ABSTRACT

Web Services (WS) are application components which help in integrating various Web based applications. WS are used by almost all web applications. With the help of WS, web applications can provide service on the internet without any restrictions to the operating system or programming language. Today the number of WS on the internet are rising and it is difficult for the user to select a well suited service among a large number of services which gives low performance to the user and it is a potential risk for a business process. In order to overcome this, based on a recently published research work in the literature, we develop a location-aware QoS based web services recommendation system. The recommendation system obtains the QoS information from the users and the location information of the user which provides personalized results to the users. The recommendation system implements this concept by using collaborative filtering (CF) technique and recommends potential favorite Web Services for a user.
# TABLE OF CONTENTS

Abstract .................................................................................................................................................. ii  
Table of Contents ............................................................................................................................... iii 
List of Figures ....................................................................................................................................... iv 
1. Introduction ...................................................................................................................................... 1  
  1.1 Web Services .............................................................................................................................. 1  
  1.2 Recommender System ................................................................................................................ 2  
  1.3 Recommender System for Web Services ..................................................................................... 3  
  1.4 Collaborative Filtering ................................................................................................................ 4  
  1.5 Scalability ..................................................................................................................................... 6  
  1.6 Literature Survey ........................................................................................................................ 6  
2. Narrative .......................................................................................................................................... 10  
  2.1 System Overview ......................................................................................................................... 10  
  2.2 System Features .......................................................................................................................... 10  
  2.3 Problem Statement ...................................................................................................................... 12  
  2.4 Proposed System ........................................................................................................................ 12
Test Case 1 ................................................................. 40
Test Case 2 ................................................................. 40
Test Case 3 ................................................................. 41
Test Case 4 ................................................................. 43
Test Case 5 ................................................................. 44

5.3 Evaluation ................................................................ 46

6. Conclusion and Future work .................................... 51

Bibliography and References ........................................ 52
List of Figures

Figure 2.1. Use Case Diagram.............................................................................. 16

Figure 3.1. Architecture of Recommender System............................................. 18

Figure 3.1.2. Flow chart......................................................................................... 19

Figure 3.2 Architecture of the System...............................................................20

Figure 3.3. Sequence Diagram.............................................................................23

Figure 4.1. Few Entries in T_LOCATION Table................................................... 28

Figure 4.2. Sorted Entries...................................................................................... 29

Figure 4.3. Entries of User Feedback in the T_Feedback Table .........................30

Figure 4.4. Sorted Entries of Feedback...............................................................31

Figure 4.5. Entries of Search Keyword in T_Search Table...............................34

Figure 4.6 Location Table Based on IP address..................................................35

Figure 4.7. User Table Showing the Interests.....................................................37

Figure 5.1. DataSet Website.................................................................................39

Figure 5.3. No Results Found for Search.............................................................40

Figure 5.4. Output for Search Option.................................................................41

Figure 5.5. (a) Feedback Form Validation..........................................................42

Figure 5.5. (b ) Feedback Form Validation..........................................................42
Figure 5.6. Submit Feedback Form………………………………………………………….. 43

Figure 5.6. Checking the Output of Feedback Form in User Profile…………………………. 44

Figure 5.7 Testing the Link Access………………………………………………………….. 44

Figure 5.8. Web Service Example ………………………………………………………….. 45

Figure 5.9 (a) Comparing the Feedback Output…………………………………………….. 46

Figure 5.9 (b) Comparing the Feedback Output…………………………………………….. 47

Figure 5.10. Output for History Based Recommendation……………………………………. 48

Figure 5.11. Output for Interest Based Recommendation…………………………………… 49

Figure 5.12. Interests of the User from the User Table………………………………………… 49

Figure 5.13. Output for User Profile………………………………………………………….. 50
List of Tables

Table 1.1. User Service Table ................................................................. 6
Table 2.2. Use Cases and their Functions .............................................. 16
Table 3.1. User Location Table .............................................................. 21
Table 3.1.2. Sorted User Location Table .................................................. 21
Table 3.3. User Service Table ............................................................... 22
Table 3.4. User Service Table ............................................................... 25
Table 5.2 Web Services List ................................................................. 39
1. BACKGROUND AND RATIONALE

1.1 Web Services

What are Web Services (WS) and how does a Web Service work?

The use of the web services technology on the internet has increased widely as it has improved the efficiency and throughput for developers in developing applications. Often we get confused between a website and a web service as both of them are rendering some kind of service to the end user over the network but there exists a difference between the two, and it is that, a website is meant for human consumption and a service which is meant for code consumption or application level consumption is called a web service. There are a lot of interpretations of web services and we can say that web services are designed to compose various software components and provide machine-machine interactions over a network and the name itself suggests that it is a type of service that is deployed on the internet [1]. It allows two different applications running on different servers to interact with each other over the network and these applications can be implemented in different languages. For example, a Java application can call methods of an application in .NET. Hence, it supports interoperability. For the web services in order to support such kind of interactions it provides an intermediary interface between the consumer and the application which is providing the implementation. The interface should be in a format that supports interoperability, hence this interoperability is provided with the support of some technologies like Hyper Text Transfer Protocol (HTTP), Universal Description Discovery and Integration (UDDI), Web Services Description Language (WSDL) for description of the interface and Simple Object Access Protocol (SOAP) for exchanging structured information. When a web service is created, the web service contract must be shared for various applications
to use it. This contract is written as an Extensible Markup Language (XML) document and the WSDL document contains the contract of the WS. When a web service is created we share WSDL document of that Web Service to the consumers. The consumers require a source to search for these Web services and all the Web services after their creation the developer must publish it to a registry called UDDI, it is like the yellow pages of WS. Consumer can query the UDDI for the required WS. The SOAP protocol is used for accessing the object of the Web service. For example, when two applications are communicating, and we suppose that one application is written in JAVA and the other in .NET, a JAVA class object cannot access the data of an application in .NET. So in order to overcome this, a class called service endpoint interface converts the JAVA object into a SOAP object and this can be generated through the WSDL. These are the various levels of communication involved in the Web Service development and deployment [2].

