FREQUENT PATTERN MINING for EDUCATIONAL DATA by USING MAPREDUCE APPROACH in HADOOP

THAN HTIKE AUNG

UNIVERSITY OF COMPUTER STUDIES, YANGON

NOVEMBER, 2021

Frequent Pattern Mining for Educational Data By Using Mapreduce Approach In Hadoop

Than Htike Aung

University of Computer Studies, Yangon

A thesis submitted to the University of Computer Studies, Yangon in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

November, 2021

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research and has not been submitted for a higher degree to any other University or Institution.

…..…… ……..……

Date Than Htike Aung

ACKNOWLEDGEMENTS

First of all, I would like to thank the Minister, the Ministry of Science and Technology for full facilities support during the Ph.D. course at the University of Computer Studies, Yangon.

Secondly, I would like to express very special thanks to Dr. Mie Mie Khin, Rector, the University of Computer Studies, Yangon, for allowing me to develop this thesis and giving me general guidance during the period of my study.

I would like to express very special thanks to Dr. Mie Mie Thet Thwin, former Rector, the University of Computer Studies, Yangon, for allowing me to develop this thesis and giving me general guidance during the period of my study.

I would like to extend my special appreciation to Prof. Dr. Aung Nway Oo, professor the University of Information Technology for the useful comments, sharing knowledge, giving advice, and insight which are invaluable to me.

I would also like to express my respectful gratitude to Dr. Thin Lai Lai Thein, Professor, Dean of the Ph.D. 10th batch, the University of Computer Studies, Yangon, for her excellent guidance.

I would like to express my deepest gratitude to my former supervisors, Dr. Nang Saing Moon Kham, Professor, Faculty of Information Science, the University of Computer Studies, Yangon, for her patient supervision, tenderness, encouragement and providing me with excellent ideas throughout the study of this thesis. I will always remember her for being a mentor to me.

I would like to express my deepest gratitude to my supervisors, Dr. Yi Mon Thet, Associate Professor, Faculty of Information Science, the University of Computer Studies, Yangon, for her patient supervision, tenderness, encouragement and providing me with excellent ideas throughout the study of this thesis. I will always remember her for being a mentor to me.

I would also like to express my respectful gratitude to Dr. Khine Khine

Oo, Professor, former Ph.D. 10th batch teacher, the University of Computer Studies, Yangon, for her excellent guidance.

I would like to express my respectful gratitude to all my teachers for their encouragement and recommending the thesis. To the reading committee teachers, especially Daw Aye Aye Khine, Associate Professor, Head of English Department, I would like to thank her for valuable supports and editing my thesis from the language point of view.

I also thank my friends from the Ph.D.10th batch for their co-operation and encouragement.

Last but not least, I am very much indebted to my family for always believing in me, for their endless love and support. They are always supportive of me during my period of studies, especially for this Doctorate Course.

ABSTRACT

Lots of structured and unstructured information can provide educational insights through information retrieval. Gathering educational information in an effective way is also a major challenge. In addition, historical records can be obtained by effectively storing the information collected. Traditional analytical methods need to be expanded. The proposed method can effectively manage large volumes of data by combining apriori, prefix tree and eclat solutions. These techniques are reported to be more effective by integrating Hadoop and Mapreduce platforms. It also builds educational data collection software on Android mobile phones designed to improve education in the home country. This software can be accessed by adding a user account. A QR code is used to verify that you are a registered student or teacher.

In the education area of Myanmar, computers, mobile and internet have become important tools for high school students. To enable the quality and the flexibility of the education, verities of education programs and methods are greatly included but with different manners. Frequent itemset mining (FIM) is most popular technique in datamining area like health care, manufacture and finance. The proposed system develops two parts. The first part is implements teacher assessment survey application. Teacher assessment survey application is to collect educational data. The second part is introduced two FIM methods. These are Apriori Prefix Tree Eclat (ATE) and Eclat Prefix Tree (ET) based on Hadoop Mapreduce platform. Proposed method ATE is developed for large dataset to handle scalability and optimization. ET method is implemented to compare compile time to proposed method ATE. The proposed method is thinking instructor, student behavior and giving managerial decision support, analysis on teacher assessment survey data. The proposed method is for the decision maker. It helps to think about supporting student behavior and management decisions. It can be used to manage the frequent itemset results of the proposed method. It can effectively analyze large pieces of educational data in a timely manner. Using this proposed method can be used to effectively analyze the frequency and results of educational data in a short period of time. In addition, the proposed method is applicable to many universities. For university can also be used in management for individual teachers or for multiple teachers. It is particularly suitable for the analysis of large educational data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF EQUATIONS

CHAPTER 1
INTRODUCTION

Since developing countries have recently raised the production of the enormous amounts of data storage, the discovery of knowledge large data analysis has become more critical for educational research. The modern day cloud technology aids teacher best. When used precisely and appropriately information can be provided for an instructor with the wisdom and evidence required to help educational politics while upgrading classroom experiences for every student.

Explosion of the Internet, social media and impact of technological innovations can train a basic part of youthful lives and future race. Mobile devices used in Myanmar alone was over $30 million in 2019.

Various education standards are being developed in electronic learning and information technology and communications subjects among the preschool to higher level school. Consequently, electronic information data is increasing every year. To manage and maintenance storages required on every educational information to provide enhance educational purpose decision making. Educational data mining method is useful to get quality of education, useful knowledge, improve management and decision. Through the actual application of teacher assessment data of university of computer studies, Yangon in Myanmar, data amount grows and continues to increase. How to know students and teachers' requirements in universities classroom and how to improve the quality of universities status for effective management are interest of administrator.

Data Mining is essential for data analysis and finding knowledge for frequent pattern. Discovering frequent pattern was used in the previous time used with sample data mining program and algorithms. Their memory capacity and processing power of computers is limited to both software program and hardware. Unfortunately, this nature is limited at future tense of analysis for large data. Large data to frequent pattern mining application is required for educational datamining. Processing to large data not only needs large memory capacity and fast process power but also requires parallel or batch processing

frame work.

In the recent year, hadoop and mapreduce is a open source processing framework suitable for academic work such as universities used for education. This proposed system has used Hadoop processing frame work. Yearly produced teacher assessment data can be enlarged , and support that big data sets. Tradition data mining algorithm is modified to translate mapreduce algorithms run on hadoop. Hadoop is distributed framework and it can be stored vast data amount in distributed environment clusters. Map reduce works mapper and reducer two important tasks. Mapper maps input key/value pairs to a set of intermediate key/value pairs. Reducer reduces a set of intermediate values which share a key to a smaller set of values. MapReduce uses parallel computing approach and HDFS is fault tolerant system. Hadoop distributed file system is exploited to find out frequent itemsets.

1.1 Problem Definition

Now a day Big Data has bloomed in various area. Especially educational data on university of computer studies in Myanmar has grown every year. Most traditional data mining program working on typical datasets are not suitable for truly Big Data. Mining to large databases is problematic. Parallel programming deals with the massive amounts of data. Traditional techniques are not efficient performance.

Daily use of personal computer has memory and I/O cost limitations. Education survey data was required for computerized and analysis. Thus these data was stored on hadoop file system. The dataset size increases, it cannot be handled by traditional frequent item set mining algorithms. Eclat Prefix Tree method which is able to mine large datasets, but can be prohibitive when dealing with large data set. Counting the frequent itemsets can be expensive when use it. In order to decrease the number of itemsest that might have been missed. It has memory problem when data set is large. These conditions are major risk for large data set mining. Therefore currently data mining algorithms are needed to solve these limitations.

1.2 Terminology

Classification, clustering, and association analysis techniques have been already applied successfully in the educational domain. Find hidden meaning and patterns. Solve educationally-related problems. Uncover patterns and trends within large amounts of educational data.

1.3 Requirements

Throughout this work, the following objectives are pursued:

- To receive the teacher assessment's data from mobile phone using a Google Cloud Service.
- To register students and teachers ID to mobile phone applications.
- To communicate to students group section to its teacher for attempt survey questions.
- To store their survey question answer from their mobile application and sends to the Google Cloud storage.
- To solve yearly progress data in server side and download them to university data storage system.
- To support hadoop mapreduce framework
- To use education data in future analysis puporse
- To solve the performance of proposed approaches by comparing other existing daata mining method.
- To conclude the work an interpretation of the evaluation's results

1.4 Motivation of the Research

Frequent itemsets mining is a major field of study in data mining. The existing algorithms both classical and recently developed makes to determine the suitable and most efficient algorithm to use. The major challenge encountered in the frequent itemset mining is generation of large result set which is accompanied by huge memory consumption. If the threshold value is relatively low an exponentially large number of itemsets are generated and so the algorithm take up much more memory and take longer time for generation of the frequent itemsets.

The main motivation is to compare the performance of algorithms with the proposed algorithm which provided massive volume of educational data in

future to use.

Teacher Assessment survey application is developed to collect Education data set. Massive volume of educational data is collected to use Educational purpose. Frequent Pattern Mining on educational dataset is important.

1.5 Objectives of the Research

The main purpose of research is to get knowledge in open source of frequent itemset mining technique on Mapreduce platform. This research method is aim at performance of frequent itemset mining for large data. The teacher assessment survey system facilitate the data collection Computer Universities.

There are three main purposes. The first is to improve performance frequent itemset mining. Because conventional frequent itemset mining do not support big data well, the proposed frequent itemset mining is designed to be even better. The second is to investigate a frequent itemset mining that can be used with Mapreduce. Finally, to implement EDM (Education Data Mining) System for teacher assessment data.

1.6 Contributions of the Research

The main problem of the paper is that many of the methods used in frequent pattern mining are less efficient, so they have to replace the proposed method. The proposed method, the Apriori Prefix Tree Eclat and the MapReduce method, is combined on the Hadoop. This method is more efficient than the previous Eclat Prefix Tree method. In other words, the proposed method improves the weaknesses of the Eclat Prefix Tree method. The main problem of frequent pattern mining algorithm is that they become less efficient as they get bigger data volume. That is why this Apriori Prefix Tree Eclat method is proposed. Then there is the second problem. This is data collection. To collect large amounts of data, teacher assessment survey data in University of Computer Studies, Myanmar should be collected annually. Therefore, it is necessary to have a data collection program. The data collection program is designed for use with Android mobile phones. The data collection program is developed to be installed on teachers 'and students' mobile phones. The

program is divided into three access levels of the user account. These include students, teachers and managers. The data obtained at the initial planning stage is to be filtered vertically or in rows and stored on Hadoop file system storage device. The original study and analysis will take data from the Frequent Itemset Mining Implementations Repository and design and compare the performance of the two frequent itemset mining methods.

1.7 Organization of the Research

This dissertation is organized with six chapters. This chapter includes an introduction, the motivation of the thesis, the problem statements, objectives, focuses and contribution of the research work. Chapter 2 includes literature review and related work, case study of educational datamining. Chapter 3 includes history in data mining technologies, what is data mining, extraction of features, market basket model, frequent and maximal Itemsets, Association rules, interesting rules, Apriori algorithm, Eclat algorithm, FP growth algorithm, mapreduce, parallelization of frequent pattern mining algorithms. Chapter 4 includes Apriori Prefix Tree Eclat, frequent pattern mining with hadoop mapreduce. Chapter 5 includes , load balancing, program demonstration , data collection and program implementation, android based data Collection and program implementation, Apriori Prefix Tree Eclat program implementation, user program copy to remote host inline command statement, user program inline command parameters instruction, general format inline command for Apriori Prefix Tree Eclat (ATE) and Eclat prefix tree(ET) statement, user program(Apriori Prefix Tree Eclat) inline command statement, user program Eclat Prefix Tree (ET) inline command statement, General Format Inline Command for Trie data to frequent pattern file statement, user program Trie dumper inline command statement. Chapter 6 includes datasets, execution time results between ATE and ET method on dataset-1, execution time results between ATE and ET method on dataset-2, execution time results between ATE and ET nethod on dataset-1and dataset-2. Chapter 7 includes conclusion and future work.

CHAPTER 2
LITERTATURE REVIEW AND RELATED WORK

Distributed data is a process which obtained building of statistical data model to mining massive data ,L.Jure work the issue[2].

Knowledge discovery process produced many steps cleaning the data to presentation of knowledge which work by Gorunescu [3].

The FIM problem is considered as the root of the pattern mining field, which encompasses multiple tasks that aim at extracting itemsets on multiple forms and for various purposes.

The FIM issue is express as field of frequent pattern mining, which includes various activities aimed at collecting itemsets in various ways and for specific purpose (Aggarwal & Han, 2014) [5].

Simple method FIM is the extraction of itemsets. The anti-monotone FIM methods have been rejected.

The problem of frequent pattern mining in different ways of methods survey as rare pattern mining in dynamic or static data that frequent or infrequent (Koh & Ravana, 2016) [14].

Starting from it, the pattern mining concept was extended with new ideas (Aggarwal & Han, 2014), considering small variations in some cases, like the one considering an item within a pattern (either frequent or infrequent) as interesting not only if it is present (postive patterns) but also if it is absent (negative patterns) in data (Savasere, Omiecinski, & Navathe, 1998;

H. Wang, Zhang, & Chen, 2008). Of course, an item and its opposite form (positive or negative) always produce a zero support value since any record cannot satisfy both at the same time.

Sergio A. Alvarez described statistical method, such as Chi-squared method for datasets and association rule compute correlation between datasets [6].

The mapreduce programming model is become parallel and clustering the input data. Intermediate key value combining, large clustering, large amount of memory fit and load balancing work present by J.Dean and; S.G. jeff.[15]

Frequent pattern mining model used compress and storing data structure for large data based which not made candidate generation. J.Han , J. Pei and Y.

Yin work Mining Frequent Patterns with- out Candidate Generation [10].

Agarwal et al. used apriori algorithm for generating frequent itemsets and association rules. Level wise search. Frequency of an itemsets are counted by scanning the database D and then candidate k+1 itemsets are generated from frequent k-itemset by applying support and threshold condition.

Zaki et al. uses eclat algorithm which requires vertical database D needs to be stored in main memory. DistEclat developed by Zaki and Gouda store diffset(vertical data representation); diffset is difference between candidate itemset of size k and prefix frequent itemsets of size k-1; support value is based on diffset gaining performance growth than Eclat; it is not efficient when the database is sparse.

2.1 Case Study of Educational Data Mining

Table 2.1 Teacher Assessment Questionnaire

Attributes	**Description**	**Ordinal**
q1	The course content, course arrangement and assignments support course objectives.	Strongly Disagree=1, Disagree=2, Neutral=3, Agree=4, Strongly Agree=5
q2	The course materials that the instructor prepared are good enough to grasp the concept of the course chapters.	Strongly Disagree=6, Disagree=7, Neutral=8, Agree=9, Strongly Agree=10
----	-----	-------
q15	Overall, this course has been efficient to advance my learning.	Strongly Disagree=71, Disagree=72, Neutral=73, Agree=74, Strongly Agree=75

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
1	Timestamp	Course	Semester	InstructorsName	Section	q1	q2	q3	q4	q5	q6	q7	q8	q9	q10	q11	q12	q13	q14	q15
2	2018-08-19 14:22:18	First Year	1	U Mya	A	4	10	15	19	25	30	35	40	44	50	55	59	64	69	75
3	2018-08-19 14:30:14	First Year	1	U Mya	A	4	9	14	19	24	29	35	40	45	50	55	60	64	69	74
4	2018-08-19 14:39:04	First Year	1	U Mya	A	4	10	15	20	25	30	35	40	45	50	55	60	65	70	75
5	2018-08-19 14:56:42	First Year	1	U Mya	A	4	10	15	20	25	30	35	40	45	50	55	60	65	70	75
6	2018-08-19 14:57:38	First Year	1	U Mya	A	4	10	15	20	25	30	35	40	45	50	55	60	65	70	75
7	2018-08-19 14:58:39	First Year	1	U Mya	A	3	8	14	17	24	29	33	39	44	47	55	59	64	69	74
8	2018-08-19 15:04:54	First Year	1	U Mya	A	5	10	15	20	25	29	34	39	44	49	54	60	65	69	75
9	2018-08-19 15:09:58	First Year	1	U Mya	A	5	10	15	20	25	30	35	40	45	50	55	60	65	70	75
10	2018-08-19 15:50:38	First Year	1	U Mya	A	5	9	15	19	24	30	35	40	44	49	55	60	65	70	74
11	2018-08-19 15:57:43	First Year	1	U Mya	A	1	8	14	20	24	29	34	40	43	48	51	57	64	66	72
12	2018-08-20 9:33:12	First Year	1	U Mya	A	1	8	14	20	24	29	34	40	43	48	51	57	64	66	72
13	2018-08-20 18:45:54	First Year	1	U Mya	A	1	8	14	20	24	29	34	40	43	48	51	57	64	66	72
14	2018-08-20 18:47:38	First Year	1	U Mya	A	1	8	14	20	24	29	34	40	43	48	51	57	64	66	72
15	2018-08-20 18:49:27	First Year	1	U Mya	A	1	8	14	20	24	29	34	40	43	48	51	57	64	66	72

Figure 2.1 Teacher Assessment Survey Database

```
11 12 13 14 15 18 19 20 21 22 40 43
23 24 25 44
7 31 32 33 41 42 45 57 58
2 3 4 5 6 26 27 34 35 38 46 47 48 49 50 51 52 54 56 59 60
8 9 10 16 17 28 29 30 36 37 53 55 61
18 19 20 21 22 40 43
23 24 25 44
7 11 12 13 14 15 31 32 34 35 41 42 45 50 51 54 60
3 10 26 27 36 38 39 46 47 49 52 53 55 56 57 58 59
2 4 5 6 8 9 16 17 28 29 30 33 37 61
18 19 20 21 22 40
23 24 25 43 44
31 32 41 45 60
3 7 11 12 13 14 15 26 27 36 38 39 42 46 48 49 50 51 52 53 54 58
59
2 4 5 6 8 9 10 16 17 28 29 30 33 34 35 37 47 55 56 57
18 19 20 21 22 40 44
7 23 24 25 31 32 43
35 36 41 45 50 52 53 54 59 60
2 3 10 26 27 33 34 37 38 39 42 46 48 49 51 55 56 58
4 5 6 8 9 11 12 13 14 15 16 17 28 29 30 47 61
18 19 20 21 22 40 43
23 24 25 34 57
31 32 41 45 52
3 7 10 11 12 13 14 15 26 27 33 36 38 39 42 44 46 48 49 50 51 54
55 56 58 59 60
2 4 5 6 8 9 16 17 28 29 30 35 37 47 53 61
18 19 20 21 22 40 43
23 24 25
```

Figure 2.2 Teacher Assessment Survey Data Set

Table 2.2 Teacher Assessment Condition and Action

No	Condition	Action	Support	Confidence	lift
1	∀X∈student,Q13(X,agree) ∧Q15(X,agree) =>	Q14(X,agree)	0.3	1.0	2.5
2	∀X∈student,Q12(X,agree) ∧Q13(X,agree) =>	Q14(X,agree)	0.3	1.0	2.5

3	∀X∈student,Q12(X,agree) ∧Q13(X,agree) ∧Q15(X,agree) =>	Q14(X,agree)	0.3	1.0	2.5
4	∀X∈student,Q12(X,agree) ∧Q15(X,agree) =>	Q11(X,agree)	0.3	0.9	2.4
5	∀X∈student,Q5(X,agree) ∧Q6(X,agree) =>	Q3(X,agree)	0.3	0.9	2.4
6	∀X∈student,Q12(X,agree) ∧Q14(X,agree) ∧Q15(X,agree) =>	Q11(X,agree)	0.3	0.9	2.4
7	∀X∈student,Q11(X,agree) ∧Q15(X,agree) =>	Q12(X,agree)	0.3	1.0	2.4
8	∀X∈student,Q11(X,agree) ∧Q12(X,agree) =>	Q15(X,agree)	0.3	1.0	2.4
9	∀X∈student,Q11(X,agree) ∧Q14(X,agree) ∧Q15(X,agree) =>	Q12(X,agree)	0.3	1.0	2.4
10	∀X∈student,Q11(X,agree) ∧Q12(X,agree) ∧Q14(X,agree) =>	Q15(X,agree)	0.3	1.0	2.4

Table 2.3 Teacher Assessment Transactions

Transaction id (t)	**Items**
t1	4,10,15,19,25,30,35,40,44,50,55,59,64,69,75
t2	4,9,14,19,24,29,35,40,45,50,55,60,64,69,74
t3	4,10,15,19,25,30,35,40,44,50,55,59,64,69,75
t4	4,10,15,20,25,30,35,40,44,50,55,60,65,70,75
t5	4,10,15,19,25,30,35,40,44,50,55,59,64,69,75

- ***example number [1]***

  "The key points of this course can be understandable"=**agree** *and* "Overall, this course has been efficient to advance my learning"=**agree** *then* "Overall, the effectiveness of the instructor is good"=**agree**

- ***example number [3]***

  "This course enhances my critical thinking skill about the subject"=**agree** *and* "The key points of this course can be understandable"=**agree** *and* "Overall, this course has been efficient to advance my learning"=**agree** *then* "Overall, the effectiveness of the instructor is good"=**agree**

- ***example number [5]***

  "The instructor uses inter active teaching Students' participation and questions"=**agree** *and*

  "The lecturer and course assignments are rightly matched"=**agree** *then* "The instructor clearly explains difficult materials"=**agree**

2.2 Case Study of Data Collection

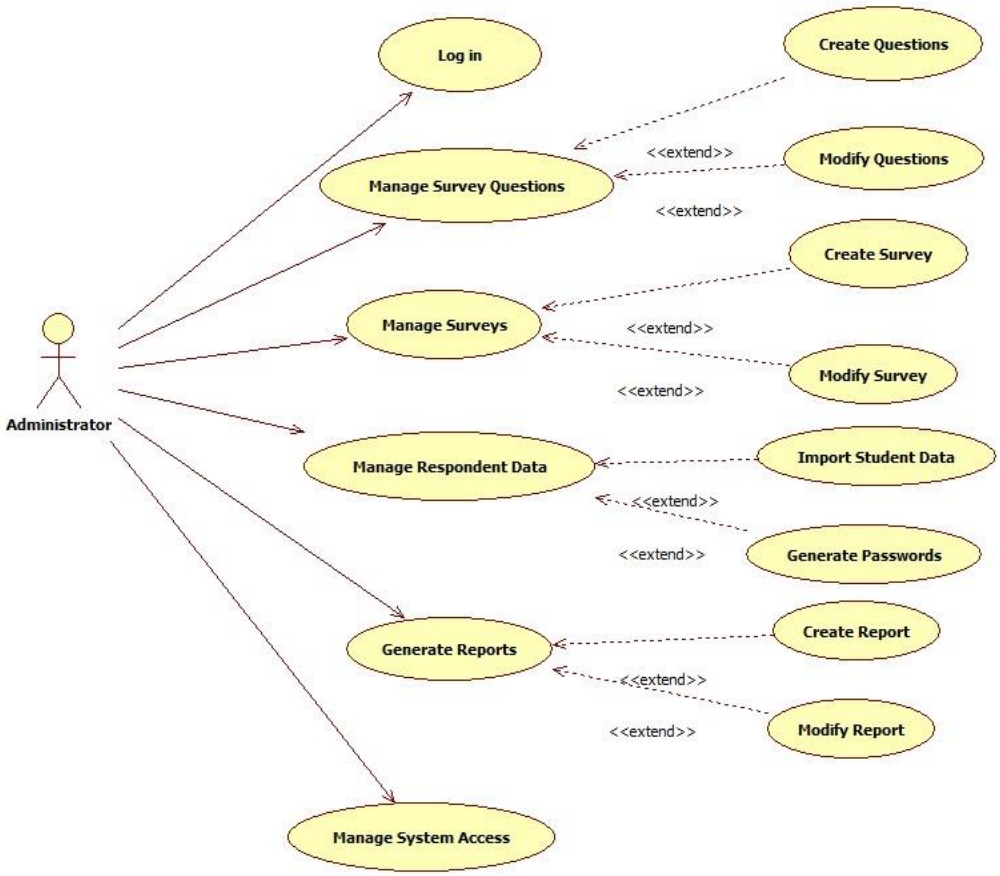

Figure 2.3 Use Case Diagram For Administrator(Teacher)

