CHAPTER - 1
INTRODUCTION

1.1 Background

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. It is necessary to analyze this huge amount of data and extract useful information from it. Data mining also involves other processes such as Data Cleaning, Data Integration, Data Transformation, Data Mining, Pattern Evaluation and Data Presentation. Once all these processes are over, we would be able to use this information in many applications such as Fraud Detection, Market Analysis, Production .The overall goal of data mining process is extract information from data set and transforms it into understanding structure for further use.

Frequent item set mining and high utility item set mining, these two concepts used here. Frequent pattern mining is a popular problem in data mining, which consists in finding frequent patterns in transaction databases. The goal of frequent item set mining is to find **frequent item sets.** . In frequent item set mining, there is a well-known property of the frequency (support) of item sets that states that given an item set, all its supersets must have a support that is lower or equal. Many popular algorithms have been proposed for this problem such as Apriori, FP Growth. These algorithms takes as input a transaction database and a parameter "**min sup**" called the **minimum support threshold.** This algorithm is very powerful to prune the search space because if an item set is infrequent then we know that all its supersets are also infrequent. In high utility item set mining there is no such property but it has some important limitations. To address these limitations, the problem of frequent item set mining has been redefined as the problem of **high-utility item set mining.** It cannot satisfy the requirement of users who desire to discover item sets with high utilities such as high profits .The problem of **high-utility item set mining** is to find the item sets (group of items) that generate a high profit in a database, when they are sold together. The user has to provide a value for a threshold called "**min utility**" (the minimum utility threshold). An item set is called high utility item set (HUI) if

its utility is no less than a user-specified minimum utility threshold **min utility**. High utility item sets (HUIs) mining is an emerging topic in data mining.

1.2 Relevance

Table 1.2.1 Summary of literature survey

Number	Paper Name	Advantage/Disadvantage	Concepts
1.	Mining High Utility Pattern in One Phase Without Generating Candidates	**Add:-** 1.Without Generating Candidates 2. Pattern growth approach to avoid the level wise candidate generation. **Disadvantage:-** 1. In Middle IHUP Tree are generated then Time is getting Large.	Mining High Utility Pattern in One Phase Without Generating Candidates". This article represents three variations of tree structure for high utility pattern mining for handling incremental databases. In this paper, three variations of tree structure have been proposed that are $IHUP_L$-tree, $IHUP_{TF}$-tree and $IHUP_{TWU}$-tree. These are very efficient for incremental and interactive high utility pattern mining. In that paper, the author used pattern growth approach to avoid the level wise candidate generation.
2.	Efficient Algorithms for Mining High Utility Item sets from Transactional Databases.	**Add:-** 1. Uncertain data are used. 2. UHUI-apriori which has to solve the problem of mining high utility item sets (MHUI) over uncertain databases, in which each item has a utility **Disadvantage:-** 1.Data Structure is used as uncertain data which has no format and rare data set are used.	Efficient Algorithms for Mining High Utility Item sets from Transactional Databases. ".In this paper, a large amount of uncertain data, i.e. sensor data, real time monitoring data has been collected and the problem of mining uncertain frequent item sets has attracted much attention in the database and data mining communities. The author proposed an efficient mining algorithm named UHUI-apriori which has to solve the problem of mining high utility item sets (MHUI) over uncertain databases, in which each item has a utility
3.	Performance Evaluation of Class Balancing Techniques for Credit Card Fraud Detection.	**Add:-** 1.To identify fraud , previous work has focused on fraud rating 2. Detect both malware and apps subjected to search rank fraud **Disadvantage:-** 1. Data set are fixed.	Fraudulent behaviors in Google Play, the most popular Android app market, fuel search rank abuse and malware proliferation. To identify malware, previous work has focused on app executable and permission analysis. In this paper, we introduce FairPlay, a novel system that discovers and leverages traces left behind by fraudsters, to detect both malware and apps subjected to search rank fraud. FairPlay correlates review activities and uniquely combines

			detected review relations with linguistic and behavioral signals gleaned from Google Play app data (87K apps, 2.9M reviews, and 2.4M reviewers, collected over half a year), in order to identify suspicious apps. FairPlay achieves over 95% accuracy in classifying gold standard datasets of malware, fraudulent and legitimate apps. We show that 75% of the identified malware apps engage in search rank fraud. FairPlay discovers hundreds of fraudulent apps that currently evade Google Bouncer's detection technology. FairPlay also helped the discovery of more than 1,000 reviews, reported for 193 apps, that reveal a new type of "coercive" review campaign: users are harassed into writing positive reviews, and install and review other apps

The proposed work of authors is discussed in detail below

A. C. Ahmed, S. Tanbeer, B. Jeong, and Y. Lee [1] : "Efficient tree structures for high-utility pattern mining in incremental databases". Recently, high utility pattern (HUP) mining is one of the most important research issues in data mining due to its ability to consider the non-binary frequency values of items in transactions and different profit values for every item. On the other hand, incremental and interactive data mining provide the ability to use previous data structures and mining results in order to reduce unnecessary calculations when a database is updated, or when the minimum threshold is changed. In this paper, we propose three novel tree structures to efficiently perform incremental and interactive HUP mining. The first tree structure, Incremental HUP Lexicographic Tree (IHUPL-Tree), is arranged according to an item's lexicographic order. It can capture the incremental data without any restructuring operation. The second tree structure is the IHUP Transaction Frequency Tree (IHUPTF-Tree), which obtains a compact size by arranging items according to their transaction frequency (descending order). To reduce the mining time, the third tree, IHUP-Transaction-Weighted Utilization Tree (IHUPTWU-Tree) is designed based on the TWU value of items in descending order. Extensive per-

formance analyses show that our tree structures are very efficient and scalable for incremental and interactive HUP mining.

B. R. Chan, Q. Yang, and Y. Shen [2]: "Mining high-utility item sets" Traditional association rule mining algorithms only generate a large number of highly frequent rules, but these rules do not provide useful answers for what the high utility rules are. We develop a novel idea of top-K objective-directed data mining, which focuses on mining the top-K high utility closed patterns that directly support a given business objective. To association mining, we add the concept of utility to capture highly desirable statistical patterns and present a level-wise item-set mining algorithm. With both positive and negative utilities, the anti-monotone pruning strategy in Apriori algorithm no longer holds. In response, we develop a new pruning strategy based on utilities that allow pruning of low utility item sets to be done by means of a weaker but anti-monotonic condition. Our experimental results show that our algorithm does not require a user specified minimum utility and hence is effective in practice.

C. J. Han, J. Wang, Y. Lu, and P. Tzvetkov [3]:"Mining top-k frequent closed patterns without minimum support" In this paper, we propose a new mining task: mining top-k frequent closed patterns of length no less than min_/spllscr/, where k is the desired number of frequent closed patterns to be mined, and min_/spllscr/ is the minimal length of each pattern. An efficient algorithm, called TFP, is developed for mining such patterns without minimum support. Two methods, closed-node-count and descendant-sum are proposed to effectively raise support threshold and prune FP-tree both during and after the construction of FP-tree. During the mining process, a novel top-down and bottom-up combined FP-tree mining strategy is developed to speed-up support-raising and closed frequent pattern discovering. In addition, a fast hash-based closed pattern verification scheme has been employed to check efficiently if a potential closed pattern is really closed. Our performance study shows that in most cases, TFP outperforms CLOSET and CHARM, two efficient frequent closed pattern mining algorithms, even when both are running with the best tuned min-support. Furthermore, the method can be extended to generate association rules and to incorporate user-specified constraints.

D. J. Han, J. Pei, and Y. Yin [4]: "Mining frequent patterns without candidate Generation" Mining frequent patterns in transaction databases, times Series databases, and many other kinds of databases have been studied popularly in data mining research. Most of the previous studies adopt an Apriori-like candidate set generation-and-test approach. However, candidate set generation is still costly, especially when there existproli c patterns and/or long patterns. In this study, we propose a novel frequent pattern tree (FP-tree) structure, which is an extended pre xtree structure for storing compressed, crucial information about frequent patterns, and develop an e client FP-tree based mining method, FP-growth, for mining the complete set of frequent patterns by pattern fragment growth. Exigency of mining is achieved with three techniques: (1) a large database is compressed into a highly condensed, much smaller data structure, which avoids costly, repeated database scans, (2) our FP-tree-based mining adopts a pattern fragment growth method to avoid the costly generation of a large number of candidate sets, and (3) a partitioning-based, divide-and-conquer method is used to decompose the mining task into a set of smaller tasks for mining con need patterns in conditional databases, which dramatically reduces the search space. Our performance study shows that the FP-growth method is ancient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm and also faster than some recently reported new frequent pattern mining methods.

E. P. Fournier-Viger, C. Wu, and V. S. Tseng [5]:"Novel Concise Representations of High Utility Item sets Using Generator Patterns" Mining High Utility Item sets (HUIs) is an important task with many applications. However, the set of HUIs can be very large, which makes HUI mining algorithms suffer from long execution times and huge memory consumption. To address this issue, concise representations of HUIs have been proposed. However, no concise representation of HUIs has been proposed based on the concept of generator despite that it provides several benefits in many applications. In this paper, we incorporate the concept of generator into HUI mining and devise two new concise representations of HUIs, called High Utility Generators (HUGs) and Generator of High Utility Item sets (GHUIs). Two efficient algorithms named HUG-Miner and GHUI-Miner are proposed to respectively mine these representations. Experiments on

both real and synthetic datasets show that proposed algorithms are very efficient and that these representations are up to 36 times smaller than the set of all HUIs.

