CROSS-DOMAIN RECOMMENDER SYSTEM

**Bachelor of Technology**

**In**

**Computer Science & Engineering**

**Under the supervision of**

***Mrs. Koyel Datta Gupta***

**By**

ADITYA KUMAR (13215002712)

BHANU KUMAR (10015002712)

ROHIT BHARDWAJ (06615002712)

VISHESH BHAT(16215002712)

**To**



**Maharaja Surajmal Institute of Technology**

Affiliated to Guru Gobind Singh Indraprastha University

C-4, Janak Puri, New Delhi-58

**CERTIFICATE**

This is to certify that the project work done on *“CROSS-DOMAIN RECOMMENDER SYSTEM*” submitted to Maharaja Surajmal Institute of Technology, Janakpuri Delhi by “*ADITYA KUMAR, BHANU KUMAR, ROHIT BHARDWAJ, VISHESH BHAT*” In partial fulfillment of the requirement for the award of degree of Bachelor of Technology, is a bonafide work carried out by him/her under my supervision and guidance. This project work is the original one and has not submitted anywhere else for any other degree.

*Mrs. Koyel Datta Gupta Mrs. Kavita Sheoran*

(Project Guide) (HOD, CSE MSIT)

**ACKNOWLEDGEMENT**

Team effort together with precious words of encouragement and guidance makes daunting tasks achievable. It is a pleasure to acknowledge the direct and implied help we have received at various stages in the task of developing the project. It would not have been possible to develop such a project without the furtherance on part of numerous individuals. We find it impossible to express our thanks to each one of them in words, for it seems too trivial when compare to the profound encouragement that they extended to us.

We are grateful to ***Mrs. Kavita Sheoran, HOD(CSE);*** for having given us opportunity to do this project, which was of great interest to us.

Our sincere thanks to ***Mrs. Koyel Datta Gupta*** for believing in us and providing motivation all through. Without her guidance this project would not be such a success.

At last we thank the almighty, who had given the strength to complete this project on time.

Finally we would like to thank our parents, all friends, and well-wishers for their valuable help and encouragement throughout the project.

*ADITYA KUMAR*

*(13215002712)*

*BHANU KUMAR*

*(10015002712)*

*ROHIT BHARDWAJ*

*(06615002712)*

*VISHESH BHAT*

*(16215002712)*

TABLE OF CONTENTS

[ABSTRACT 1](#_Toc436081798)

[INTRODUCTION 2](#_Toc436081799)

[INTRODUCTION TO PROJECT 4](#_Toc436081800)

[CROSS-DOMAIN RECOMMENDATION 5](#_Toc436081801)

[ARCHITECTURE 10](#_Toc436081802)

[SYSTEM REQUIREMENTS 12](#_Toc436081803)

[SYSTEM DESIGN 13](#_Toc436081804)

[TESTING 25](#_Toc436081805)

[CONCLUSION 29](#_Toc436081806)

[REFERENCES 30](#_Toc436081807)

# ABSTRACT

The ongoing rapid expansion of the Internet greatly increases the necessity of effective recommender systems for filtering the abundant information. Extensive research for recommender systems is conducted by a broad range of communities including social and computer scientists, physicists, and interdisciplinary researchers. Despite substantial theoretical and practical achievements, unification and comparison of different approaches are lacking, which impedes further advances. We compare and evaluate certain algorithms and examine their roles in creating a Cross Domain Recommender System. In addition to algorithms, physical aspects are described to illustrate macroscopic behavior of recommender systems.

The vast amount of data available on the Internet has led to the development of recommendation systems. This project proposes the use of soft computing techniques to develop recommendation systems. It addresses the limitations of current algorithms used to implement recommendation systems, evaluation of experimental results, and conclusion. This report provides a detailed summary of the project “Cross Domain Recommender System.” The report includes a description of the topic, system architecture, and provides a detailed description of the work done till point. Included in the report are the detailed descriptions of the work done and tools used.