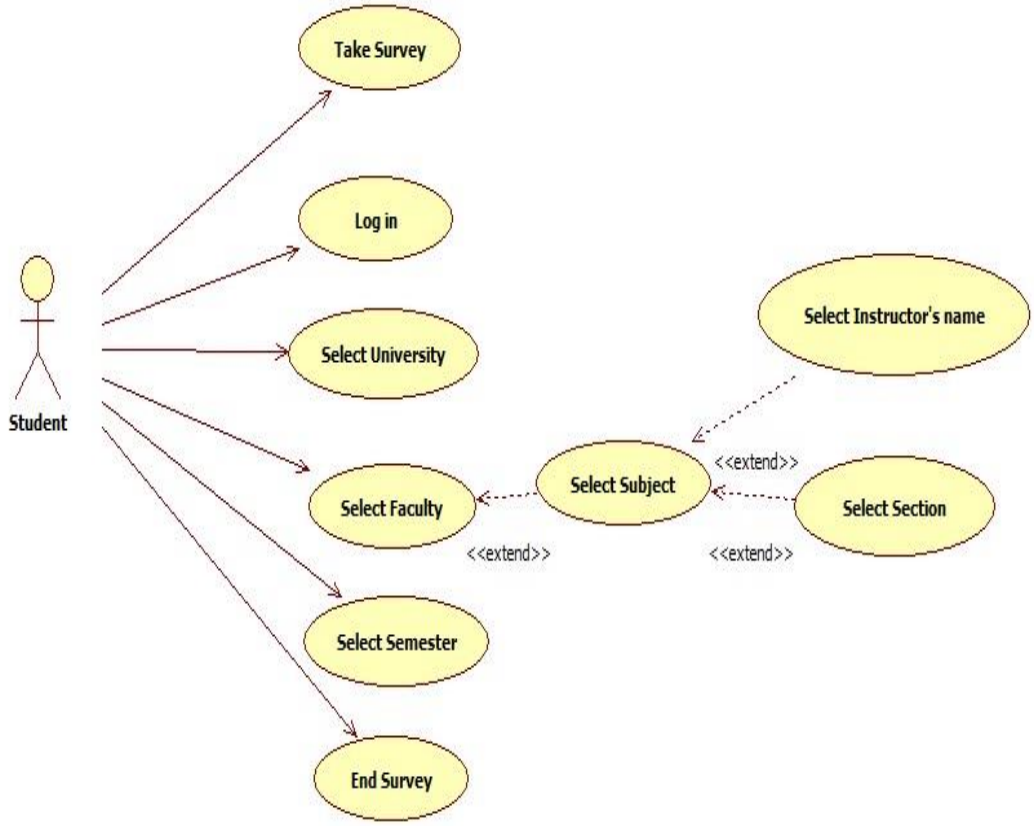

Figure 2.4 Use Case Diagram For Administrator

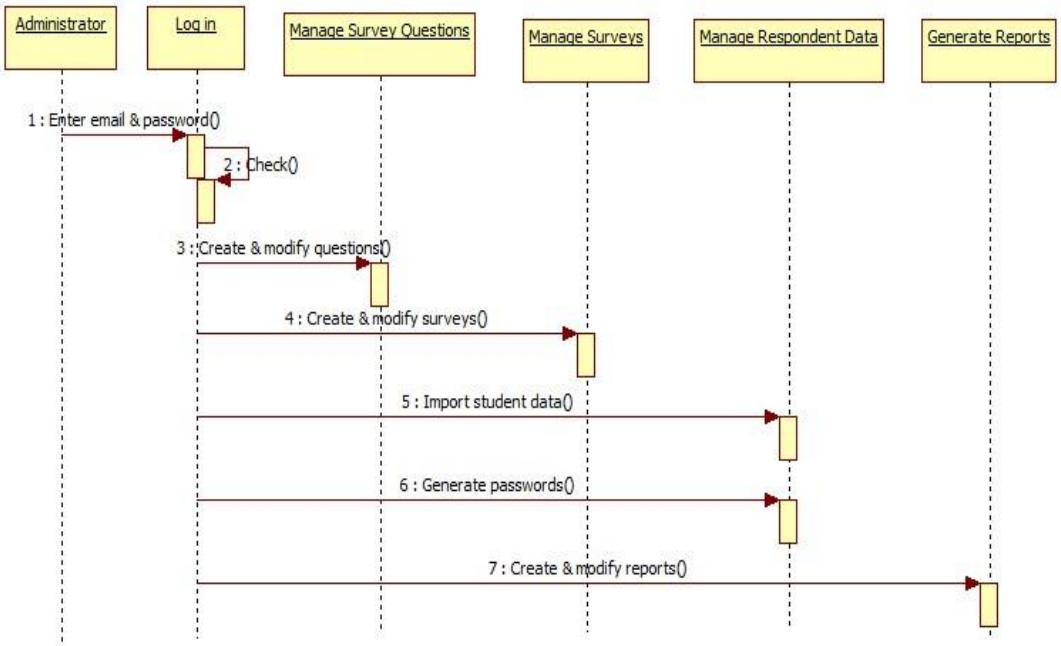

Figure 2.5 Use Case Diagram For Student

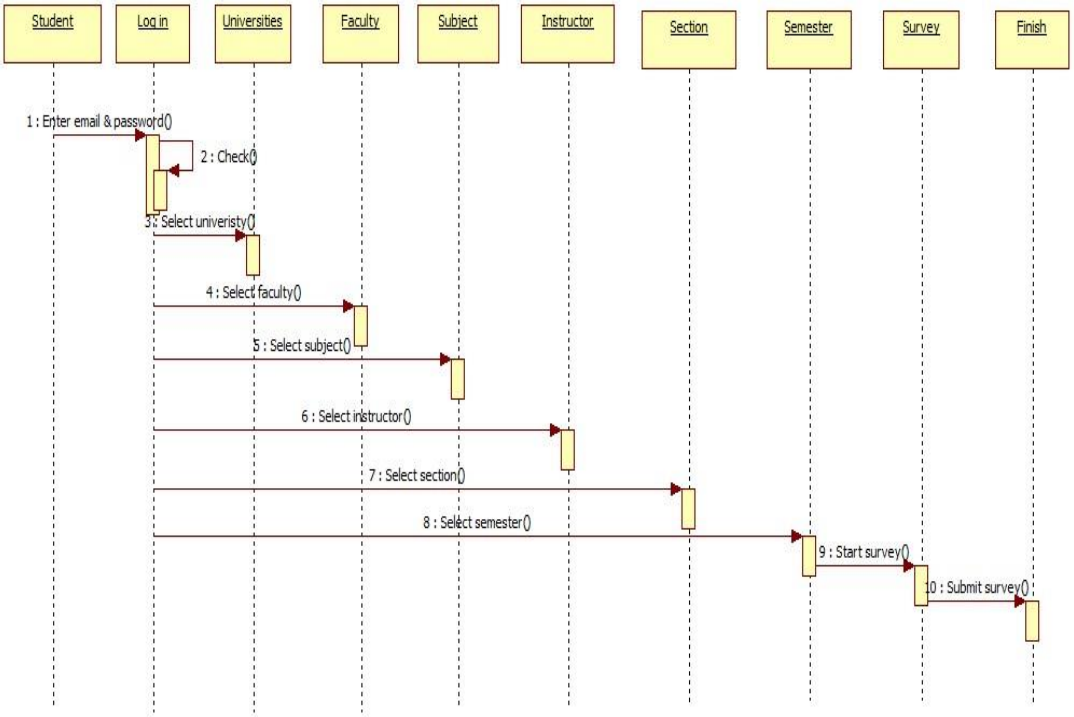

Figure 2.6 Sequence Diagram For Administrator(Teacher)

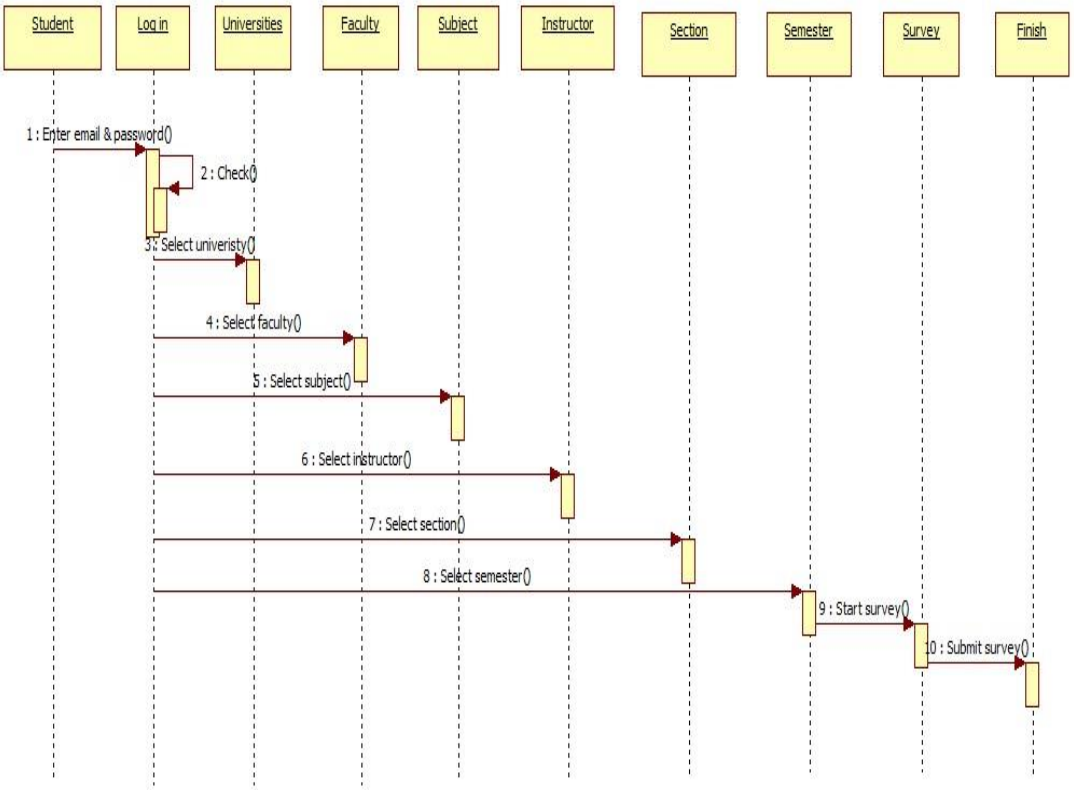

Figure 2.7 Sequence Diagram For Student

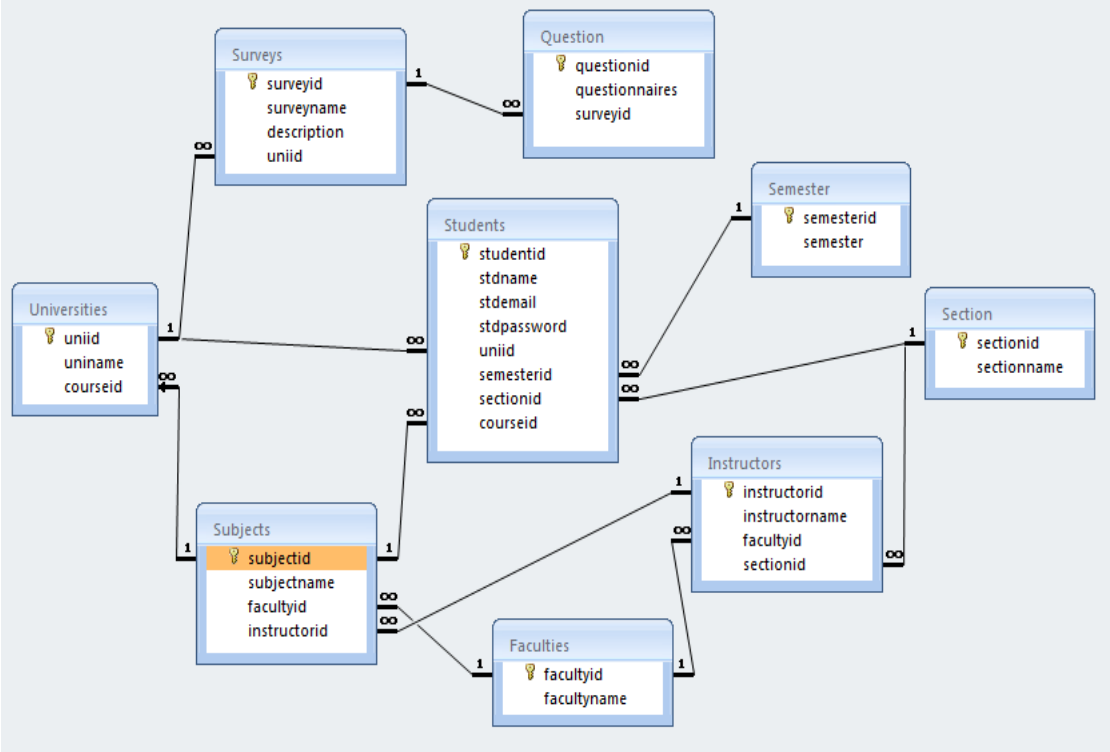

Figure 2.8 Teacher Assessment Survey Database Design

2.3 Chapter Summary

This chapter includes literature review and Related Work and case study of educational datamining. The previous researchers described large data mining model , frequent pattern mining , Mapreduce programming model and problem issue of frequent itemset on large dataset. Case study on educational datamining discuss teacher assessment questionnaire, survey data stored on teacher assessment database and teacher assessment datasets. Finally, This chapter discusses teacher assessment condition and action, transaction and discuss examples of rule mining on teacher assessment datasets. This chapter also include Data Collection of Teacher Assessment Survey for use case diagram, sequence diagram and database design. In the next chapter, this research will describe the data mining technologies and knowledge discovery process steps and Mapreduce framework.

CHAPTER 3
THEORETICAL BACKGROUND

Data mining has many methods that applies to large and complex database. Knowledge discovery process use these methods to eliminate types of errors and discover the hidden pattern. There are many methods driving force on big data analysis. Data mining technologies is most important field of study.

3.1 History in Data Mining Technologies

Historical point of view, there are three crucial points in the development of information. First of them was around 3100 BC in ancient Sumer where independent writing system was invented. The second crucial point ~~is~~ connected with Johannes Gutenberg who invented printing press in 1439. The last point denotes current period which is also called as an information age. The information age is characterized by the shift from industrial production to one based on information and computerization. Current economy, social relation, politics is based on information technology and processing of a big amount of data.[1] With the rise of Internet usage in last decades, amount of produced information gets bigger every year. It implies that new approaches to the effective processing of such amount of information had to be invented.

3.2 What is data mining?

Data mining term was first used by statisticians. Today, it is used with the process of building a statistical model, which describes distribution from which the data are obtained [2]. Knowing the model is essential to make decisions or to be able to make predictions about the data.

Regarding computer science, data mining can be characterized as “the process of discovering patterns and knowledge from large amounts of data”[3].

A similar term is a knowledge discovery from data. This process consists of several steps [3]:

1. Data cleaning (to remove inconsistent data
2. Data integration (to combine multiple data sources)
3. Data selection (to select data relevant to analysis)

4. Data transformation (to transform data into a form suitable for analysis)
5. Data mining (to extract data patterns)
6. Pattern evaluation (to select interesting patterns)
7. Knowledge presentation (to visually present knowledge to users)

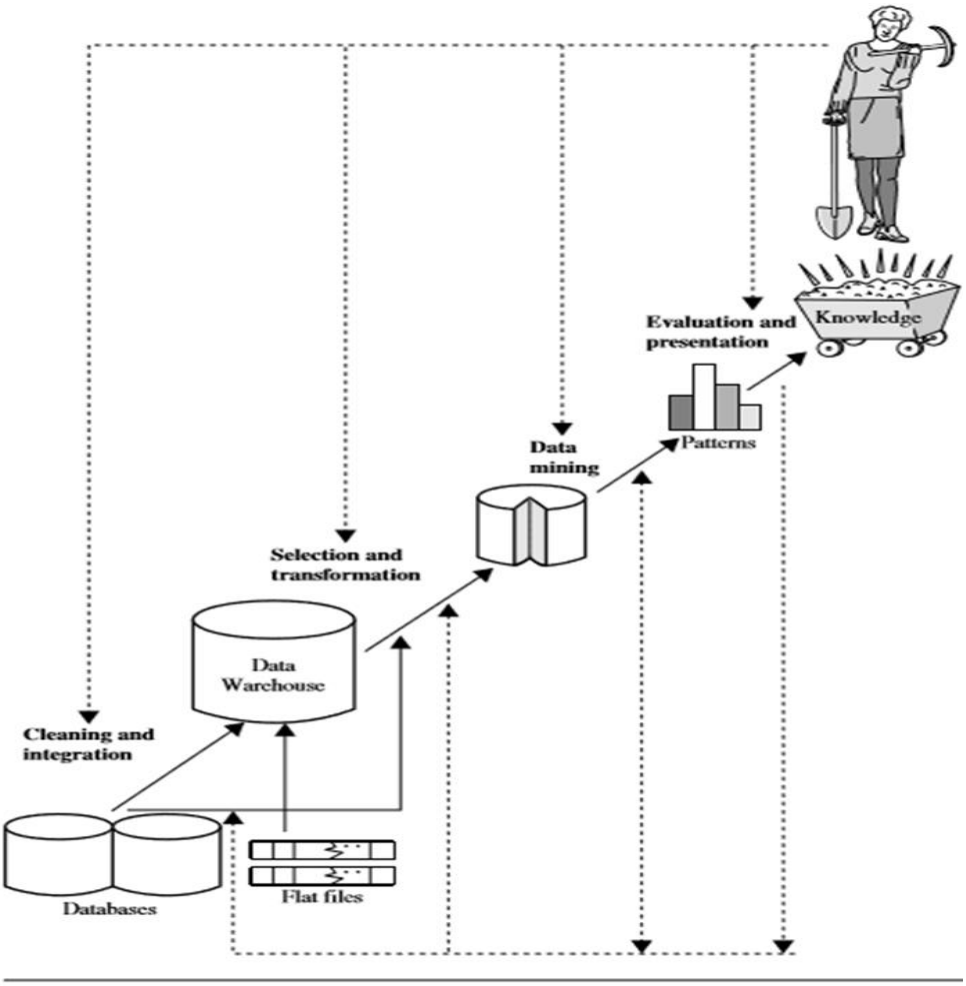

Figure 3.1 Data Mining as a Step in the Process of Knowledge Discovery

Data mining is the key step in this knowledge mining process. Steps before it can be described as a form of preprocessing of data and in steps six and seven mined knowledge is handled (presented or stored). That's why this whole process is sometimes interchanged with data mining term [3].

Knowledge discovery is used for obtaining different new information. Different attitudes on how to obtain, it exists. Models describing a data can be created using a summarization, where a whole object is represented by some summary. Or another approach is an extraction of important pieces from the

data [2], which in fact is some specific form o summarization.

3.3 Extraction of Features

When analyzing data, the model could be interested in some concrete features. Part of creating the data model is a feature extraction from data. Subsequently, for querying about the model only extracted features are used.[2]

The finding of similar items is one example of important pieces extraction. Frequent pattern mining also belongs into this category, and it is described in later chapters.

3.4 Market Basket Model

One of the first usages of finding frequent patterns was an analysis of records, what customers usually buy in a store chain. Data for this analysis consists of baskets and items. Items represent things or services, which it's possible to buy in the chain. Baskets contain several items and indicate that they were bought together. This relationship is called market-basket model, and generally, it's a many-to-many relationship.[2]

The purpose of such analysis is to know which products are often bought together. This knowledge allows different marketing strategies. One kind of goods can be made cheaper to attract people to come to the shop. And other items often bought with it can be raised.[3]

Generally, this model is sometimes explained using transaction databases, where transactions are analyzed. So transactions take a role of a basket. And the results of analysis are sets of items which are in many transactions. In this text both terms (baskets and transactions) are used.

3.5 Frequent and Maximal Itemsets

As was said before, baskets consist of items. A frequent itemset is set of items, which are often together. Support threshold is a number defining in how many baskets an itemset has to be to be considered frequent. It could be stated as the number of transactions or as a fraction (or percentage) of the total number of transactions. The support of the itemset is the number in how many baskets the itemset is.[4]

- Frequent itemset A is maximal if there is no other frequent

itemset B such that A ⊂ B.[3]

- A closed frequent itemset is an itemset, which support isn't the same as any of its supersets.[3]

3.6 Association Rules

Association rules describe a relationship between two frequent itemsets.

Let's say A ⇒ B is association rule, where A and B are frequent itemsets. If A is in the basket, it says that B is in there too. Frequent patterns for the same support threshold are used for discovering association rules.

Confidence is an indication how often is the rule true. It says what is the probability that B will be in the basket if A is in it.[4]

$$conf\,(A \Rightarrow B) = support(A \cup B)/support(A) \quad (3.1)$$

The basic principle of frequent itemsets is that every subset of it is also frequent. So from frequent itemsets association rules can be easily generated. For frequent itemset I generate all its proper subsets. Then for every subset s

$$s \Rightarrow (I \setminus s) \quad (3.2)$$

is association rule which passes minimal support threshold (it was required when finding frequent itemsets).[2]

3.7 Interesting Rules

Selection of interesting rules is subjective and depends on the business use of the mined knowledge. So mostly, people can decide it, but to be able to manage it in relatively short time, the number of rules to go through, have to be small.

First criteria to select potentially interesting rules is their confidence. So generally, confidence has to be higher than our specified threshold. This filtering selects rules, which are with some probability true, but it doesn't mean that presence of first itemset increase the chance that second itemset will be in transaction too.[3]

But support and confidence have some disadvantages in the filtering of interesting rules. It is because these measurements are not normalized. So even they could be relatively big, for rarely occurring items provided information is

not accurate.[4]

Correlation of the rule can be also used to filter out uninteresting rules. For rule A ⇒ B it defines a correlation between the two itemsets.

Different correlation measurements are possible.

$$lift\ (A\ \Rightarrow\ B)\ = \frac{support\ (A \cup B)}{support(A) \times support(B)} \quad (3.3)$$

For independent itemsets lift is equal to 1. Compared to change that B is in the basket, positive lift (bigger than 1) means, that the occurrence of A in the basket increases the chance B will be in it too. Negative lift indicates that if A is in a basket, it's less likely that B will be bought as well.[3]

Another more sophisticated method of measuring correlation is, for example, $\chi 2$ statistical test. It can be computed by following formula[5].

$$\chi^2(A\ \Rightarrow\ B)\ = n(lift\ -\ 1)^2 \frac{support\ \ confidence}{(conf - support)(lift - conf\ )} \quad (3.4)$$

3.8 Apriory Algorithm

The A-priory algorithm was developed by R. Agrawal and R. Srikant in 1994[6]. It belongs to the family of join-based algorithms. They used level-wise approach for finding frequent itemsets. It means that for every k is true, that all frequent itemsets of length k are found before frequent itemsets o length (k + 1).

To find frequent (k + 1) itemsets the candidate itemsets are found first. An itemset is a candidate if at least two of its subsets are frequent, but it's not known that all of them are (which is the condition to be a frequent itemset).[4]

The correctness of this algorithm is based on a principle called monotonicity of itemsets, which says that if an itemset is frequent, then every its subset is frequent too. This property of frequent itemsets is also called downward closure property.[2]

A simple approach to generating candidate itemsets could be to pick a pair of itemsets and join them. For example, to get candidate set of length k + 1, frequent itemsets of length k and length 1 are joined. This approach is used by an older algorithm called AIS (authored by Agrawal, Imielinski and Swamiin 1993[7]).

The A-priory algorithm optimizes a process of generating of can- didate itemsets. To create a candidate of length k + 1 two candidates of length k are used. So to get length k + 1, just k − 1 items have to be same in both

itemsets.[3] Items in itemsets are ordered to keep generating of candidates simpler. It means that to check that k − 1 items in an itemset are the same, sequential comparing of first k − 1items can be used.

Algorithm 1 ([4]) shows how Apriory algorithm works. Generating of frequent 1-itemsets on line 1 is very simple. All transactions in T are read and for every item is counted how many times it was read. Items which were read more than (or equal) s times creates 1-itemsets F_1.

Candidate itemsets are generated by using the technique described above. In case of C_2, there is no difference because candidates are all pairs of 1-itemsets from F_1. For larger itemsets, this process is more complicated. To avoid generating the same candidate itemset multiple times candidate ck+1 is created from two k-itemsets only if the largest item in the first one is smaller than the largest item in the second one.[6]

Algorithm 1: A-priory algorithm
Data: T - transactions, t - support threshold
Result: all frequent itemsets - $F_1,...,F_k$
1 Generate frequent 1-itemsets F_1 (support $\geq t$);
2 k := 1 ;
3 **while** F_k *isn't empty* **do**
4 Generate C_{k+1} candidates from F_k ;
5 Remove from C_{k+1} itemsets, which have any infrequent subset ;
6 Count support for C_{k+1} in T ;
7 F_{k+1} := itemsets from C_{k+1} with support $\geq t$;
8 k := k+1 ;
9 **end**

Removing of candidates which could not be frequent, because all its subsets aren't frequent, is called pruning[6]. All subsets of candidate itemsets don't have to be checked to do this. The A-priory algorithm uses the fact that frequent itemsets shorter by one are known. So for candidate k-itemset f_k, only subsets of length k − 1 have to be checked that are in F_{k-1}. That's because all

shorter subsets are also subsets of k − 1 subsets. So if k − 1 subsets are known to be frequent, all their subsets need to be frequent too.

