
# Midterm Project – CS634 Data Mining

Instructed by Prof. Jason Wang


| | |
|---|---|
| First-name: | Rahul Gautham |
| Last-name: | Putcha |
| NJIT-ID: | 31524074 |
| UCID | RP39 |
| Email-address: | rp39@njit.edu |

# TABLE OF CONTENT

# List of Tables and Figures

# Chapter 1. General Introduction

With the amount of data increasing at an astounding rate, it has become difficult for people to get useful insights from the large dataset in real-time. Data mining has provided us with a vast set of algorithms that are being used in a variety of applications and have proved to be reliable from time to time.

## 1.1 The Frequent itemset generation problem

One of the common problems in data mining, and the one we are going to face in this project is the frequent itemset generation problem. As presented in the paper, Agrawal et al [2], describes this problem as follows:

*Given examples that are sets of items and a minimum frequency, any set of items that occurs at least in the minimum number of examples is a frequent itemset.*

## 1.2 About Association rules

One of the reliable and most used data mining algorithms, the Apriori Algorithm, is used for generating association rules. Association rules are useful for analyzing and predicting customer behavior. They also play an important part in customer analytics, market basket analysis, product clustering, catalog design, and store layout.

The focus of this project is on Association rule mining, where we generate interesting association rules from a user's past shopping transactions. The algorithms we are focusing on, especially for generating the frequent-itemset over the past transactions, will include the Brute-force method and Apriori algorithm. These algorithms provide a rule-based machine learning method for discovering interesting relations between variables in large databases.

Before we delve into the details of these algorithms, there is a need to familiarize ourselves with a few of the basic terminologies.

## 1.3 Basic Terminologies

In this section, we will focus on a few vocabulary terms and concepts which will help us in understanding the algorithm better.

| | **Term** | **Definition** |
|---|---|---|
| 1. | $Support_{(X)}$ | The support of item X is defined as the,<br>$Support_{(X)} = \frac{\text{(number of transactions containing X)}}{\text{(the total number of transactions)}}$<br>It gives the popularity of an item X. |
| 2. | $Confidence_{(X \rightarrow Y)}$ | The confidence of association rule X→Y is given,<br>$Confidence = \frac{\text{Support(X and Y))}}{\text{Support(X)}}$<br>It gives the likelihood of item Y being bought when item X is bought. |

**Tab 1.1 Basic Terminologies required for Frequent-itemset generation problem**

In the next section, we will look into the algorithms used to generate Frequent-itemset and association rules.

# Chapter 2. Algorithms for Frequent-itemset generation

In this chapter, we look at the algorithms that we will be using for the remainder of the project. These algorithms are based on Association Rule Mining and Apriori algorithm as mentioned in Agrawal et al.

We first begin with the Brute-force method and discuss the drawbacks of using it. Later in this chapter, we will discuss how the Apriori algorithm is used to resolve the drawbacks specific to the Brute-force method.

## 2.1 Brute-Force method

The brute-force method depicts the first logic of how humans generally think when resolving a problem. In the case of a frequent itemset generation problem, given a list of transactions and the shopping items, we usually generate a list of candidates out of the shopping list and check out which one is frequent. If we have more than one frequent item in the list, we take a combination of two items and check how frequent items are when bought together with another. Again, if we see more than one item sets then we now take a combination of three items from the shopping list and carry out the elimination. We repeat this process till we find that no more itemset is generated in the next step. A sample of the algorithm is given as follows.

```
Support(C)
    for transactions t in T
        D_t ← {c in C : c ⊆ t}
        for candidates c in D_t
            count[c] ← count[c] + 1
    return count

Bruteforce(T, ε)
   C_0 ← {large 1 - itemsets} # shopping item list
   L_0 ← {c in C_0 : support(c) ≥ ε}
   k ← 0
   while L_k is not empty
      C_k+1 ← GenerateItemset(C_k , k) # Note we take in the candidate set itself for generating the next candidate
      count ← Support(C_k+1 )
      L_k+1 ← {c in C_k+1 : count[c] ≥ ε}
      k ← k + 1
   if L_0 is empty then return L_0
   return Union(L_k-1)

GenerateItemset(L, k)
    result ← list()
    for all p ⊆ L, q ⊆ L where p_1 = q_1, p_2 = q_2, ..., p_k-2 = q_k-2 and p_k-1 < q_k-1
       c = p ∪ {q_k-1}
       if u ⊆ c for all u in L
          result.add(c)
    return result
```

**Fig 2.1. Brute-force method**

For each iteration generate a new combination of candidate itemsets taken from the shopping list and carry out the elimination of items (or item sets) that are not frequent. We can already picture this process

as cumbersome as it sounds by its name. Therefore, we shift our focus to the Apriori algorithm and then discuss how resolves the search space of candidate generation.

## 2.2 Apriori Algorithm

The brute-force method is a useful algorithm for generating frequent item sets. But as the dataset on which it works become larger and larger, the searching of frequent itemset becomes tedious and time-consuming even for a computer.