**CHAPTER-1**

# INTRODUCTION

The huge and ever increasing amount, complexity and heterogeneity of available digital information overwhelm the human processing capabilities in a wide array of information seeking and e-commerce tasks. To cope with information overload recommender systems have been introduced to filter those items –Web pages, images, videos, audio– that are of low relevance or utility for the user, and present only a small selection better suiting the user’s tastes, interests, and priorities. Often these suggestions are presented while the user is browsing an information service, and without requiring her to launch explicit search queries, as is usually done in information retrieval systems. Recommender systems are an active research field and are being used successfully in numerous e-commerce and leisure Web sites such as Amazon, Netflix, YouTube, iTunes, and Lasf.fm. The vast majority of these systems offer their recommendations only for items belonging to a single domain. Hence, for example, Netflix suggests movies and TV series, and Last.fm makes personalized recommendations of music artists and compositions. In both cases the recommendations are computed using user feedback (ratings) about items in the target domain. In e-commerce sites like Amazon, nonetheless, it would be useful to exploit the user’s evaluations about diverse types of items in order to generate a more general model of the user preferences. In fact, there could be dependencies and correlations between preferences in different domains and instead of treating each type of items (e.g. electronics and music) independently, user knowledge acquired in one domain could be transferred and exploited in several other domains. Moreover, although it is not the main goal of cross-domain recommendation, a system could offer joint, personalized recommendations of items in multiple domains, e.g. suggesting not only a particular movie, but also music CDs, books or videogames somehow related with that movie. Analogously, in a touristic application it would be valuable to suggest a cultural event to a customer who has booked a room in a recommended hotel, or in an e-learning system, to present a student with bibliographic references related to a video-lecture that has been recently recommended. Some systems already offer joint recommendations of items in different domains, but in general, in order to build a recommendation in one domain, they exploit only user preferences on that target domain. Cross-domain recommendation is thus an emerging research topic in the Recommender Systems area. One of the first studies on cross-domain recommendation was that presented by Winoto and Tang in. In that work, the authors identify three important issues to investigate: a) verifying the existence of global correlations of user preferences for items in different domains, b) designing models able to exploit user preferences on a source domain for predicting user preferences on a target domain, and c) developing evaluations appropriate for cross-domain recommendations. Winoto and Tang speculate that, although cross-domain recommendations may tend to be less precise than single-domain recommendations, the former will be more diverse, which may lead to a higher user satisfaction and engagement. Moreover, cross-domain recommendation techniques may have other advantages, such as addressing the cold-start problem and mitigating the sparsity problem. By identifying relations between items in two different domains, one could suggest a user with items in a novel, unexplored domain, simply exploiting her preferences for items in other known domains. Despite the above benefits and work, this research topic is quite new and still largely unexplored. Hence, to the best of our knowledge, in the literature an agreed definition of the cross-domain problem has not emerged so far and there is not a general classification of existing approaches. In fact, the issue of how to exploit information from various domains to provide recommendations has been addressed from distinct perspectives in diverse disciplines such as User Modelling, Information Retrieval , Knowledge Management, and Machine Learning]. In this paper we present a survey of the state of the art on cross-domain recommendation, analyzing how diverse domains can be related, and providing a taxonomy that lets characterizing, categorizing and comparing the revised work, and identifying interesting unexplored research issues. The reminder of the paper is structured as follows. We propose a formal statement of the cross-domain recommendation task, starting from a discussion about the notion of domain and the potential sources of relations between domains.

A Recommender system is a fully functional software system that applies at least one implementation to make recommendations. In addition, recommender systems feature several other components, such as a user interface, a corpus of recommendation candidates, and an operator that owns/runs the system.

Recommender systems have changed the way people find products, information, and even other people. They study patterns of behavior to know what someone will prefer from among a collection of things he has never experienced. The technology behind recommender systems has evolved over the past 20 years into a rich collection of tools that enable the practitioner or researcher to develop effective recommenders.

Recommendation systems use a number of different technologies.

We can classify these systems into two broad groups.

**• Content-based** systems examine properties of the items recommended. For instance, if a Netflix user has watched many cowboy movies, then recommend a movie classified in the database as having the “cowboy” genre.

**• Collaborative filtering** systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users. However, these technologies by themselves are not sufficient, and there are some new algorithms that have proven effective for recommendation systems.

# INTRODUCTION TO PROJECT

The project began with a theoretical phase where previous thesis projects about recommender systems were studied along with relevant theory. As one of the project requirements was building an application prototype, the initial phase also included a brief study of the sites available and their developer documentation and guides. The majority of time was spent on the prototyping activities, which were initiated early. During the design process, the user tests were planned and a literature search was initiated. This was followed by a literature study that was conducted in parallel with the implementation process. Some of the literature regarding how to evaluate system was used to revise the design of the user tests before finishing the implementation process. The user tests were conducted during a two week period during which the main findings of the literature study were summed up. After analysing the data and literature, suggestions were made relating to the recommendation process and possible future work.