Counting of support for candidate itemsets in basic A-priory al- gorithm doesn't use any "trick" to improve the efficiency. So for each transaction all candidates itemsets are tested if they are subsets of the transaction.

The Apriory algorithm has good performance for short, frequent patterns in sparse datasets (such as the market-basket model). For dense datasets with long, frequent patterns the performance degrades. It is due to the fact that it reads the database as many times as the length of the largest frequent itemset.[8]

3.9 Eclat Algorithm

Eclat (Equivalence Class Transformation) algorithm, which is developed by Mohammed J. Zaki in 1997 [11], uses a different approach to represent transaction database.

In this format, transactions are stored i the form of TID-itemset records. It means that each record contains the trans- action ID (TID) and set of items included in the transaction. Unlike them, Eclat uses vertical data format where records are stored in item- TID_set form. It means that each record contains item name and set of transaction IDs (also called tidlist), where the item is contained.[3]

Figure 3.2 shows a difference between formats of data mentioned above (same transactions are used in both cases).

Vertical representation of data allows straight counting of itemset support. The size of the intersection of tidlists of items in the itemset is a support of that itemset.[4] The intersection of sets could be very fast, and for sorted itemsets, it can be done in time linear to the size of intersected operands.

Instead of doing k − 1 intersections to compute tidlist for k-itemset, the same trick as in case of the A-priory algorithm can be used – to gen- erate candidate of length k two frequent (k − 1)-itemsets can be used.

Same can be done to compute support of itemset using the intersection of tidlists. The intersection of two tidlists of (k − 1)-itemsets to get the support of k-itemset is more efficient than using of k intersections.[4] On the other hand, memory has to be big enough to keep computed tidlists for larger

itemsets. Different variations of Eclat traverses the lexicographical tree in different ways. Both depth-first and breadth-first traversals are possible.[4]

Original work [11] describes traversing of candidates using equivalence class clustering. It is prefix based equivalence relationship. So for each k two itemsets are in the same class if they share the same prefix of length k. Such clustering is used to describe the fact that traversing of the lattice of candidates can be divided into independent problems and solved separately (which is beneficial in case there is not enough memory to keep all intermediate tidlists).[12] And it is also used to generate potential maximal itemsets.[11]

Vertical format

'I1': {'T100', 'T400', 'T500', 'T700', 'T800', 'T900'}
'I2': {'T100', 'T200', 'T300', 'T400', 'T600', 'T800', 'T900'}
'I3': {'T300', 'T500', 'T600', 'T700', 'T800', 'T900'} 'I4': {'T200', 'T400'}
'I5': {'T100', 'T800'}

Horizontal format

'T100': {'I1', 'I2', 'I5'}
'T200': {'I2', 'I4'}
'T300': {'I2', 'I3'}
'T400': {'I1', 'I2', 'I4'}
'T500': {'I1', 'I3'}
'T600': {'I2', 'I3'}
'T700': {'I1', 'I3'}'
'T800': {'I1', 'I2', 'I3', 'I5'}
'T900': {'I1', 'I2', 'I3'}

Figure 3.2 Example of Transactions in Horizontal and Vertical Format Data [3]

Vertical mining: Eclat algorithm. The algorithm called Eclat (Equivalence Class Transformation) employs prefix-based classes to reduce the search space.

Algorithm 2 Eclat (F_k ,t)

Data: F_k - frequent patterns of size k, t - support threshold

Result: FP - frequent patterns

1 **forall** I_i *in* F_k **do**

2 $F_i := \{\}$;

3 **forall** I_j *in* F_k, *where* I_i *has the same* $(k-1)$-*prefix and* $I_j > I_i$ **do**

4 $I_{ij} := I_i \cup I_j$;

5 $tidset(I_{ij}) := tidset(I_i) \cap tidset(I_j)$;

6 **if** $tidset(I_{ij}) > t$ **then**

7 **Add** I_{ij} into F_i;

8 **end**

9 **end**

10 $FP := FP \cup F_i$;

11 Eclat(F_i, s) ;

12 Include in FP all itemsets returned by previous step ;

13 **end**

Each of the equivalence class can be matched to a subtree of the lexicographical tree. So, in fact, using the equivalence class clustering is the same as using a lexicographical tree. Depth first exploration of the lexicographical tree is used in this implementation. Also, new frequent patterns are generated in the same way as in case of Apriory two frequent (k − 1)-itemsets sharing the same prefix are joined to create a pattern of length k. Pruning of candidate itemsets is not necessary, because the computation of support for the candidate means to perform one itemset intersection.

In line 5 of algorithm 5 computation of transaction id list (tidlist) for new itemset is done. As mentioned above implementation decision about this has to be made if to compute tidlists in every iteration by intersecting tidlists of contained items or if to store tidlists for itemsets in Fk in memory to speed up computation. In the second case, only tidlists for itemsets in the path to the

root from the current node in the lexicographical tree has to be kept on memory. This choice is not explicitly stated in the pseudo code so tidset() for itemsets on the right side of ":=" statement could mean accessing of tidlists in a global structure (or structure passed on each recursive call) or computing of tidlist from individual items in the itemset.

To improve the memory usage when storing tidlists for unfinished nodes in the tree, Mohammed J. Zaki and Karam Gouda in [8] introduced new data representation called diffset. With this approach, only differences in TIDs from tidlists of itemsets which are joined to generate new itemset has to be kept for the newly generated itemset.

The prefix tree (or trie) is a data structure with fast ordered tree used for retrieval (Navarro et al., 2002). The prefix tree stores data items with index from the starting of a string (i.e. prefix).

So, the insertion of any string into a Trie starts from the root node. It represents an empty string. All prefixes of length one are direct children of the root node. All prefixes of length 1 are stored at until level 1, all prefixes of length 2 are sorted at until level 2 and so on. In addition, all prefixes of length 2 become children of the nodes existing at level one. Tries are generally used on groups of strings, rather than a single string. The recursive version of the algorithm is presented in the following pseudo code.[17]

```
ConstructTrie(items)
{
Create empty node root
FOR i = 1 TO count(items) DO
{
AddString(root, items [i], i);
}
}
AddString(node, string, index)
{
IF (length(string) > 0) THEN
{
IF (string[0] is not the key of a child of node)
```

```
THEN
Create new node child with the value
string[0]
ELSE
Set child as the child of node with the key
string[0]
AddString(child, substring(string, 1), index);
}
ELSE //node is a terminal node
{
node.index = index;
}
}
```

3.10 FP-Growth Algorithm

FP-Growth algorithm was developed by J. Han, J. Pei and Y. Yin in 2000.[10] It used the suffix-based exploration of patterns. It means that to find a new frequent itemset from the already known frequent itemset I, the new itemset extends I by item, which is lesser or equal than all items in I. Thus I is a suffix of the new itemset.[4]

To efficiently determine, which items can be used to extend known frequent itemset to form a larger one, Frequent pattern tree (FP-tree) data structure is used.

FP-tree consists of a tree and frequent-item header table. The root of the tree is labelled Null, and children are item prefix-subtrees. Each node in item prefix-subtree consists of item identifier, count and link to next node with the same item. The count represents how many transactions contain set of items from the path from the root of the tree to the node. If there is no other node with the same item, the link is null.[10]

Algorithm 3 shows how to construct FP-tree.[4] Two passes over database are needed. In the first one, all transactions are read to count support for each item. In the second pass, frequent items from each transaction are inserted into the tree. After this insertion, there is a path starting in the root that

contains items from the transaction.

FP-tree is a compressed representation of transaction database. It contains only frequent items and transactions with same prefixes share common nodes in the tree.[3] Also, items are inserted in sorted order from the most frequent item to the least one to maximize the number of shared prefixes.

Figure 3.3 shows an example of FP-tree for given list of transactions. Algorithm 4 shows how FP-growth works. The key advantage is that it needs to read data only twice and then it uses FP-tree structure to quickly find parts of transactions from the database, which can extend given frequent "suffix". Frequent-item header table allows getting all nodes containing given item in constant time. And then to directly read possible frequent prefixes for frequent itemsets containing the item.

The given FP-tree is divided into smaller conditional databases (similar to a projected database). Each such database is associated with one frequent itemset (pattern suffix), and then this smaller part of data is mined separately in the same way. So to generate new frequent itemsets not all database have to be read, which reduces required time.[3]

Algorithm 3: FP-tree construction

Data: T - transactions, $supp\_thr$ - support threshold

Result: FP-tree F

(*children_names*(F,N) returns children names of node N in tree F)

```
1  F := tree with one node named null;
2  count support of items in T;
3  forall transaction t in T do
4  N := root of F;
5      Remove from t items with support ≤ supp_thr;
6      Sort items in t in descending order by their support;
7      forall item i in t do
8      if item i in children_names(F,N) then
9              N := child of N named i
10             N.count := N.count + 1
11     else
12             add a new child of N named I;
13             N := node added in previous step ;
14             N.count := 1 ;
15             Add N into frequent-item header table and update
                pointer link;
16     end
17     end
18 end
```

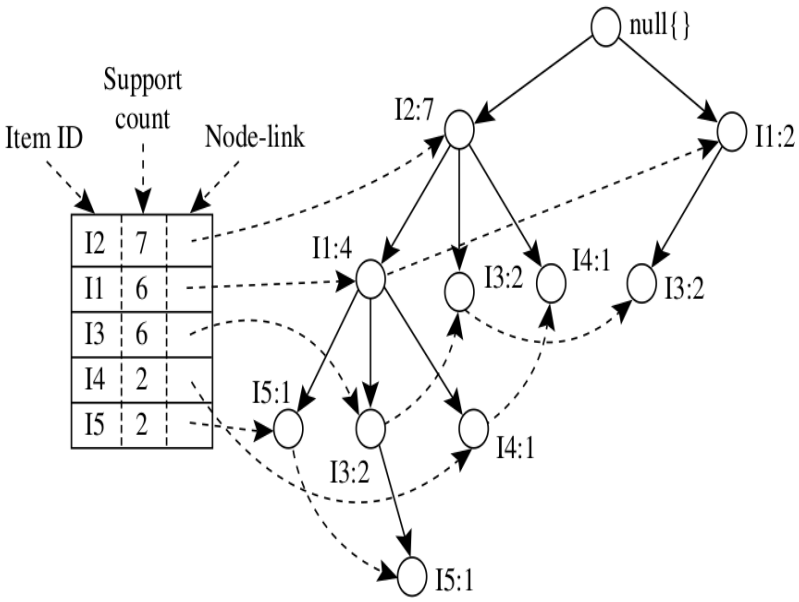

Figure 3.3 FP-tree from items I1, I2, I3, I4, I5 and 9 transactions [4]

3.11 MapReduce

To be able to process a big amount of data, two approaches to scaling are possible [13]:

horizontal – adding or removing compute nodes to a system (e.g., adding a new computer to a distributed system)

vertical – adding or removing of resources from a single "node" (e.g., adding more memory, CPU with more performance and so on)

The invention of algorithms described in chapter 4 is a way of vertical scaling. It means that a new more efficient approach how to find frequent itemset. However, these new algorithms still run on one computer, using a single thread.

A different approach to analyze bigger data is the usage of parallelization (horizontal scaling). A popular programming model for data-intensive computing is MapReduce software framework developed by Jeffrey Dean and Sanjay Ghemawat.[4][14]

It is designed to get parallelism from "computing clusters" instead of "supercomputer". Such clusters are created from commodity hard- ware connected by network together. To be able to share data among nodes and to store such big amount of data, a new form of filesystem was developed. This "distributed file system" also provides replication to protect against failures of storage media. Files are divided into small units called chunks. Moreover, each chunk is replicated several times to different nodes. A typical size of the chunk is 64 MB and is replicated three times (but normally this is possible to be configured by the user). To be able to find what chunks creates given file, there is master node (or name node) file which tracks chunks for the file. Finally, all master nodes can be found in a directory of a file system. The distributed file system is designed for rarely updated data, and it is not suitable for even "big data", but which are changed frequently.[2]

MapReduce style of computing was first used by Google in its internal system. It got its name because of the fact that it combines two capabilities from functional languages: map and reduce.[15] Now several implementations exist. One of them is open source Hadoop with Hadoop Distributed File System which is developed by Apache Foundation.[2]

Operation of MapReduce can be described in following steps[2]:

1. Each map task reads one or more chunks from the distributed file system and transform it into *key-value* pairs.
2. Master *controller* collects key-value pairs, sorts them by the key and distributes them to Reduce tasks – in a way that one Reduce task process all *key-value* pairs with an equal key.
3. A reduce task combines values from passed *key-value* pairs and produce some output.

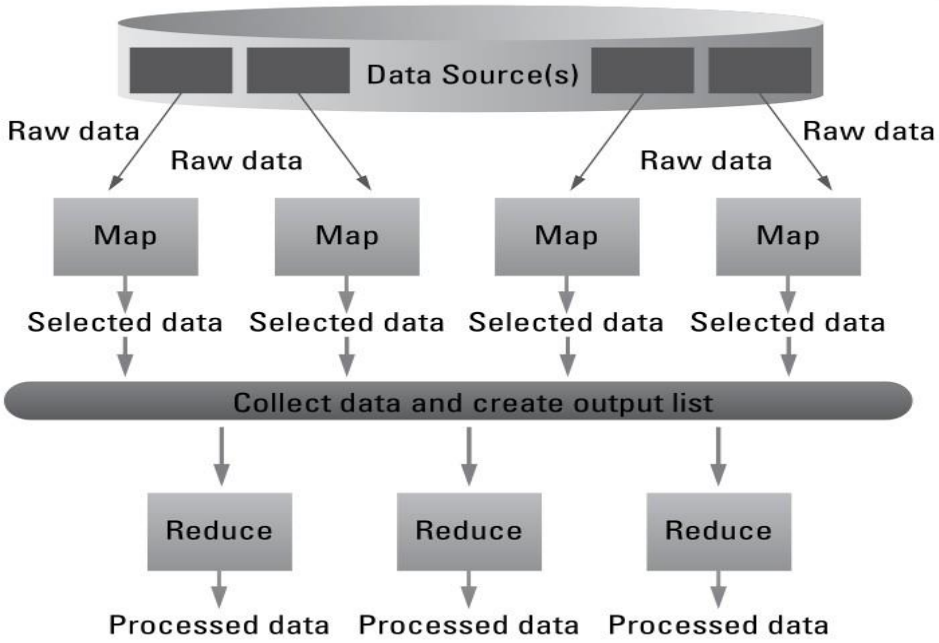

Figure 3.4 Schema of MapReduce Computation [15]

MapReduce offers a simple way how to write parallel programs because a user of this framework provides only a code, which creates

key-value pairs from data (map function). And a code, which combines values from pairs with the same key (reduce function). So parallelism is implicitly used, and program developer does not have to bother with low-level details such as memory address spaces, data location and so on. All is done automatically by MapReduce framework. Typically distributed file system is used to distribute key-value pairs to compute tasks. So accessing data using disk between Map and Reduce is one of the disadvantages of this approach.[4]

3.12 Parallelization of Frequent Pattern Mining Algorithms

SON algorithm (authored by Savasere, Omiecinski, and Navathe - also called Partition algorithm[16]) is one of the algorithms which can be parallelized. It tries to solve a disadvantage of the A-priory algorithm to reduce the number of database scans. It divides the dataset into non-overlapping parts (called partitions) and finds frequent itemsets in these parts independently. Support threshold for these parts is set smaller for n partitions and original threshold p the new threshold is p/n. All found frequent itemsets are then joined to create candidate itemsets. In the second pass over the database, support for candidate itemsets is counted. The correctness of this algorithm relies on the fact that each globally frequent itemset has to be frequent in at least one partition.[4][16]

Finding of frequent itemsets in individual partitions can be easily performed in parallel. The same applies to support counting of candi- date itemsets. It can be described as a sequence of two MapReduce runs[4]:

- **Map function 1** – processes assigned chunks of the database file in the distributed file system and find frequent itemsets there. Support threshold is appropriately smaller to the number of partitions. It outputs key-value pairs (F,1), where F is found frequent itemset and value is unused in next processing.
- **Reduce function 1** – outputs keys from assigned key-value pairs. These keys are candidate itemsets.
- **Map function 2** – takes all candidate itemsets from reduce tasks and processes assigned chunks of the database file. And in the portion of the file, it counts support for candidate itemsets. The output pairs are (I,c), where I is processed itemset and c is sup- port of I.
- **Reduce function 2** – all pairs with the same key are processed by one reduce function. It sums all values for that key and if the result is higher than support threshold the reduce function outputs the key. The support for such itemset in the key can also be included in the output.

Itemsets produced by reduce function 2 are frequent itemset in the database file.

Apriori , FP-growth and Eclat algorithms as described in this thesis can

use the lexicographical tree to describe the process of finding frequent itemsets. Each of these algorithms minimizes the size of arguments passed into recursive function calls for a subtree when processing the node of the tree. Processing of the subtree is an independent problem. So it could be run in parallel to the processing of other subtrees.

3.13 Chapter Summary

This chapter stated theory background of data mining, knowledge discovery process steps. This chapter included The discovery of frequent patterns, associations, and correlation relationships. These are efficient and scalable algorithms for frequent itemset mining. These algorithms are Apriori algorithms, frequent pattern growth–based algorithms such as FP-growth, and algorithms that use the vertical data format. Finally, this chapter presents Mapreduce process and parallelization of frequent pattern mining algorithm. The next chapter will described proposed system of research.

CHAPTER 4

IMPLEMENTATION OF THE PROPOSED SYSTEM

System design is implemented in Figure 4.1. Firstly, input section consists of four parameters with datasets, number of maps and support count via the command line. AprioriMapReduce phase processed from input data set the first stage. Then the next stage generates prefix tree and check prefix. If it is no prefix then go to Generate Prefix Tree stage repeat again. Check Prefix condition is yes then next to prefix tree process. EclatMapReduce process partitions (v?) the data and generates frequent pattern the next stage. Finally output the frequent itemset.

4.1 Apriori Prefix Tree Eclat

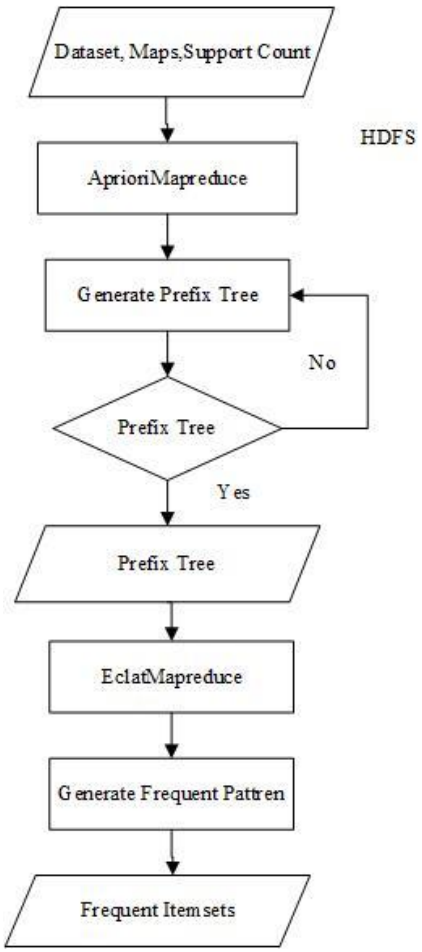

Figure 4.1 Proposed System Design of Apriori Prefix Tree Eclat

1) The proposed system contains input dataset, number of maps and support count.
2) AprioriMapReduce process generated k-Frequent itemsets by using input large transaction list. Mappers received <key,value> pairs of transaction list. Reducer combines all local frequencies. Reducer redistributed global frequencies of item/itemsets to all mappers as candidate for the next time. Then repeated k-times process to get k-frequent itemsets. The candidates <key,value> pairs are partition into across the mappers can solved memory problem.
3) PrefixTree process generated k+1 Frequent itemsets (local frequent superset). Reducer combine all local frequencies to global frequencies list. Then redistributed to complete prefix groups of itemsets to all mappers.
4) EclatMapReduce mining part utilizes the diffsets to mine the prefix groups as a conditional database for frequent itemsets.

4.2 Frequent Pattern Mining with Hadoop Mapreduce

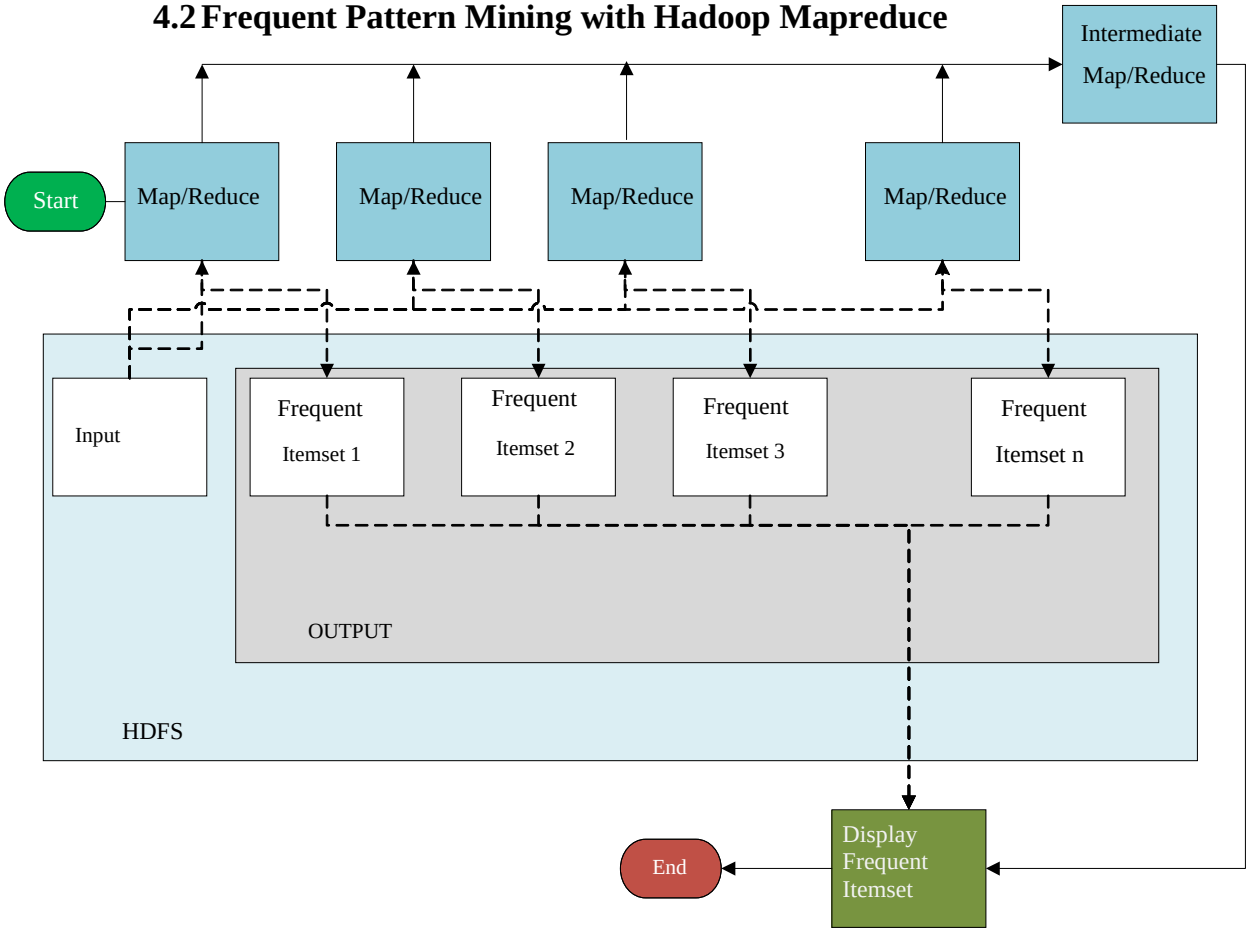

Figure 4.2 Data Flow Diagram Showing Iteration of Proposed Method

1) Generate frequent 1-itemsets –The HDFS of the Hadoop environment is input dataset storage source to support scalable and quick access data for MapReduce operations [16]. Then input data is split into various chunks and Mapper process data maps to support output. The mapper output is produced as <key, value> pair. The combiner combined all the maps <key, value> pair together. Then reducer received maps <key, value> from combiner and calculate the support values. Then the support values are satisfied the minimum support count that values are taken as the frequent 1-itemsets.