Apriori algorithm focus on the core principle that state,

*If an itemset is infrequent, then all its supersets must also be infrequent. … [Apriori Principle]*

Taken in a contrapositive manner,

*If superset is frequent, then all of its subsets itemset are also frequent.*

This principle helps in reducing the problem search space, thereby increasing the performance and simplifying the process of generating the frequent itemset by a large magnitude.

We can start updating our previous solution of the Brute-force method to the Apriori algorithm by generating the next candidate list from the previously generated frequent itemset. In this way, we also follow the Apriori principle mentioned above.

```
Support(C)
    for transactions t in T
        D_t ← {c in C : c ⊆ t}
        for candidates c in D_t
            count[c] ← count[c] + 1
    return count

Apriori(T, ε)
    C_0 ← {large 1 - itemsets} # shopping item list
    L_0 ← {c in C_0 : support(c) ≥ ε}
    k ← 0
    while L_k is not empty
        C_k+1 ← AprioriGen(L_k , k) # Note we take in the previously generated frequent list
        count ← Support(C_k+1 )
        L_k+1 ← {c in C_k+1 : count[c] ≥ ε}
        k ← k + 1
    if L_0 is empty then return L_0
    return Union(L_k-1)

AprioriGen(L, k) # a.k.a, GenerateItemset(L, k)
    result ← list()
    for all p ⊆ L, q ⊆ L where p_1 = q_1, p_2 = q_2, ..., p_k-2 = q_k-2 and p_k-1 < q_k-1
        c = p ∪ {q_k-1}
        if u ⊆ c for all u in L
            result.add(c)
    return result
```

**Fig 2.2. Apriori algorithm (a modified Brute-force algorithm)**

This is the reason why the Apriori algorithm is faster than the Brute-force method. We are generating the next combination of items in the candidate list from the previously generated frequent itemset rather than build it from scratch using the entire shopping list.

## 2.3 Association rules generation

Generating the frequent-itemset generating association rules are fairly simple. The algorithm for generating association rules is given below.

```
AssociationRule(FREQ_Itemsets, min_confidence)
  R ← { }                          # set of all association rules
  for all  I in FREQ_Itemsets
    P_I ← PowerSet(I) excluding the empty set and the I itself
    for all S in P_I
      if min_confidence <= Confidence(S → I − S)
        add rule S → I − S to R
  return R
```

**Fig 2.3. Algorithm for generating Association Rules**

Here, for every Frequent item set, say I, that is generated we will start by generating all possible subsets except for the empty set and the itemset itself. With the subsets we have, for every subset, say S, we will generate the rule S→(I – S) only if the Confidence $_{S\rightarrow(I-S)}$ is greater than the user specified threshold.

## 2.4 Trick for getting Support $_{(\{x1, x2, x3,\ldots,xn\})}$ from the Transaction database

Since it is necessary to have a good design for a database, databases must always scale in height and stay fixed at width as much as possible. In a later section, we will see the complete design for the database. In this section we will give a way for us to calculate Support$_{(X)}$, where X is a set of any non-zero number of elements. For the sake of simplicity, we showcase the algorithm using Relational algebra as follows.

T = transaction_details table

Level 1 support: $\mathcal{F}_{count\ (*)}\ (\prod_{Tid}\ (\sigma_{label=x1}\ (T)))$

Level 2 support: $\mathcal{F}_{count\ (*)}\ (\prod_{Tid}\ (\sigma_{label=x1}\ (T)) \cap \prod_{Tid}\ (\sigma_{label=x2}\ (T))\ )$

Level 3 support: $\mathcal{F}_{count\ (*)}\ (\prod_{Tid}\ (\sigma_{label=x1}\ (T)) \cap \prod_{Tid}\ (\sigma_{label=x2}\ (T)) \cap \prod_{Tid}\ (\sigma_{label=x3}\ (T))\ )$

… … … …

Level n support: $\mathcal{F}_{count\ (*)}\ (\prod_{Tid}\ (\sigma_{label=x1}\ (T)) \cap \prod_{Tid}\ (\sigma_{label=x2}\ (T)) \cap \ldots \cap \prod_{Tid}\ (\sigma_{label=xn}\ (T))\ )$

**Fig 2.4. Relational algebra for calculating Support$_{(X)}$**

Above algorithm involve intersections of several SQL query searches and aggregating the combined result we find the count of transaction with involving X. We repeat the process for all the sets in the item sets for getting their support during the execution of a frequent-itemset algorithm.

With this, we end this chapter by encompassing all algorithms that are required for us in solving the association rule mining problems. The next chapter will look at the system and user-specific requirements for this project.

# Chapter 3. Requirement Specifications

This chapter will present some details on what is required for a user to run this application.