F. P. Fournier-Viger and V. S. Tseng [6]:"Mining Top-K Sequential Rules" Mining sequential rules requires specifying parameters that are often difficult to set (the minimal confidence and minimal support). Depending on the choice of these parameters, current algorithms can become very slow and generate an extremely large amount of results or generate too few results, omitting valuable information. This is a serious problem because in practice users have limited resources for analyzing the results and thus are often only interested in discovering a certain amount of results, and fine-tuning the parameters can be very time-consuming. In this paper, we address this problem by proposing TopSeqRules, an efficient algorithm for mining the top-k sequential rules from sequence databases, where k is the number of sequential rules to be found and is set by the user. Experimental results on real-life datasets show that the algorithm has excellent performance and scalability.

G. J. Liu, K. Wang, and B. Fung [7]:"Direct Discovery of High Utility Itemsets without Candidate Generation" Utility mining emerged recently to address the limitation of frequent itemset mining by introducing interestingness measures that reflect both the statistical significance and the user's expectation. Among utility mining problems, utility mining with the itemset share framework is a hard one as no anti-monotone property holds with the interestingness measure. The state-of-the-art works on this problem all employ a two-phase, candidate generation approach, which suffers from the scalability issue due to the huge number of candidates. This paper proposes a high utility itemset growth approach that works in a single phase without generating candidates. Our basic approach is to enumerate itemsets by prefix extensions, to prune search space by utility upper bounding, and to maintain original utility information in the mining process by a novel data structure. Such a data structure enables us to compute a tight bound for powerful pruning and to directly identify high utility itemsets in an efficient and scalable way. We further enhance the efficiency significantly by introducing recursive irrelevant item filtering with sparse data, and a lookahead strategy with dense data. Extensive experiments on sparse and dense, synthetic and real data

suggest that our algorithm outperforms the state-of-the-art algorithms over one order of magnitude.

H. Y. Lin, C. Wu, and V. S. Tseng[8]:"Mining High Utility Itemsets in Big Data" In recent years, extensive studies have been conducted on high utility itemsets (HUI) mining with wide applications. However, most of them assume that data are stored in centralized databases with a single machine performing the mining tasks. Consequently, existing algorithms cannot be applied to the big data environments, where data are often distributed and too large to be dealt with by a single machine. To address this issue, we propose a new framework for mining high utility itemsets in big data. A novel algorithm named PHUI-Growth (Parallel mining High Utility Itemsets by pattern-Growth) is proposed for parallel mining HUIs on Hadoop platform, which inherits several nice properties of Hadoop, including easy deployment, fault recovery, low communication overheads and high scalability. Moreover, it adopts the MapReduce architecture to partition the whole mining tasks into smaller independent subtasks and uses Hadoop distributed file system to manage distributed data so that it allows to parallel discover HUIs from distributed data across multiple commodity computers in a reliable, fault tolerance manner. Experimental results on both synthetic and real datasets show that PHUI-Growth has high performance on large-scale datasets and outperforms state-of-the-art non-parallel type of HUI mining algorithms.

I. Y. Li, J. Yeh, and C. Chang[9]:"Isolated items discarding strategy for discovering high utility item sets" Traditional methods of association rule mining consider the appearance of an item in a transaction, whether or not it is purchased, as a binary variable. However, customers may purchase more than one of the same item, and the unit cost may vary among items. Utility mining, a generalized form of the share mining model, attempts to overcome this problem. Since the Apriori pruning strategy cannot identify high utility item sets, developing an efficient algorithm is crucial for utility mining. This study proposes the Isolated Items Discarding Strategy (IIDS), which can be applied to any existing level-wise utility mining method to reduce candidates and to improve performance. The most efficient known models for share mining are ShFSM and DCG, which also work adequately for utility mining as well. By applying IIDS to

ShFSM and DCG, the two methods FUM and DCG+ were implemented, respectively. For both synthetic and real datasets, experimental results reveal that the performance of FUM and DCG+ is more efficient than that of ShFSM and DCG, respectively. Therefore, IIDS is an effective strategy for utility mining.

J. T. Quang, S. Oyanagi, and K. Yamazaki [10]:"ExMiner: An efficient algorithm for mining top-k frequent patterns" Conventional frequent pattern mining algorithms require users to specify some minimum support threshold. If that specified-value is large, users may lose interesting information. In contrast, a small minimum support threshold results in a huge set of frequent patterns that users may not be able to screen for useful knowledge. To solve this problem and make algorithms more user-friendly, an idea of mining the k-most interesting frequent patterns has been proposed. This idea is based upon an algorithm for mining frequent patterns without a minimum support threshold, but with a k number of highest frequency patterns. In this paper, we propose an explorative mining algorithm, called ExMiner, to mine k-most interesting (i.e. top-k) frequent patterns from large scale datasets effectively and efficiently. The ExMiner is then combined with the idea of "build once mine anytime" to mine top-k frequent patterns sequentially. Experiments on both synthetic and real data show that our proposed methods are more efficient compared to the existing ones.

1.3 Project Undertaken

In Data Mining frequently generate a huge set of HUIs to such as state- art-algorithms but their mining performance is degraded consequently Further in case of long transactions in dataset or low thresholds or min Utility value are set then this condition become a worst. To improve this condition Top k generated the hybrid algorithm TKO With TKU without need to set the low thresholds or min Utility value and Execution time in long transactions or denser data set is also low

1.4 Organization of Report

- **Chapter 1 – Introduction :** In this chapter survey of efficient algorithm for mining.

- **Chapter 2 -Background**: This chapter shows the study of journal/conference papers or other magazines related to HUI mining.
- **Chapter 3 –Specialization:** Inthis chapter we explain the existing system, previous work, the different techniques and algorithms.
- **Chapter 4 –System Design:** In this chapter we explain the proposed system in detail and the problem statement
- **Chapter 5 –Implementation:** In this chapter we explain the Modules, Data flow diagrams and UML diagrams.
- **Chapter 6 –Result and Evaluation:** The conclusion is summarized with a brief discussion of the given approach. List of publications in international journals, international conferences used for this research work is also provided at the end.
- **Chapter 7 –Conclusion:** The conclusion is summarized with a brief discussion of the given approach. List of publications in international journals, international conferences used for this research work is also provided at the end.

CHAPTER 2
BACKGROUND

2.1 Basic Concepts

There is a huge amount of data available in the Information Industry. This data is of no use until it is converted into useful information. Existing top-k high utility item set (HUI) mining algorithms generate candidate item sets in the mining process, their time & space performance might be severely affected when the dataset is large or contains many long transactions and when applied to data streams, the performance of corresponding mining algorithm is especially crucial. To address the above issues by another structure for top-k high utility item set, where k is the coveted number of HUIs to be mined. Two sorts of proficient calculations named TKU (mining Top-K Utility thing sets) and TKO (mining Top-K utility thing sets in one phase) are used for mining to get high utility item sets without the need to set min utility. These two algorithms have some drawbacks and takes more time for execution. To overcome above drawbacks, hybrid concept of HUIs has been used which provides correct results. To Get the HUI (High Utility Item Set) fixed the Parameter which Parameter are rating after detection of Fraud Detection means the Fraud user and Product mapping with review and rating given to the particular user.

2.2 Exiting System

FREQUENT item set mining is a fundamental research topic in data mining (FIM) mining. However, the traditional FIM may discover a large amount of frequent but low-value item sets and lose the information on valuable item sets having low selling frequencies. Hence, it cannot satisfy the requirement of users who desire to discover item sets with high utilities such as high profits. To address these issues, utility mining emerges as an important topic in data mining and has received extensive attention in recent years. In utility mining, each item is associated with a utility (e.g. unit profit) and an occurrence count in each transaction (e.g. quantity). The utility of an item set represents its importance, which can be measured in terms of weight, value, quantity or other information depending on the user specification. An item set is called high utility item set

(HUI) if its utility is no less than a user-specified minimum utility threshold min_ utility. HUI mining is essential to many applications such as streaming analysis, market analysis, mobile computing and biomedicine.

2.3 Previous Work on Lifetime

Without setting min_ utility threshold value, to find out high utility itemsets becoming a great challenge. Thus, there is significant importance to find out appropriate min_ utility value. Important problem of extracting frequent item sets from a large uncertain database, interpreted under the Possible World Semantics. This issue is technically challenging, since an uncertain database contains an exponential number of possible world. In this existing system, the incremental mining algorithms, which enable probabilistic frequent item, set results to be refreshed. This reduces the need of re-executing the whole mining algorithm on the new database, which is often more expensive and unnecessary. In this system, a high utility item set growth approach that works in tree structured algorithm. Incur problem of producing large no. of candidate item sets for HUI. The situation may become worse when the database contains lots of long transactions or long high utility item sets. We build up two algorithms, namely utility pattern growth (UP-Growth) and UP-Growth+, for mining high utility item sets with a set of effective strategies for pruning candidate itemsets. The TKU and TKO algorithm are used here. Existing top-k high utility item set (HUI) mining algorithms generate candidate item sets in the mining process; their time & space performance might be severely affected when the dataset is large or contains many long transactions; and when applied to data streams, the performance of corresponding mining algorithm is especially crucial.

Algorithms

1. **TKO (Top K in One Phases)**
2. **TKU (Top K in Utility Phases)**

CHAPTER 3

SOFTWARE REQUIREMENT SPECIFICATION

3.1 Introduction

3.1.1 Project Scope

1. The main concepts of HUI are k value are not fixed while Minimum utility value are not fixed mainly the data are collects with HUI items sets.
2. Second concepts are how to include the concepts of top k pattern mining along with TKO pattern mining along with TKO and TKU model Although TKO and TKU is widely used in mining.

3.1.2 User Classes and Characteristics

User Classes

1. Admin
2. User

Characteristics

First **User** register with details and after login they have two operation such as the select the Product and Search product if user Search product by algorithms such as TKO and TKU and hybridalgorithm (Combination of TKO with TKU)

3.1.3 Assumptions and Dependencies

Dependencies such as the software which used in application apache tomcat 7, JDK 1.7 and My Sql version 5.