Most recommender systems work on single domains, i.e., they recommend items related to the same domain where users have expressed ratings. However, the integration of different domains into one recommender system could allow users to span over different types of items. For instance, in our project users that have watched movies could like to be recommended with on-demand movies and related books. This report focuses on cross-domain collaborative recommender systems, whose aim is to suggest items related to multiple domains. We first formalize the cross-domain problem in order to provide a common framework for the classification and the evaluation of state-of-the-art algorithms. We later define a new class of cross-domain algorithms based on neighborhood collaborative filtering, either item-based or user-based. The main idea is to first model the classical similarity relationships as a direct graph and to later explore all possible paths connecting users or items in order to find new, cross-domain, relationships.

# 1.2 CROSS-DOMAIN RECOMMENDATION

Cross-domain recommendation Cross-domain recommendation has been addressed recently from different perspectives, and in diverse research areas. Based on such perspectives, in this section we provide a general definition of the cross-domain recommendation task, starting from a discussion about the notion of domain, and a description of the types of relations between domains.

**1.2.1 Notion of domain**

In the literature on cross-domain recommendation there is no a consensus on the notion of domain. Authors have utilized the term domain sometime for referring to types of items (e.g. movies vs. books) or in other cases to groups of similar items with common characteristics (e.g. movies vs. TV shows). This may be due to the lack of public datasets including users’ evaluations on diverse types of items, to be used in the evaluation of the proposed cross-domain recommendation approaches. In fact, several authors have considered artificial data splits to simulate different domains but using data coming from one, single domain, dataset. For instance, group movies based on their genres (e.g. drama, comedy, thriller), and consider these groups as different movie domains. We present a review of collaborative filtering approaches for cross-domain recommendation, distinguishing three types of domains: system domains, data domains, and temporal domains. System domains are the different datasets upon which the recommender systems are built, and in which some kind of transfer learning is performed. Data domains are the different representations of user preferences, which can be implicit (e.g. clicks, time) or explicit (e.g. ratings). Finally, temporal domains are subsets in which a dataset is split based on timestamps. We consider a notion of domain similar to Li’s definition of system domain, which is the one more frequently used in the literature. We define domain as a set of items that share certain characteristics that are exploited by a particular recommender system. As we shall explain, these characteristics can be manifold, e.g. content attributes, ratings, and tags.

**1.2.2 Statement of the cross-domain recommendation task**

To date the large majority of the proposed approaches to cross-domain recommendation deals with collaborative filtering (CF). Collaborative filtering strategies exploit user preferences (usually expressed as explicit ratings for items), and ignore any content- based description (attributes) of the items. This feature of CF represents a great advantage when the items belong to heterogeneous sources. In the following we propose a definition of the cross-domain recommendation task that is valid for both content based and collaborative filtering approaches. Without loss of generality we consider the task of cross-domain recommendation when only two domains are involved. Using the notation, let , be the sets of users and , be the sets of items with “characteristics” (user preferences and item attributes) in the domains and respectively. We define two cross-domain recommendation tasks:

 Exploit knowledge about users and items in the source domain for improving the quality of the recommendations for items in the target domain.

 Making joint recommendations for items belonging to different domains, i.e., suggesting items in to users in  .

In this context, as in single-domain recommendation, we assume that the cross-domain recommendation task involves personalization. We thus ignore those recommendation strategies, such as popularity-based, that do not take into account the target user’s preferences, and simply suggest items positively evaluated by a large number of users. Moreover, we also exclude those cross-type systems that are initially built with user evaluations about items of different types (e.g. movies and books). In these systems the users’ preferences already comprise the domains of interest, and thus any collaborative filtering strategy can be used to provide recommendations of items, which can be considered as belonging to a single domain. We assume a recommendation scenario in which user and item profiles are distributed in multiple systems (domains), and in which we have to establish a mechanism to link or transfer domain knowledge (e.g. content attribute mappings, semantic similarities, rating patterns) between such systems. In this context, classic nearest neighbour strategies are not valid since it is not possible to directly compute rating-based similarities between pairs of users/items.

**1.2.3Types of explicit relations between domains**

According to the type of overlap between domains, identify four situations in which a cross-domain recommendation task may be conducted: a) no-overlap,  ; b) user overlap,  ; c) item overlap,  ; and d) full overlap,  . In all these situations but the first –no-overlap–, we could obtain effective recommendations with a classic collaborative filtering strategy by considering all the user/item ratings belong to a single, common domain. Nonetheless, assuming a memory-based approach to CF, when the overlap is small, user and item similarities, and consequently generated recommendations, may tend to be inaccurate. To address the above mentioned no overlap situation, we have to develop approaches that find or build some type of explicit/implicit relations between domains, which would be used as semantic bridges connecting different domains in a recommender system. In the following we extend the ideas to deal with the domain overlap issue in a more flexible way. Instead of focusing only on users/items with rating information in the considered domains, we define (overlap) relations between domains through “characteristics” that are shared by the user/item profiles in the different domains. In a vector space model users can be represented as vectors in which each component is associated to certain characteristic, that is:



where is the set of characteristics used to describe the user preferences. Analogously, items can be represented as:



where is the set of item characteristics. In both cases, as we shall explain, the nature of these characteristics can be quite diverse, e.g. ratings, content attribute-value pairs, social tags, explicit semantic relations, and implicit latent factors.