2) Generate candidate 2-itemsets and n-itemsets – Next the mapper generated the candidate 2-itemsets that are using the frequent 1-itemsets. The candidate 2-itemsets count input data of each item is verified that is provided to the mapper. Then combiner combined them to calculate the count values of the 2-itemsets and provided to the reducer. The reducer again reduces from combiner produced data and counts the support values of 2-itemsets. This process is repeated until all the possible candidate n-itemsets are generated no possible frequent itemset is available in previous iteration.

3) Finally Mapreduce process is generating all the frequent n-itemsets. The output is all the selected itemset value and its support count. It is written in an output file.

The data flow diagram illustrates two iterations of MapReduce in Hadoop ~~is~~ as shown in Figure 4.2

4.3 Implementation of the Frequent Itemset Mining on Mapreduce

Educational data mining is the knowledge discovery process of finding knowledge from large volume of educational data. Large amount of data can't cannot be handled with traditional data mining tools. This chapter illustrates the proposed approach~~ed~~ that implements big data analytics tool for educational purposes. There are large data that stored to Hadoop file system HDFS to speed up processing in data mining work. These data is used in mapreduce data mining algorithm. First algorithm finds out the prefixes of length (1,2,3,..) then sends to workers. Then the each worker generated the frequent itemset and

estimating its work done. Lower depth of high pruning ratio is accurate estimation of a worker total work. In the first step, the frequent itemset increases with the prefix length, distributed load amount decreases. Approximation of total work of workers meet high pruning ration at lower depths. Note that mining the number of frequent itemsets with prefix length increases in the first step, distributed process decreases.

Implementation of frequent itemsets mining is worked by mapreduce. Each mappers and reducers communicated to each other in step three. In this implementation to reduce network communication, this system encoded the itemset using compress trie string representation to each batch of mined pattern. Basically, delimiters represented the traversed tree downwards or upwards.

$$\begin{vmatrix} a\ (300) \\ a\ b\ c\ (100) \\ a\ b\ d\ (200) \\ b\ d\ (50) \end{vmatrix} \rightarrow a\ (300)|b|c(100)\$d(200)\$\$\$b|d\ (50)$$

The proposed system calculates superset of the closed itemsets in the transaction dataset. Only the closed sets in subtree maps the individual mappers. It is possible to mine the correct closed itemsets.

4.4 Load Balancing

The load balancing in 2 way approached: the relation between the workload and (1) distributed prefixes length, (2) the assignment scheme.

Let the set of k-prefixes P_k = {p_1 p_2, ... ,p_m} be partitioned to n workers, $P_k^1, P_k^2, \ldots\ldots, P_k^n$, where P_k^j is the set ofprefixes assigned to worker j, then the prefixes are assignedto worker nodes using the following methods:

Round-Robin: p_i is assigned to the worker.$P_k^{(i\ mod\ n)}$

Equal Weight: When p_i is assigned to a worker, support(p_i) is added the score of that worker. p_{i+1} is assigned to a worker with the lowest score. Assignment is order dependent.

Block partitioning: {p_1, ... ,$p\lceil m/n \rceil$} are assigned toP_k^1, {$p(\lceil m/n \rceil + 1), \ldots, p(2 \times \lceil m/n \rceil)$} are assigned to P_k^2,and so on.

Random: Each p_i is assigned to a random worker.

For this set of experiments we use 4 different datasets: Abstracts provided by

De Bie [9], T10I4D100K, Mushroom and Pumsb are from FIMI repository [1]. Properties of these datasets are given in Table I.

4.5 Program Demonstration

Program demonstration has the initial data collection to data mining process flow. There are two programs for this approached system. The two programs wrote the java programming language. The first one is writing for android and the other for Cent OS (Orical Big Data Light OS). First program provided data collection for teacher assessment data from the whole computer universities in Myanmar. The collected data is stored in google cloud data storage. Long term storage space used own data repository server that deployed in university of computer studies, Yangon. The proposed data mining program is used input data from University of Computer Studies dataset repository, Yangon. Lab computer has used Cent OS and HDFS system. The implementation of the two methods are shown in the next section.

4.6 Dataset Collection and Program Implementation

The collection program of the proposed system design is based on cloud database and mobile android application. The android teacher assessment apk installed on not below android 7 version. Teachers, Students and Administer must register before ~~use of~~ using this application. Students register verification used QR code scanner with their student identification card. Teacher and Administer register verification used with preregister by administration staff records from their relevant Universities. Teacher assessment survey questions can be used in three languages as Myanmar Unicode, Myanmar Zawgyi, and English fonts. Thus, teacher assessment users can easily understand in survey questions. Moreover, their answers are accurately confident to their understandable questionnarie. The figure 4.3. shows mobile devices which uses teacher assessment apk and cloud system.

Mobile phone application has used internet connection for data collection. Data collection mobile app installed on smart phone that provided teacher assessment survey system. The teacher assessment survey app ~~is~~ has used google cloud platform Figure 4.4.

4.7 Android- Based Data Collection and Program Implementation

Figure 4.3 shown mobile application connection to cloud system. Data Collection android mobile application users can used with data connection from their android mobile phone.

Figure 4.3 Mobile Application

Figure 4.4 shown Android mobile phone using Google cloud platform.

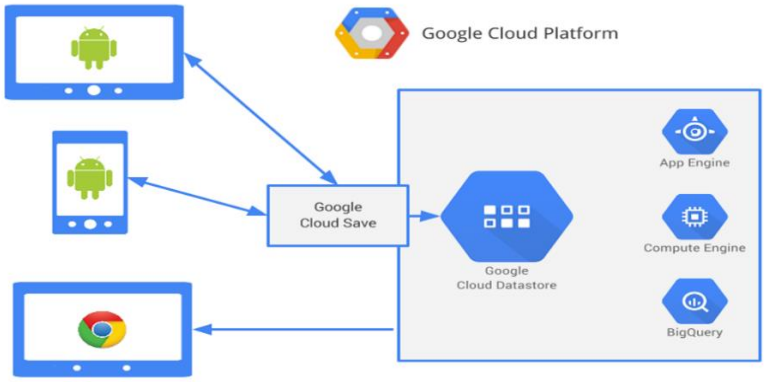

Figure 4.4 Google Cloud Platform

Teacher assessment survey app user interface ~~as~~ is shown in the following.

1. Main user from Figure ~~4.5.~~
2. Teacher Register form Figure 4.6.
3. Student Register form Figure 4.7.
4. User Login form Figure 4.8.
5. Teacher Assessment Survey form in Myanmar language Figure 4.9.
6. Teacher Assessment Survey form in English language Figure 4.10.
7. Teacher score of course Figure 4.11.
8. Total score of the whole teacher on each course Figure 4.12.

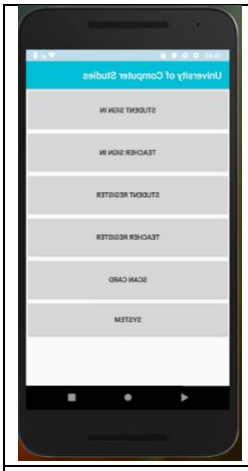	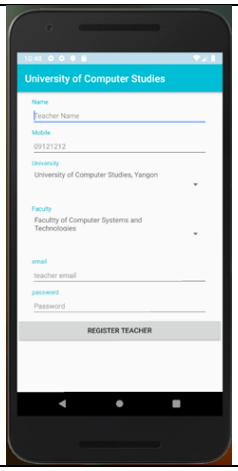	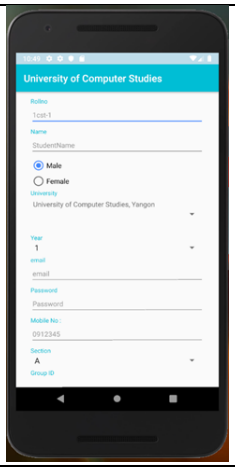	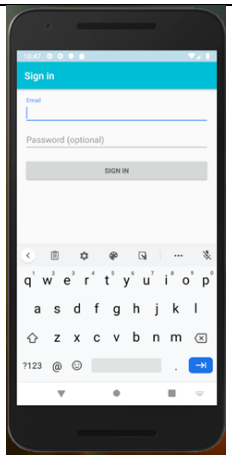
Figure 4.5	**Figure 4.6**	**Figure 4.7**	**Figure 4.8**
			
Figure 4.9	**Figure 4.10**	**Figure 4.11**	**Figure 4.12**

4.8 Android- Based Data Collection Program for Main

Figure 4.13 is a main page of data collection program. Firstly, Users touched with this page. It contains student sign in, teacher sign in, student register, teacher register, scan card and system.

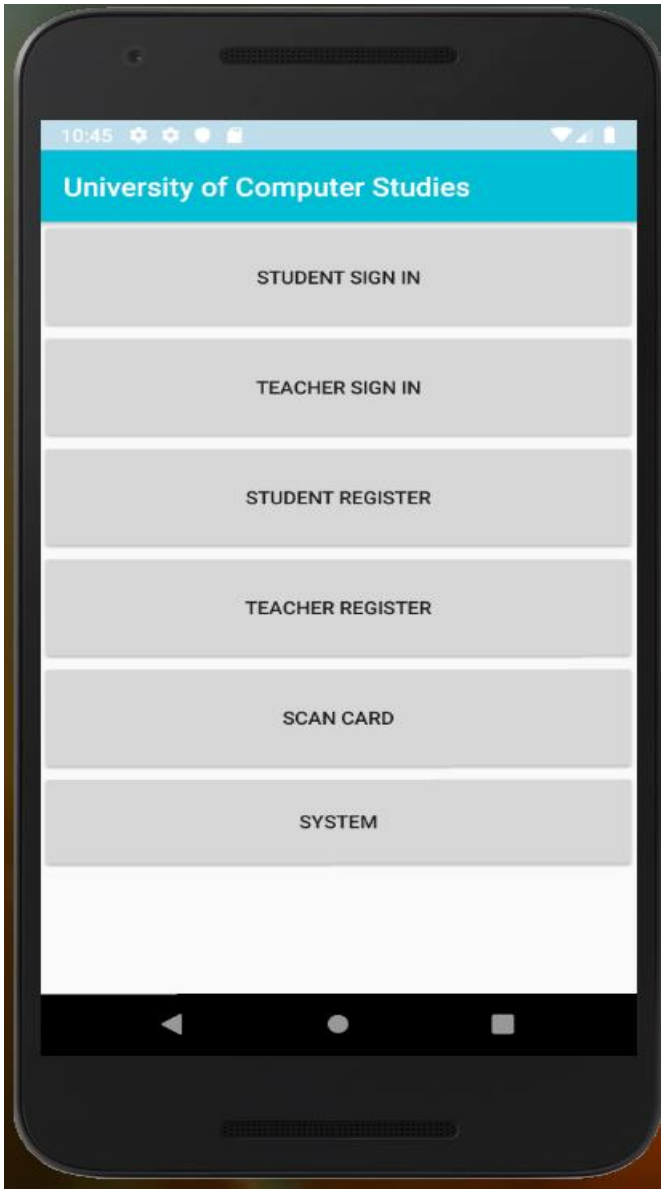

Figure 4.13 Teacher Assessment Survey Main

4.9 Android- Based Data Collection Program for Sign in

Figure 4.14 shown student/teacher sign page. It is allowed to enter registered students in University. This page contained User id for email address and password.

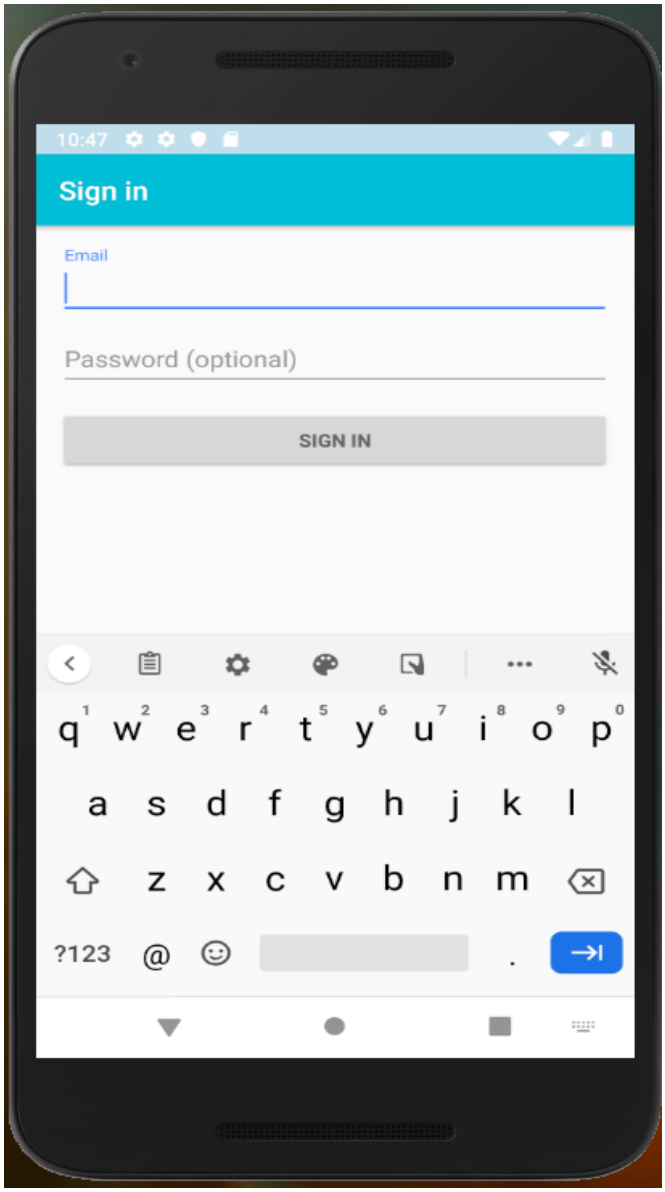

Figure 4.14 Teacher/Student Login

4.10 Android- Based Data Collection Program for Teacher Register

Figure 4.15 shown teacher register page in android mobile application. New teacher can register with this page. New teacher entry required information in this page and when complete click the register teacher button.

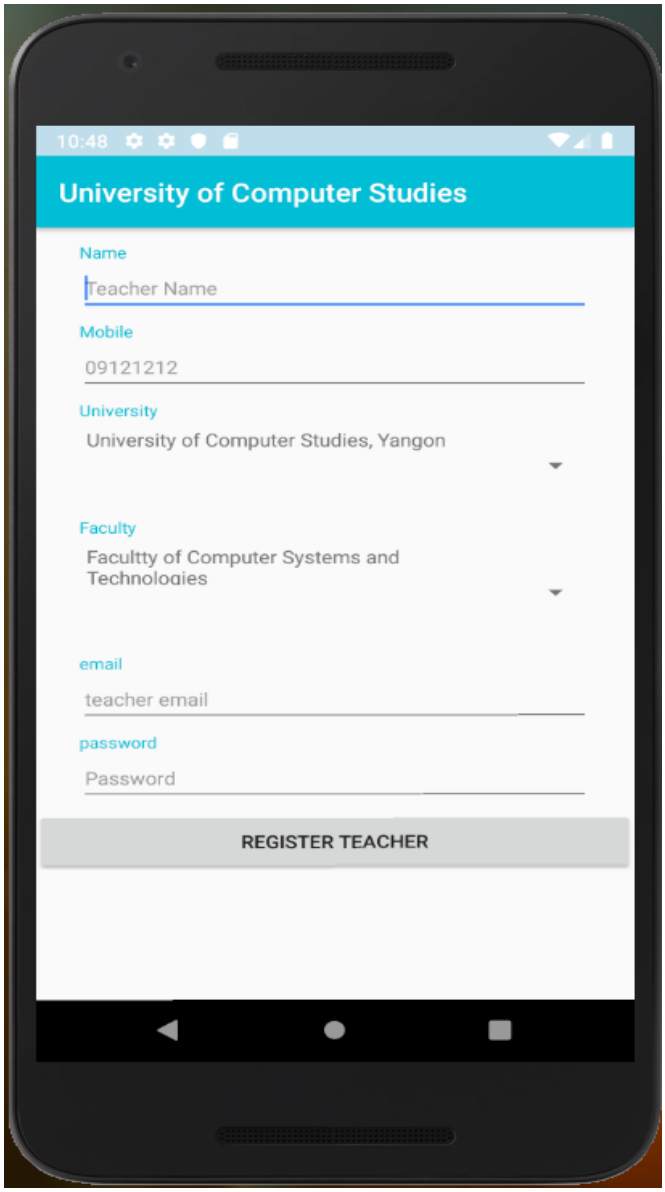

Figure 4.15 Teacher Register

4.11 Android- Based Data Collection Program for Student Register

Figure 4.16 shown student register page in android mobile application. New student can register with this page. New student entry required information in this page and when complete click the register student button.

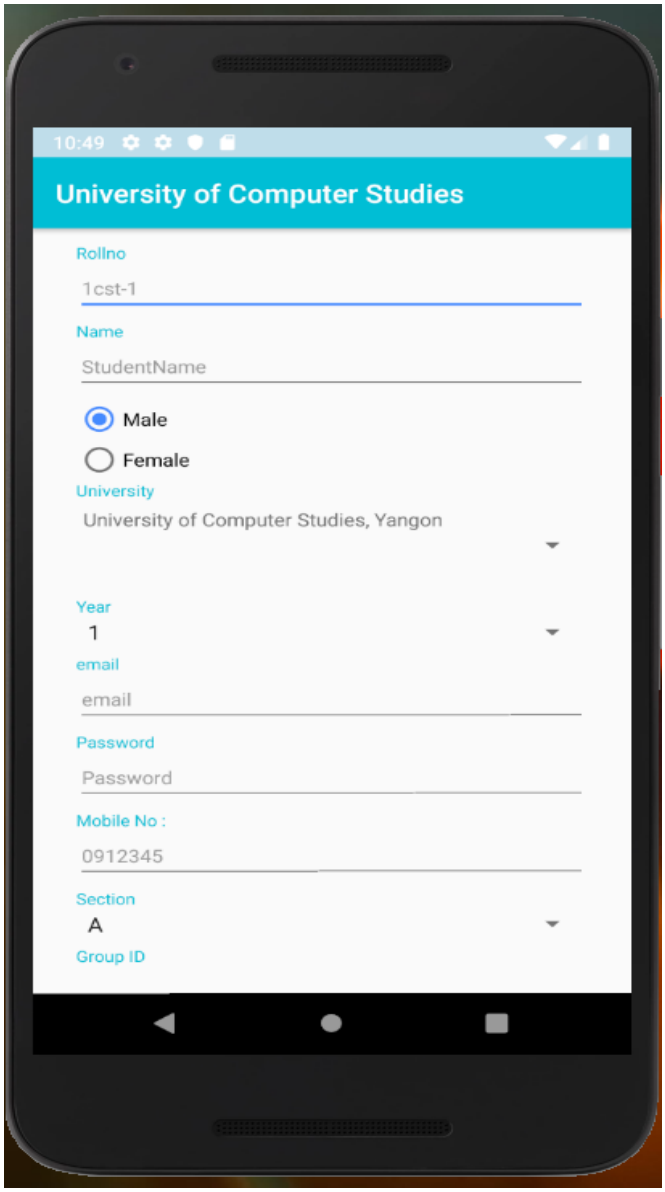

Figure 4.16 Student Register

4.12 Android- Based Data Collection Program for QR Code Scan

Figure 4.17 shown QR code scan page. When user press the scan button scanner is appeared and scan the QR code image from student identification card.

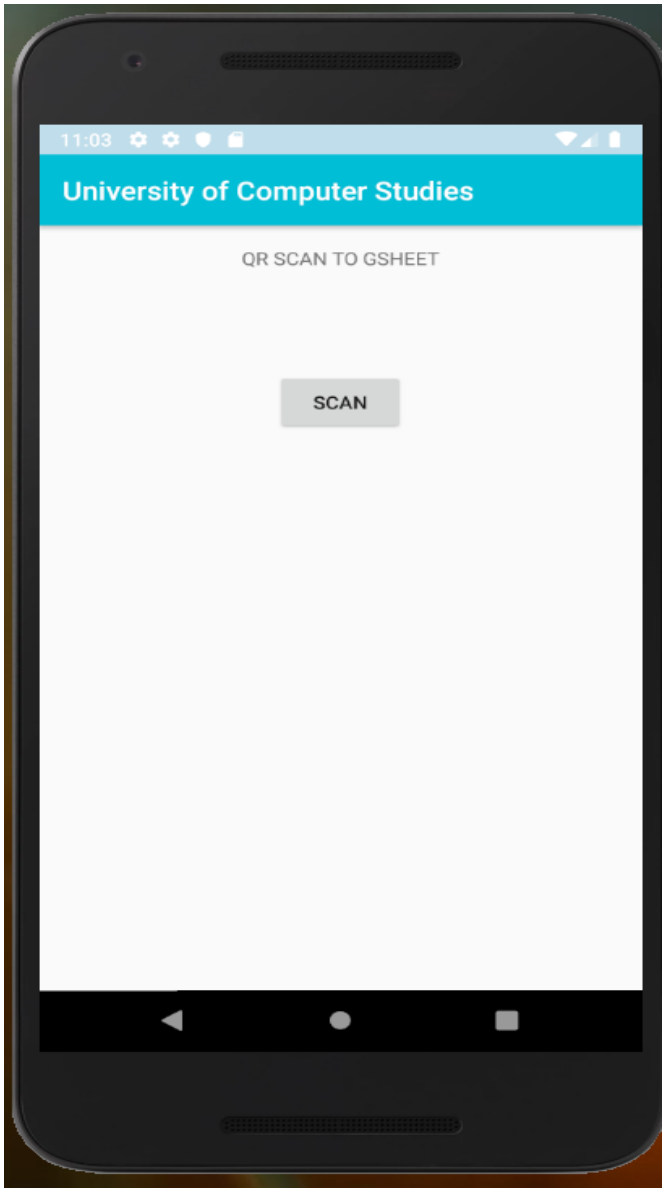

Figure 4.17 QR Code Scan

4.13 Android- Based Data Collection Program for Student Home

Figure 4.18 shown student home page. Student home page contains home, user profile, survey questions, edit profile and logout buttons. User can view his profile using profile button. User answered survey questions by clicking questionnaire button. User can edit his profile information by clicking edit profile button and logout with using logout button.

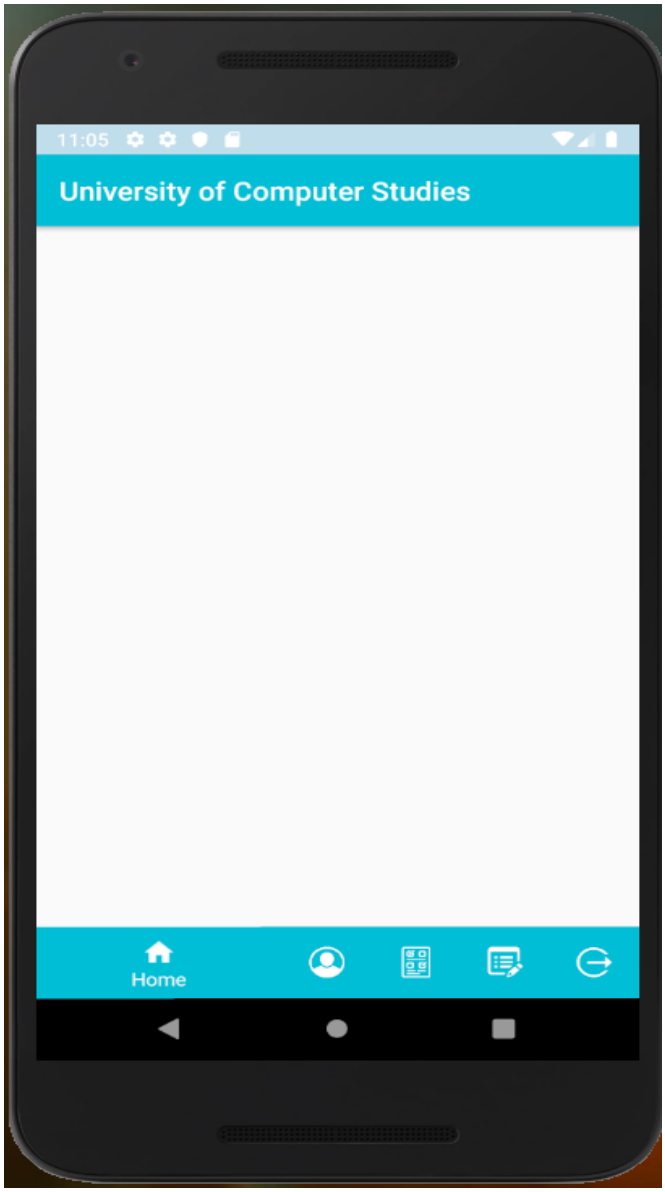

Figure 4.18 Student Home

4.14 Android- Based Data Collection Program for Student Profile

Figure 4.19 shown student profile page. When user click the profile button then user profile page will appear.

