPOSE**

This document provides a detailed description of Software Requirements Specification (SRS) for Recommendation System. It is prepared according to “IEEE Recommended Statements for Software Requirements Specification - IEEE Standard 830 – 1998”.

The Software Requirements Specification (SRS) document is intended to provide the requirements of the Recommender System project and the expectations of the stakeholders. The document includes the project perspective, data model and constraints of the overall system.

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

* Project managers review the document and determine whether the planned system fulfills the requirements. They notify the developers to fill up missing parts.
* Developers provide consistency by using the documentation.
* Testers use the documentation for verification and validation.
* End users make use of this document to learn about the scope of the system and its capabilities.
  1. **SCOPE**

GODEGO is a source\_code recommendation system, which provides users(developer) a code which they may want,based on the query which the user provided. Every logged in user should have access to the CODEGO.

In CODEGO recommendation system we proposed to do the Online source code repositories and question answering communities, such as GitHub or Stack Overﬂow, have facilitated the task of ﬁnding suitable (with respect to a developer query) source code.

The system were crawling the intented source\_code from websites(Github,Stack\_overflow,Geeks for Geeks ) then according to the information extracted, it should provide the intented source-code on reusability metrics based recommendations according to the user perspective.

* 1. **Objectives:-**

The project’s main aim is to provide accurate source\_code recommendations to the user(developer). This project is beneficial for the user, they may browse source\_code that

without consuming time and even they can encounter more items which they want from the recommendations.

For the company, they make the website more attractive, so they draw more users to the website and the system makes the users of the website spend more time online and also make an IDE more attractive so that to reduce software costs, and improve software quality as well as programmer productivity.

1.5**. OVERVIEW**

This document includes six topics:

● Overall description

● Specific requirements

● Data Model and Description

● Behavioral Model and Description

● Planning

● Conclusion

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

**2. Overall description**

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

**2.1** **Product perspective**

CODEGO system depends on data come from websites. This program has different type of users, so there is functionality differences between users will occur with respect to item data.

The recommendation systems used by companies have differences for efficiency. Our recommendation system should work efficiently according to source\_code data. So, there exists a user interface that is suitable for code recommendations website. Our system will be working in background.

Once the recommendation system finds an accurate result, it will be shown on the interface. In terms of hardware, recommendation system will be embedded in a website. To use or benefit from recommendation system, user should enter from a personal computer with internet connection. In terms of software, our recommendation system will run on personal computers.That is, it will run on any device with internet connection.

The system will work both Windows and Unix operating systems. Moreover, it will be implemented making use of database management. This brief information of interfaces is explained in more detailed below

**2.1.1 System interfaces**

First of all, the application needs to have source\_code data to make recommendation to user and user query to evaluate. User query is necessary to make a really relevant recommendation. The system will use this information to make inference about recommendation by getting user’s actions. Reusability score of a source\_code data for ranking purposes is a derived value obtained by an algorithm.

**2.1.2 User interfaces**

There will be one type of user. Therefore there are no differences between users in terms of functionality, visualization and interface. The user interface is only depend on the websites designers or application designers. The window application and the website that our system is working on the background of it can offer different opportunities to its users. For example, a website can offer the right to choose background image to each of its users. However the architecture will probably be the same for all users.On the other hand, the interface will be in some number of steps such as user login, search options etc. But the first step should be user login to activate the CODEGO. Then the recommendations will be specified according to the algorithm that depends on user query.

Up to now, we mentioned about how the recommendation will be done, but making recommendation more recognizable is also an important point. After getting a recommendation for the user, the system should display it on the interface.