1.2 Recommender Systems

Recommender systems are applied to various applications now a days. E-commerce sites like amazon and eBay, social networking sites like Facebook, Twitter and YouTube are some examples of the variety of applications in which recommender systems are being used. As there is huge amount of data to be searched, it is difficult to get a limited and accurate set of results when a user searches for a particular information, hence the recommender system will make use of the “likes” and “dislikes” of various users and generates recommendations based on his/her interests and the advantage of a recommender system is to reduce the user’s time for searching the required information by narrowing down the choices that the recommender algorithm predicts a user might be interested in.
Recommender systems employ Information Filtering technique that focuses on providing the recommendations of the items to the users that are likely to be of the user’s interest. A recommender system is defined as: “if $U$ is the set of users and $I$ is the set of all possible items that can be recommended, then there exists a function from $U \times I$ to $R$ where $R$ is a totally ordered set of nonnegative integers or real numbers within a certain range” [3].

1.3 Recommender System for Web Services

Web Services have been widely accepted over the internet, they have been employed by individual developers and companies for building services through this application. As the abundance of web services have increased, designing an effective method for recommendation and selection of web services has gained importance. Google developers, yahoo pipes, programmable web etc. are some of the existing web service recommendation public sites.

In order to predict Web Services for a user, user preferences, user location and web service properties should be considered, like QoS which has been considered as a major factor in service selection. QoS includes response time, price, correctness, etc. Among these properties some values like response time, etc. must be measured at the client side in order to get accurate feedback from the end user's point of view, which results in obtaining accurate results. Some QoS factors like reliability needs to be calculated by observing for long period of time. For the recommendation system it becomes difficult to get QoS data for all the services due to huge number of web services. These problems are overcome by giving personalized predictions to the user based on past user experiences or the feedback data. And the users can select the service which gives them optimal performance. QoS properties are highly related to the location of the user as the QoS experience varies from one user to other for a Web Service [4]. From this, we
understand that location information is important for recommending WS and also there are restrictions on using services based on the availability factor of that service. For example, a service created in India cannot be used and shipped to a different country due to trade compliance. This involves a great risk for service-oriented software. According to Google Transparency Report, user observed web service QoS values are highly dependent on the location of users [5]. To enhance this prediction accuracy, we have tried to implement a proposed system which takes into account of user location information and QoS information of web services called location-aware Web Service recommender system (LoRec). This system works in the following way. When a user of LoRec registers into the system and share their past experience with Web Services they have experience with, based upon which predictions are calculated and recommendations are generated for the user.

1.4 Collaborative Filtering

The growth of the Internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering. One of the techniques used for dealing with this problem is called collaborative filtering. The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with similar tastes to themselves. Collaborative filtering is a technique used by the recommender systems to make predictions and recommend potential favorite items to a user by finding similar users to that user, CF is based on user-item matrix. The underlying assumption of the collaborative filtering approach is that if a person $A$ has the same opinion as a person $B$ on an issue, $A$ is more likely to have $B$'s opinion on a different issue $x$ than to have the opinion on $x$ of a person chosen randomly [6]. Breese et al. [7] divide the CF algorithms into two broad classes:
memory based algorithms and model-based algorithms. Memory based collaborative filtering includes user-based approaches, item-based approaches and their fusion. User-based approaches predict the ratings of users based on the ratings of other similar users, and item-based approaches predict the ratings of users based on the similarity of the item. Memory-based algorithms are easy to implement, require little or no training cost, and can easily take ratings of new users into account but do not scale well to a large number of users and items due to the high computation complexity.

Model-based CF algorithms, on the other hand, learn a model from the rating data using statistical and machine learning techniques. Examples include clustering models [8], latent semantic models [8], [9], latent factor models [10], and so on. These algorithms can quickly generate recommendations and achieve good online performance. However, these models must be rebuilt when new users or items are added to the system [1].

The importance of collaborative filtering methods is: guessing the usefulness of a particular item for a particular user depending upon the previous history of other users. Consider Table 1.1, where 'U' and 'S' represent user and item respectively. U1, U2, U3, U4, U5 represent users and S1, S2, S3, S4, S5 represent items. Assume that we want to suggest a particular item for U1 (user 1), S3 (item 3) is recommended for the U1 (user 1) who is “similar” to U2 (user 2) as both of them have given high rating for the item S2, User 1 is likely to like item S3. The recommendations are made by comparing the “likes” and “dislikes” of users who have taste that is similar to that of the current user.
Table 1.1. User Service Table

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>3</td>
<td>4</td>
<td>?</td>
<td></td>
<td>2</td>
</tr>
<tr>
<td>U2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>U3</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td></td>
<td>3</td>
</tr>
<tr>
<td>U4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

1.5 Scalability

The data available in the websites is very huge. As the data consists of both data of the users and the data related to the Web Services on the website. It is very important for the website to function properly with the increasing number of items and users day by day. With the increasing users to the website, scalability becomes an important factor to be considered as the site needs to be scaled well with the increasing traffic.