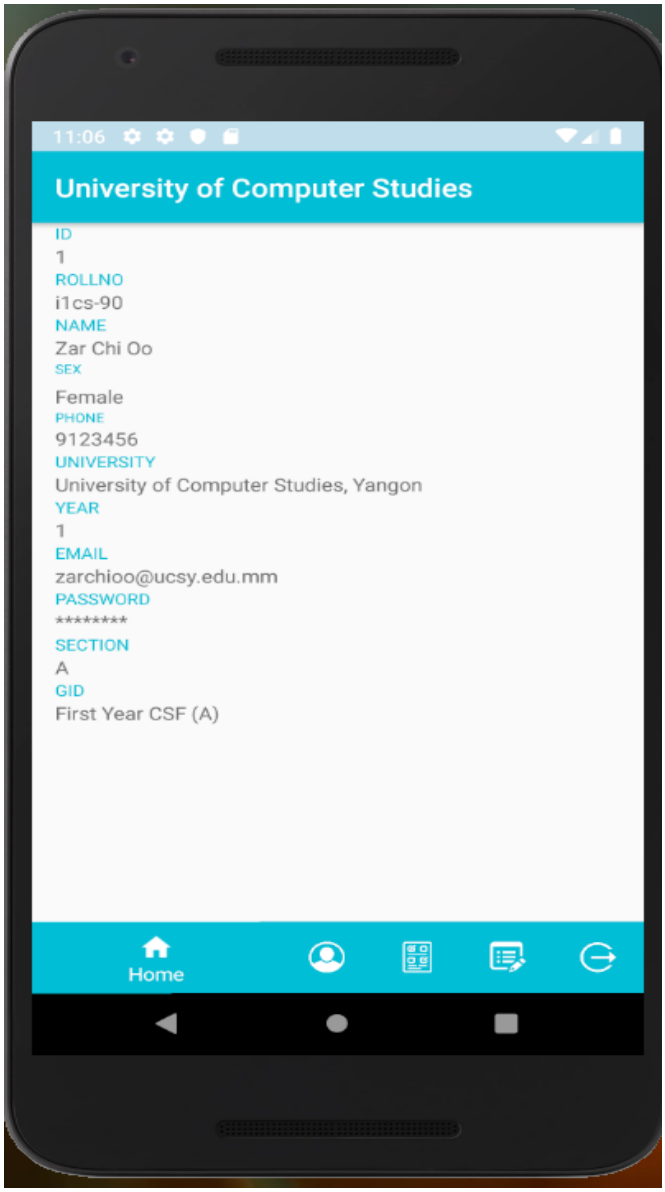

Figure 4.19 Student Profile

4.15 Android- Based Data Collection Program for Student Profile Update

Figure 4.20 shown android based data collection program for student profile update.

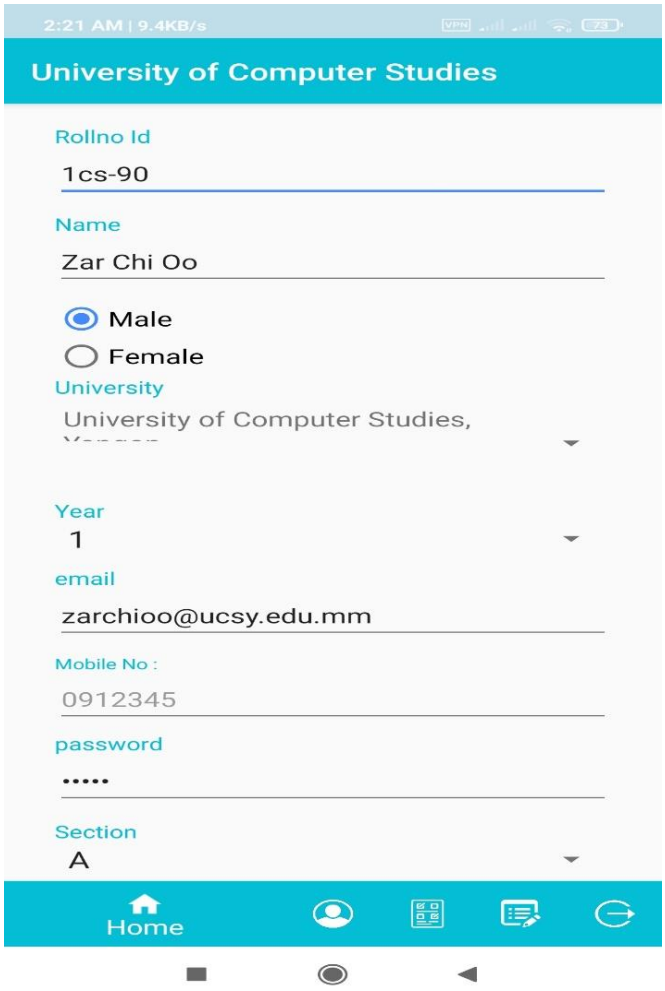

Table 4.20 Student Profile Update

4.16 Android- Based Data Collection Program for Teacher Assessment Survey Question Attempt

Figure 4.21 shown teacher assessment survey question attempt page list from each teachers for each student groups. User can answer survey questions by clicking each question list.

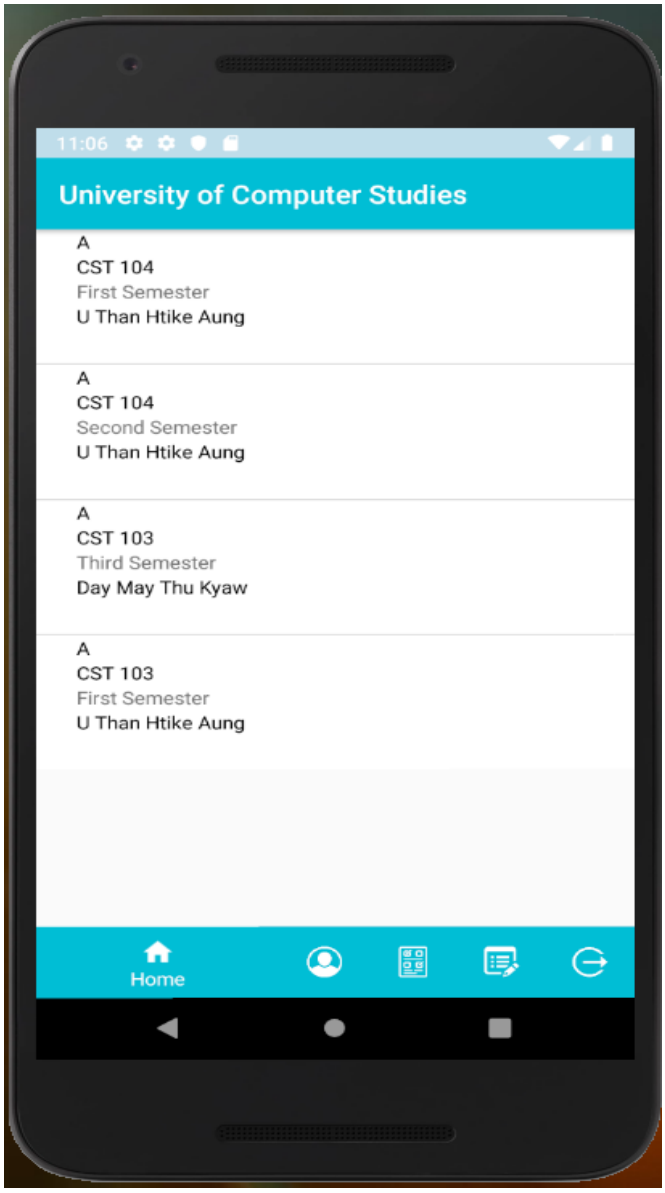

Figure 4.21 Teacher Assessment Survey Question Attempt

4.17 Android- Based Data Collection Program for Teacher Assessment Survey Question for Languages

Figure 4.22 shown teacher assessment survey question for languages page. User can optionally choice in English, Myanmar Unicode and Myanmar ZawGyi survey questions by clicking each button.

Figure 4.22 Teacher Assessment Survey Choice Language

4.18 Android- Based Data Collection Program for Teacher Assessment Survey Question in Myanmar Language

Figure 4.23 shown teacher assessment survey question in Myanmar Language page. User can answer survey questions by clicking each option button in each question and finally click submit button.

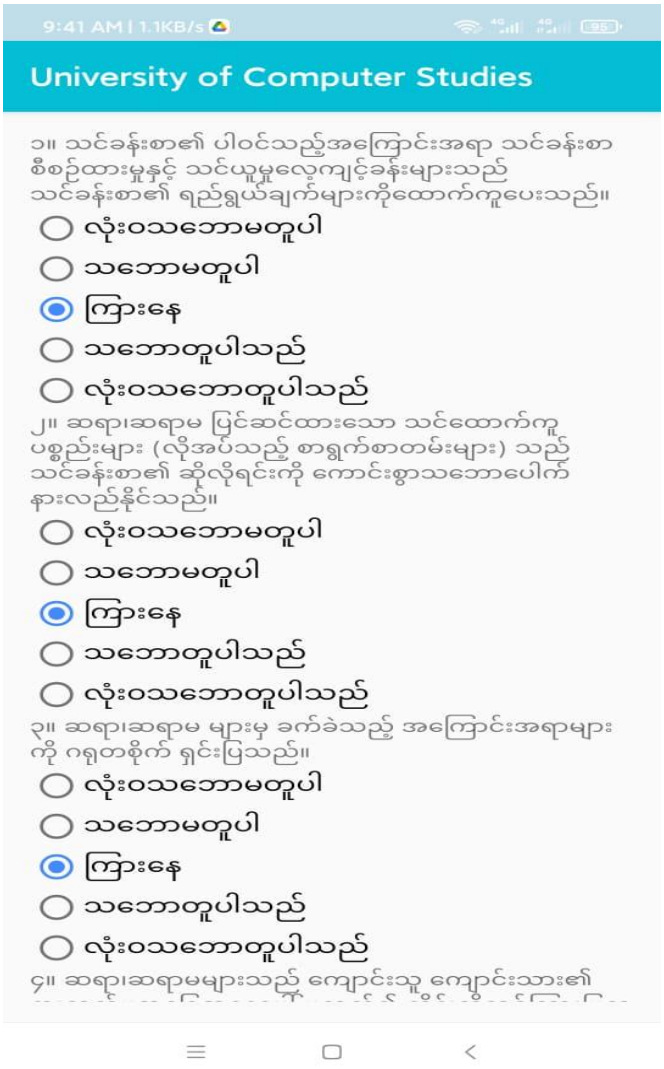

Figure 4.23 Teacher Assessment Survey Question in Myanmar

4.19 Android- Based Data Collection Program for Teacher Assessment Survey Question in English Language

Figure 4.24 shown teacher assessment survey question in English Language page. User can answer survey questions by clicking each option button in each question and finally click submit button.

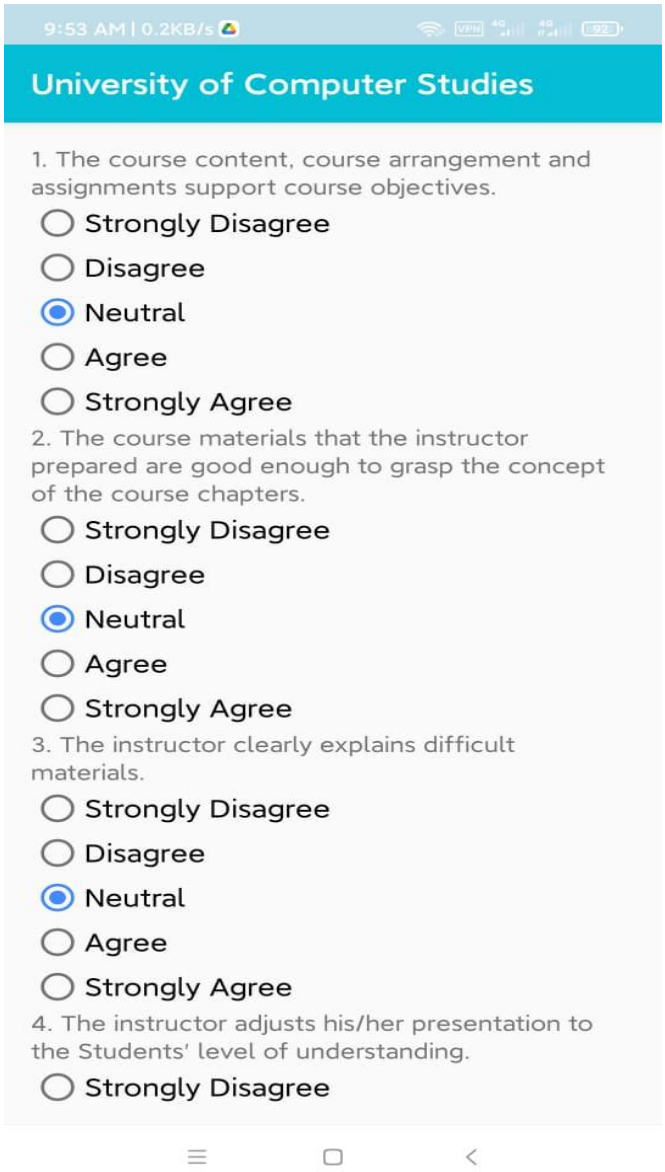

Figure 4.24 Teacher Assessment Survey Question in English

4.20 Android- Based Data Collection Program for Teacher Home

Figure 4.25 shown teacher home page. User can answer survey questions by clicking each option button in each question and finally click submit button. This page contains nine buttons.

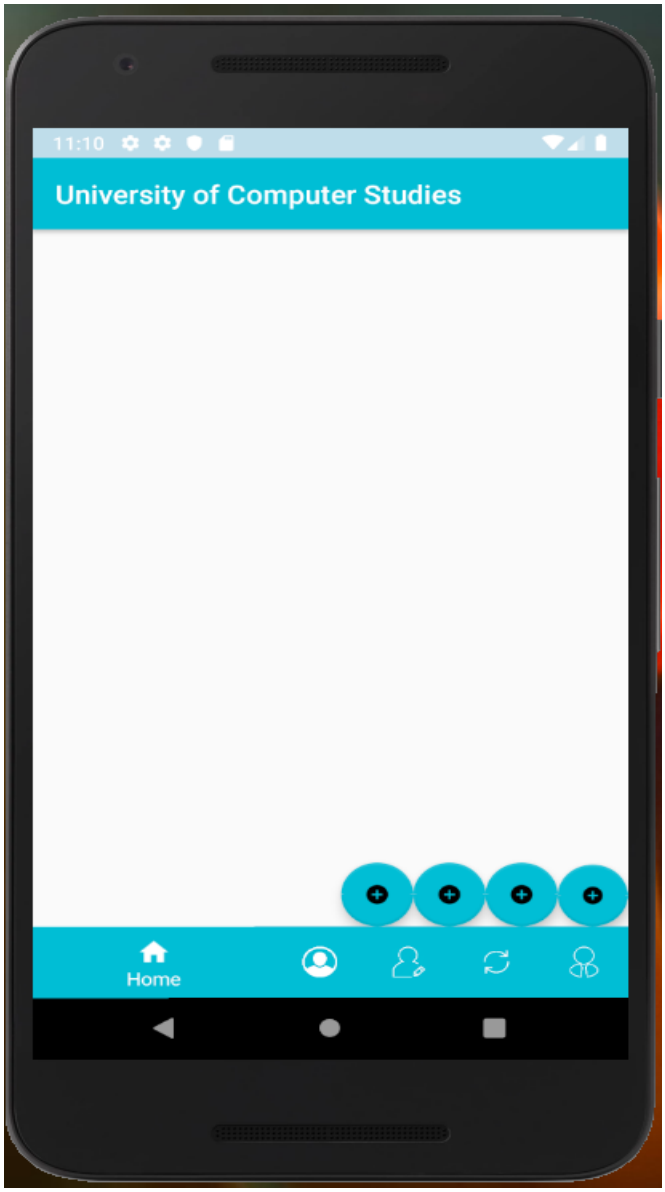

Figure 4.25 Teacher Home

4.21 Android- Based Data Collection Program for Teacher Profile

Figure 4.26 shown teacher profile page. When teacher click profile button profile page will display.

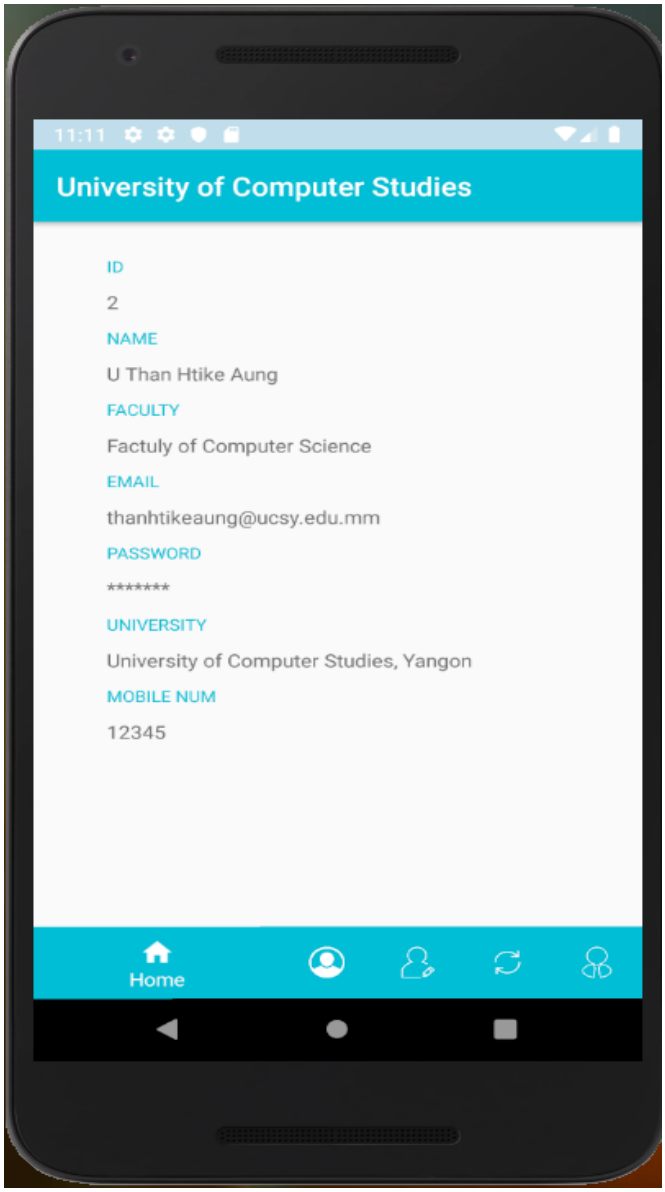

Figure 4.26 Teacher Profile

4.22 Android- Based Data Collection Program for Teacher Profile Update

Figure 4.27 shown teacher profile update page. When teacher click profile update button profile page will display. User can change their data using profile update.

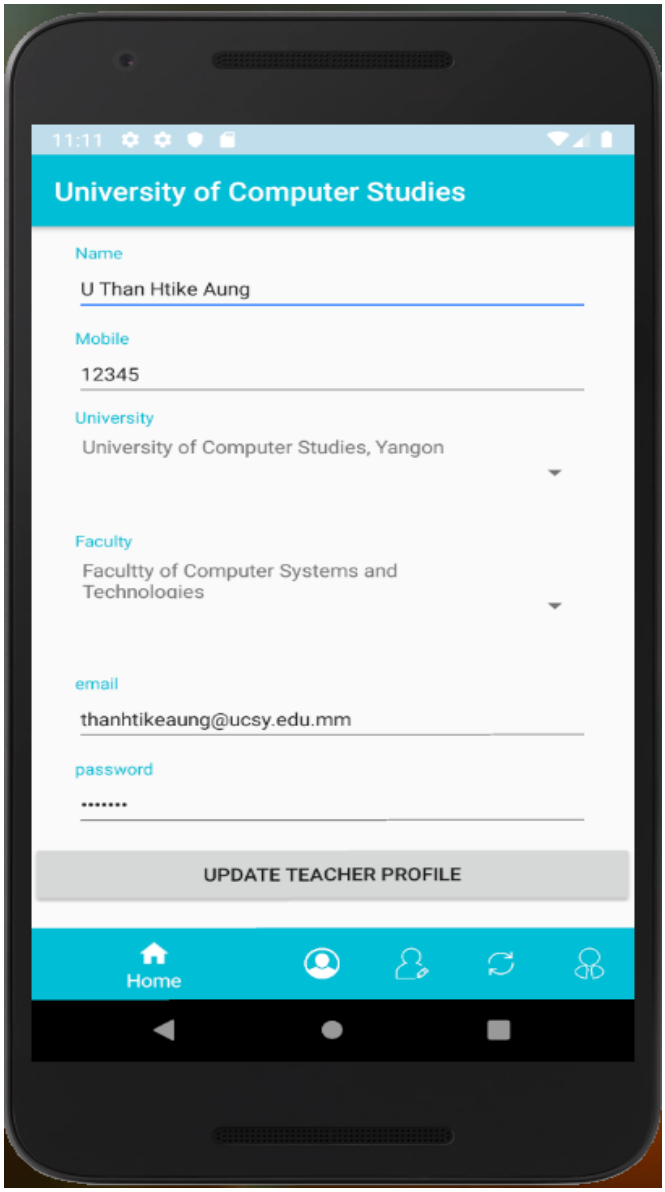

Figure 4.27 Teacher Profile Update

4.23 Android- Based Data Collection Program for Create New Teacher Workload

Figure 4.28 shown new teacher workload create page. New user can create his work load by using new teacher work load create page and click create workload button.

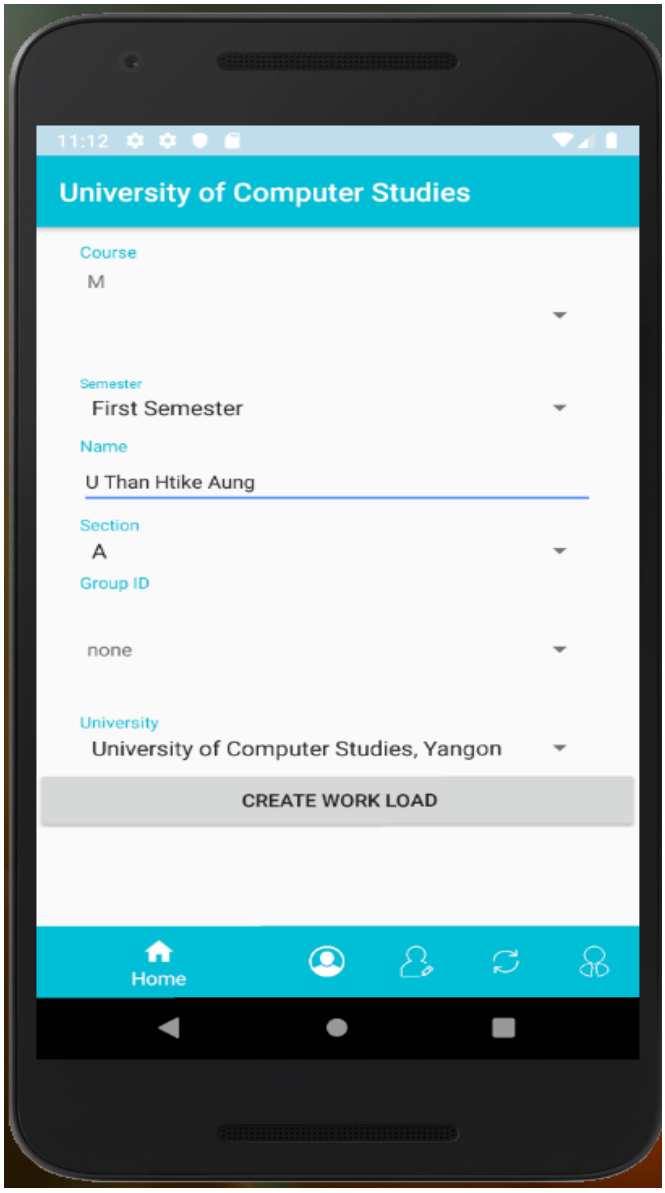

Figure 4.28 Create New Teacher Workload

4.24 Android- Based Data Collection Program for Teacher Workload Lists

Figure 4.29 shown new teacher workload lists page. User can update teacher assessment survey list by clicking each list to his page.