**2.1.3 Hardware interfaces**

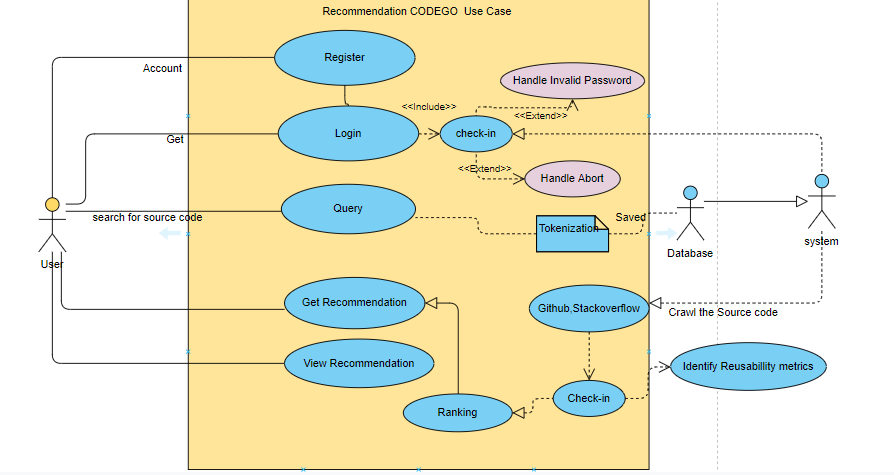
The CODEGO can work on any internet connected device. These devices should have some limit requirements to make the application run effectively. We expect that the processor speed and internet speed are high.

**2.1.4 Software interfaces**

First of all the system will work on any platform. Internet connection is a must to reach the system. Moreover, most of the application will be coded by .Net C#,javascript. SQL database is used to saved the tags or keywords of the user query after Tokenization.

**2.2 Product functions**

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



**2.2.1 Acquire and Index New Items**

This function inserts new items by indexing them like the previously saved items. This functionality may be achieved by a query language. After adding this item to database, this item will also ranked according to its album id for users who listens that album.

**2.2.2 Get Customer Profile and Statistics**

This function provides customer profile information and statistics to supplier of the whole system. Actually, this function provides the whole information needed to make a recommendation.

**2.2.3 Monitor User Actions/Feedback**

The system should monitor user actions as the system receives implicit feedback through it.

**2.2.4 View Recommendations**

To make user tend to make use of an opportunity of a recommendation, monitoring is an important point. Here supplier uses this functionality to get attention of the user on a recommended item by displaying it in an attractive way.

**2.2.5 Provide Feedback**

The users must be able to read reviews and provide feedback to the system. This feedback is crucial to provide more accurate recommendations in time.

**2.3 Constraints**

● Since we need user profile data while developing the product, to find real time and sufficient data can be a problem for developer because of regulatory policies.

● Millions of data will be needed to test the software. At this stage developers will need huge amount of disk space and clusters.

● The application gathers real time user profile information from user accounts. Therefore, it must be reliable and keep those data in safe. Moreover, the system will produce new data about users depending on their behavior on the web. Security of this resulting data must be provided by the software also.

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

**2.4 Assumptions and dependencies**

As stated in the previous section which is constraints, there are several requirements like source\_code data, user data, database management tool etc. To accomplish activating recommendation system, these requirements should be provided. However, the case of having all these software and hardware provide, still we might have some difficulties to test the system without strong internet connection. Losing connection is an important problem, because our system will work on online platform. If the internet connection cannot be supplied as requested, the whole sales policy will fail and our system will be useless. We stated that we will use C# as a programming language to code our system. However, we can also study in other platforms. For example, we might study in Visual Studio.

**3.Specific requirements**

**3.1 Interface Requirements**

The user needs to click the link to the website. Then he/she needs to register to the system by

providing a password and an email, otherwise he/she won’t be able to use the Recommender

System. Then, to benefit from the Recommender System he/she needs to be active on the website

by listening and downloading music, adding his/her favorites to the list or sharing them on

facebook/twitter.

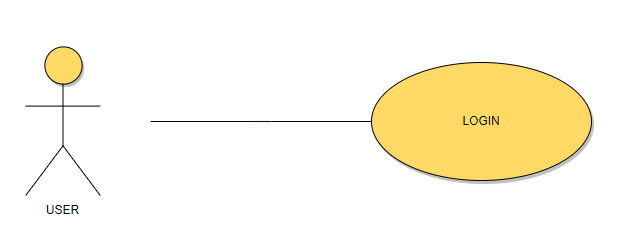
3.2 **Functional Requirements**

This section outlines the use cases for each user registered to the website(larger system).