1.6 Literature Survey

X. Chen et al. [11] say that effective QoS based recommendation is becoming more and more important and previous approaches have failed to consider QoS variance according to the user’s location and also provide limited information on the performance of service candidates. This paper proposes a novel collaborative filtering algorithm designed for large scale Web Services.
The recommendation approach makes use of region-based CF algorithm and consists of two phase process. The first phase, the users are divided into different regions based on their physical location and previous QoS experience on Web Services. In the second phase, when a user is requesting Web Services, it finds similar users for the current user and makes predictions for Web Services which have the best predicted QoS values for the unused services.

**J. Yin** et al. [12] stresses that QoS values are important and propose a novel collaborative QoS prediction framework. Let us assume that there are m users and n Web Services, and they contribute to an m x n web service QoS matrix R, and each entry r_{ui} represents a QoS value recording the specific usage information of web service i executed by user u and predicts missing QoS values of Web Services by using the concept of localization and matrix factorization. This approach assumes that users nearby share similar web service invocation experience and makes of matrix factorization framework for predicting missing QoS values.

**J. Zhu** et al. [13] propose a novel clustering-based QoS prediction framework, in which various Landmarks (computers) are deployed in the internet to monitor QoS information of the available Web Services by invoking these services at regular intervals and then cluster the computers based on the QoS information that has been obtained. It then clusters these small groups into a large existing cluster, and try to form hierarchy of clusters, this is done by measuring the latency between the landmark and the cluster, from this QoS predictions are made from the QoS information that has been gained from the landmarks.

**G. Kang** et al. [14] propose a Web Services recommendation approach which recommends Web Services to a user based on the user’s history. The system measures the similarity between the user’s functional interests and web services and based on the similarity in the functional and non-
functional characteristics Active Web Service Recommendation System, ranks the services so that a list is generated which has top recommendations for the user.

Z. Zheng et al. [15] present a collaborative filtering approach for predicting QoS values of Web Services. It proposes a prototype called Web Service Recommendation (WSRec) which makes use of user-collaborative mechanism for collecting Web Services QoS information from different users, and based on the collected QoS information, predictions are made using the collaborative filtering approach. WSRec is a centralized server which consists of web service QoS data for various Web Services contributed by service users and makes recommendations for the user requesting a web service.

J.E. Haddad et al. [16] address the issue of recommending Web Services by considering into account transactional properties like compensable or not, QoS characteristics, and also the functional requirements of Web Services according to the requirements of the user. The web service composition can be viewed as a three stage process. In the first step, the user submits a query that he/she wants a composite web service to satisfy. In the second step, Web Services that satisfy the user requirements will be displayed to the user and the user can select from those results or they could be automatically decided by the system. The third step is executing the selected WS component. This paper has focused on designing a composite web service by designing an algorithm which integrates QoS and transactional properties that will ensure proper execution. In this, mainly five QoS criteria (execution price, execution duration, reputation, successful execution rate, and availability) have been used and a local QoS-driven service selection related to these criteria has been chosen.

In this paper, risk notion has been calculated for each of the scenarios or based on the user preferences, if some user prefers minimum price then it calculates the risk for a particular web
service and recommends those web service which has potentially low risk. So predictions are
given based on analysis of risk to the user.

L. Shao et al. [17] propose that non-functional properties such as quality of service (QoS),
should be taken into consideration when making recommendations to the customers. But there
are a lot of Web Services that can be found on the internet for which we do not have any idea
about its QoS factors, for such Web Services for which the user does not have any idea about,
predictions are made on the quality of such unknown Web Services. This paper makes use of the
concept of similarity mining through collaborative filtering for making predictions to the users
from other consumer experiences.
2. NARRATIVE

2.1 SYSTEM OVERVIEW

With the increase in number of Web Services it has become quite difficult for a developer to find reliable services on the internet, hence we are trying to implement a recommender system for recommending these Web services to the user. E-commerce sites like Amazon and EBay are some examples of recommender system which provide recommendations to the user based on history of their search and also product history. Facebook, Twitter, Netflix and Google Developers are some examples of recommender sites. These applications have incorporated a key feature of encouraging users to share their knowledge and experience, which will help in improving the accuracy of the recommender engine results. Our system employs this idea of user feedback and provides a platform to the users for sharing observed QoS values based on which the system generates personalized recommendation to the user. The more number of QoS records the more accurate the recommendation results.

2.2 System Features

The primary key for generating good recommendation results is the user feedback, that is the rating provided by the users. Hence user and rating are the two focal points of this entire project prior to the algorithm. Therefore, we define few terms related to the entire system flow of this project.
- **User**: One who makes use of our system, gives feedback and for whom the system suggests Web services.

- **Web services**: It is the item in our recommender system which provides a services to the user.

- **Feedback**: Rating is given to the web service by the user based on the user’s experience with that service.

- **Profile**: Each user has a profile which shows the ratings he/she has given to Web services, the search history of the user, and the services he is following.

- **User-Item Matrix**: This matrix is an excel sheet called Userdataset.csv which contains user id, web service id and rating given to that particular web service by various users.

- **Training user**: They are registered users of our system who have shared feedback on QoS of the Web services.

- **Active user**: If a training user requests web service recommendations then he/she becomes an active user.

The Recommender System contains the dataset file which has a set profiles of various users which contains rating data and which the recommender system process this data to give recommendations to the users.

The Recommender System has the following approach:

1. Learn the **“likes”** and **“dislikes”** or feedback from the user.

2. Compare the user’s preferences to other users.

3. Recommend those Web services which are similar to the ones a user has preferred.
Recommendation is done on the basis of fetching the user’s preferences. This kind of information is important for generating personalized recommendations to the user. Most recommender system learn about a user by using explicit feedback and keep improving the user recommendations.