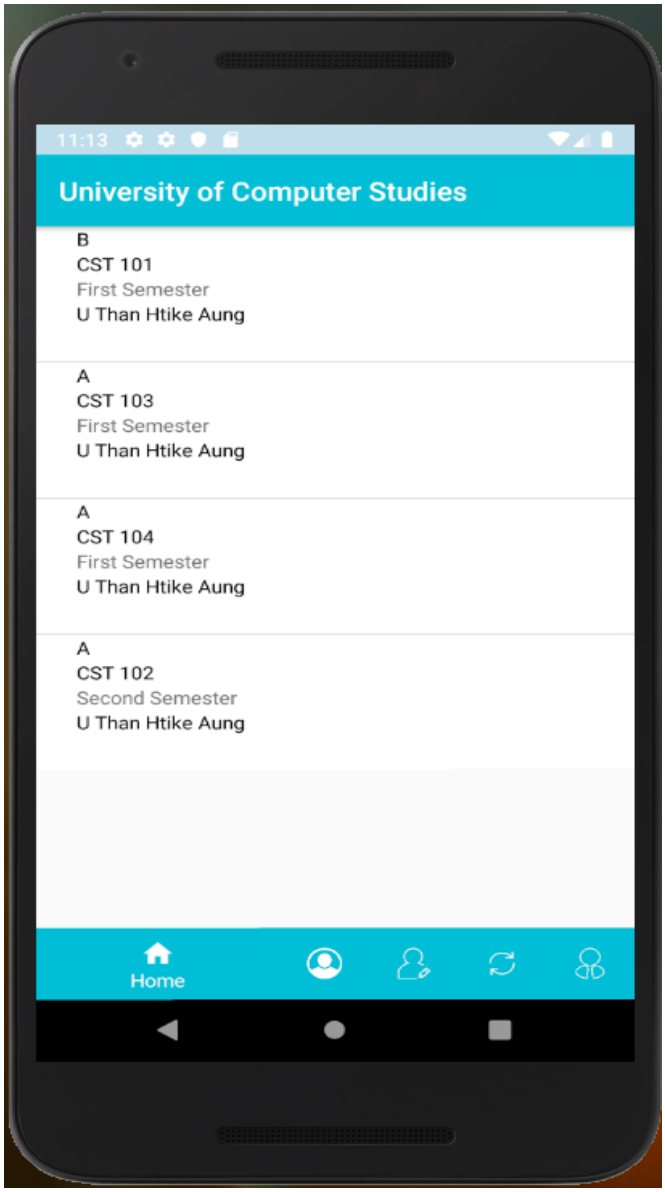

Figure 4.29 Teacher Workload Lists

4.25 Android- Based Data Collection Program for Update Teacher Workload

Figure 4.30 shown new teacher workload update page. User can update his work load by using teacher work load update page and click update workload button.

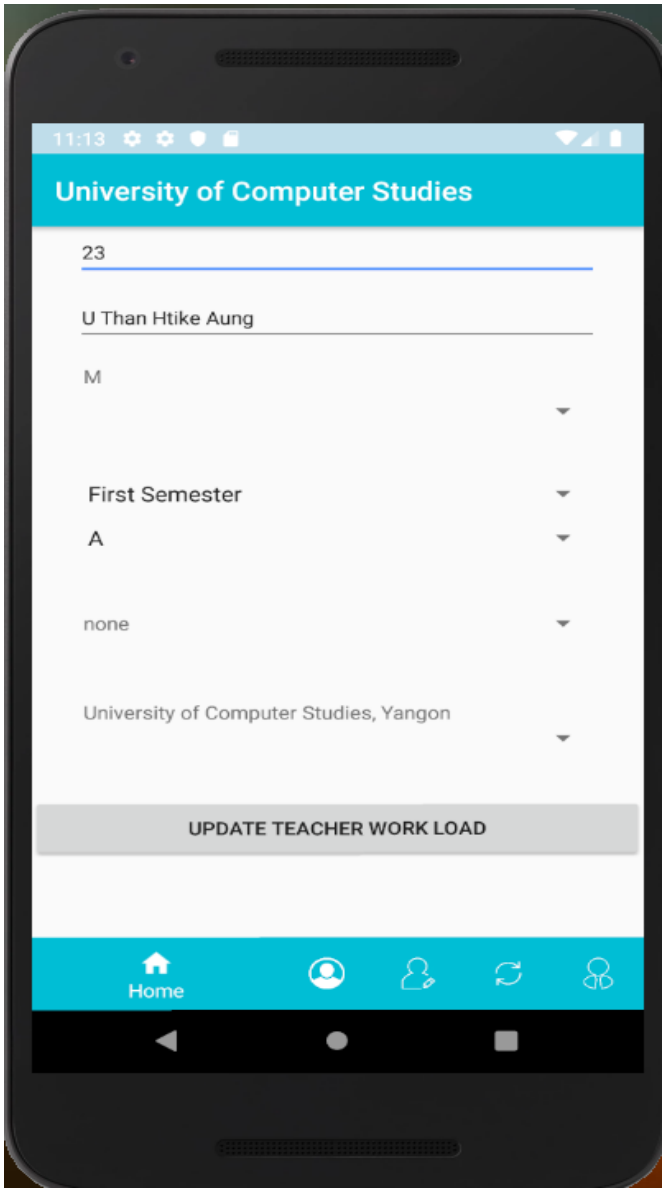

Figure 4.30 Update Teacher Workload

4.26 Android- Based Data Collection Program for Teacher Workload Lists To Create Teacher Assessment Survey For Student

Figure 4.31 shown teacher assessment survey page for each student. User can send survey to each student by clicking each list.

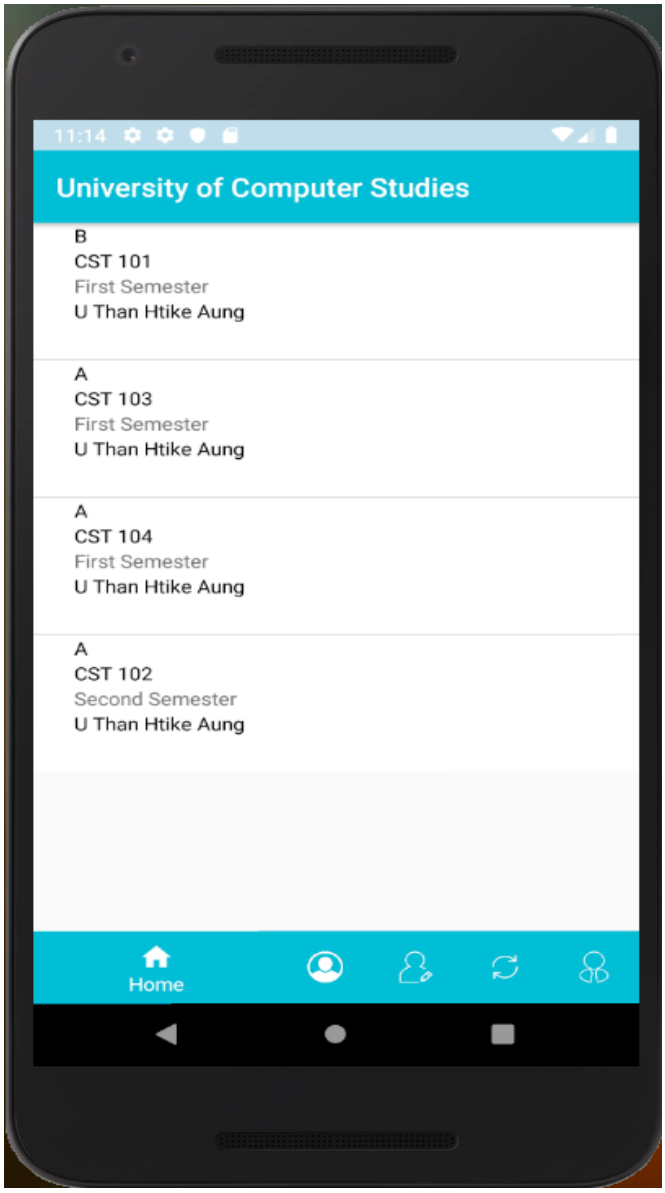

Figure 4.31 Teacher Workload Lists To Create Teacher Assessment Question For Student

4.27 Android- Based Data Collection Program for Teacher Assessment Survey Post To Students

Figure 4.32 shown teacher assessment survey post to students page for each student. User can send survey to each student by clicking post list to student.

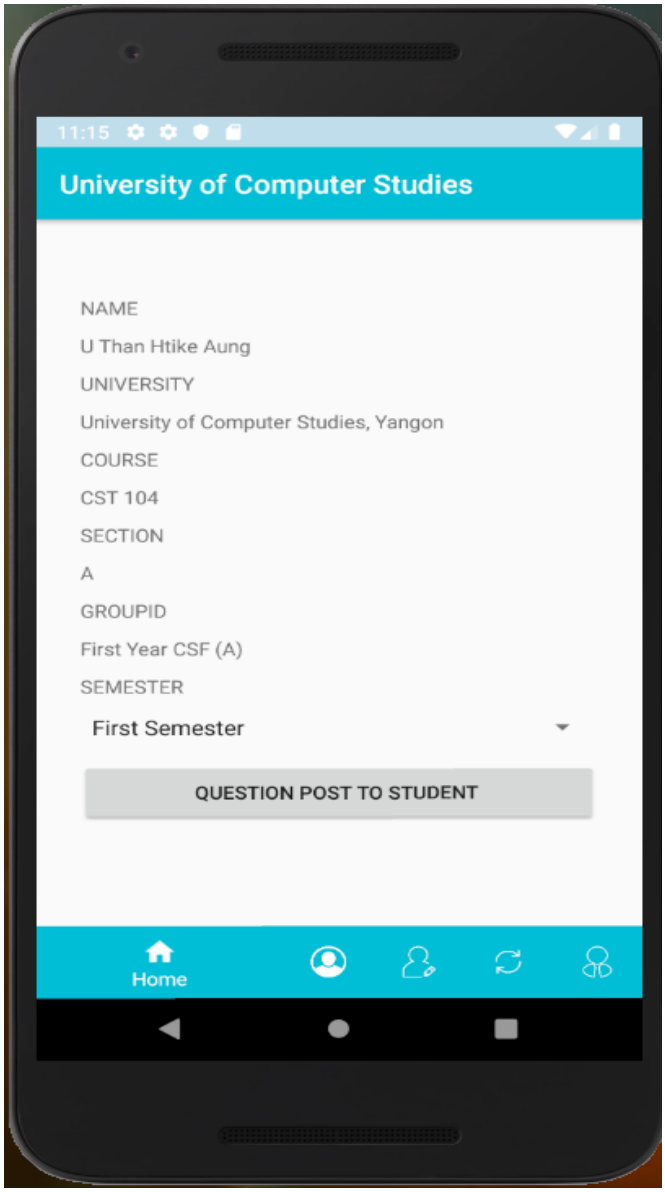

Figure 4.32 Teacher Assessment Survey Post To Students

4.28 Android- Based Data Collection Program for Display Created Teacher Assessment Survey Lists

Figure 4.33 shown display created teacher assessment survey lists page for each student.

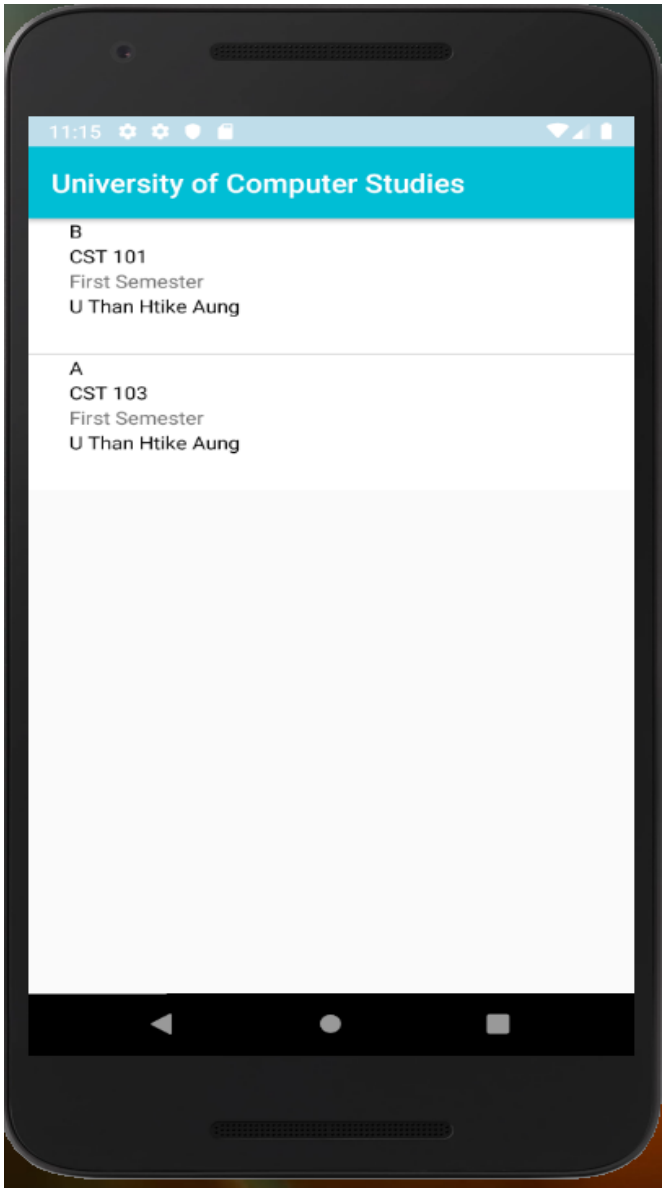

Figure 4.33 Teacher Assessment Survey Lists

4.29 Android- Based Data Collection Program for Teacher Workload Detail

Figure 4.34 shown teacher workload detail page. This page include course, semester, Teacher name , Section and University.

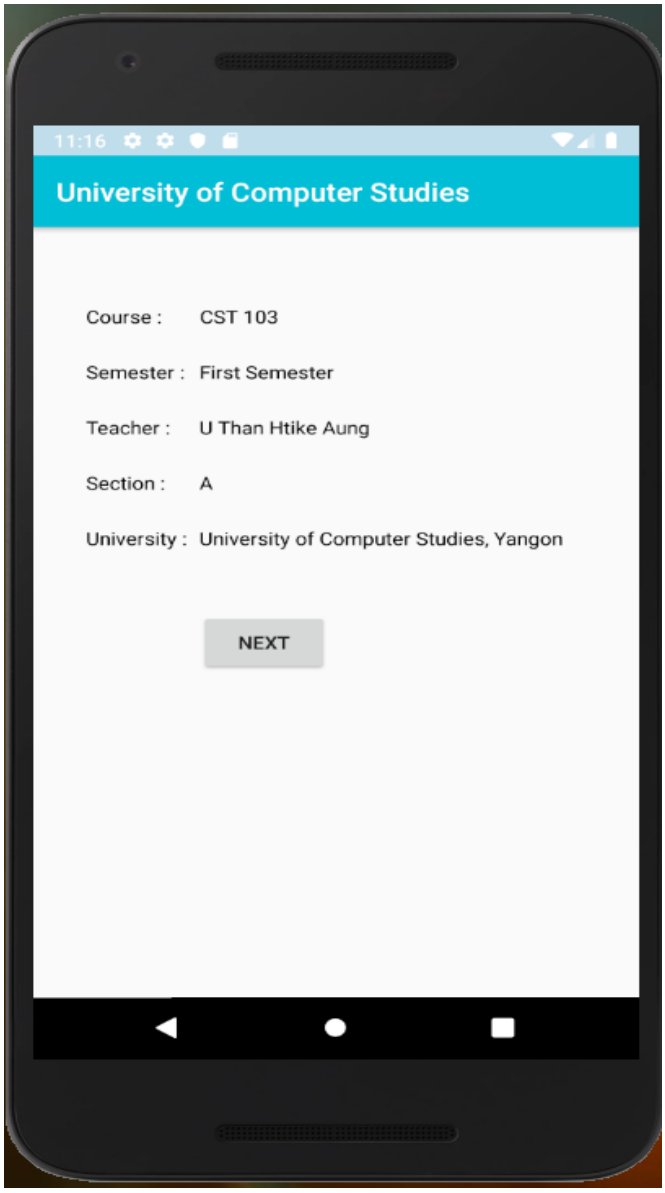

Figure 4.34 Teacher Workload Detail

4.30 Android- Based Data Collection Program for Update Teacher Workload

Figure 4.35 shown update teacher workload page. This page can update teacher workload data using this page.

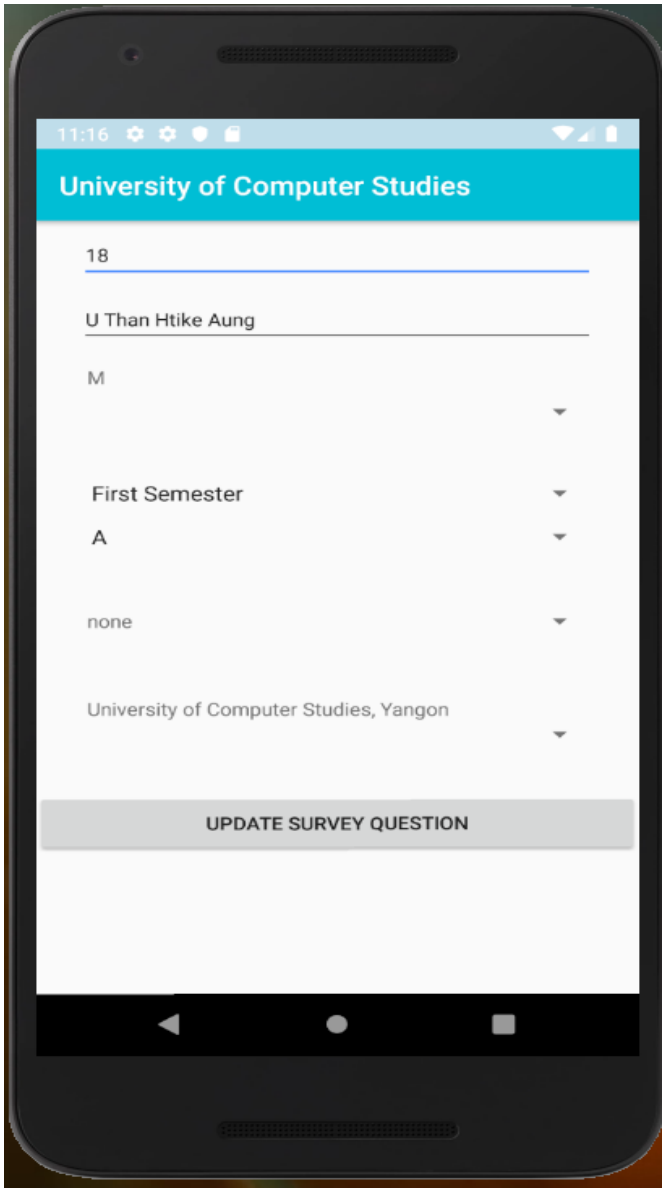

Figure 4.35 Update Teacher Workload

4.31 Android- Based Data Collection Program for Individual Teacher Assessment Survey Result

Figure 4.36 shown individual teacher assessment survey result page. When user click Teacher Assessment Survey Result button teacher assessment survey result will display for individual teacher.

Figure 4.36 Individual Teacher Assessment Result

4.32 Android- Based Data Collection Program for All Teacher Assessment Survey Result

Figure 4.37 shown for teacher All teacher assessment survey result page. When user click All Teacher Assessment Survey Result button then teacher assessment survey result will display for all teachers.

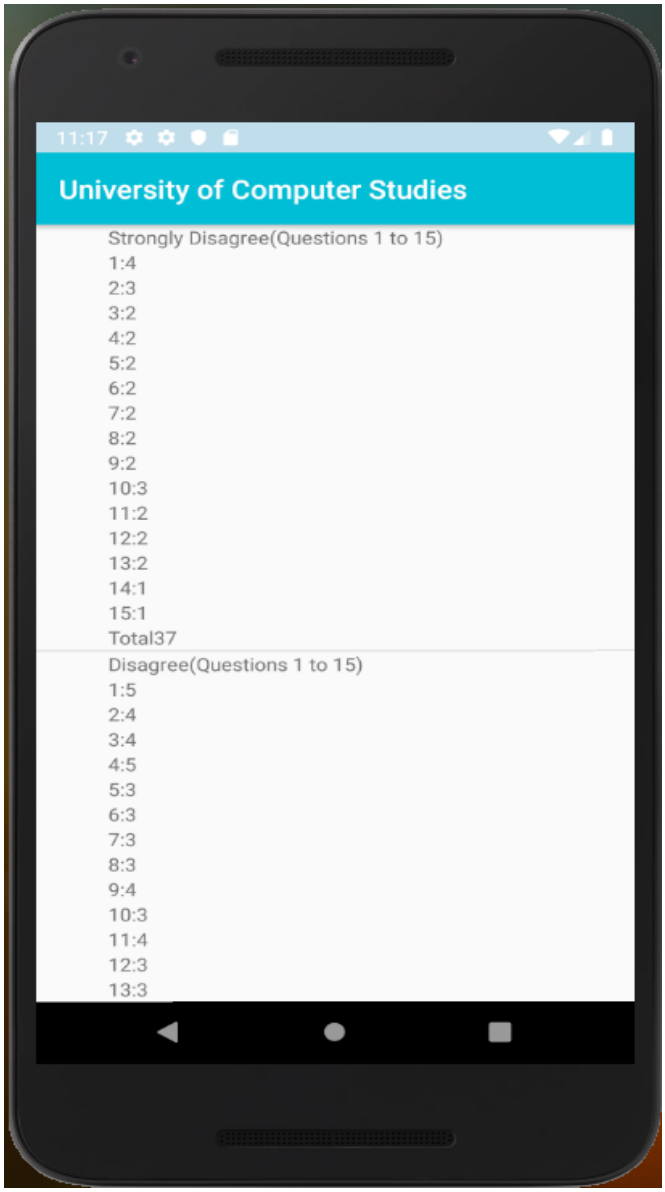

Figure 4.37 For All Teacher Assessment Survey Result

4.33 Android- Based Data Collection Program for Find Teacher Assessment Survey Result

Figure 4.38 shown find teacher assessment survey result page. Each registered teachers can find their teacher assessment survey result using find teacher assessment survey result page.

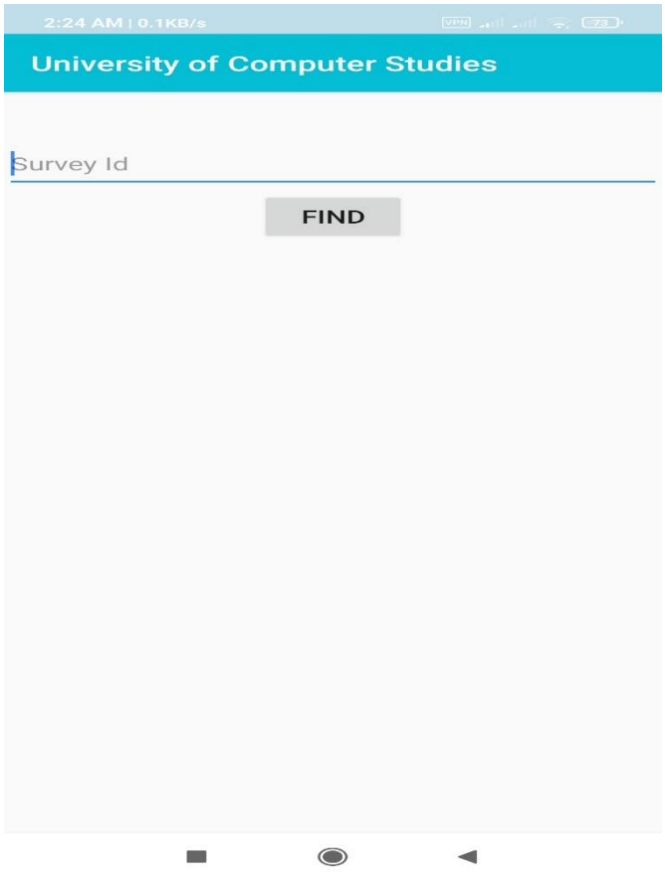

Figure 4.38 Find Teacher Assessment Survey Result

4.34 Android- Based Data Collection Program for Read All Teacher Workload Information

Figure 4.39 shown All teacher workload information read page. When user click Read All button All teacher workload information will display.