3.2.1 **Login Use Case**

Use case: Login

Diagram



**Figure3. Login Use Case**

**Brief Description**

The user logins to the system to see what recommendations he/she has.

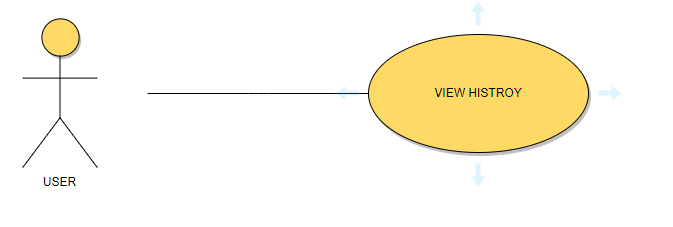
**Initial Step-By-Step Description**

1. The user needs to open the website.

2. The user needs to type his/her username and password to login.

3.2.3 **View History Use Case Use case: View History**

Diagram:



**Figure4. View History Use Case**

**Brief Description**

The user can listen to music on the website online.

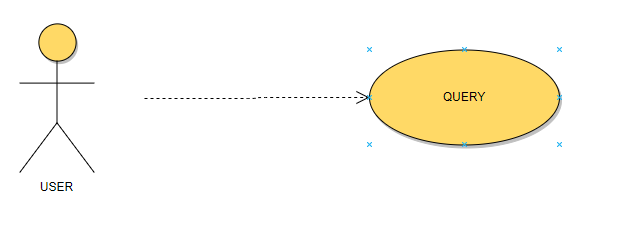
**Initial Step-By-Step Description**

1. The user needs to open the website.

2. The user needs to login to the system.

3. After being logged in, the user can see his/her past actions.

**3.2.1.2. USE CASE: USER QUERY**



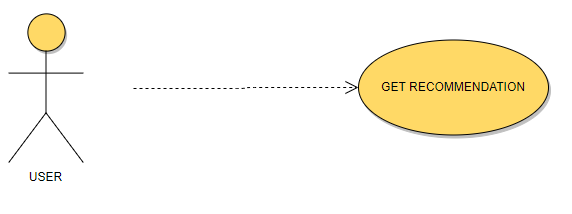
**Brief Description**

The user will QUERY TO THE SYSTEM of the source code which they want to be retrieved.

**Initial Step-By-Step Description**

1. first the user will provide the query.
2. System will parse the query,fetch the keyword.
3. Extarct the pattern.

**3.2.1.2. USE CASE: GET RECOMMENDATION**



**Brief Description**

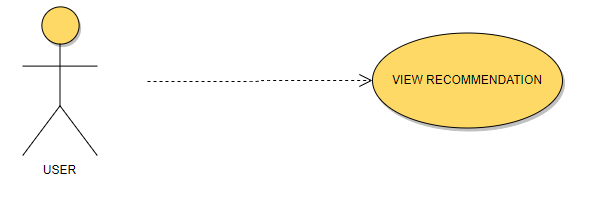
The user will get the recommendations of the source code which they want to be retrieved.

**Initial Step-By-Step Description**

1. first the user will provide the query.
2. System will parse the query,fetch the keyword,then
3. System provide the recommendations on the bases of the user query.

**3.2.4 View Recommendations Use Case Use case: View recommendations**

Diagram:



**Figure5. View Recommendations Use case**

**Brief Description**

The user will be able to view recommendations of the source code on the webpage.

**Initial Step-By-Step Description**

1. The user needs to open the website.

2. The user needs to login to the system.

3. After listening music, the user will be able to view music recommendations selected using by

their past actions and preferences

**3.3 Non-functional Requirements**

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

**3.3.1 Performance requirements**

As a larger system, the website has a monthly traffic of over 4 million users. Since

recommendation system is planned to be designed for the use of every user in the larger system,

it is easy to say that this system will have a monthly traffic of over 4 million users. However, not all 3 million users are logged by the website or not all logged information will be given for the use of the system. By the company, daily 1 million logs will be given to be handled. User information of the past 60

days will be the reference point of the system, that’s why 60 million logs need to be handled in

order to implement the whole recommender system.