2.3 Problem Statement

Different from the existing methods, which suffer from low prediction accuracy, we implement an effective CF algorithm for web service recommendation with the consideration of the region factor. We implement a location-aware QoS based Web services recommendation approach, in which we gain the QoS information and give personalized results to the user’s. We use the process of filtering the results obtained from collaborative filtering (CF) technique based on the user’s location information which significantly improves the recommendation accuracy by predicting and recommending potential favorite items for a user.

2.4 Proposed System

In the proposed system, we use the concept of grouping users based on the location of the users which we fetch from the IP addresses and also sort the web services data based on the ratings, i.e., we arrange our data according to the ratings in a descending order through SQL querying and we pass the resulted grouped data for filtering using User based Collaborative Filtering technique.
The algorithm predicts what the user might prefer and recommends those web services. We fetch this information and filter the recommendations based on the location which is done by fetching the active user’s location and output only those web services which are matching to the user’s location or the country, generating personalized recommendations to the user. Recommendations are made by predicting the QoS values for the active user using Cosine Similarity metric which measures the similarity between the active user and the training users.

2.5 Purpose

This recommender system aims at making the recommendations efficient to the user, by giving recommendations to users based on location and QoS feedback, as studies have shown that users in a particular region experience difference in QoS for the same service accessed from a different region [1].

2.6 Software Requirements Specification

The software requirements specifications are mentioned below:

This section describes the hardware and software requirements specific to the project

2.6.1 System Requirements

Below are the specifications of software and hardware needed for the execution of the project.

The software used and needed for the demonstration of the project are:

- Operating System: Any Operating System
- Language: JAVA for implementing the Algorithms, Servlets and JSP
• User Interface : HTML5
• Database : H2
• Server : Apache Tomcatv8.0

2.6.2 Hardware Requirements

• The hardware requirements for the project are:
  • Processor : Pentium IV
  • RAM : 500 MB
  • HARDDISK : 40 GB

2.7 Functional Requirements

The Functional Requirements of the project are as follows:

i) Grouping the user data and Web Services data

The users of our system are grouped based on the location information and the Web Services data are grouped based on the QoS similarity by fetching information from the database.

ii) Collaborative Filtering

In this, collaborative filtering takes user’s preferences and returns predicted preferences for other users. We build a user based recommender system where we consider the following.

User Similarity:

User Similarity takes the feedback data or rating data into account and uses Cosine Similarity for calculating the similarity between the users.

Neighborhood Selection:

User Neighborhood considers a number of nearest neighbors to the active user.
Prediction by the Recommender:

After the similarities are computed to the active user, predicted rating for unrated Web Services is computed and recommendations are given.

2.8 Non-functional Requirements

The Non-functional requirements are:

Performance

- Effective usage of memory
- Efficient use of the processor to improve the robustness of the process
- Faster execution of operations with quick response time

Reliability

Software delivers services according to the specifications consistently. ‘Consistency’ is key to reliability. It goes without saying that it is one of the most important attributes.

Availability

Time bound service, specific delivery is the necessary criteria for effective functioning of a software which cannot be done away with.

Security

Defending the software from intrusion and unauthorized use is central to enhancing security.
Maintainability

For a project to be successful over time it has to be flexible, extensible and scalable. Maintainability thus is an important requirement and our project satisfies it.

Portability

Our software is platform independent i.e., it can run on any different platforms with ease. Hence it also helps in cost reduction.

Figure 2.1. Use Case Diagram
Table 2.2. Use Cases and their Functions

<table>
<thead>
<tr>
<th>USECASE</th>
<th>FUNCTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Feedback</td>
<td>Give rating to the Web Services</td>
</tr>
<tr>
<td>Search for Web Services</td>
<td>To search for Web Services</td>
</tr>
<tr>
<td>Give Recommendation</td>
<td>Generating recommendations to the user</td>
</tr>
</tbody>
</table>

3. Design

3.1 Detailed Design

Documentation is the culmination of any engineering activity and so is the case with software design. It is essential for the design document to truly represent the complete design so that the manufacturing team can proceed to build the product. Design is, therefore, vital as all phases of development cycle are part of the design process.

The human-computer interaction is facilitated by menu selection. The system is, therefore, totally menu driven. Our menu structure is designed in such a way that it provides fast and easy access to a large number of items making it easily navigable.

The only thing that matters to the user is the output. Hence it is essential for an efficient, well organized output design to enhance the association with the user. The user is facilitated by making it more readable and organized by providing help menus and error messages wherever necessary.

25
The user logs in to the system by entering the login id and password. After the user logs in, he/she is directed to the home page and the user can select from one of the four links, which are home, recommendations, search and give feedback to the Web Services. Figure 4.1 represents the block diagram for the system.
Figure 3.1. Architecture of Recommender System

Figure 3.1 represents the architecture of Recommender System which describes how the system works. When user log on to a website which is offering recommendations, they can either search for services or they can give feedback to the service they have used. When the user makes a request for recommendation. The query will result in the recommendation algorithm fetching feedback data from the database and performs calculations and predicts which services the user will like and the predicted result is given as output to the user.