Upon the above representation domains will be linked if or , i.e., if there are user or item characteristics shared by the two domains. The larger the number of shared characteristics, the more robust the relation between domains. In a real situation, due to the heterogeneity of domain representations, which may be provided by various systems, we will have to establish a number of functions mapping characteristics between domains, i.e. , and . For instance, in a movie and book recommender system, a mapping function could be applied to the genre*, f(comedy movie)=humor book* , or could identify a user registered in both systems, . We note that this mapping is a particular case of a more general approach in which both features are mapped to a new one. We do not enter into this for simplicity purposes.

User and item characteristics together with their relations depend on the implemented recommendation strategy. In the following we describe representative examples.

**1.2.4 Recommendation Technique:**

Our system is a content based recommendation system.Movie domain can be seen as a set of movies where each food has a set of reviews. Content-based approaches treat the recommendation problem as a search for related items. Given a rated movie, the algorithm constructs a search to find other related items with the similar reviews. If a user likes horror, for example the system might recommend other movies having horror like The Conjuring. However, this is the simplest logic behind content-based recommendation systems. In our system movies are defined by their important features and represented by vectors. Thus, feature weights are crucial in these vectors. Similarity is computed based on item attributes using appropriate distance measures. Content-based recommendation systems share in common describing the items that may be recommended, creating a profile of the user that describes the types of items the user likes, and comparing items to the user profile to determine what to recommend. My content-based recommendation system can be seen as a combination of three distinct parts; movie profiling, user profiling and recommending movies and books according to the previous feedback of the users. Moreover, movie domain can be seen as a set of features where each feature is actually a genre of that movie. Therefore, both item and user profiles are kept as vectors of features. As our system is a content based recommendation system, it tries to find best matches between the user profile and the movie profiles.

**1.3 Item Profile Representation**

In content based recommendation systems, every item is represented by a set of features or an attribute profile. A variety of distance measures between the feature vectors may be used to compute the similarity of two items. For example Euclidian or cosine similarity supposes that all the features have equal importance. However, human judgment of similarity between two items often gives different weights to different attributes. Moreover, document frequency is more commonplace to use for this purpose.



Here N denotes for the total number of movies in a collection, and dft is for the total number of movies that have the feature/genre “t”. Thus the idf of a rare term is high, whereas the idf of a frequent term is likely to be low. This method is called “inverse document frequency” and we have assigned the calculated similarity as weights of the features.

**1.4 Knowledge Acquisition Technique & User Profile Representation**

The type of the user profile derived by a content-based recommender depends on the learning method employed. Decision trees, neural nets, and vector-based representations can be all used. In this project we have used decision vector based representations constructed with the help of user ratings. At this step, our system uses explicit data collection. Specifically, after each recommendation, user can explicitly state whether the recommendation is satisfying or not. The next recommendations will be mostly based on this user review.

**CHAPTER-2**

# ARCHITECTURE

**2.1.1 The implementation environment**

We implemented my project with Matlab. The reasons that we choose this language is that it is simple, intuitive, easy to use, easy to control multiple users at the same time; has a easy integration platform, support for several services & legacy systems etc. We developed it on Linux Operating System.

**2.1.2 Information Retrieval**

Information retrieval (IR) is a data search technology which includes crawling, processing and indexing of content, and querying for content. Web crawling is the process by which we gather pages from the Web, in order to index them and support a search engine. The objective of crawling is to quickly and efficiently gather as many useful web pages as possible, together with the link structure that interconnects them. I have done web crawling to create my dataset. We have written some code to extract the movie information from the data set.. Keeping them in a text file, then we have extracted the books related to each main movie type. We store those books in a database table. Lastly, we extracted the movies with their ratings. The movies are also parsed into their ratings. Another database table is created for the ratings and their calculated weights.

**2.1.3 Item Profiling**

Our system first constructs item profiles from the information it collects from data set. It uses this information to build movie profiles through assigning feature weights to each feature in the movie domain. Weights are calculated with the method we previously explained.