3.2 Functional Requirements

3.2.1 System Feature 1(Functional Requirement):-

"Top K" use as Data Set standard Set such as Product Data Set (Food Mart)

3.2.2 System Feature 2(Functional Requirement):-

1. User Search Dynamic HUI according to Parameter
2. User Select the product according to Category.

3.3 External Interface Requirement:-

3.3.1 User Interface:-

1. Home Page

2. User Registration
3. User Login
4. Admin Login
5. User Search Product
6. User Select Product

3.3.2 Hardware Interface:-

The entire software requires a completely equipped computer system including monitor, keyboard, and other input output devices.

3.3.3 Software Interface

The system can use Microsoft as the operating system platform. System also makes use of certain GUI tools. To run this application we need JDK 1.8 and above as java platform and Apache tomcat as server. To store data we need MySQL database.

3.3.4 Communication Interface:-

For Communication our system use Java and Servlet APIs (Controller)

3.4 Non Functional Requirements:-

3.4.1 Performances Requirements:-

The performance of the system lies in the way it is handled. Every user must be givenproper guidance regarding how to use the system. The other factor which affects the performance is the graph of TKO With TKU and TKO and TKU.

3.4.2 Safety Requirements

To ensure the safety of the system, perform regular monitoring of the system so as trace the proper working of the system. Authenticated user is only able to access system and Admin also get the HUI Item set.

3.4.3 Security Requirements

Any unauthorized user cannot access the system. Password authentication can introduced.

Factor access Control is used for user side

3.4.4 Software Quality Attributes

Accuracy

The level of accuracy in the proposed system will be higher. All operation would be done correctly and it ensures that whatever information is coming from the center is accurate. Result is organic results. To achieve more accurate results for a focused Execution time is less because of combination of algorithms (TKO and TKU).

Reliability

The reliability of the proposed system will be high due to the above stated reasons. The reason for the increased reliability of the system is that now there would be proper Result accurate in newly combine Hybrid Algorithms

3.5 System Requirements

3.5.1 Software Requirement

1. Operating System -Windows
2. Application Server - Apache Tomcat
3. Front End - HTML, JDK 1.7, JSP
4. Scripts - JavaScript.
5. Server side Script - Java Server Pages.
6. Database - My SQL 5.0
7. IDE - Eclipse.

3.5.2 Hardware Requirements

1. Processor - Pentium –III
2. Speed - 1.1 GHz

3. RAM - 256 MB(min)
4. Hard Disk - 20 GB
5. Floppy Drive - 1.44 MB
6. Key Board - Standard Windows Keyboard
7. Mouse - Two or Three Button Mouse
8. Monitor - SVGA

3.6 Analysis Models: SDLC Model to be applied

The spiral model is similar to the Incremental model,with more emphasis placed on risk analysis. The spiral model has four phases: Planning, Risk Analysis, Engineering and Evaluation. A software project repeatedly passes through these phases in iterations (called Spirals in this model). The baseline spiral, starting in the planning phase, requirements is gathered and risk is assessed. Each subsequent spiral builds on the baseline spiral. Its one of the Software development model like Waterfall, Agile and V-model.

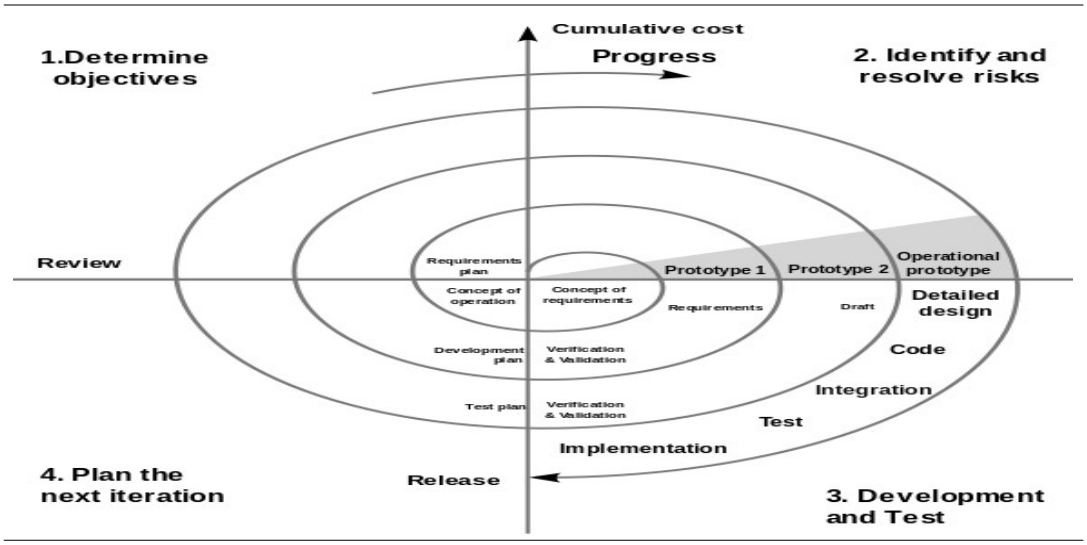


Figure 3.6.1 Spiral Model

CHAPTER 4
SYSTEM DESIGN

1. Use case diagram

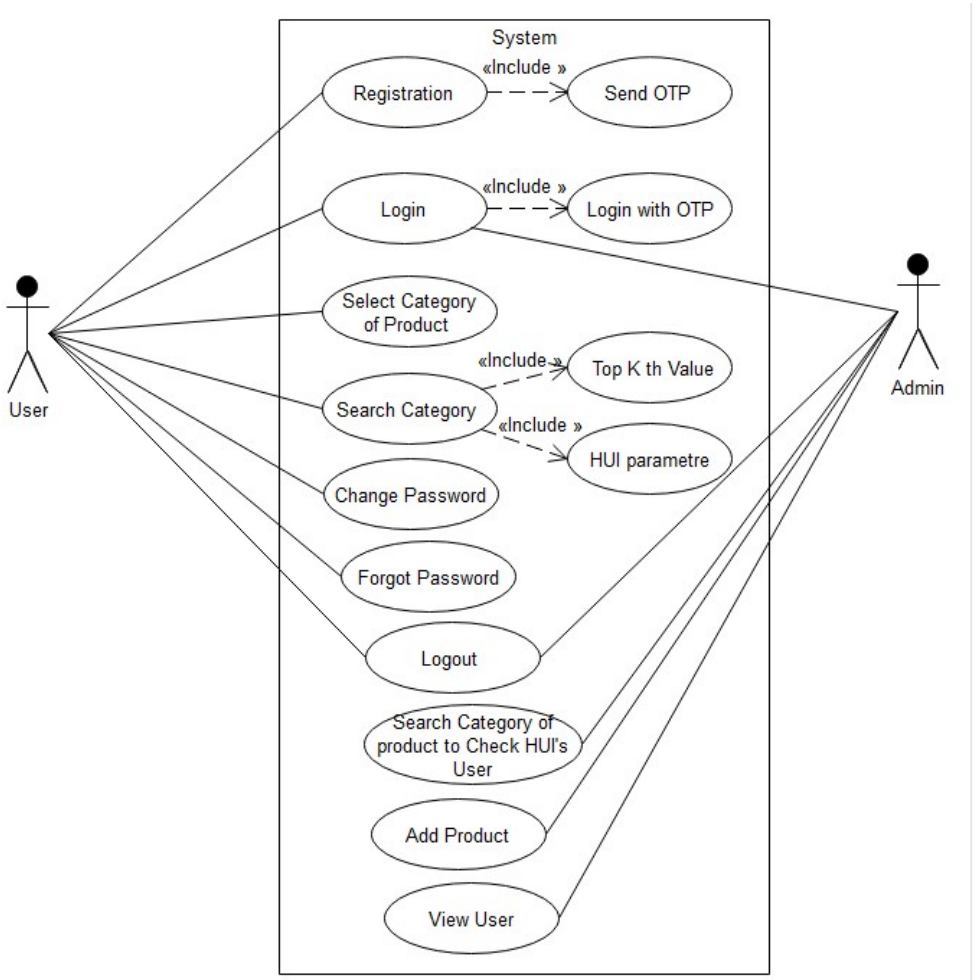


Figure 4.1 Use Case Diagram

This Use case defines the intereaction of a single user and the admin with the system. It also helps to define the functionality of the whole Hybrid Algorithm.