3.1.2. Flow chart
3.2 Architecture of the Proposed System
The architecture of the proposed system is shown in Figure 3.2 which shows the system flow, how it is achieving the implementation of the functionalities in the project. When the user logs in to the system, he becomes an active user if he is asking for recommendations. The active user’s location is found out from the IP address. The IP address provides information about a user’s country, city etc. The system has a dataset which contains feedback data from various users. The dataset which contains feedback data and user data is passed for collaborative filtering. QoS predictions are made from these aggregations, through which recommendations are generated for the active user. For example, there are four users U1, U2, U3, U4 in our system and U1 is requesting for recommendations.
First, we group the users based on their countries. In the table, we can see that U1 and U2 belong to USA and U3 and U4 belong to INDIA. So we have grouped users based on their locations which is shown in the table 3.2.

### Table 3.1. User Location Table

<table>
<thead>
<tr>
<th>User</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>USA</td>
</tr>
<tr>
<td>U2</td>
<td>INDIA</td>
</tr>
<tr>
<td>U3</td>
<td>INDIA</td>
</tr>
<tr>
<td>U4</td>
<td>USA</td>
</tr>
</tbody>
</table>

After grouping the users, we check users’ feedback to the web services as represented in Table 3.2. We pass this data for collaborative filtering where the similarity metric is calculated between the active user and the training users. Here the active user is U1 and the training users are U2, U3, and U4. Using Cosine similarity, the similarity is calculated and the potential neighbors to U1 are identified. From the table 3.3, we can see that U3 and U4 do not fall in the neighborhood of U1. So based on these calculations the system recommends service S3 to U1 as U2 has given rating 3 for S3 which is good and U2 has not provided any rating for S4. Hence S3 is recommended for U1.

### Table 3.2. Sorted User Location Table

<table>
<thead>
<tr>
<th>Users</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>U3,U4</td>
<td>INDIA</td>
</tr>
<tr>
<td>U1,U2</td>
<td>USA</td>
</tr>
</tbody>
</table>
3.3 Sequence Diagram

A sequence diagram is an interaction diagram that shows how the processes operate with one another and in what order. Hence it models the flow of logic within the system in a visual manner. They are used for both analysis and design purposes and also focus on identifying behavior within the system, hence, aptly called behavioral diagrams. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.
3.4 Algorithm

The algorithm used in the project is based on Cosine Similarity metric.

In this algorithm, we compute the similarity between the ratings $x, y$ of two users where $x$ is the ratings of the web services given by the active user or the current user and $y$ is the ratings given by training user for web services. Cosine Similarity is used for computing the similarity values. There are several ways of calculating this similarity. For example, Pearson Correlation can also be used. Typically, user based collaborative filtering has used Pearson correlation to compare users. Early work tried Spearman correlation and (raw) cosine similarity, but found Pearson Correlation to work better, but there are two issues with computing similarity between users using Pearson correlation. They are, one is what to consider for those items that one user has given rating for but the other user has not. To overcome this, we consider only those ratings for the services both the users have given rating for, and thus maintain consistency in this approach.
The other issue is the fact that when two users have less number of items in common, their similarity measure will be high, which is not a good way to show they are perfectly similar. The typical way to deal with this is significance weighting; multiply the similarity, decreasing similarity linearly until the users have at least 50 rated items in common. This improves the performance of recommenders using Pearson correlation. Cosine similarity can be computed exactly like Pearson correlation (considering only items in common). This does not have the self-damping benefits of considering all items [18].

The correlation used in the project "centers" its data, shifts the user's preference values so that each of their means is 0. This is necessary to achieve expected behavior on all data sets. This correlation implementation is equivalent to the cosine similarity since the data it receives is assumed to be centered where mean is 0. The correlation may be interpreted as the cosine of the angle between the two vectors defined by the users' preference values.

Given two vectors of attributes, \(A\) and \(B\), the cosine similarity, \(\cos(\theta)\), is represented using a dot product and magnitude as

\[
similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \times \sqrt{\sum_{i=1}^{n} (B_i)^2}}
\]

The numerator is defined as the sum of the product of ratings given to those items by both the active user and the training user. The denominator is defined as the sum of the squares of ratings given by active user to the items multiplied by sum of the squares of ratings given by training user to the items. The similarity value lies between 0-1, a non-negative value, where 0 indicates
no correlation and 1 indicates positive correlation. Let us consider the following table where we have ratings given by four users on five services.

**Table 3.4. User Service Table**

<table>
<thead>
<tr>
<th></th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>S4</th>
<th>S5</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>3</td>
<td>4</td>
<td>?</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>U2</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>U3</td>
<td>2</td>
<td>1</td>
<td>5</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>U4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>

We calculate the similarities between U1 and {U2, U3, U4}.

\[
\text{Sim} (U1, U2) = \frac{32}{\sqrt{29}} \ast \sqrt{38} = 0.96
\]

\[
\text{Sim} (U1, U3) = \frac{16}{\sqrt{29}} \ast \sqrt{14} = 0.79
\]

\[
\text{Sim} (U1, U4) = \frac{10}{\sqrt{25}} \ast \sqrt{5} = 0.53
\]

From the above similarity values we can see that similarity between U1 and U2 is stronger compared to other users.

After the similarity measures are calculated between the active user and the training users, we use the neighborhood approach for identifying which users are lying in the neighborhood of the active user. We set a threshold value between 0 - 1 usually 0.5, if the similarity value is less than threshold value then that user is not in the neighborhood of the active user and if greater than threshold value, the user falls in the neighborhood of the active user. From these calculations, the
algorithm predicts which services the active user might prefer and give recommendations accordingly. For the example described, service S3 is recommended for U1.

4. IMPLEMENTATION

4.1 Java Technology

JSP technology provides a very simplified and fast way to create a dynamic web content. It dynamically generates these web pages based on HTML, XML and others. It allows the developers a great flexibility and easy maintainability, and provides information rich, dynamic pages. These are of a great use for the development of web based applications which are platform independent [19].