● **Accuracy**

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

● **Failure handling**

System components may fail independently of others. Therefore, system components must be built so they can handle failure of other components they depend on.

● **Openness**

The system should be extensible to guarantee that it is useful for a reasonable period of time.

● **Security**

Sensitive information should be kept in safe.

**3.3.2 Design constraints**

**● Language**

The product should be integratable with source code websites such as (Github,Stackoverflow,quora). Also we will handle with a lot of parameter and object. At this point, we need an object-oriented and commonly used language. As stated above, So, Java programming language will be used for these aims.

● **Hardware Constraints**

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

● **Software System Attributes**

➔ Usability

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

➔ Reliability

The system should have accurate results and fast responses to user’s changing habits.

➔ Security

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

**4. DATA MODEL AND DESCRIPTION**

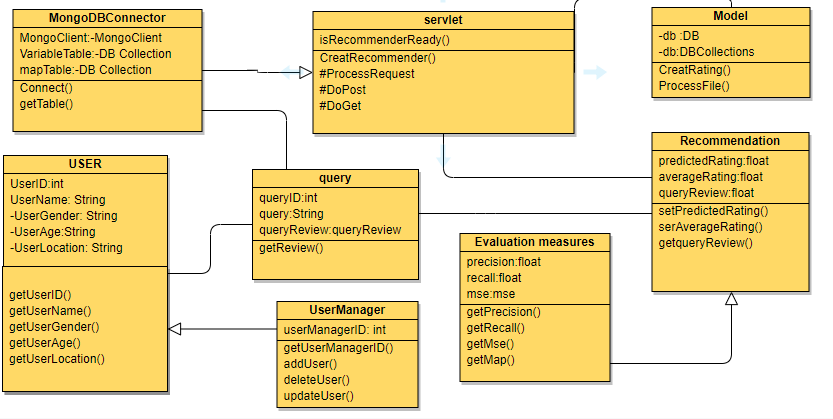
**4.1. DATA DESCRIPTION**

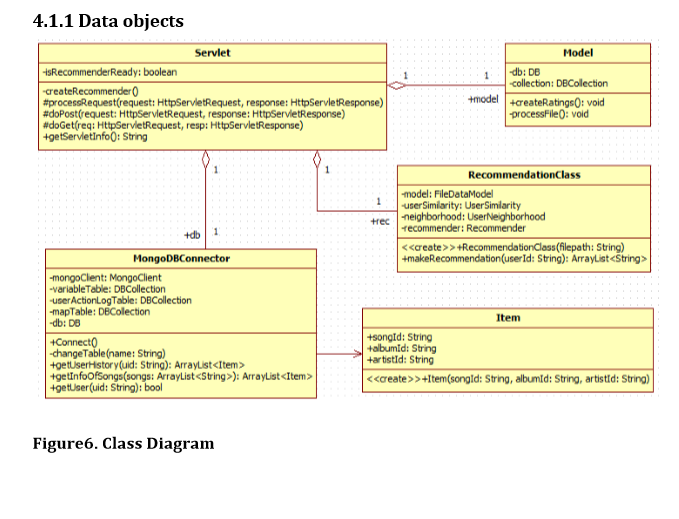
Data objects in the source\_code Recommender that will be managed/manipulated by the software are described in this section with their attributes using class diagrams

**4.1.1. DATA OBJECTS**

Source\_code recommendation system roughly has 5 types of data objects, namely User, query , Recommendation, UserManager, EvaluationMeasures.

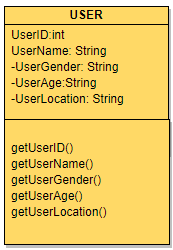
This subsection of the document explains system's classes and their relations with each other. Most of the system functionalities are represented in Figure 11.