**2.1.4 User Profiling**

After the registration to the system, a set of random movies are asked to the user in order to be rated from 0 to 5. If a movie has already been rated before, the new rating will be considered in the next sessions. We made such an estimation that if the rating is smaller than 3, the user does not liked that movie. Therefore, the ratings smaller than 3 are considered as negative ratings, the ratings equal to 3 are considered as neutral ratings, and the remaining (4, 5) are considered as positive ratings. Briefly, user profile vectors are constructed with the positively rated ingredients.

**2.1.5 Recommendation and User Feedback**

The recommendations are done by calculating the movies which are most similar to the profile of the user. However, only the movies which are positively rated are considered in this similarity process. Moreover, users can evaluate the recommendations by stating whether they liked the recommendations or not. We made estimation here. I assume that the user likes the recommendation according to the books in which that recommended movie can be compared. This part is the user feedback step of our recommendation system. After these evaluations, the next recommendations are built on both personally liked (positively rated) and the previously liked recommendations, which are affects the similarity score positively.

**2.2 Evaluation of The System**

We firstly implemented our system considering that all the features have the same weight. However we were not satisfied with the recommendations. We have read something about Inverse Document Frequency and used it; then the results changed dramatically. The second version of our project was without the evaluation of recommendations (without user review part). We have used the system with a few people. Because of the limited time, it was not clear to see its development in making good recommendations. We have added the recommendation evaluation module later. We have used it with a few people, too; but it is making better recommendations day by day.

# SYSTEM REQUIREMENTS

|  |  |
| --- | --- |
| PROCESSOR | : CORE i5 |
| CLOCK SPEED | : 2.2 GHz |
| SYSTEM BUS | : 64 Bit |
| RAM | : 6 GB |
| HDD | : 500 GB |
| MODEM | : 128 KBPS |
| OPERATING SYSTEM | : LINUX |

**CHAPTER-3**

# SYSTEM DESIGN

Design of software involves conceiving, planning out and specifying the externally observable characteristics of the software product. We have data design, architectural design and user interface design in the design process. These are explained in the following section. The goal of design process is to provide a blue print for implementation, testing and maintenance activities.

The primary activity during data design is to select logical representations of data objects identified during requirement analysis and software analysis. A data dictionary explicitly represents the relationships among data objects and constraints on the elements of the data structure. A data dictionary should be established and used to define both data and program design.

Design process is in between the analysis and implementation process. The following design diagrams (Data Flow Diagrams and E-R Diagrams) make it easy to understand and implement

The design process for software system has two levels.

1. System Design or Top Level Design.
2. Detailed Design or Logical Design.

**System Design or Top Level Design:**

In the system design the focus is on deciding which modules are needed for the system, the specification of these modules and how these modules should be interconnected.

**Detailed Design or Logical Design:**

In detailed design the interconnection of the modules or how the specifications of the modules can be satisfied is decided. Some properties for a software system design are

* Verifiability.
* Completeness.
* Consistency.
* Trace ability.
* Simplicity/Understandability.

**3.1** DATABASE DESIGN

The overall objective in the development of database technology has been to treat data as an organizational resource and as an integrated whole. DBMS allow data to be protected and organized separately from other resources. Database is an integrated collection of data. The most significant form of data as seen by the programmers is data as stored on the direct access storage devices. This is the difference between logical and physical data.

Database files are the key source of information into the system. It is the process of designing database files, which are the key source of information to the system. The files should be properly designed and planned for collection, accumulation, editing and retrieving the required information.

The organization of data in database aims to achieve three major objectives: -

* Data integration.
* Data integrity.
* Data independence.

The proposed system stores the information relevant for processing in the MS SQL SERVER database. This database contains tables, where each table corresponds to one particular type of information. Each piece of information in table is called a field or column. A table also contains records, which is a set of fields. All records in a table have the same set of fields with different information. There are primary key fields that uniquely identify a record in a table. There are also fields that contain primary key from another table called foreign keys.

**3.2** UML APPROACH

UML stands for Unified Modeling Language. UML is a language for specifying,visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

Software design is a process that gradually changes as various new, better and more complete methods with a broader understanding of the of the whole problem in general come into existence. There are various kinds of methods in software design. They are as follows :

* Usecase Diagram
* Activity Digram
* Collabration Diagram
* Class Diagram

The top-level diagram is often called a “context diagram”. It contains a single process, but it plays a very important role in studying the current system. The context diagram defines the system that will be studied in the sense that it determines the boundaries. Anything that is not inside the process identified in the context diagram will not be part of the system study.

**3.2.1 Use case Diagrams**

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor.

An actor represents an outside entity- either human or technological. In this example its human (Stick man). Notice the curved rectangle on the diagram this represents the system boundary everything inside that is part of that system, and everything outside are actors (basically not part of system).