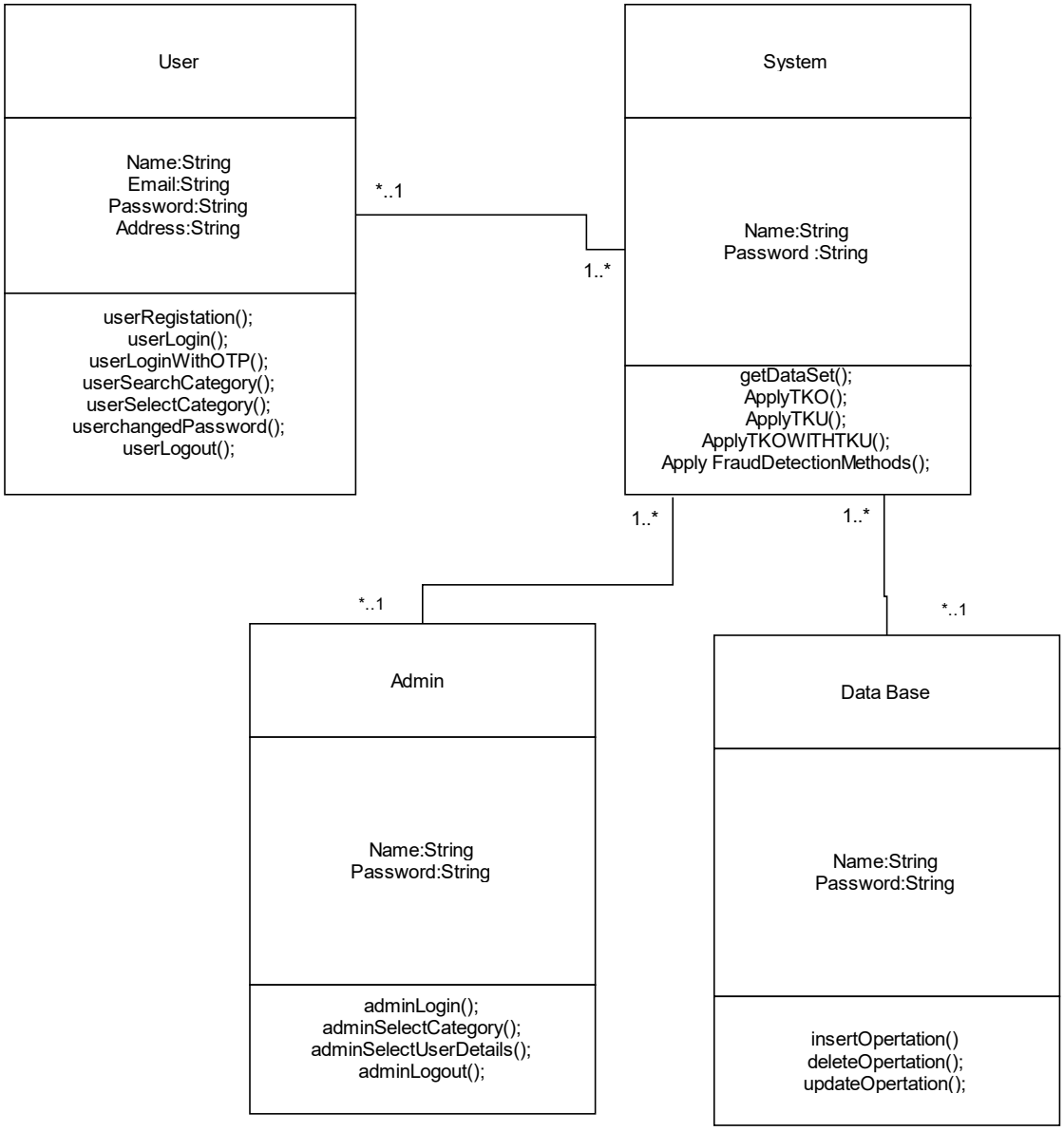


Figure 4.2 Class Diagram

Above Class Diagram states the classes and the functions that will be used in the system. Here the attributes of User class, System class, Admin class and the Database Class is defined.

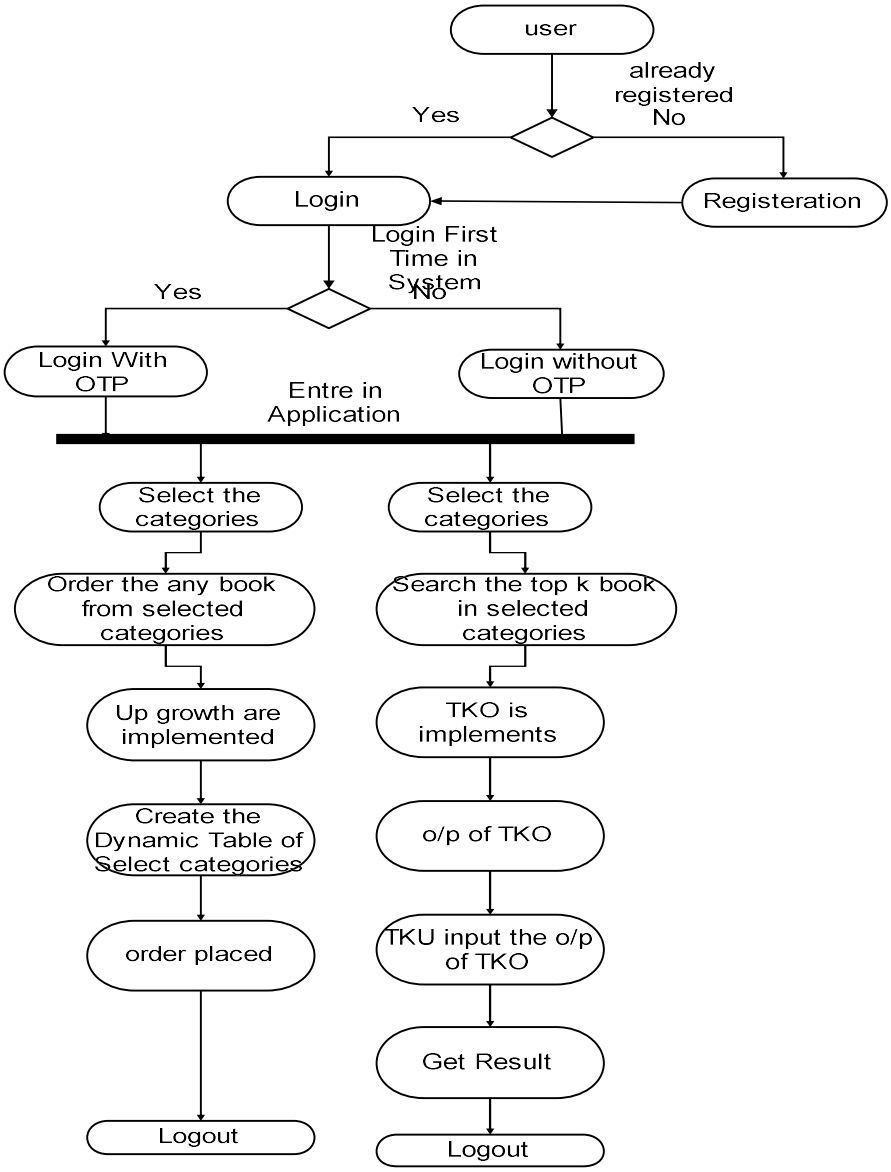


Figure 4.3 Activity Diagram

This Activity Diagram is used in this system to define the execution of the flow from one activity to another activity.The activity is decribed as various operations that represent complete flow of system.

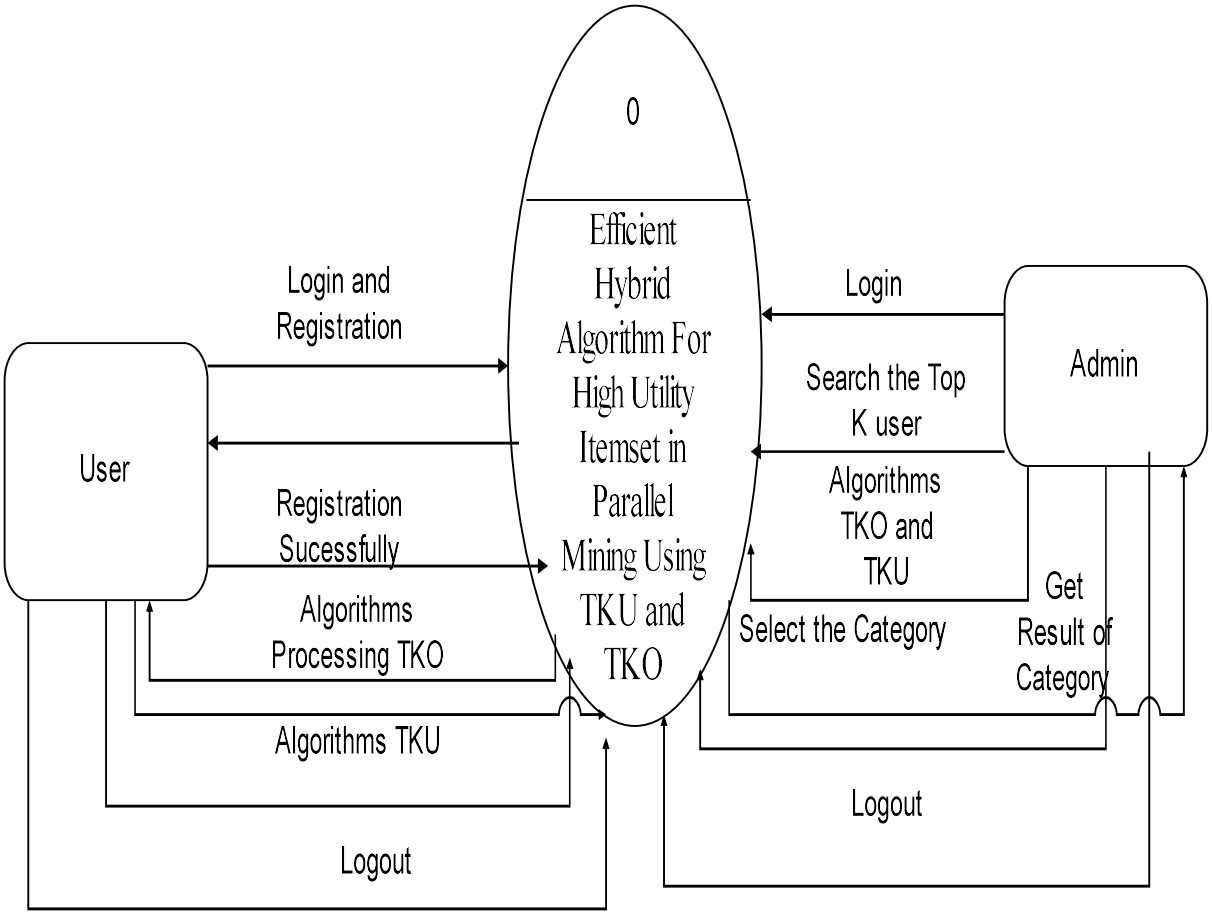


Figure 4.4 DFD Level 0

Data Flow diagram is a way of representing data flow.Data Flow also provide information about the outputs and inputsof each entitiy and the process itself.