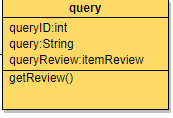
**4.1.1.1 User**

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



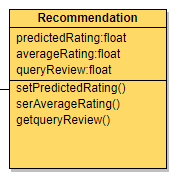
**4.1.1.2 Query**

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



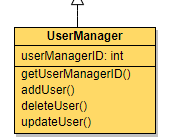
**4.1.1.3 Recommendation**

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



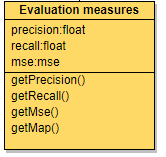
**4.1.1.4 UserManager**

This object handles the user addition and deletion operations



**4.1.1.5 EvaluationMeasures**

This object handles the operations about evaluating the data.



* **DFD**

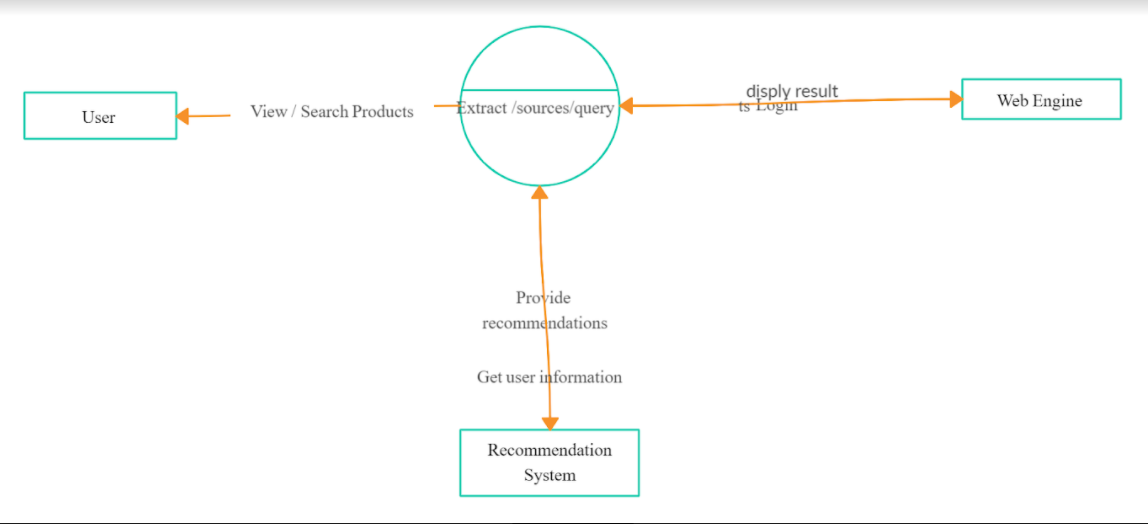
Data Flow Diagram

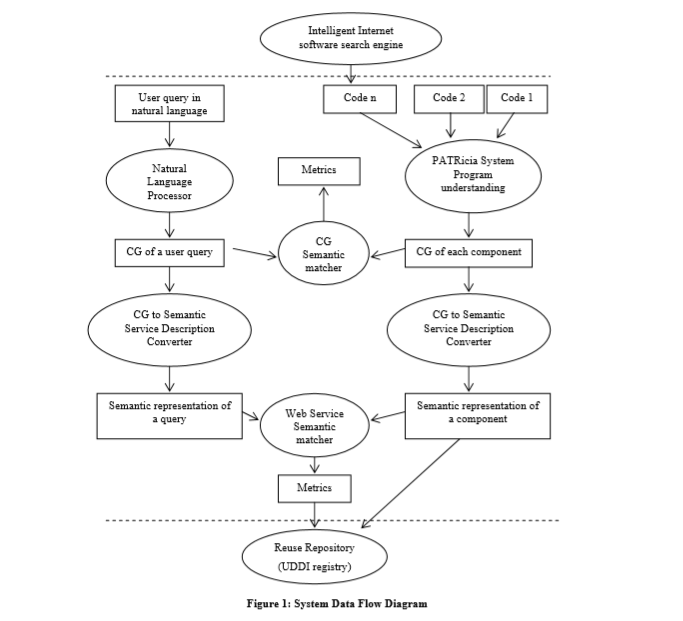
-User

-Data(Extarct/sources/query)