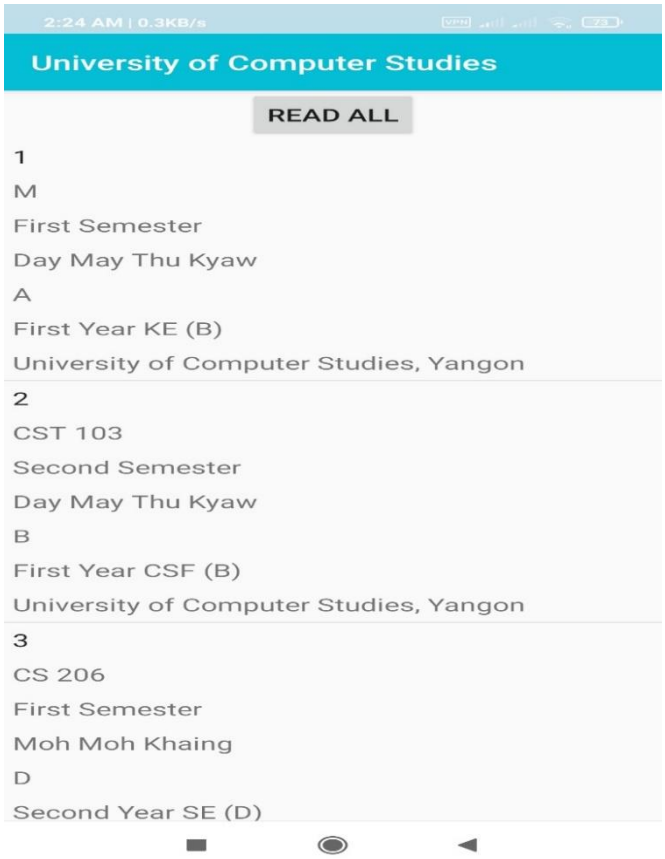

Figure 4.39 Read All Teacher Workload Information

4.35 Android- Based Data Collection Program for System Home

Figure 4.40 shown android based data collection program system home page. System administrator can start with ,STUDENT, TEACHER, UNIVERSITY, FACULTY, COURSE, GROUP AND ATTEMPT for teacher assessment survey .

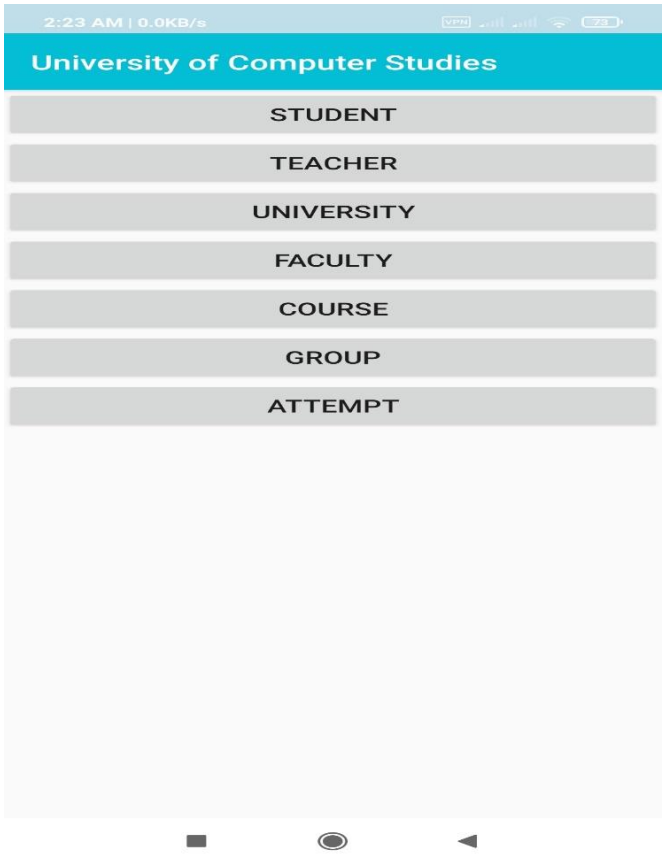

Figure 4.40 System Home

4.36 Android- Based Data Collection Program for System Data View, Insert, Update, Delete

Figure 4.41 shown android based data collection program for system administrator page contain CREATE/INSERT, READ, UPDATE, READ ALL and DELETE button. System administrator can create, read update, view student, teachers university, faculty, course, group and attempt teacher assessment survey data.

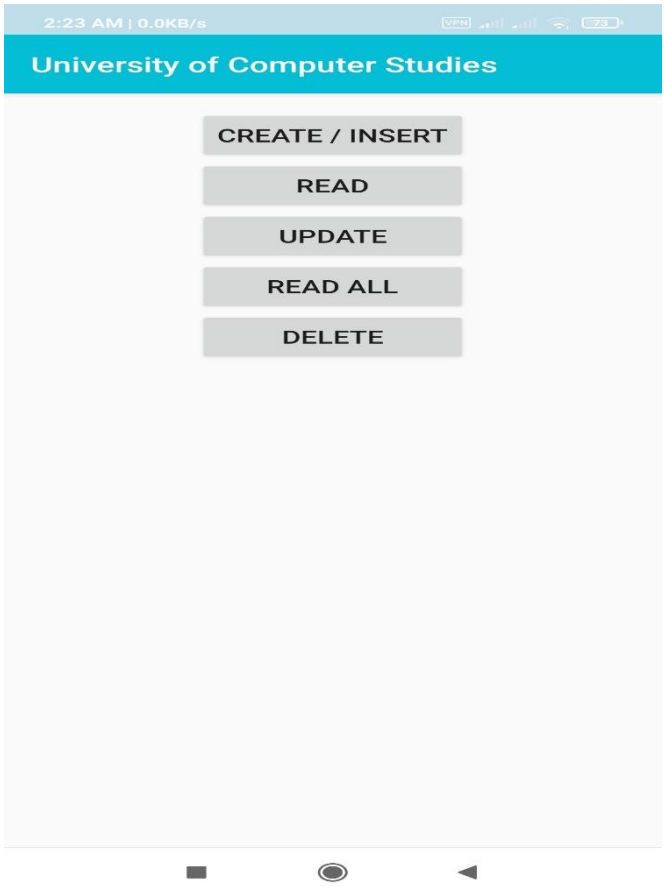

Figure 4.41 System View, Insert, Update, Delete

Table 4.1 shows Teacher Assessment dataset of University of Computer Studies, Yangon

Dataset	**Number of Transactions**	**Number of different Items**	**Average Transaction Width**
tass (teacher assessment survey data)	10011	75	15

Table 4.1 shows Teacher Assessment dataset of University of Computer Studies, Yangon (tass.dat). This is survey datasets of 10011 transactions in tass file which contains all the transactions surveying between 01/12/2014 to 09/12/2019 for University of Computer Studies, Yangon.

Attribute Information:

1. Timestamp (Integer)

2. Course (categorical)
3. Semester (categorical)
4. InstructorsName (categorical)
5. Section (categorical)
6. UniversityName (categorical)
7. q1, q2,....,q15 (questions 15) (categorical)
8. email (categorical)
9. gid (group ID) (Integer)
10. tcid (Teacher ID)(Integer)

4.37 Apriori Prefix Tree Eclat Program Implementation

Apriori Prefixtree Eclat (ATE)program(gattingstarted_mapreduce.jar) is ~~copy~~ (copied) to remote host oracle@10.0.2.15:/home/oracle.

4.38 User Program Copy to Remote Host Inline Command Statement

The comm4and is the following [Directory] Scp (stands for secure copy) [filename] [remote directory] [oracle@quickstart Desktop]$ scp gettingstarted_mapreduce.jar oracle@10.0.2.15:/home/oracle as shown in Figure 4.11.

Figure 4.42 shows inline command input of copy to remote location.

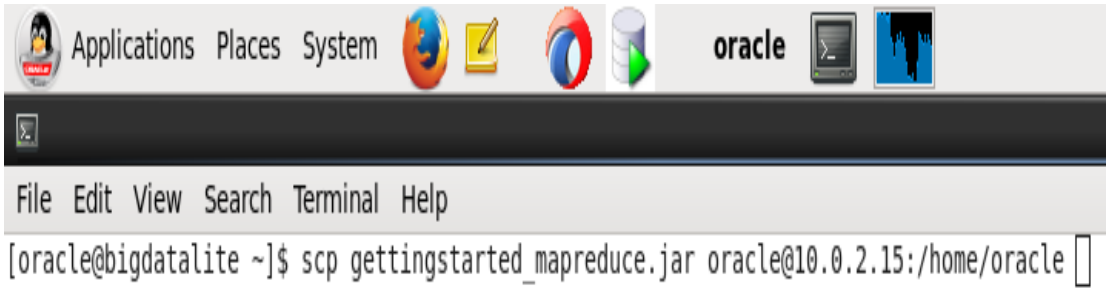

Figure 4.42 Program Run File Copy

4.39 User Program Inline Command Parameters Instruction

The following parameters are inline command parameters instruction for ATE and ET programs.

-i input-file: Mine the transaction database input-file. It should be already in computer home directory on HDFS.

-o output directory: Frequent itemsets and in-between files will be created in this folder. If the directory exists, it will be deleted first.

-s 60: Minimum support threshold for the frequent itemsets.

-m 2: Number of mappers that is going to be used. It is recommended to select this as the number of available machines.

-p 3: The depth of the prefix tree (length of the prefixes) to mine before distributing the search space further.

4.40 General Format Inline Command for Apriori Prefix Tree Eclat (ATE) and Eclat Prefix Tree(ET) Statement

[hadoop] [jar] [user jar file] [class file] [input file name] [output file name] [minimum support count][number of mappers depth of prefix tree]

4.41 User Program(Apriori Prefix Tree Eclat) Inline Command Statement

[oracle@bigdatalite ~]$ hadoop jar gettingstarted_mapreduce.jar be.uantwerpen.adrem.bigfim.BigFIMDriver -I /user/oracle/input/tass.dat -o /user/oracle/output -s 70 -m 2 -p 3

4.42 User Program Eclat Prefix Tree (ET) Inline Command Statement

[oracle@bigdatalite ~]$ hadoop jar gettingstarted_mapreduce.jar be.uantwerpen.adrem.disteclat.DistEclatDriver -i /user/oracle/input/tass.dat -o /user/oracle/output -s 70 -m 2 -p 3

Figure 4.43 shows inline command input of ATE program. Figure 4.4 shows runtime information display in program console.

Figure 4.43 Apriori Prefix Tree Eclat Program Run in Command Line

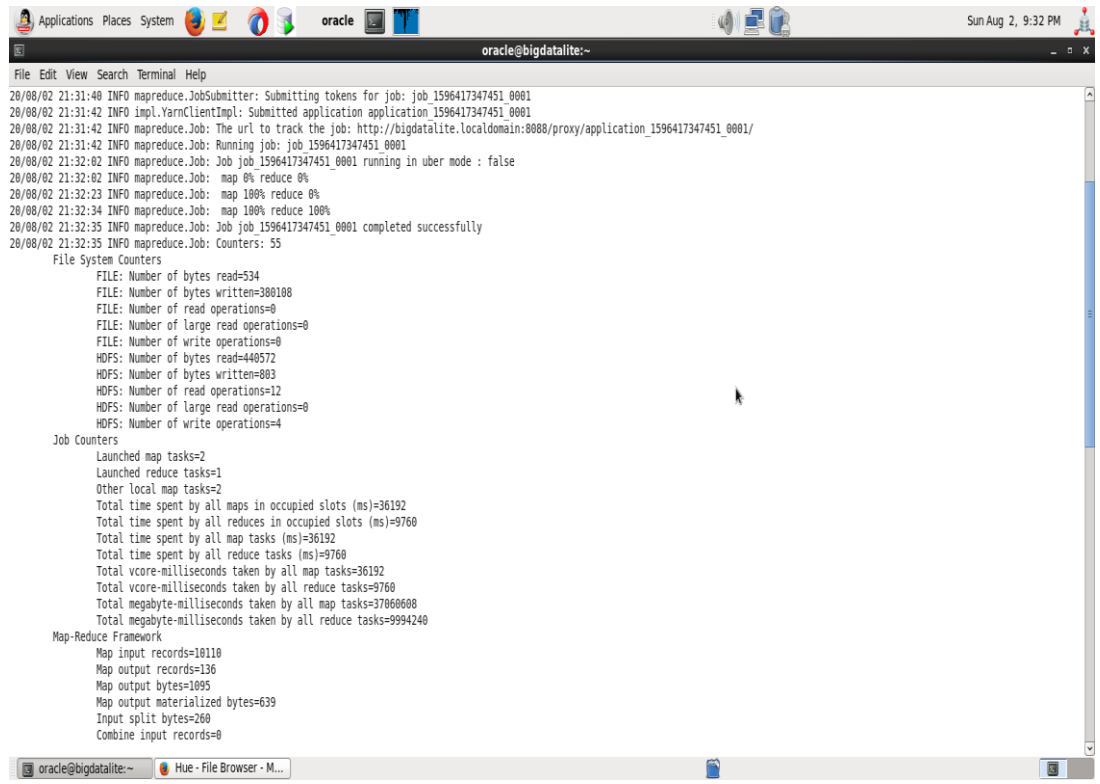

Figure 4.44 Apriori Prefix Tree Eclat Program Mining Run Time Information

4.43 General Format Inline Command for Trie Data to Frequent Pattern File Statement

[hadoop] [jar] [user jar file] [class file] [input file name] [output file name]

The above command is part-r-00000 file Trie Data format to user readable frequent pattern data file format.

4.44 User Program Trie Dumper Inline Command Statement

[oracle@bigdatalite ~]$ hadoop jar gettingstarted_mapreduceb.jar be.uantwerpen.adrem.eclat.util.TrieDumper /home/oracle/Desktop/output/part-r-00000 /home/oracle/Desktop/output/out.txt

Figure 4.45 show Frequent Itemset stored with Tri Format in part-r-00000 file in HDFS after ET and ATE programs are executed. Then, part-r-00000 file is generated to frequent itemset data file using TriDumper program.

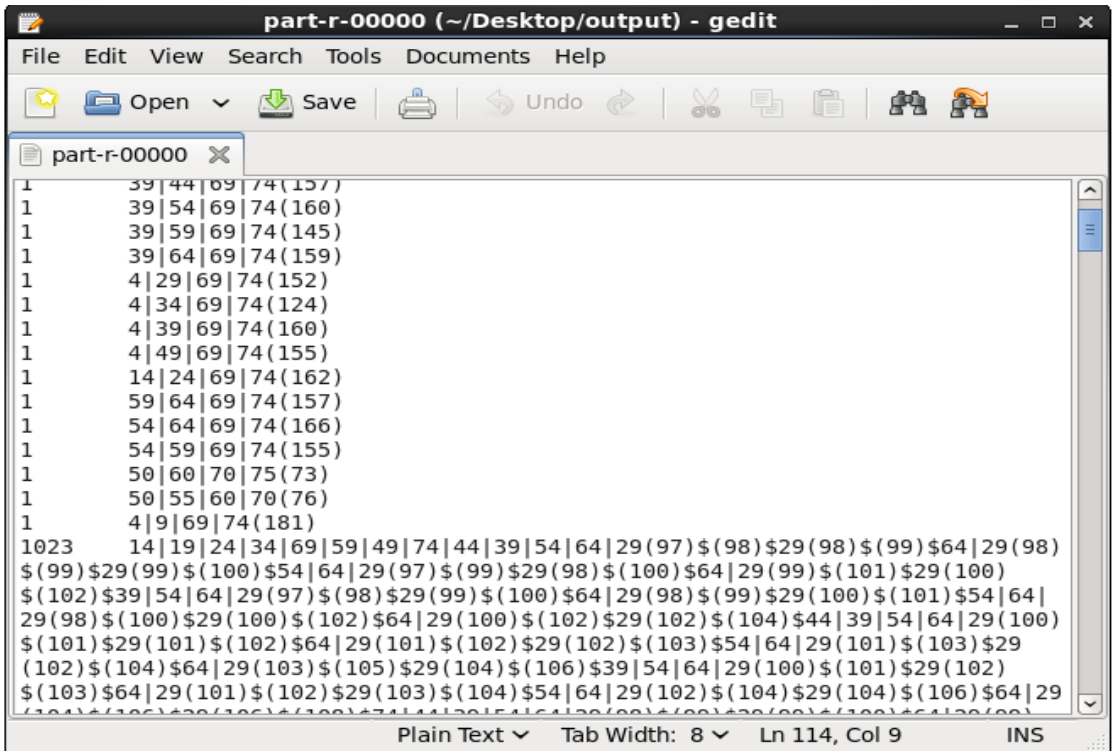

Figure 4.45 Show Tri Format in part-r-00000 file

Figure 4.46 shows inline command input of TrieDumper program.

Figure 4.46 Command part-r-00000 to Frequent Pattern File

Figure 4.47 shows final output is shown frequent itemset with support count.

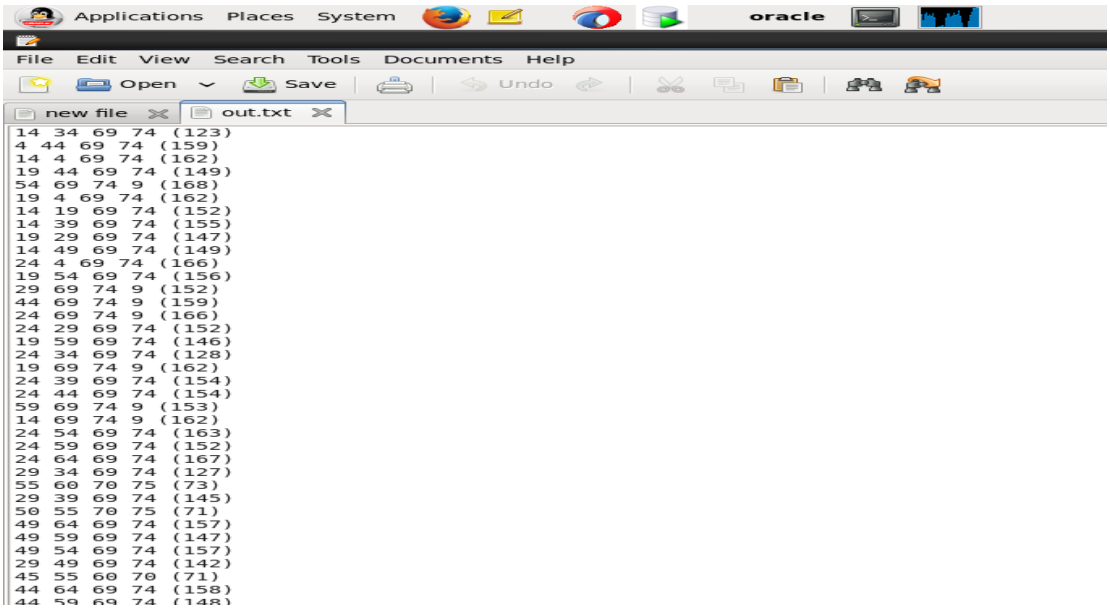

Figure 4.47 Result of Frequent Pattern File

4.45 Chapter Summary

This chapter present proposed system design of Apriori Prefix Tree Eclat. This proposed system is developed with input, AprioriMapReduce, PrefixTree, EclatMapReduce, Generate Frequent Pattern and Frequent Itemsets. This chapter also presents Frequent Pattern data flow diagram using Hadoop Mapreduce. The next chapter will be described implementation of the proposed system.

This chapter describes implementation of proposed system. This proposed system is developed Cent Operating System. This chapter also mentions data collection program. It is a android mobile program using google cloud service to be stored teacher assessment survey data. This dataset input into Apriori Prefix Tree Eclat program. Finally, proposed system generate frequent itemsets.

CHAPTER 5
EXPERIMENTAL RESULTS

This chapter discuss the scalable evaluation of the proposed system. The hadoop mapreduce based Apriori Prefix Tree Eclat (ATE) compared with Eclat Prefix Tree (ET) has used the three kinds of datasets, such as sample.dat, T10I4D100K.dat and ttas.dat.

Experimental data processing computer has used Core i5-2390T CPU, 16Gb RAM, Transcend's SATA III 6Gb/s SSD230S storage. The proposed teacher assessment dataset is applied to produce frequent item and scalable performance. The output results are illustrated by each type of datasets with their parameter value tables and graph.

The resulted frequent item sets are attained to local storage system for later use. This chapter goal is to evaluate scalable performance of the hadoop mapreduce based Apriori Prefix Tree Eclat.

There are several types of frequent pattern mining. One of them is frequent itemset mining. Frequent itemsets is one of the examples of frequent patterns. The result of teacher assessment datasets can be seen in hidden information and knowledge. The datasets becomes large in every academic year in universities in Myanmar. Thus, Teacher Assessment survey data is increasingly progressing to become large dataset. The proposed system can process large datasets by using hadoop mapreduce Apriori Prefix Eclat.

5.1 Datasets

Table 5.1 shows characteristics of datasets it (which is) used for experimental analysis in this proposed system. The T40I10D100K (dataset-1) has hundred thousand of transactions, thousands of different items and average transaction width with forty [ref]. The kddcup99 (dataset-2) also has one million of transactions , hundred and thirty- five different items and transaction width with sixteen.

Table 5.1 . Characteristics of the datasets used for experiment analysis

Dataset	Number of Transactions	Number of different Items	Average Transaction Width
T40I10D100K (dataset -1)	100000	1000	40
kddcup99 (dataset -2)	1000000	135	16

5.2 Execution Time Results between ATE and ET Method on Dataset-1

Table 5.2. shows execution time comparison on different methods used of dataset-1. ET and ATE methods use two parameters. It is support count= 0.2 and number of maps= 1,2,3,4,5. The execution time of proposed method ATE is less than ET method at each test results output.

Table 5.2 . Elapse Time (sec.) of mapreduce phases on dataset-1 (support count = 0.2) and (number of maps=1 ,2,3,4,5)

No of Mappers	Map=1	Map=2	Map=3	Map=4	Map=5
Support Count (%)	0.2	0.2	0.2	0.2	0.2
ET method (Time in sec)	79	80	80	81	81
ATEproposed method (Time in sec)	74	74	75	76	76

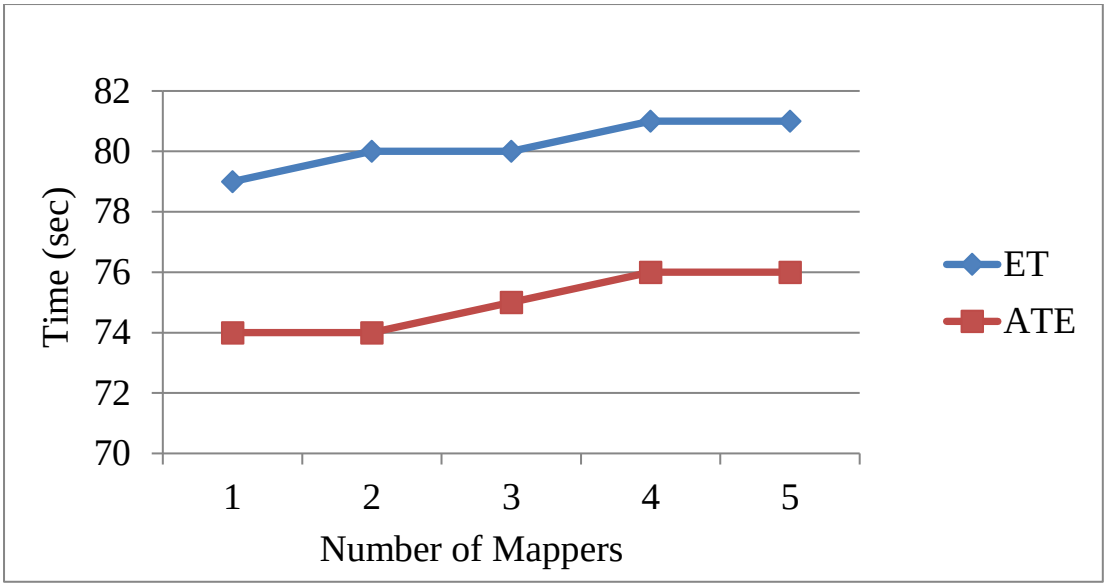

Figure 5.1 Execution Time and Mappers between ATE and ET with dataset-1

5.3 Execution Time Results between ATE and ET Method on Dataset-2

Table 5.3. shows execution time comparison on different methods used of dataset-1. ET and ATE methods use two parameters. It is support count= 0.1,0.15,0.2,0.25,0.3 and number of maps= 2. The execution time of proposed method ATE is less than ET method at each test results output.