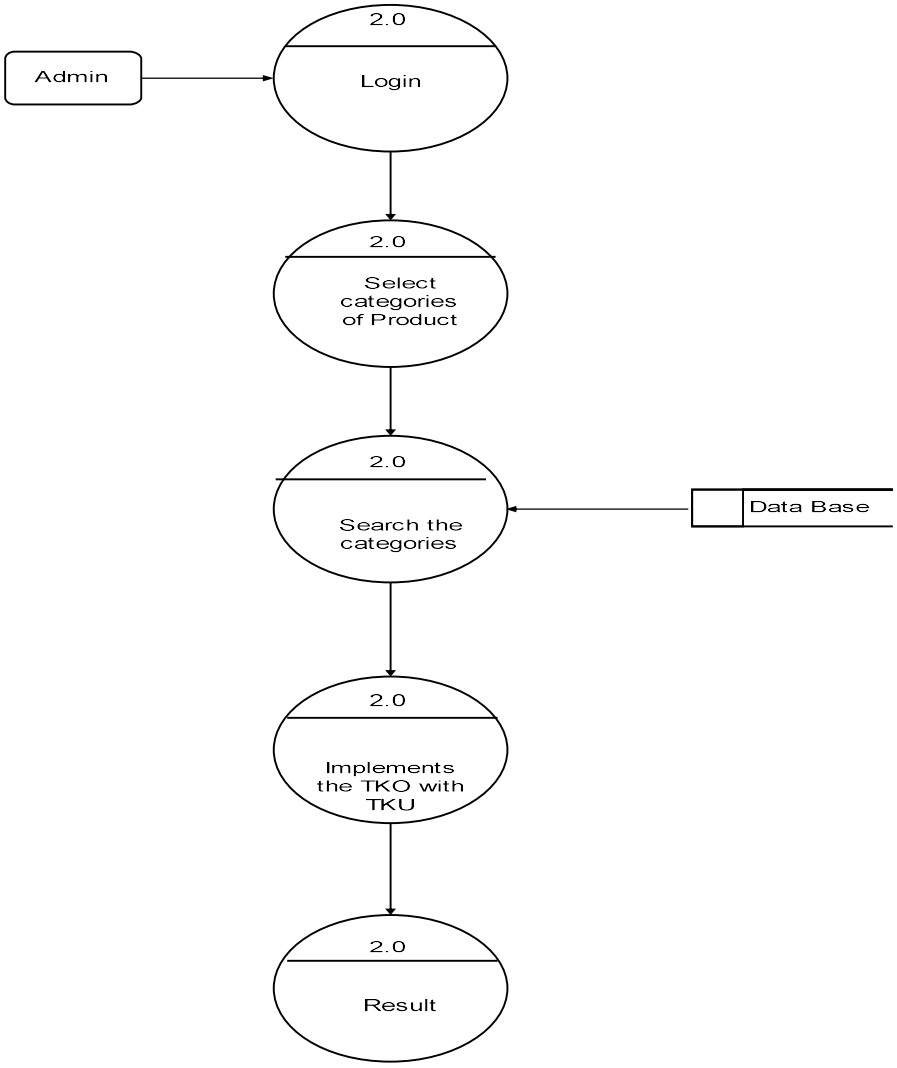


Figure 4.5 DFD Level 1

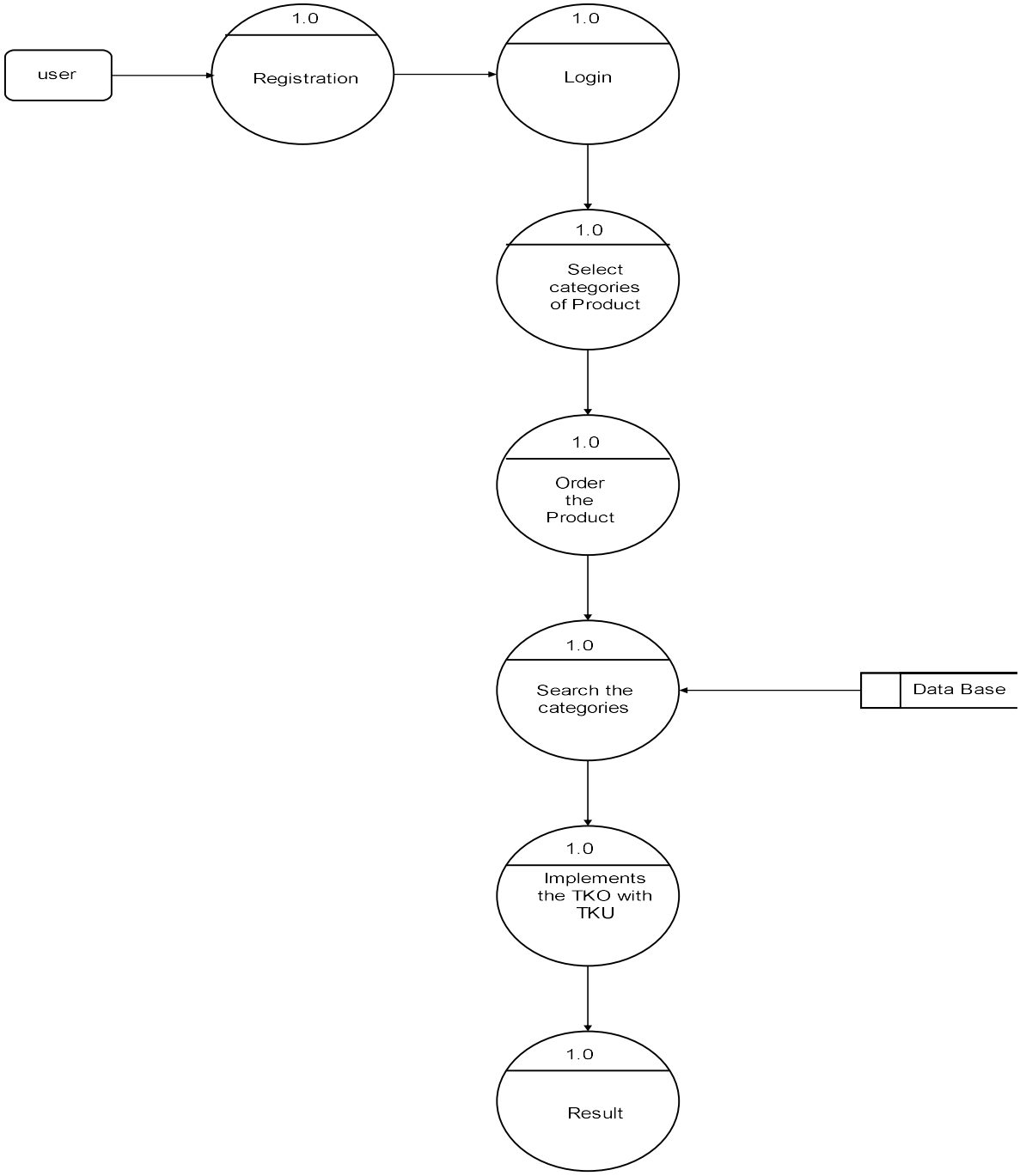


Figure 4.6 DFD Level 2

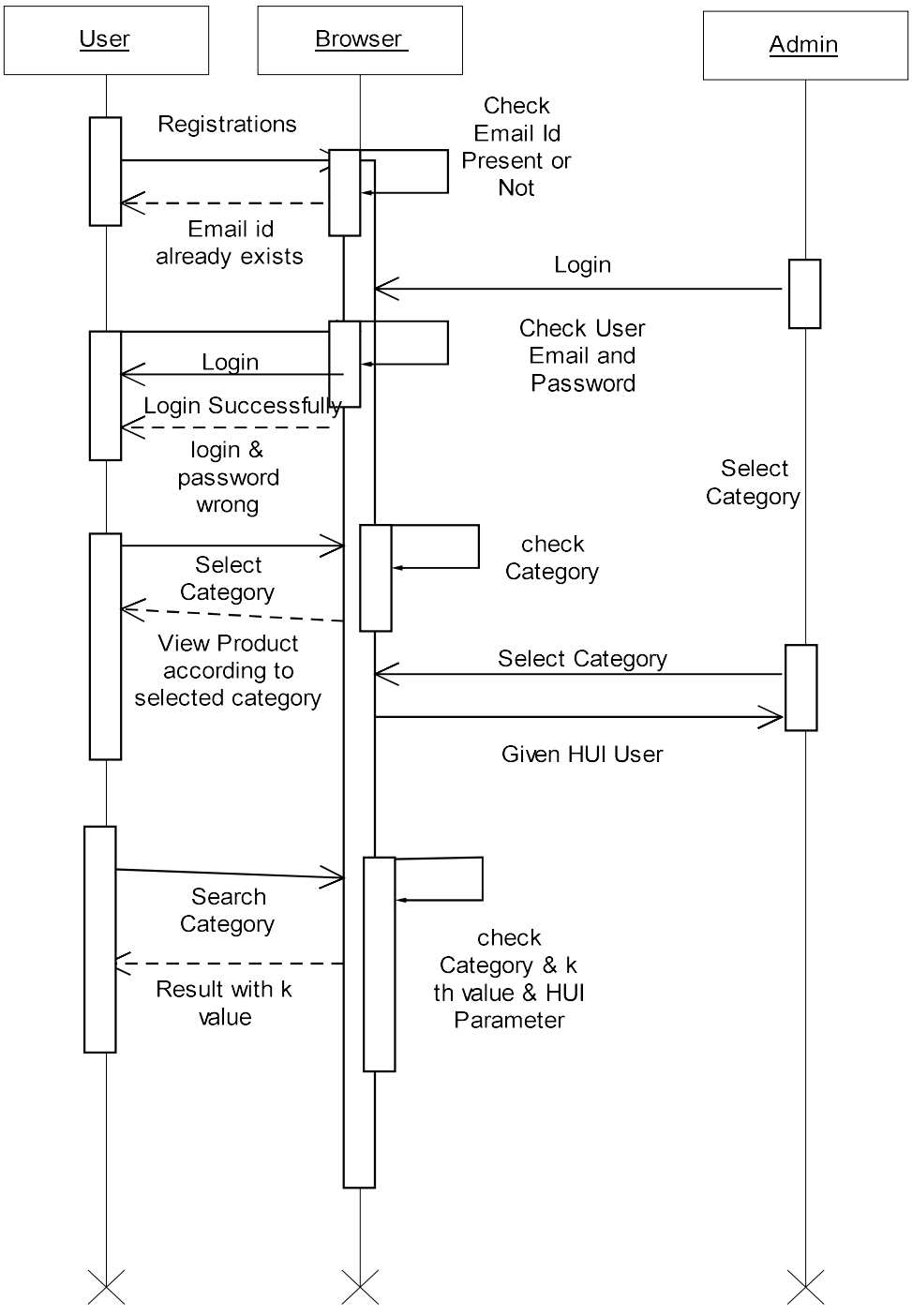


Figure 4.7 Sequence Diagram

A sequence diagram shows object interactions arranged in time sequence.It depicts object and the sequence of messages between the objects needed to carry functionality.