-Web engine

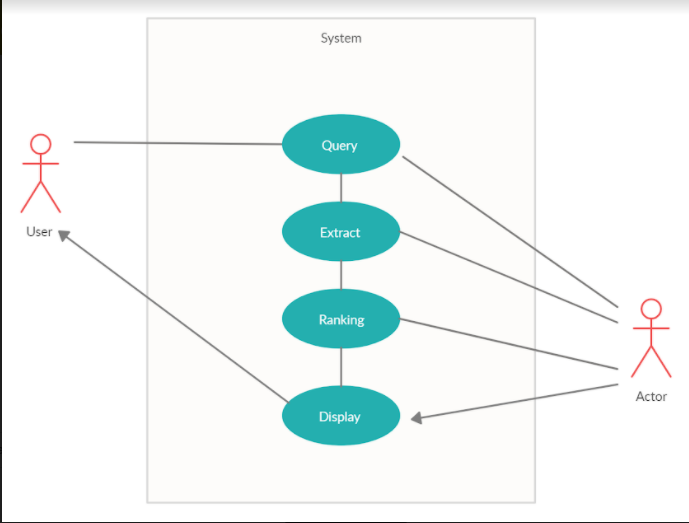
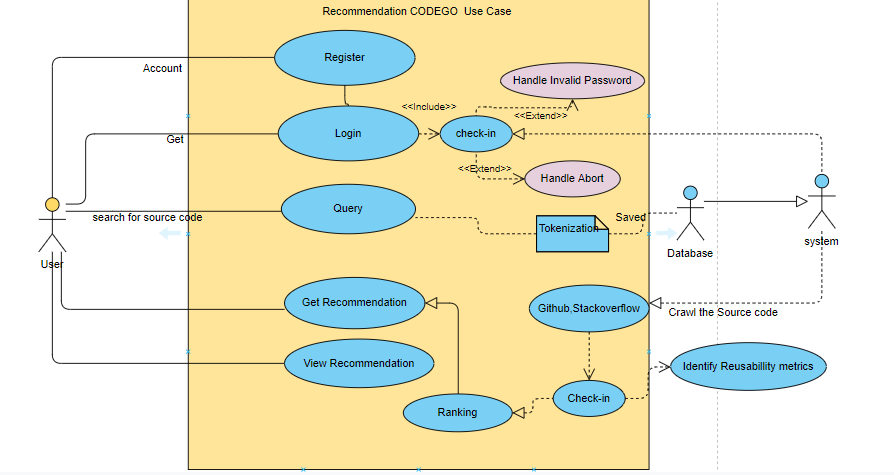
-Recommendation system





* **UML:-**

Recommendation systems UML diagram to show the USER perspective Views



* **DOMAIN MODEL**
* **ERD**

**SYSTEM DIAGRAM**



**4.1.2 Data dictionary**

|  |  |  |
| --- | --- | --- |
| **Name** | **Type** | **Description** |
| Age | Int Age | Age of the user |
| Gender | Character | Gender of the user |
| Occupation of the user | Int Job | Occupation of the user |
| userId | Int Unique Id | userId of the user of the website |
| Source\_code ID | Int An Id | Id for the code |

**5 Behavioral Model and Description**

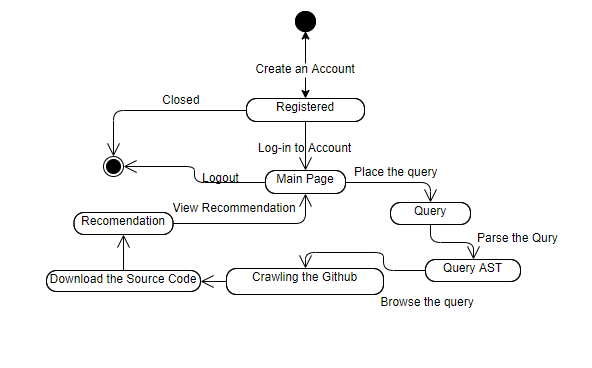
**5.1 Description for software behavior**

This subsection describes the major events and states of our software. When the user open