Use case diagrams can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do.

Usecase Diagram consists of usecases and actors and shows the interaction between the usecase and actors.

* The purpose is to show the interactions between use cases and actor.
* To represent the system requirements from user’s perspective.
* It must be remembered that the use-cases are the functions that are to be performed in the module.
* An actor could be the end-user of the system or an external system.

**3.2.2 Sequence Diagram**

The purpose is to show the of functioning through a use case. In other Words, we call it mapping processes in terms of data transfers from the actor through corresponding objects.

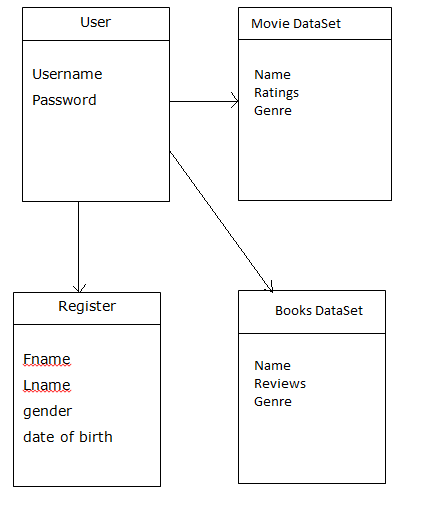
* To represent the logical flow of data with respect to a process.
* It must be remembered that the the sequence diagram display Objects and not the classes.

**3.2.3 Class Diagram**

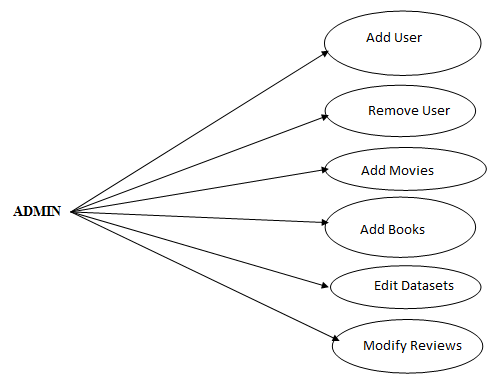
* This is one of the most important of the diagrams in development.
* The diagram break the class into three layers. One has the name ,the second describes its attributes and the third its methods. The private attributes are represented by a padlock to left of the name.
* The relationships are drawn between the classes.
* Developers use the Class Diagram to develop the classes.
* Analyses use it to show the details of the system.

Architects look at class diagrams to see if any class has too many functions and see if they are required to be split.