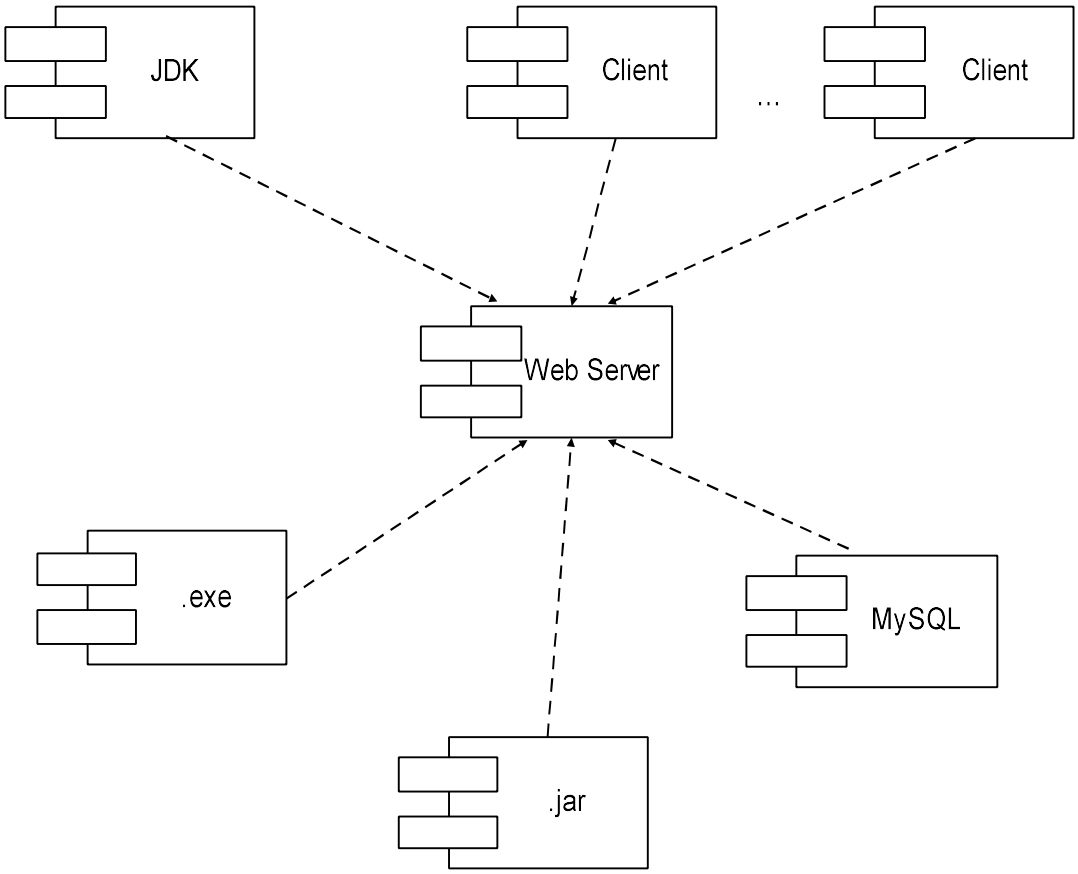


Fig 4.8 Component Diagram

It is used to relationship between different components in a system.There is user development approach that revolves around components based development (CBD). Such as .exe ,.jar,Mysql database,client, jdk.