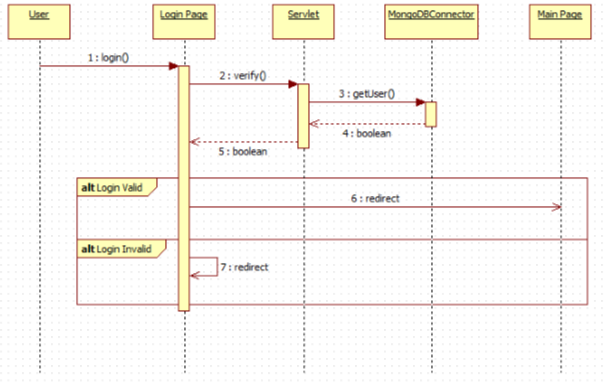
the webpage at first, user will see a login screen. In this screen user will required to provide

username and password. If the information is valid the user will be able to move on to the main

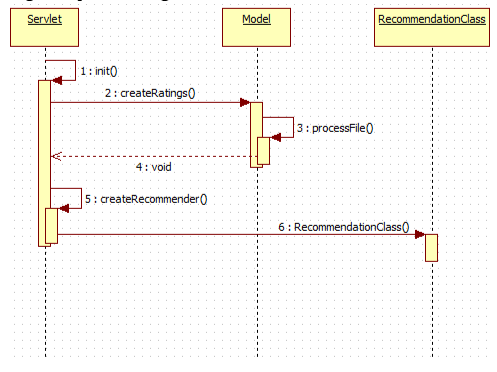
page. On this page user will see the browsing tab where able to browse from the query every user will be able to view recommendations. Users will be able to logout of the system.



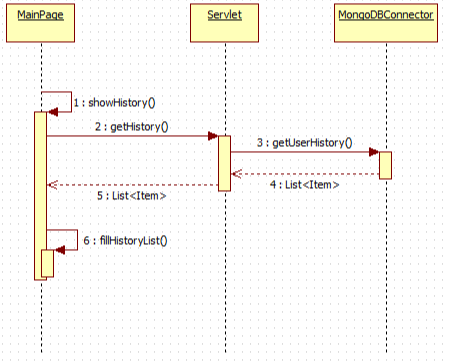
**5.2 State Transition Diagrams**



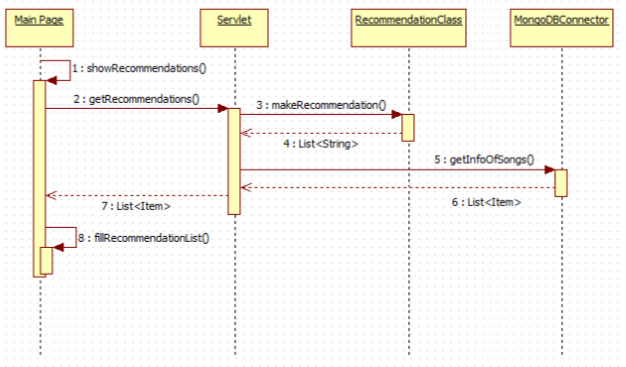
**Figure 8.Login Sequence Diagram**



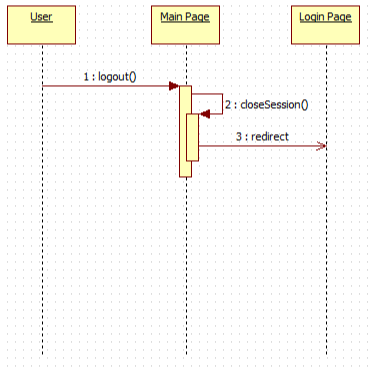
**Figure 9.Generate Recommendations Sequence Diagram**