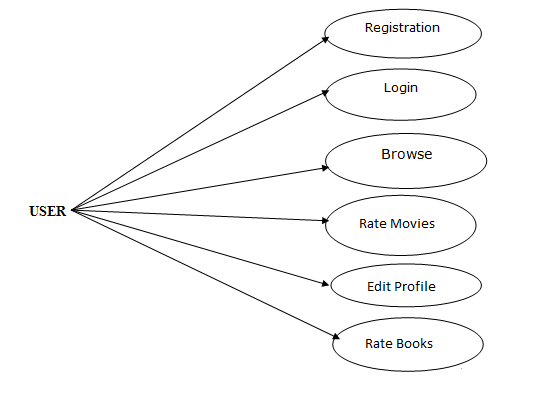
1. **CLASS DIGRAM**



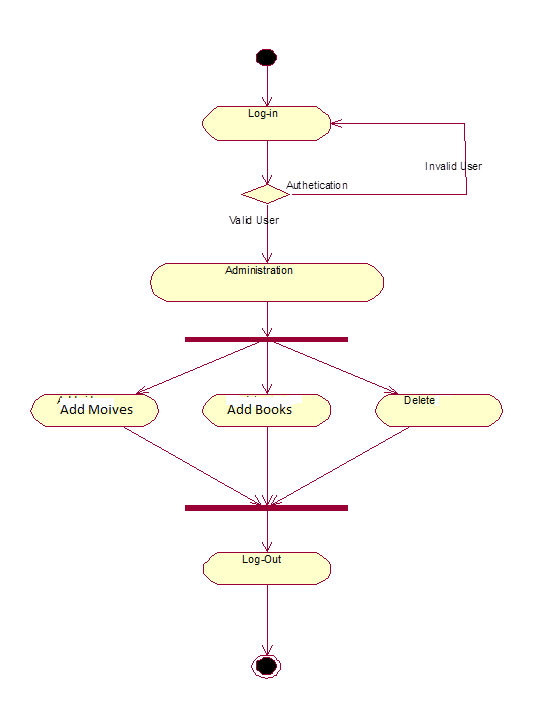
**Use Case for Administrator**



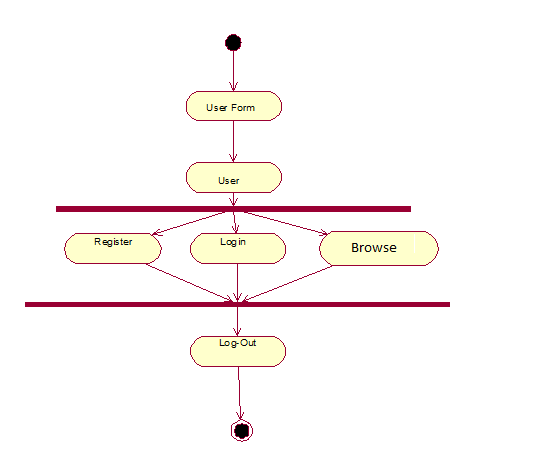
**Use Case Diagram for Normal User:**



**Activity Diagram for Admin**



**Activity diagram for the users**



**3.2.4 Data Flow Diagram (0 level)**

A graphical tool used to describe and analyze the moment of data in system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output may be described logically and independently of the physical components associated with the system. The DFD is also known as a data flow graph or a bubble chart. A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The DFD is also known as a data flow graph or a bubble chart.

**Types of DFD’s**

Data Flow Diagrams are of two types as follows:

(a)Physical DFD

(b)Logical DFD

**Physical DFD:**

Structured analysis states that the current system should be first understand correctly. The physical DFD is the model of the current system and is used to ensure that the current system has been clearly understood. Physical DFDs shows actual devices, departments, and people etc., involved in the current system.

**Logical DFD:**

Logical DFDs are the model of the proposed system. They clearly should show the requirements on which the new system should be built. Later during design activity this is taken as the basis for drawing the system’s structure charts.

This diagram shows the Automatic Teller System Software and the hardware that it interacts with. The arrows show the direction and type of data flowing between the software and each hardware element.



**CHAPTER-4**

# **TESTING**

**4 SYSTEM TESTING**

Testing is a set activity that can be planned and conducted systematically. Testing begins at the module level and work towards the integration of entire computers based system. Nothing is complete without testing, as it is vital success of the system.

• Testing Objectives:

There are several rules that can serve as testing objectives, they are

1. Testing is a process of executing a program with the intent of finding an error
2. A good test case is one that has high probability of finding an undiscovered error.
3. A successful test is one that uncovers an undiscovered error.

If testing is conducted successfully according to the objectives as stated above, it would uncover errors in the software. Also testing demonstrates that software functions appear to the working according to the specification, that performance requirements appear to have been met.

There are three ways to test a program

1. For Correctness
2. For Implementation efficiency
3. For Computational Complexity.

Tests for correctness are supposed to verify that a program does exactly what it was designed to do. This is much more difficult than it may at first appear, especially for large programs.

Tests for implementation efficiency attempt to find ways to make a correct program faster or use less storage. It is a code-refining process, which reexamines the implementation phase of algorithm development.

Tests for computational complexity amount to an experimental analysis of the complexity of an algorithm or an experimental comparison of two or more algorithms, which solve the same problem.

• Testing Correctness

The following ideas should be a part of any testing plan:

1. Preventive Measures
2. Spot checks
3. Testing all parts of the program
4. Test Data
5. Looking for trouble
6. Time for testing
7. Re Testing

The data is entered in all forms separately and whenever an error occurred, it is corrected immediately. A quality team deputed by the management verified all the necessary documents and tested the Software while entering the data at all levels. The entire testing process can be divided into 3 phases

1. Unit Testing
2. Integrated Testing
3. Final/ System testing

**4.1 UNIT TESTING**

As this system was partially GUI based WINDOWS application, the following were tested in this phase

1. Tab Order
2. Reverse Tab Order
3. Field length
4. Front end validations

In our system, Unit testing has been successfully handled. The test data was given to each and every module in all respects and got the desired output. Each module has been tested found working properly.

**4.2 INTEGRATION TESTING**

Test data should be prepared carefully since the data only determines the efficiency and accuracy of the system. Artificial data are prepared solely for testing. Every program validates the input data.

**4.3 VALIDATION TESTING**

In this, all the Code Modules were tested individually one after the other. The following were tested in all the modules

1. Loop testing
2. Boundary Value analysis
3. Equivalence Partitioning Testing

In our case all the modules were combined and given the test data. The combined module works successfully without any side effect on other programs. Everything was found fine working.

**4.4 OUTPUT TESTING**

This is the final step in testing. In this the entire system was tested as a whole with all forms, code, modules and class modules. This form of testing is popularly known as Black Box testing or system testing.