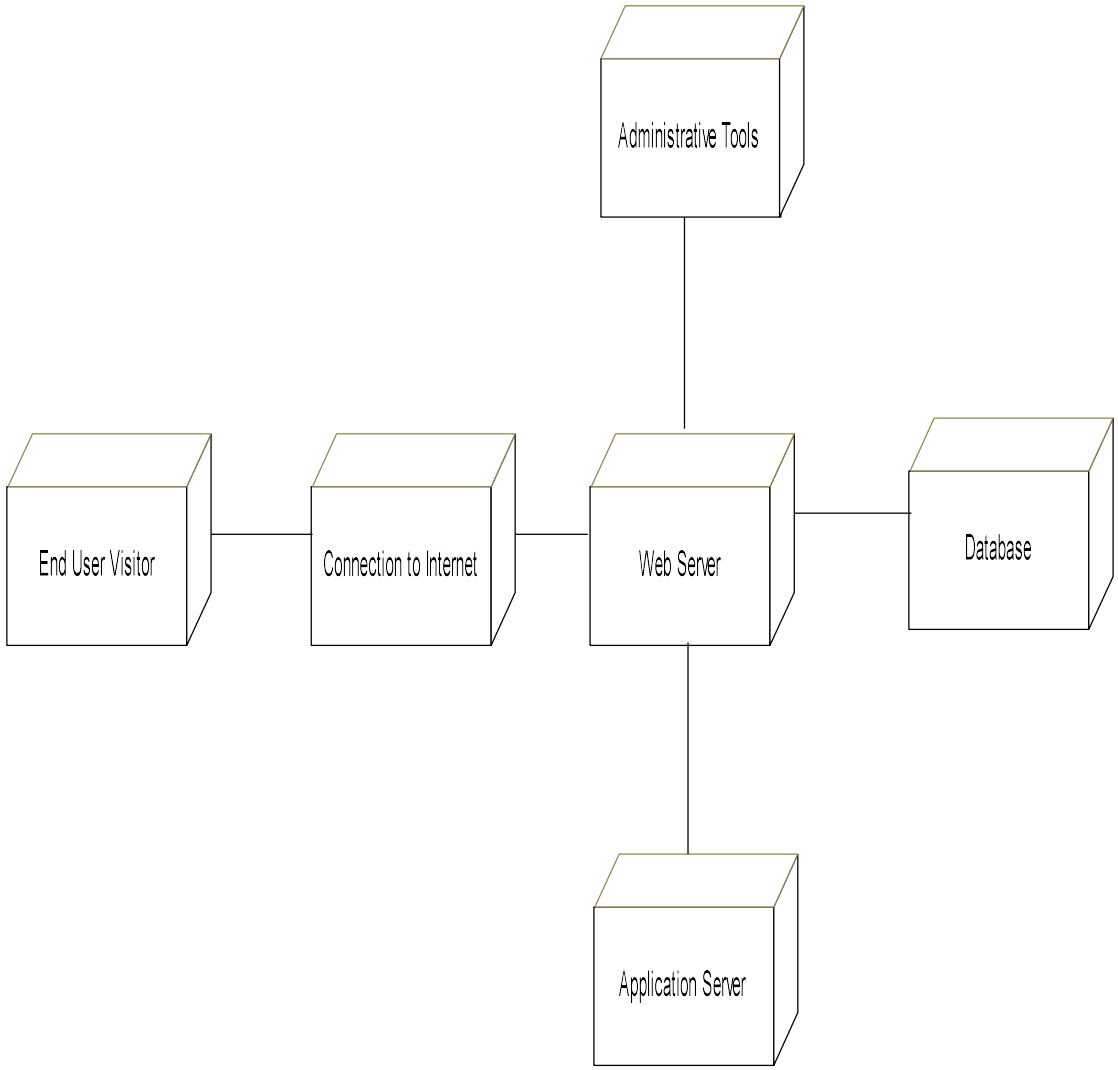


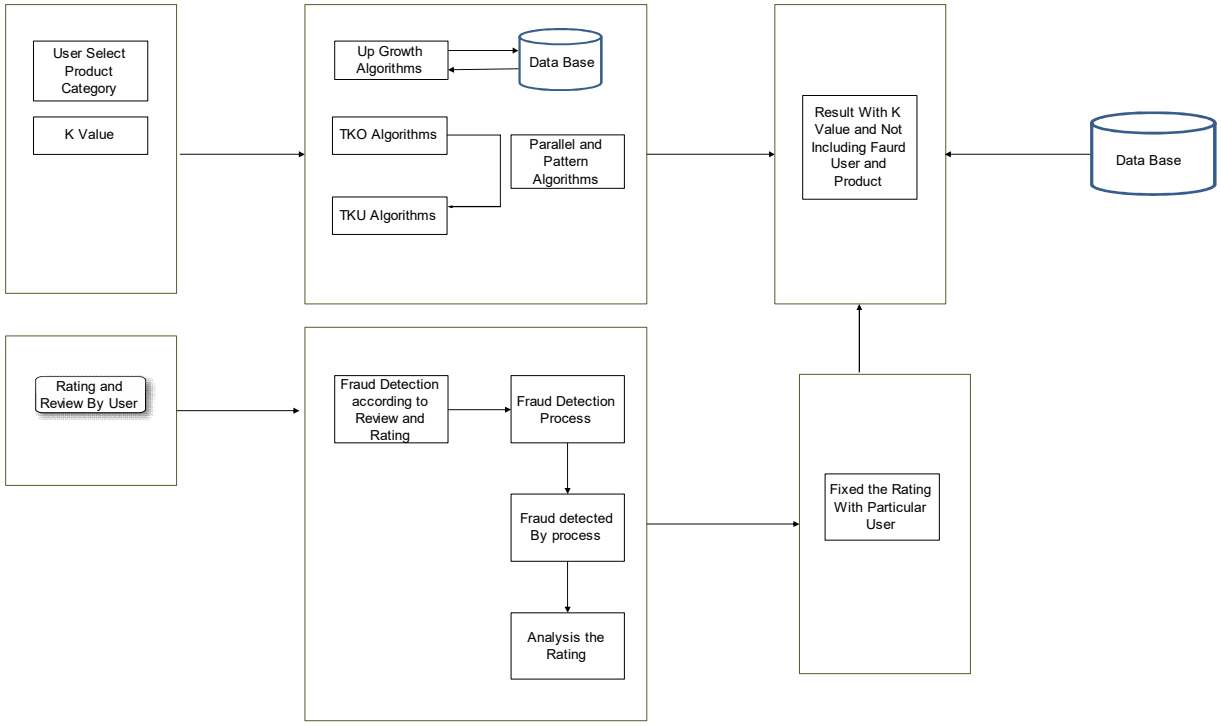
Figure 4.9 Deployment Diagram

It shows shows the execution architecture of the system,include hardware components like internet or software environment like web server and the middleware connecting them . they are used for representation of physical hardware and software of a system.

CHAPTER 5
PROPOSED SYSTEM

5.1 Proposed System

In the proposed framework, we address the problems mentioned above by proposing another system for calculating the means and means responsible for a high utility configured in parallel extraction using TKU and TKO. Two types of production calculations called TKU (extraction of sets of utility elements Top-K) and TKO (sets of themes of extraction Top-K are proposed in one phase) to extract these series of elements without the need to establish a utility minimum. But the TKO algorithm have the main disadvantage of not mainly accumulating the result of TKO given the value of the garbage in the set of high utility items isthe result of the TKU algorithm is increased but the execution time is high, so the alternative solution is to find the efficient algorithm in the proposed combination of the TKO and TKU algorithm system. It can be said that the result of TKO Top K in one phase is given at the entrance of TKU Top K in the utility result of TKO and TKU is increased and the execution time is low. In the proposed system, a new algorithm is generated for combining the name TKO and TKU as TKO WITH TKU or TKMHUI Top k Main set of utility elements.



Module:

Module 1 - Administrator (Admin)

The administrator preserve database of the transactions made by customers. In the daily market basis, each day a new product is let go, so that the administrator would add the product or items, and update the new product view the stock details.

Module 2 - User (Customer)

Customer can purchase the number of items. All the purchased items history is stored in the transaction database.

Module 3 - Construction of Up Tree

In Up Tree Dynamic Table is generated by algorithms. Mainly the Up growth is considerable to get the PHUI itemset.

Module 4 - TKO and TKU Algorithms

In Combination of TKO and TKU algorithms first the TKO (Top k in one phase) algorithms is called and then output of TKO is given as the input of TKU (Top k in utility phases) then the actual result is TKU Result.

Algorithms

TKO with TKU (Hybrid Algorithms)

Input: All HUI tree $T_s$and header tables Hs in the current window, an itemset based itemset (base –itemset is initialized as null), as list TKValueList, minimum utility value min_utility.

Output: TKHUIs

Begin

1. Find top-k maximal total utility value of itemset in Hs to TKValueList
2. Add a field add-information to each leaf-node
3. For each item Q in HL do from the last item of HL and HL is one HS
4. //Step 1:Calculate utility information of the node Q
5. Float twu=0,BU=0,SU=0,NU=0;
6. For each header table H in Hs do
7. For each node N for the item Q in the tree T corresponding to H do
8. BU+=T.N.bu;
9. SU+=T.N.su;
10. NU+=T.N.nu;// N.nu is a utility for item Q in the list N.piu
11. End For
12. twu=BU+SU

Step 2: Generate new itemset and create new sub tree and header table

13. If(twu>= minUti) then
14. base-itemset={Q};
15. create a sub HUI tree subT and a header table subH for base-itemset.
16. sub-Mining(SubT,SubTbase-itemset,TKValueList,min_uti);

17. Remove item Q from itemset base-itemset;
18. End if
19. // Step 3:Pass add-information on node Q to parent node
20. Move each node’s bac-info to its parent;
21. End For
22. Delete itemset whose value are less than minUti from TKHUIs;
23. Return TKHULS;
24. END

CHAPTER - 6
RESULT AND DISCUSSION

Table 6.1 TKO Algorithm with K Value

Number	Product Category	K Value	Time (milliseconds)
1.	Top Measures	1	65
2.	National	2	47
3.	Special	2	46

Table 6.2 TKU Algorithm with K Value

Number	Product catego-ry	K value	Time (milliseconds)
1.	Top Measures	1	246
2.	National	2	186
3.	Special	2	133

Table 6.3 Hybrid Algorithm with K Value

Number	Product category	K value	Time (milliseconds)
1.	Top Measures	1	230
2.	National	2	77
3.	Special	2	70

6.2 Discussion

Summary of Discussion as given below

1. Collection of High utility item Set (HUI) According to Selected parameter.
2. As a result, Top K Algorithm is applied in different data sets, such as data sets, such as mushrooms, chess and accidents with different k, respectively. In this graph TKUWITHTKO is the best performance among HUI top-k mining algorithms, in this TKO chart with TKU Spend 120 almost time-based for seconds to complete the mining process while REPT and TKU last longer than 160 seconds and TKO 70 seconds. In comparison, the TKO and TKU algorithms of the table are compared with various parameters, such as TWU and CHUD, and the Garbage Find values.
3. In below Table 1 as per the observations of the current status of project is given by the comparison between the Algorithms TKO, TKU and TKO with TKU with following parameter such as the Anti-monotonicity, fallow TUU high utility pattern, Closed high utility data set and the Most important point Garbage value generator and Minimum Utility Value.

As a result, Top K Algorithm is applied in different data sets, such as data sets, such as mushrooms, chess and accidents with different k, respectively. In this graph TKUWITHTKO is the best performance among HUI top-k mining algorithms, in this TKO chart with TKU Spend 120 almost time-based for seconds to complete the mining process while REPT and TKU last longer than 160 seconds and TKO 70 seconds. In comparison, the TKO and TKU algorithms of the table are compared with various parameters, such as TWU and CHUD, and the Garbage Find values.

In below Table 1 as per the observations of the current status of project is given by the comparison between the Algorithms TKO, TKU and TKO with TKU with following parameter such as the Anti-monotonicity, fallow TUU high utility pattern, Closed high utility data set and the Most important point Garbage value generator and Minimum Utility Value.

Table 6.4 Comparison between Algorithms

Sr. No.	**PARAMETER**	**TKO**	**TKU**	**TKO With TKU**
01	Anti-monotonicity	Yes	No	Yes
02	Fallow TWU high utility pattern	No	Yes	Yes
03	CHUD (Closet high utility dataset)	Yes	Yes	Yes
04	Finding Garbage Value in algorithm	Yes	No	No
05	Min until value	0	0	Set by Data set.

CHAPTER 7
CONCLUSION

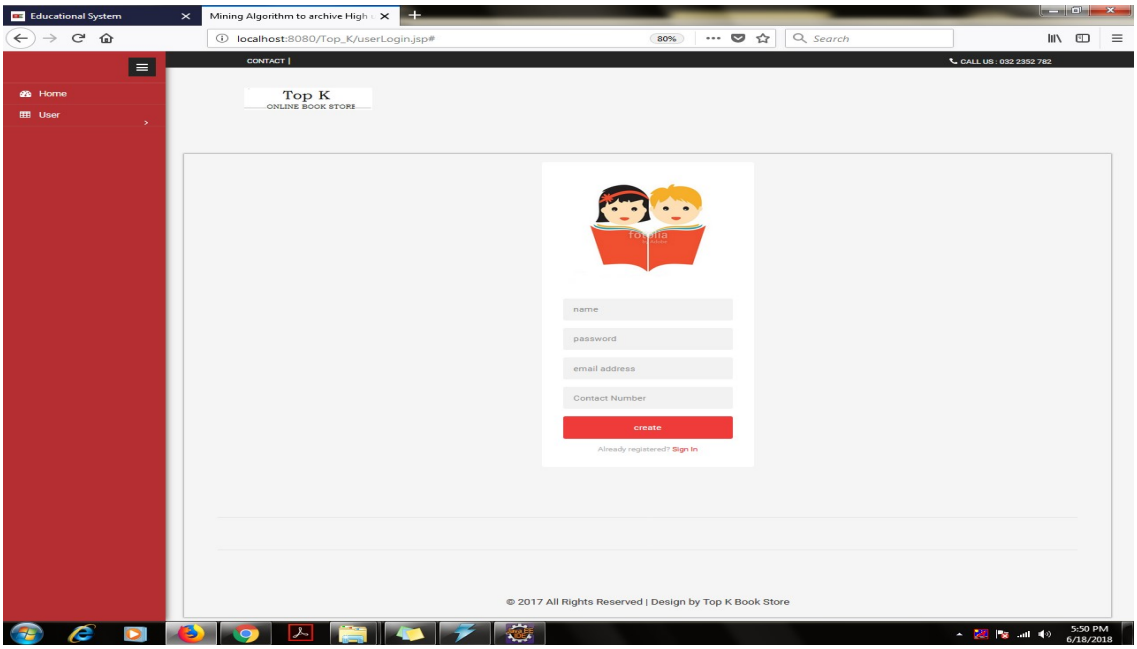
In this paper, we looked at the question of the best sets of high-use mining mines, where k is the coveted number of highly useful sets of things to extract. The most competent combination of TKO WITH TKU of the TKO and TKU calculations is proposed to extract such sets of objects without establishing utility limits. Instead TKO is the first single phase algorithm developed for top-k HUI mining called PHUI (high potential set of utility elements) and PHUI is given to TKU in the utility phases. Empirical evaluations on different types of real and synthetic data sets display the proposed algorithms have good scalability in large data sets and the performance of the proposed algorithms are close to the optimal case of the state of the combination of both phases in an algorithm.