**Figure 10.View History Sequence Diagram**



**Figure 11.View Recommendations Sequence Diagram**



**Figure 12.Logout Sequence Diagram**

**6 Planning**

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

**6.1 Team Structure**

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

|  |  |
| --- | --- |
| **Task** | **Members** |
| Graph Traversal Algorithm Component | AYESHA JAVED,AYESHA MALIK |
| Database Component | MAHNOOR ALTAF,KIREN RIASAT |
| Machine Learning Algorithm Component | AYESHA JAVED,MAHNOOR ALTAF |
| UI Component | AYESHA MALIK,KIREN RIASAT |
| Weka API Component | AYESHA JAVED |
| Recommender Component | KIREN RIASAT |

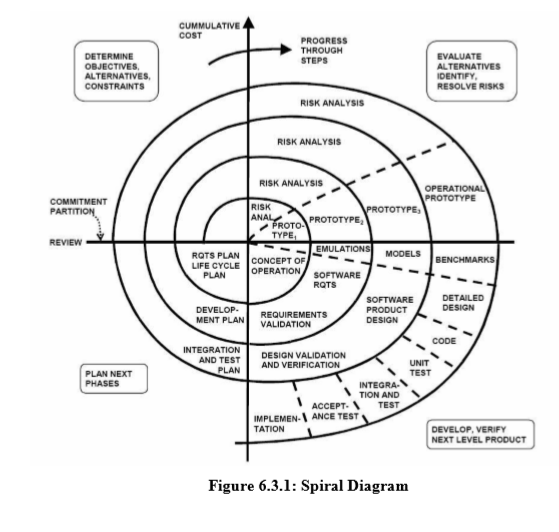
**6.2. ESTIMATION**

**Estimation**

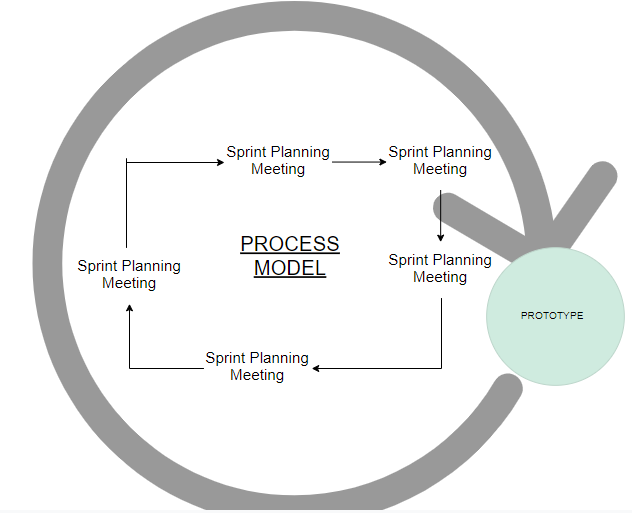
|  |  |
| --- | --- |
| **Estimation Date** | **Date Task** |
| 04.9.2019 | Deciding Project |
| 13.9.2019 | Project Idea Proposal |
| 22.9.2019 | Elevator Pitch |
| 30.9.2019 | Software Requirements Specification (SRS) |
| 30.9.2019 | Researching Requirement Systems and Tools |
| 07.10.2019 | Trying Software Tools |
| 14.10.2019 | Documenting Design Report |
| 18.10.2019 | Design Report |
| 18.10.2019 | Trying Software Tools |
| 04.10.2019 | Documenting Updated Reports |
| 16.10.2019 | Updated Reports |
| 20.11.2019 | Implementing Base Recommendation System |

**6.3. PROCESS MODEL**

In this project, spiral model,the spiral model is a software development process combining elements of both design and prototyping-instages, in an effort to combine advantages of top-down and bottom-up concepts



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



**7. Conclusion**

Software Requirement Specification is an important part of the process of project development. Moreover, it is a prerequisite for creating the following design documentation.

This Software Requirement Specification document is prepared to give requirement details of the

project, “Source\_code Recommender System”. First, definition of the problem and the general

description of the system are given. Then, all the functional, non-functional and interface

requirements, data and behavioral models are stated in a detailed manner. Finally, structure of the

development team, basic planning and estimation of the development process and the model of

the development process are explained. This document will be helpful at constituting a basis for

design and development of the system to be developed. Hence This document has been created through the help of various researches and depending on the demands of the customer. However, some little specifications are prone to be changed in the future.