To deploy and execute the JSP, a compatible web server with servlet container is required. JSP may be viewed as a high level abstraction of Java Servlets which are translated during the runtime; each servlet is cached and re-used until the original JSP is modified. It can be described as using the (Model View Controller) MVC architecture with Java Beans and Servlets being used as model and controller respectively. The compiled pages, as well as any dependent Java libraries, use the Java bytecode. These must be executed within a Java Virtual Machine (JVM) that integrates with the server's host Operating System to provide an abstract platform-neutral environment [20].

5.2 H2 Database

H2 is a lightweight open source java database which can also be run on client server mode. This database supports API's such as SQL and JDBC. We know indexing is very important as it becomes vital when dataset increases in size. Hence for memory optimization, it is possible to
create both in-memory tables (hash table, tree) and disk-based tables (b tree) in this. Like in other databases, data manipulation operations are transactional. Strong security is one of its key features as it uses SHA-256 for user password authentication. As a result user passwords are never transmitted over the network. The fact that it keeps up with the pace and is supported by open source community makes it reliable enough to be used in many open source and commercial products.

LIST OF TABLES USED IN THE SYSTEM:

- **T_Feedback (service_id, user_id, question, answer, lst_updt_ts)**: This table has four fields service_id, user_id, question and answer and it stores the feedback (ratings) data given by the users of the system.

- **T_Location (city, country, user_id, lst_upd_ts)**: This table has four fields city, country, user_id and last update timestamp. This table stores the location of the users based on their IP address.

- **T_Search (search_txt, user_id, lst_upd_ts)**: This table has three fields, search_txt, user_id and last update time stamp. This table stores the keywords entered in the search space by the users in a session.

- **m_user (name, username, password, interests, location, lst_updt_ts)**: This table has six fields, name, username, password, interests, location and last update timestamp. It stores the details of all the users of our system.

- **m_service (name, content, country)**: This table has three fields, name of the web service, the web service link and the country of the web service.

- **m_countries (country_code, country_name)**: This table has three fields, list of all the countries in general and the id for each country.
4.3 Modules of the System

There are mainly four modules in the system

1. User feedback recommendation
2. History based recommendation
3. Location based recommendation
4. Interest based recommendation

4.3.1 Implementing recommendations based on User Feedback

Step 1: Grouping users based on the location

The country and city of the user are gathered from the user’s IP address and during communication store the country of the user in a table T_location. Then group users with the same location and store the result in a file, and also check if the user’s registered country and the country resulted by tracking the IP address are matching, if not, then consider based on the IP address of the user’s communication.
The following code is used to group users based on their location:

```java
// To Cluster User based on location (Country)

String filePath = request.getRealPath("/WEB-INF/userDataset.csv");

Map<Integer, String> locationMap = Utility.toClusterUser(filePath);

toClusterUser :

if (rs1.next()) {
    int countryId = rs1.getInt("id");
    users = locationMap.get(countryId); //for one country we are appending user
    if (null != users)
        users = users + "," + user;
    locationMap.put(countryId, users); //key-country id, value-user id
}

else {
    locationMap.put(countryId, String.valueOf(user));
}
Figure 4.1 shows the results before grouping and Figure 4.2 represents userDataset which contains the grouped information of the users i.e., which user id maps with country id. In our table Country id 1 is for USA, so users 65, 97, 66, 98, 129, 33, 34 and 1 belong to USA.

![Excel Spreadsheet](image)

Figure 4.2. Sorted Entires

**Step 2: Grouping Web Services based on their QoS**

Web Services are grouped based on the QoS feedback from various users. This information is fetched from the T_feedback table which has the service id of the web service, user id, question and the rating for that question. This data is ordered in ascending order based on the feedback value. The user id, the web service id and their corresponding rating are organized and these values are appended to serviceDataset file.

Figure 4.3 represents data from T_feedback before grouping the information:
Figure 4.3. Entries of user feedback in the T_Feedback table

The following code is used to create serviceDataset file:

```java
try {
    ResultSet rs = CommonDAO.retriveRecords(feedbackQry);
    FileWriter = new FileWriter(filePath);
    /*
    * fileWriter.append("USER_ID"); fileWriter.append(",");
    * fileWriter.append("SERVICE_ID"); fileWriter.append(",");
    * fileWriter.append("QUESTION"); fileWriter.append("",");
    * fileWriter.append("ANSWER"); fileWriter.append("\n");
    */
    while (rs.next()) {
        FeedbackVo feedbackVo = new FeedbackVo();
        feedbackVo.setServiceId(rs.getInt("SERVICE_ID"));
        feedbackVo.setUserId(rs.getInt("USER_ID"));
        feedbackVo.setQuestion(rs.getString("QUESTION"));
        feedbackVo.setAnswer(rs.getInt("ANSWER"));
        feedbackSet.add(feedbackVo);
        fileWriter.append(String.valueOf(feedbackVo.getUserID()));
        fileWriter.append(",");
        fileWriter.append(String.valueOf(feedbackVo.getServiceId()));
        fileWriter.append(",");
        fileWriter.append(String.valueOf(feedbackVo.getAnswer()));
        fileWriter.append("\n");
    }
}
```
Figure 4.4 shows the list of user id, web service id and the ratings given by the corresponding users after grouping the data.

Figure 4.4. Sorted Entries of Feedback

Step 3: Applying collaborative filtering technique on the serviceDataset file

The service dataset file is passed as an argument to the data model which is the interface to the information about user preferences. FileDataModel constructor receives the File instance containing the preferences data.

Datamodel model = new FileDataModel (new File(filepath1));

Step 4: Correlation between users

Here similarity metric between the current and the other users are calculated by fetching data from the service dataset based on Pearson Correlation similarity (This correlation
implementation is equivalent to the cosine similarity since the data it receives is assumed to be centered where mean is 0 and becomes equivalent to Cosine Similarity), which considers two users ratings \(x, y\) at a time and calculates the similarity between them. Here \(x\) represents feedback given by active user and \(y\) represents ratings given by training users and the similarity value ranges from 0 to 1.

The code uses the FileDataModel instance to create an instance of the `org.apache.mahout.cf.taste.impl.similarity.PearsonCorrelationSimilarity` class. This class provides an implementation of the Pearson correlation.