REFERENCES

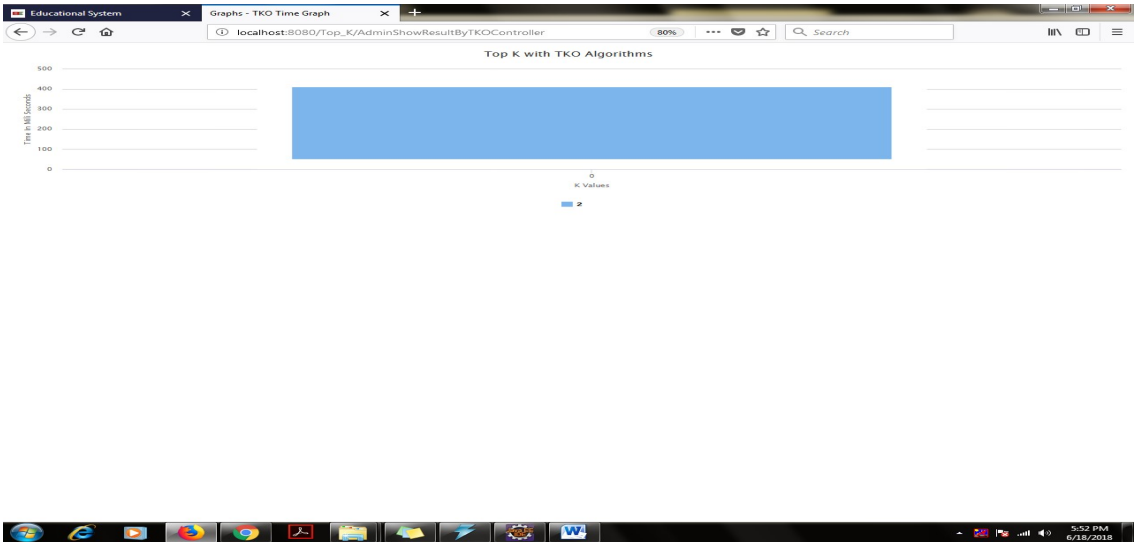
1. C. Ahmed, S. Tanbeer, B. Jeong, and Y. Lee, “Efficient tree structures for high-utility pattern mining in incremental databases,” IEEE Trans. Knowl. Data Eng., vol. 21, no. 12, pp. 1708–1721, Dec. 2009.
2. R. Chan, Q. Yang, and Y. Shen, “Mining high-utility itemsets,” in Proc. IEEE Int. Conf. Data Mining, 2003, pp. 19–26.
3. J. Han, J. Wang, Y. Lu, and P. Tzvetkov, “Mining top-k frequent closed patterns without minimum support,” in Proc. IEEE Int. Conf. Data Mining, 2002, pp. 211–218.
4. J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” in Proc. ACM SIGMOD Int. Conf. Manag. Data, 2000, pp. 1–12.
5. P. Fournier-Viger, C. Wu, and V. S. Tseng, “Novel concise representations of high utility itemsets using generator patterns,” in Proc. Int. Conf. Adv. Data Mining Appl. Lecture Notes Comput. Sci., 2014, vol. 8933, pp. 30–43.
6. P. Fournier-Viger and V. S. Tseng, “Mining top-k sequential rules,” in Proc. Int. Conf. Adv. Data Mining Appl., 2011, pp. 180–194.
7. J. Liu, K. Wang, and B. Fung, “Direct discovery of high utility itemsets without candidate generation,” in Proc. IEEE Int. Conf. Data Mining, 2012, pp. 984–989.
8. Y. Lin, C. Wu, and V. S. Tseng, “Mining high utility itemsets in big data,” in Proc. Int. Conf. Pacific-Asia Conf. Knowl. Discovery Data Mining, 2015, pp. 649–661.
9. Y. Li, J. Yeh, and C. Chang, “Isolated items discarding strategy for discovering high-utility itemsets,” Data Knowl. Eng., vol. 64, no. 1, pp. 198–217, 2008.
10. T. Quang, S. Oyanagi, and K. Yamazaki, “ExMiner: An efficient algorithm for mining top-k frequent patterns,” in Proc. Int. Conf. Adv. Data Mining Appl., 2006, pp. 436 – 447.

Appendix

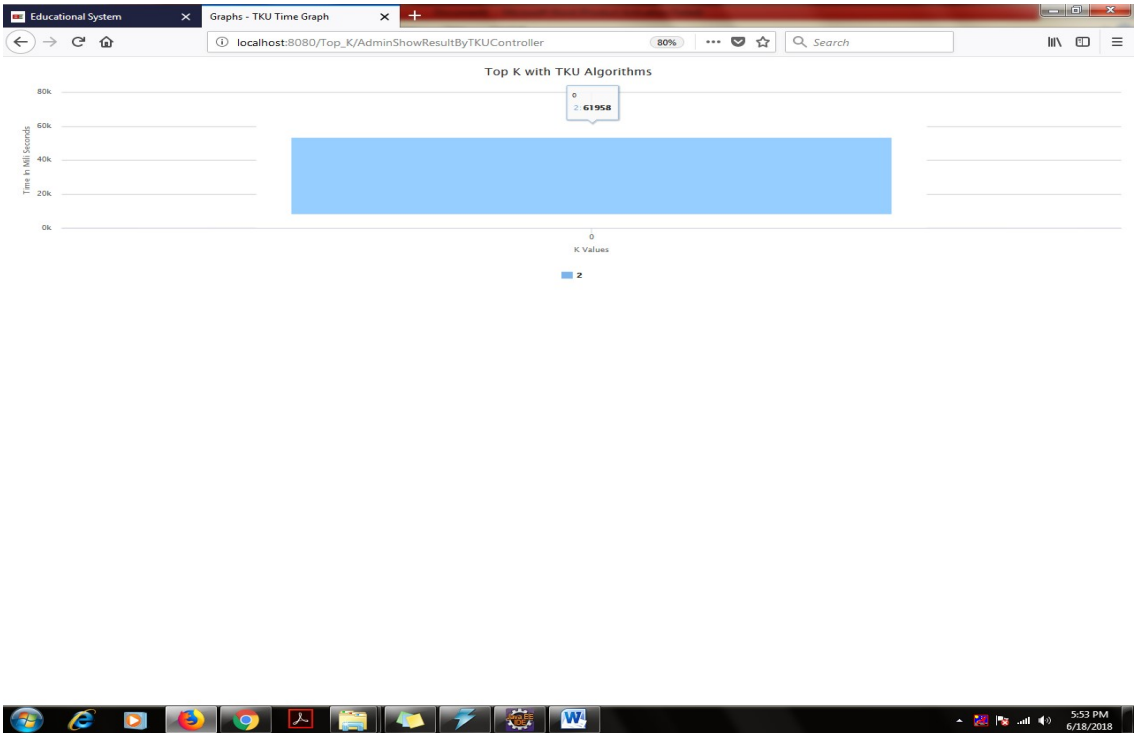
Screenshots



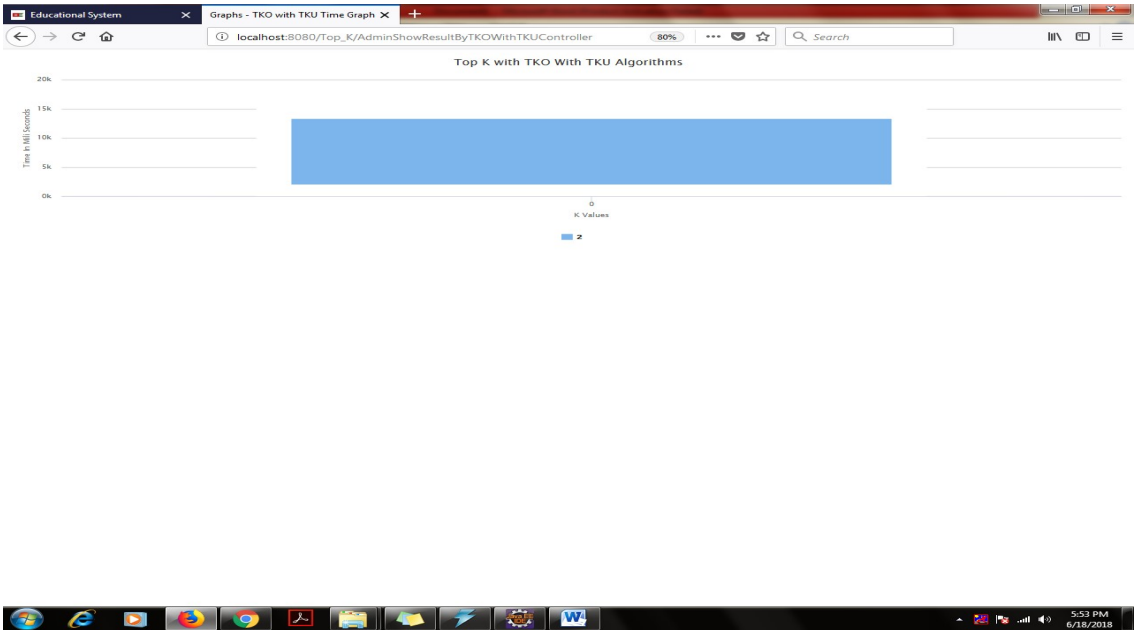
I. Login page



II. TKO Graph



III. TKU Graph



IV. Hybrid Graph