Black Box testing methods focus on the functional requirement of the software. That is, Black Box testing enables the software engineer to derive sets of input conditions that will fully exercise all functional requirements for a program. Black Box testing attempts to find errors in the following categories; incorrect or missing functions, interface errors, errors in data structures or external database access, performance errors and initialization errors and termination errors.

**future enhancements**

The developed system is flexible and changes can be made easily. The system is developed with an insight into the necessary modification that may be required in the future. Hence the system can be maintained successfully without much rework. One of the main future enhancements of our system is to include Login Facilities, Specified User Profiles and Larger Dataset.

# **CONCLUSION**

The project report entitled "**CROSS-DOMAIN RECOMMENDER SYSTEM**" has come to its final stage. The system has been developed with much care that it is free of errors and at the same time it is efficient and less time consuming. The important thing is that the system is robust. We have tried our level best to make the system as precise as possible. Also provision is provided for future developments in the system. The entire system is secured. This online system will be approved and implemented soon.

# REFERENCES

* Billsus, D., & Pazzani, M. (1998). Learning collaborative information filters. Paper presented at the Proceedings of the Fifteenth International Conference on Machine Learning, Madison, Wisconson, USA. 46-54. Retrieved from <http://portal.acm.org/citation.cfm?coll=GUIDE&dl=GUIDE&id=657311>
* Burback, R. (1998). A cyclical methodology, introduction. Retrieved March 10, 2008, from Ronald LeRoi Burback
* Christakou, C., & Stafylopatis, A. (2005). A hybrid movie recommender system based on neural networks. Paper presented at the 5th International Conference on Intelligent Systems Design and Applications, 500-505. doi:10.1109/ISDA.2005.9
* Droux, S. (2008). Vogoo - recommendation engine and collaborative filtering. Retrieved March 30, 2008, from <http://www.vogooapi.com/>
* E. Peis, J. M. Morales-del-Castillo (2008) Semantic Recommender Systems. Analysis of the state of the topic. Web: www.hipertext.net, ISSN 1695-5498.
* Gediminas Adomavicius, Alexander Tuzhilin (2005). Toward the Next Generation of Recommender
* Systems: A Survey of the State-of-the- Art and Possible Extensions. IEEE, Transactions on Knowledge and Data Engineering, 734-749.
* GroupLens research. (2008). Retrieved March 29, 2008, from <http://www.grouplens.org/>
* Klir, G., & Folger, T. (1988). Equivalence and similarity relations. Fuzzy sets, uncertainty, and information (1st ed., pp. 82-85). Upper Saddle River, NJ, USA: Prentice-Hall, Inc.
* Kumar P, Gopalan S, Sridhar V (2005). Context enabled multi-CBR based recommendation engine for e-commerce. IEEE International Conference on e-Business Engineering, 237-255.
* Machine learning. (2008). Retrieved March 27, 2008, from <http://en.wikipedia.org/wiki/Machine_learning>
* Olfa Nasraoui, Mrudula Pavuluri (2004). Accurate web recommendations based on profile-specific url-predictor neural networks.. ACM, Proceedings of the 13th international World Wide Web conference on Alternate track papers and posters, 300-301.
* Pereira, R., Ricarte, I., & Gomide, F. (2006). Fuzzy relational ontological model in information search systems. In E. Sanchez, & F. Harmelen (Eds.), Fuzzy logic and the semantic web (1st ed., ). New York, NY, USA: Elsevier Science Inc. Retrieved from http://books.google.com/books?hl=en&lr=&id=Cidej8b4ESIC&oi=fnd&pg= PA395&dq=%22Pereira%22+%22Fuzzy+Relational+Ontological+Model+in+In formation+Search+...%22+&ots=mt7470caQ3&sig=Yx8CZ49DEqnzS7ICaI5D\_p DDPko#PPA395,M1
* Sean, O. (2008). Taste collaborative filtering for java. Retrieved March 26, 2008, from <http://taste.sourceforge.net/>
* Stuart E. Middleton, Nigel R. Shadbolt (2004). Ontological user profiling in recommender systems. ACM Transactions on Information Systems, 54-88.
* Taste, (2008). Collaborative filtering for Java. Retrieved on August 15, 2008, from <http://taste.sourceforge.net/>
* Thesaurus.com. (2008). Retrieved September 10, 2008, from <http://thesaurus.reference.com/>
* Toby Segaran (2007). Programming Collective Intelligence: Building Smart Web 2.0 Applications, Page11
* Yahoo developer network. (2008). Retrieved April 1, 2008, from

http://developer.yahoo.com/