```java
UserSimilarity similarity = new PearsonCorrelationSimilarity(model);
```

**Step 5: Finding Neighborhood**

In user based recommender system, selecting a neighborhood of similar users near a given user is essential for providing recommendations. The neighbors are selected based on a threshold value which is passed as an argument, all the neighbors whose similarity value have exceeded the threshold value are considered to be a neighborhood.

```java
UserNeighborhood neighborhood = new ThresholdUserNeighborhood(0.5, similarity, model);
//Here 0.5 is the threshold value and the similarity contains the value computed from pearson correlation
```

**Step 6: Generating the Recommendation List**

We get a list of recommendations and give top 5 Web Services to each user

```java
UserBasedRecommender recommender = new GenericUserBasedRecommender(model, neighborhood, similarity);
List<RecommendedItem> recommendations = recommender.recommend(userId, 5);
for (RecommendedItem recommendation : recommendations) {
```
FeedbackVo feedbackVo = new FeedbackVo();

feedbackVo.setServiceId(Integer.parseInt(String.valueOf(recommendation.getItemId())));

recommendService.add(feedbackVo.getServiceId());

Step 7: Filtering the services based on the location of the active user

After getting the list of recommended services, we need to filter the services for location. Only those services are given which are a match to the current user’s location. The following code segment represents the way in which the recommended services are filtered based on the location of the current user.

The following code represents filtering the resulted web services based on location of the active user:

```java
//To fetch current user country Name
String country = null;
ResultSet rs2 = CommonDAO.retrieveRecords(countryQry + countryId);
if (rs2.next()) {
    country = rs2.getString("country_name");
}

Set<String> contentStr = resultList.keySet();
//To find Services, not related to current location
List<String> removeContent = new ArrayList<String>();
for (String content : contentStr) {
    ResultSet rs1 = CommonDAO.retrieveRecords(serviceContentQry + content + "'");
    if (rs1.next()) {
        if (!rs1.getString("country").equals(country)) { //comparing service country with user country
            removeContent.add(content);
        }
    }
}
//remove service not related to current location
for (String str : removeContent) {
    resultList.remove(str);
}
```

4.3.2 Implementing History Based Recommendation Search Option

We get the user id details who is logged in to the system and retrieve the records of that user from T_search table. We match the search keywords of the user which we have stored in the
T_search table with the m_service table which has the list of Web Services and display the results

Figure 4.5 is the list of keywords that are entered by different users of the system which are stored in table T_search.

![Figure 4.5. Entries of Search Keyword in T_Search table](image)

The following code represents the logic for history based recommendations

```java
try {
    User Search history Based Recommendation.
    HttpSession session = request.getSession(true);

    String qry = "Select * from t_search where user_id=" + userId;

    while (rs.next()) {
        searchSet.add(rs.getString("search_txt"));
    }

    Set<String> resultSet = new HashSet<String>();

    for (String str : searchSet) {
```
String qry1 = "Select * from m_service where upper(content) like "’%" + str.toUpperCase() + "’’;
    //System.out.println(qry1);

ResultSet rs1 = CommonDAO.retriveRecords(qry1);
    while (rs1.next()) {
        resultSet.add(rs1.getString("content"));
    }
}

4.3.3 Implementing Search Option

When a user searches for any web service, he/she enters a keyword which is stored in the
database in table T_search. We match the keyword with the web service from the table
m_service which has the collection of Web Services.

The following code represents how the search option is being implemented:

String srchTxt = request.getParameter("searchTxt");
    // System.out.println(srchTxt);

String qry = "select * from m_service WHERE upper(content) like "’%" + srchTxt.toUpperCase() + "’’;

ResultSet rs = CommonDAO.retriveRecords(qry);

4.3.4 Implementing Location Based Recommendation

1. Get the user id details of the active user
2. Get the country location from the IP address of the active user
3. Fetching the records from m_service table which belong to the location of the user
country

Figure 4.6 represents the IP address location information of different users in the system.
The following code is for fetching those web services which map with the current location of the user:

```java
for (int str : locationSet) {
    ResultSet rs2 = CommonDAO.retriveRecords(countryQry + str);
    String countryName = null;
    if (rs2.next()) {
        countryName = rs2.getString("country_name");
        ResultSet rs1 = CommonDAO.retriveRecords(serviceQry + countryName.toUpperCase() + "%'");
        while (rs1.next()) {
            resultSet.add(rs1.getString("content"));
        }
    }
}
```

### 4.3.5 Implementing Interest Based Recommendation

1. Fetching the record of the active user from m_user table

2. Identify the interests of the user

3. Map these interests to the Web Services list in the m_service table
As shown in Figure 4.7, we can see the records of m_user table where we can see the interests of the users.

![User table showing interests](image)

**Figure 4.7. User table showing interests**

The following code is used for fetching those web services which match with the interests of the active user:

```java
for (String str : strArray) {
    str = str.toUpperCase();
    String qry1 = "select * from m_service where upper(content) like '" + str + "'";
    ResultSet rs1 = CommonDAO.retriveRecords(qry1);
    while (rs1.next()) {
        resultSet.add(rs1.getString("content"));
    }
}
```
5. TESTING AND EVALUATION

Testing is a method of assessing the functionality of a software program. This assessment can be done while running the program called dynamic testing and while examining the program's code and documentation called static testing. Testing is needed to make the product error free and avoid inconsistency between functional/non-functional requirements and actual requirements.

Each smallest testable part for an application like classes, procedures, interfaces, functions etc. is called a unit. Unit testing is a method by which individual units of source code are tested to determine if they are fit for use. This is done to make sure that code meets its design and requirements and behaves as expected. It ensures that each individual part of the program is working correctly.

System testing is testing conducted on complete, integrated system to examine the compliance of system with its specific requirements. It is most often the final test to verify that the system to be delivered meets the specification and purpose. It falls within the scope of black box testing requiring no knowledge of the inner design of the code or logic. It investigates both functional and non-functional.

5.1 Data Set

A data set is a collection of data, usually presented as in table 5.2. In this project the data is taken from the dataset provided by WS-DREAM site. The dataset contains Real-World QoS evaluation results from 339 users on 5825 Web Services.
The users can download the information for free from this site and use it for testing. This raw data is used for further computations. Table 5.2 shows the example list of web service from the data set provided by the WS-DREAM website.

![Figure 5.1. DataSet Website](image)

**Table 5.2. List of Web Services**

<table>
<thead>
<tr>
<th>WSID</th>
<th>WSDL Address</th>
<th>Provider Name</th>
<th>Country Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.</td>
<td><a href="http://arcweb.esri.com/services/v1/RouteFinder.wsdl">http://arcweb.esri.com/services/v1/RouteFinder.wsdl</a></td>
<td>esri.com</td>
<td>United States</td>
</tr>
</tbody>
</table>
5.2 Test Cases

Test Case 1: No Results Found Test Case for Search option

This test case is for testing when the keyword in Figure 5.3 doesn’t match with the database then the No Results are found text is displayed.

Figure 5.3. No Results Found for Search

Test Case 2: Checking Search Option functionality

This test case tests if the search option is working properly. The search option is in the web services page which is used to search for a particular web service. Results are shown in Figure 5.4.
Test Case 3: Validations for Feedback form

This test case in Figure 5.5(a), 5.5(b) is used for testing the options of feedback form, if the user tries to submit without giving rating to the service, then it gives a prompt to select one of the options of bad, good or very good.
Figure 5.5. (a) Feedback Form Validation

Figure 5.5. (b) Feedback Form Validation
Test Case 4: Testing for Functionality of Feedback form

This test case is used for testing if the feedback given by a user are updated properly. The user can give feedback to the Web Services. We can see in the user profile that the feedback given by the user has been updated as seen in Figure 5.6 (b).

Figure 5.6. (a) Submit Feedback Form
Test Case 5: Testing for accessing the link

When the user clicks on a link he should be able to access that particular web service. Results are shown in Figures 5.7 and 5.8.
Figure 5.7. Testing the link access

Figure 5.8. Web Service Example
5.3 EVALUATION

Generating Recommendations: Giving recommendations is the main motive of this project by using collaborative filtering technique as shown in Figure 5.9, we can see two user profiles, Hrudaya and Sachin are getting different recommendations based on their similarity and their location information.

Result 1: Comparing User 1’s and User 2’s recommendations based on feedback

![Figure 5.9. (a) Comparing the Feedback Output](image-url)
User 2:

![Image of the interface showing user feedback based recommendations]

**Figure 5.9. (b) Comparing the Feedback Output**

**Result 2: History Based Recommendations for a user**

As shown in Figure 5.10, the recommendations are given to the user based on the search history of the user. The screenshots show that when the user just presses the search option, the entire list of Web Services in their history are displayed.
As shown in Figures 5.11 and 5.12, the user, named Hrudaya, has selected interest in movies and when she selects interest based recommendations, it gives the results related to the interest of the user.
Figure 5.11. Output for Interest Based Recommendation

Figure 5.12. Interests of the User from the User Table
Result 4: The profile of the user

In Figure 5.13, we can see the feedback given by a user to the services, search history of the user and the services he is following.

Figure 5.13. Output for User Profile
7. CONCLUSION AND FUTURE WORK

With the increase in the number of web services, developers are facing difficulties in finding appropriate services which fit their requirements. In order to make the developers work easy, we have implemented a recommender system. In this project, we are trying to give recommendations to users based on historical QoS (feedback) records and location information of the user, through which the user can select a well suited service. The existing approaches lack location based recommendations and also do not provide a platform to the users for giving ratings for a web service. We have overcome this in our project. Our system has different kinds of recommendations where the user can select recommendations based on categories like Personalized, History, Location and Interest.

Future work includes improving the Web service recommendation in terms of clustering method, improving the security level, improving the user interaction with our system and making the recommendations more personalized.
[1] Xi Chen ; Shenzhen Res. Inst., Chinese Univ. of Hong Kong, Hong Kong, China ; Zibin Zheng ; Qi Yu ; Lyu, M.R. Web Service Recommendation via Exploiting Location and QoS Information.

[2] Introduction to Web Service:
http://javabrains.koushik.org/courses/javaejaxws/lessons/Introduction-to-Web-Services