Table 5.3 Elapse Time (sec.) of mapreduce phases on dataset-1 (support count= 0.1,0.15,0.2,0.25,0.3) and (number of maps=2)

No of Mappers	Map=2	Map=2	Map=2	Map=2	Map=2
Support Count (%)	0.1	0.15	0.2	0.25	0.3
ET method (Time in sec)	81	81	80	74	74
ATEproposed method (Time in sec)	74	74	74	71	71

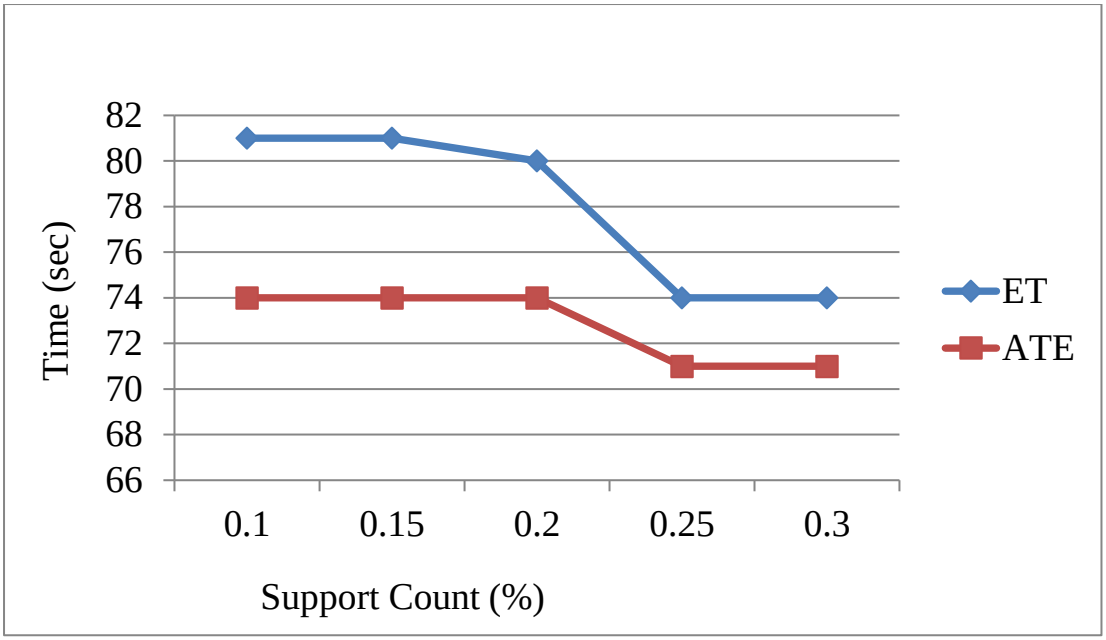

Figure 5.2 Execution Time and Support Count Between ATE and ET with dataset-1

Table 5.4. Shows execution time comparison on different methods used of dataset-2. ET and ATE methods use two parameters. It is support count= 0.2 and number of maps= 1,2,3,4,5. The execution time of proposed method ATE is less than ET method at each test results output.

Table 5.4 Elapse Time (sec.) of mapreduce phases on dataset -2 (support count = 0.2) and (number of maps=1 ,2,3,4,5)

No of Mappers	Map=1	Map=2	Map=3	Map=4	Map=5
Support Count (%)	0.2	0.2	0.2	0.2	0.2
ET method (Time in sec)	212	219	224	225	229
ATEproposed method (Time in sec)	209	211	216	218	222

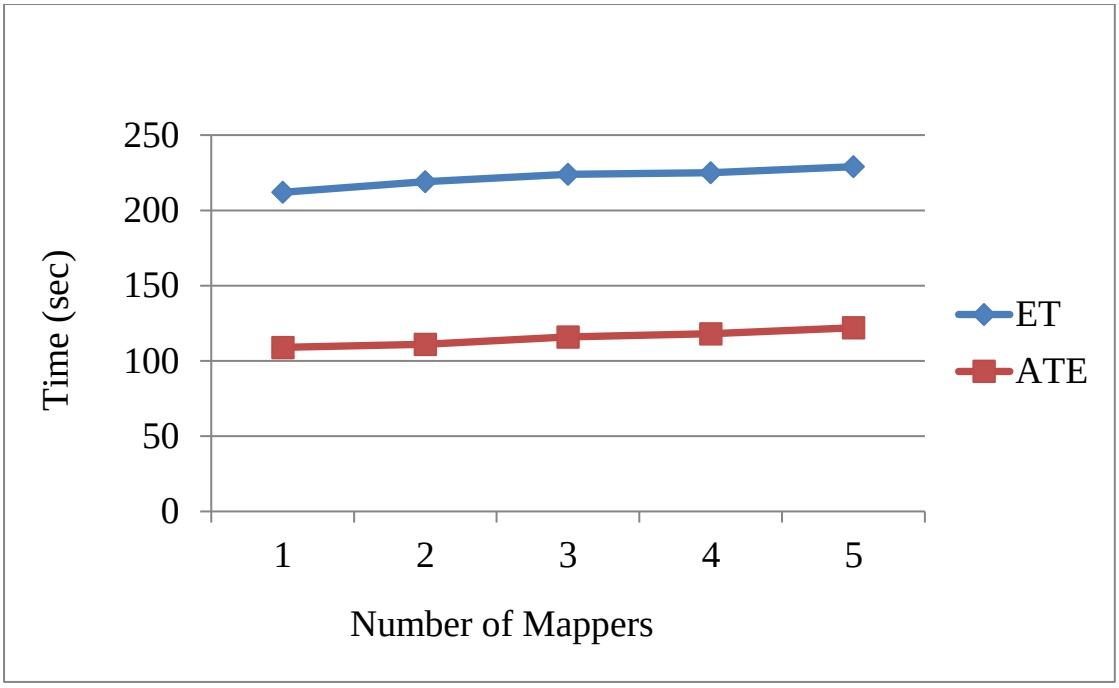

Figure 5.3 Execution Time and Mappers Between ATE and ET with dataset-2

Table 5.5. Shows execution time comparison on different methods used of dataset-2. ET and ATE methods use two parameters. It is support count= 0.1,0.15,0.2,0.25,0.3 and number of maps= 2. The execution time of proposed method ATE is less than ET method at each test results output.

Table 5.5 Elapse Time (sec.) of mapreduce phases on dataset -2 (support count = 0.1,0.15,0.2,0.25,0.3) and (number of maps=2)

No of Mappers	Map=2	Map=2	Map=2	Map=2	Map=2
Support Count (%)	0.1	0.15	0.2	0.25	0.3
ET method (Time in sec)	222	222	219	216	216
ATEproposed method (Time in sec)	114	114	111	100	100

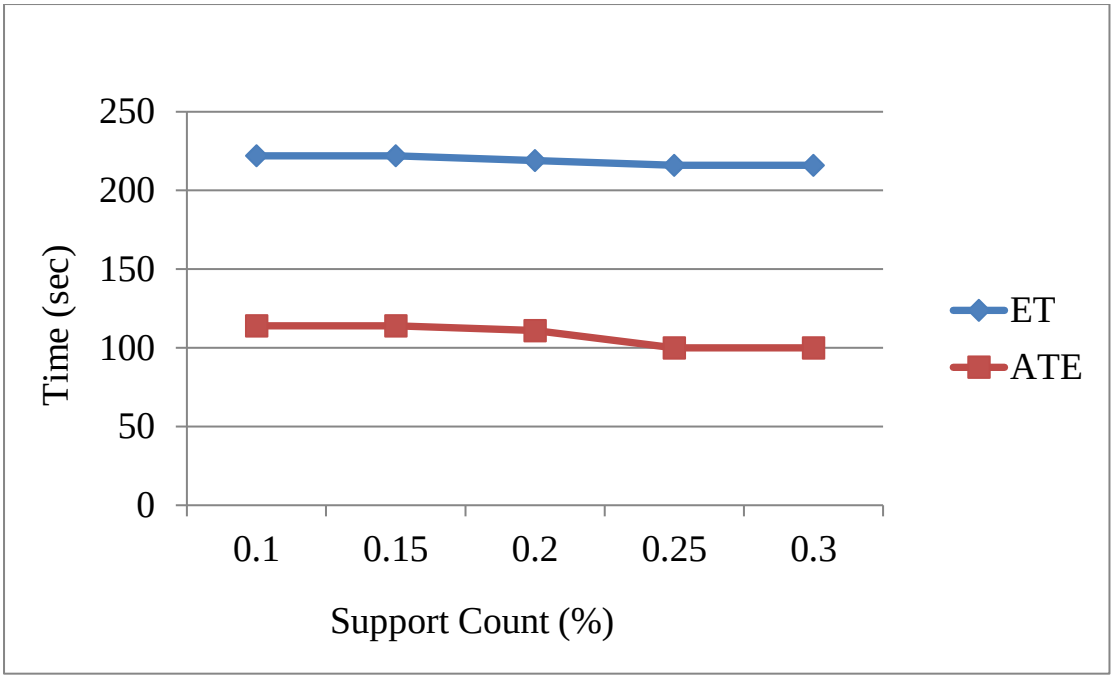

Figure 5.4 Execution Time and Support Count Between ATE and ET with dataset-2

5.4 Execution Time Results Between ATE and ET Method on Dataset-1 and Dataset-2

Table 5.6 Elapse Time (sec) comparison between two methods on dataset -1 and dataset-2 (support count=0.3, number of maps=2)

Dataset	in dataset-1	dataset-2
No of Mappers	Map=2	Map=2
Support Count (%)	0.3	0.3
ET method (Time in sec)	74	216
ATE proposed method (Time in sec)	71	100

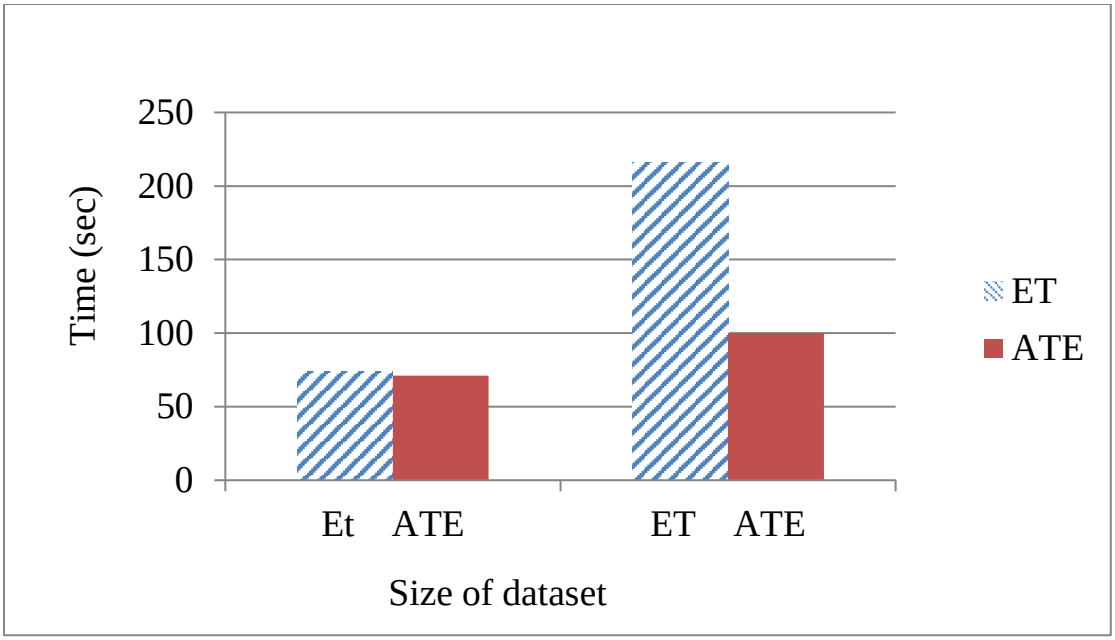

Figure 5.5 Execution Time and Size of Dataset Between ATE and ET with dataset-2

Experiments are performed on T10I4D100K and kddcup99 dataset and execution time required for generating number of frequent itemset is compared based on number of mappers and Minimum Support. Results shown that ATE is faster than ET on both T10I4D100K and kddcup99. ET algorithm is not working on large datasets. ET is not scalable enough and it faces memory problems as the dataset size increases.

Experiments performed on T10I4D100K dataset in order to compare execution time with different Minimum Support and number of mappers on ET and ATE. Figure 5.1. and Figure 5.2 shows time comparison for various methods on T10I4D100K dataset which shows that ATE has faster performance over ET algorithm. Execution time decreases as Minimum Support value increases which shows the effect of Minimum Support on execution time.

Execution time decreases as Minimum Support value increases which shows effect of Minimum Support on execution time. Execution time increases as number of mappers increases which shows effect of communication cost between mappers and reducers increases.

Results have been shown that ATE algorithm works on Large Data.

Experiments are performed on kddcup 99 dataset. ET algorithm faced memory problem with kddcup99. Results of ATE are compared with ET algorithm which is scalable.

Figure 5.3. and Figure 5.4 shows time comparison for various methods on T10I4D100K dataset which shows that ATE has faster performance over ET algorithm. Execution time decreases as Minimum Support value increases which shows effect of Minimum Support on execution time.

Execution time increases as number of mappers increases as communication cost between mappers and reducers increases.

Results have been shown in Figure 5.5 that ATE algorithm works on Large Data. Experiments are performed on kddcup 99 dataset. ET algorithm faced memory problem with kddcup99. Results of ATE are compared with ET algorithm which scalable.

5.5 Chapter Summary

This chapter discusses on the experimental comparison results of Apriori Prefix Tree Eclat (ATE) and Eclat Prefi Tree (ET) execute time. The input parameters are support count, number of maps and size of dataset. Finally, proposed system is better performance than other system.

CHAPTER 6
CONCLUSION AND FUTURE WORKS

In this system, . with the development of technology and the growth of computer users, the amount of data storage has also increased. FIM is essential part of data analysis and data mining. FIM extract information from a database that stores information. This system is divided into two parts. The first section describes the teacher assessment survey system; The second part builds a data mining model for large datasets.

Firstly, Teacher Assessment Survey System is an Android mobile application. It can be used in all computer universities. Teacher Assessment Survey software is built into a cloud database and can be accessed anywhere from the internet. The data obtained from it will become invaluable large datasets over time. They are intended for future use in the ATE model described in the second part. It can also be used for university student research if needed.

Finally, Teacher Assessment Survey data will turn into big data over time. Conventional tools cannot be used to analyze this data. Therefore, it is necessary to model and research to analyze large data. The proposed system described in above is a model that can be used for large data. It combines simple Apriori, Eclat and Prefix Tree. Also built to work with Hadoop and Mapreduce frameworks. Today's large data sets facilitate research using open source Hadoop and Mapreduce frameworks. It is reduces file size and saves storage space.

Chapter 2 described reviews study. Early in the study, large data were constructed using statistic models. FIMs were also designed as distributed systems. Clustering models were then modeled with Mapreduce. The main problem with this is the increased memory requirements. Although the Eclet model can be used with distributed parallel processing, there are inconveniences as the amount of data increases. The main problem is that when candidate data increases, they cannot be counted. Therefore, to solve this problem, this proposed system must be implemented using the Apriori Prefix Tree Eclat method. This proposed system is described in Chapter 4. Examples of teacher assessment survey data are also provided as case studies.

In Chapter 3, the main theory background of frequent itemset mining

algorithms are described, and its algorithms are Apriori. Eclat, FP-Growth and Mapreduce method. Chapter 4 presents detail outline for frequent itemsets mining. In this program, systematically described the tree structure then Apriori and Eclat algorithms in combination with Mapreduce. This chapter presents details the user usage of the Teacher Assessment Survey System and command line notes for Apriori Prefix Tree Eclat. Chapter 5 described the comparison experimental results of proposed system ATE and ET.

6.1 Summary

This system is mainly divided into two main sections. The first part is to conduct a teacher evaluation survey of teachers working in universities through a mobile application. This is because the data will become a large dataset in the future. This information is essential in the field of education and should be used by research students and administrators as needed.

The second part requires a program to organize this information. Therefore, it is necessary to build a way to face large data. Therefore, ATE was built as a way to extract large amounts of data and save memory. The proposed ATE is analyzed with FIM data for comparison with ET.

In future educational survey data storage will be increased in volume. Thus, large data set analysis can be used proposed frequent item set mining algorithm based on MapReduce Hadoop framework.

Apriori Prefix Tree Eclat (ATE) finds frequent itemsets having large itemset size. This algorithm can be resolved memory problem. Hadoop MapReduce platform can be used extensively for mining large amount of educational dataset.

6.2 Future Works

This research is concerned with educational big data mining process for large data. In the future, various type of data can be resolved with more accurate datamining algorithms.

6.3 Benefits and Limitations

The proposed methods can be used for large datasets by using hybrid methods. The Apriori algorithm finds frequent itemsets until their list of

transactions fit into the memory. Then construct prefix tree. Finally, mining the prefix tree using the Eclat algorithm complete frequent itemset mining.

The proposed method (ATE) can be used for large databases. When database is small use another method just for speed up. It produces more data than the original databases, if the data is too large, sampling can be a better option. Teacher Assessment Survey System mobile application is a very useful data collection software program for Educational Data Mining. This proposed method can be used only DAT file types.

AUTHOR'S PUBLICATIONS

[p1] Than Htike Aung and Nang Saing Moon Kham, University of Computer Studies, Yangon Myanmar, "Application of Apriori algorithm and K-Means Clustering algorithm based on Cloud based Big Data", 15th International Conference on Computer Applications (ICCA 2017). University of Computer Studies, Yangon Myanmar. Feb 16-17 ,2017,Page 58-61

[p2] Than Htike Aung and Nang Saing Moon Kham, University of Computer Studies, Yangon Myanmar, "Cloud Based Big Data Application of FP-Growth Algorithm and K-Means Clustering Algorithm Based on MapReduce Hadoop", 1st International Conference on Advanced Information Technologies (ICAIT), Nov 1-2 ,2017,Page 14-19

[p3] Than Htike Aung and Nang Saing Moon Kham, University of Computer Studies, Yangon Myanmar, "Frequent Pattern Mining for Stream Data by Using Hadoop GM-Tree and GTree", 16th International Conference on Computer Applications (ICCA 2018), Feb 22-23 2018, Page 65-71

[p4] Than Htike Aung and Nang Saing Moon Kham, University of Computer Studies, Yangon Myanmar, "Frequent Pattern Mining for Dynamic Database by Using Hadoop GM-Tree and GTree ", 1st International Conference on Big Data Analysis and Deep Learning, June 2018.pp 149-159

[p5] Than Htike Aung and Nang Saing Moon Kham, University of Computer Studies, Yangon Myanmar, Yangon Myanmar, "Cloud Based Teacher's Assessment Data in Educational Data Mining ", The 11th International Conference on Future Computer and Communication (ICFCC 2019), March 1 ,2019, pp 159-164

[p6] Than Htike Aung, Nang Saing Moon Kham and Soe Soe Mon, University of Computer Studies, "Application of RFID Based Student Attended System Using Cloud Storage and IoT", 17th International Conference on Computer Applications (ICCA 2019), March 2, 2019, Page 50-53

[p7] Than Htike Aung, Nang Saing Moon Kham and Soe Soe Mon, International Journal of Computer Applications, Foundation of Computer Science (FCS), NY, USA, Improved Frequent Pattern Mining for Educational Data by using Mapreduce Approach in Hadoop ", Volume 175 - Number 37, 13-20, December 2020

BIBLIOGRAPHY

[1] BIRKINSHAW, Julian. Beyond the Information Age [online]. 2014 [visited on 2018-05-20]. Available from: https : / / www . wired . com / insights/2014/06/beyond-information-age/.

[2] LESKOVEC, Jure; RAJARAMAN, Anand; ULLMAN, Jeffrey D. Mining of Massive Datasets, 2nd Ed. Cambridge University Press, 2014. ISBN 978-1107077232.

[3] GORUNESCU, Florin. Data Mining - Concepts, Models and Techniques. Springer, 2011. Intelligent Systems Reference Library. ISBN 978-3- 642-19720-8. Available from DOI: 10.1007/978-3-642-19721-5.

[4] AGGARWAL, Charu C.; HAN, Jiawei (eds.). Frequent Pattern Mining. Springer, 2014. ISBN 978-3-319-07820-5. Available from DOI: 10 . 1007/978-3-319-07821-2.

[5] ALVAREZ, Sergio A. Chi-squared computation for association rules: preliminary result. In: Technical Report BC-CS-2003-01. 2003. Available also from: http : / / www . cs . bc . edu / ~alvarez / ChiSquare / chi2tr.pdf.

[6] AGRAWAL, Rakesh; SRIKANT, Ramakrishnan. Fast Algorithms for Mining Association Rules in Large Databases. In: VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, Septem- ber 12-15, 1994, Santiago de Chile, Chile. 1994, pp. 487–499. Available also from: http://www.vldb.org/conf/1994/P487.PDF.

[7] AGRAWAL, Rakesh; IMIELINSKI, Tomasz; SWAMI, Arun N. Mining Association Rules between Sets of Items in Large Databases. In: Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, Washington, D.C., May 26-28, 1993. 1993, pp. 207–216. Available from DOI: 10.1145/170035.170072.

[8] ZAKI, Mohammed Javeed; GOUDA, Karam. Fast vertical mining us- ing diffsets. In: Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Washington, DC, USA, August 24 - 27, 2003. 2003, pp. 326–335. Available from DOI: 10.1145/956750.956788.

[9] AGARWAL, Ramesh C.; AGGARWAL, Charu C.; PRASAD, V. V. V. Depth first generation of long patterns. In: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining, Boston, MA, USA, August 20-23, 2000. 2000, pp. 108–118. Available from DOI: 10.1145/347090.347114.

[10] HAN, Jiawei; PEI, Jian; YIN, Yiwen. Mining Frequent Patterns with- out Candidate Generation. In: Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data, May 16-18, 2000, Dallas, Texas, USA. 2000, pp. 1–12. Available from DOI: 10.1145/342009.335372.

[11] ZAKI, Mohammed Javeed; PARTHASARATHY, Srinivasan; OGIHARA, Mitsunori; LI, Wei. New Algorithms for Fast Discovery of Association Rules. In: Proceedings of the Third International Conference on Knowledge Discovery and Data Mining (KDD-97), Newport Beach, California, USA, August 14-17, 1997. 1997, pp. 283–286. Available also from: http://www.aaai.org/Library/KDD/1997/kdd97-060.php.

[12] ZAKI, Mohammed Javeed. Scalable Algorithms for Association Mining. IEEE Trans. Knowl. Data Eng. 2000, vol. 12, no. 3, pp. 372–390. Available from DOI: 10.1109/69.846291.

[13] LIN, Goh Chun; DESMOND, Koh Eng Tat; HTOON, Naing Tayza; THUAT, NV. A Fresh Graduate's Guide to Software Development Tools and Technologies. Chapter-6: Scalability, School of Computing, National University of Singapore. 2012.

[14] DEAN, Jeffrey; GHEMAWAT, Sanjay. MapReduce: Simplified Data Processing on Large Clusters. In: 6th Symposium on Operating Sys- tem Design and Implementation (OSDI 2004), San Francisco, California, USA, December 6-8, 2004. 2004, pp. 137–150. Available also from: http://www.usenix.org/events/osdi04/tech/dean.html.

[15] HURWITZ, Judith; NUGENT, Alan; HALPER, Dr. Fern; KAUFMANN, Marcia. Big Data for Dummies. John Wiley & Sons, Inc., 2013. ISBN 978-1-118-50422-2.

[16] SAVASERE, Ashok; OMIECINSKI, Edward; NAVATHE, Shamkant B. An Efficient Algorithm for Mining Association Rules in Large

Databases. In: VLDB’95, Proceedings of 21th International Conference on Very Large Data Bases, September 11-15, 1995, Zurich, Switzerland. 1995, pp. 432–444. Available also from: http : / / www . vldb . org / conf/1995/P432.PDF.

[17] A. Tabarcea, V. Hautamäki, P. Fränti,”AD-HOC GEOREFERENCING OF WEB-PAGES USING STREET-NAME PREFIX TREES”, 6th International Conference on Web Information Systems and Technologies,April-2010, DOI: 10.1007/978-3-642-22810-0_19 ·

LIST OF ACRONYM

EDM	Educational Data Mining
LA	Learning Analytics
ATE	Apriori Prefix Tree Eclat
ET	Eclat Prefix Tree