## 3.1 System requirement

Below is the System requirement for running the project file belonging to this application.

| | Package | Version Description |
|---|---|---|
| 1. | **Operating System:**<br>Windows/Linux/Max | Windows Server 2008 and later Windows Vista and later<br>Mac OS X 10.6 and later CentOS, Ubuntu, Fedora,<br>Gentoo, Arch, SUSE (please see XAMPP specification) |
| 2. | **Software Used:**<br>XAMPP or LAMP<br>• MariaDB (MySQL)<br>• PHP<br>• Apache HTTP Server<br>HTML, CSS, JS<br>IDE (Visual Studio code) | <br><br>MySQL Server 5.0 or higher<br>PHP 5<br>Version 2.2.x<br>HTML 5, CSS 3, JavaScript ES6<br>(optional) |

**Tab 3.1. System requirements**

For more details on the space requirements please comply with XAMPP or LAMP specifications.

## 3.2 Function requirement (Specific to the Midterm Project only)

Following are the Functional requirements that are met in this project. These requirements are specific to the Midterm project description only.

| Req. | Title |
|---|---|
| FR1 | Display 30 user-selectable shopping items |
| FR2 | Enable user to make transactions containing shopping items |
| FR2.1 | User can update a cart for storing items selected for current transaction |
| FR2.2 | Store transactions in a database specific to a single user<br>(for at least 5 users with each containing at least 20 transactions) |
| FR3 | User is shown with list of all of their past transaction |
| FR4 | User is provided a form for inputting minimum support and minimum confidence |
| FR5 | Perform frequent-itemset generation (by means of Apriori Algorithm using info. from FR5) |
| FR5.1 | Generate Association rules by using info. from FR4 and FR5 |
| FR5.2 | User gets to see the Association rules generated from FR5.1 |
| FR5.3 | User gets to see the execution time for running the algorithm specified in FR5 |
| FR6 | Perform frequent-itemset generation (by means of Brute-force method using info. from FR5) |
| FR6.1 | Generate Association rules by using info. from FR4 and FR6 |
| FR6.2 | User gets to see the Association rules generated from FR6.1 |
| FR6.3 | User gets to see the execution time for running the algorithm specified in FR6 |

**Tab 3.2. Functional requirements**

# Chapter 4. Setting the Server and Project Files

This project is set up on an Apache server using the XAMPP opensource application. It is an alternative for LAMP (Linux Apache MariaDB PHP) which is also suitable for setting the project file specified for this application software. For more details on XAMPP, please visit https://www.apachefriends.org/index.html.

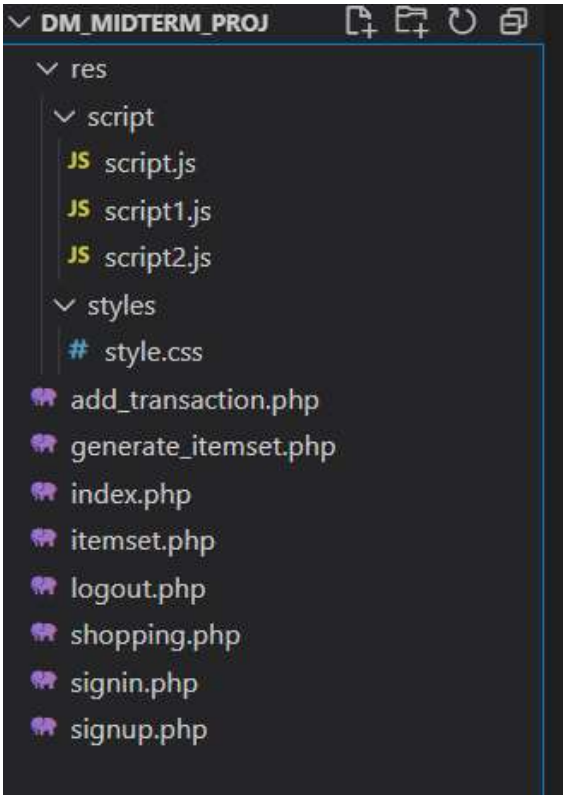



**Fig 4.1. Application Folder structure**

We will now present the code for setting up our project files. The application folder structure is shown in the above image taken from the *Visual code studio* editor. We will start the setup from the backend and move into the frontend which integrally holds the main code that provides the functioning of our application.

## 4.1 Project code (MySQL), for the admin setup only

In this section, we will provide the code for the MySQL admin only and later discuss the design aspect of the database by giving an ER-Diagram. If you want to delve right into the design presentation via ER-diagram, please see the chapter on *Designing the Shopping Database*.

SQL Code:

```
CREATE DATABASE `cs634dmadmin`;
USE `cs634dmadmin`;
```

```
CREATE TABLE `shopping_list` (
  `item_id` int(5) NOT NULL,
  `label` varchar(25) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

--
-- Setting our shopping list
--
INSERT INTO `shopping_list` (`item_id`, `label`) VALUES
(1, 'Jewelry'),(2, 'Beads'),(3, 'God Statue'),(4, 'Skin cream'),(5, 'Pendant'),(6, 'Eng. Book'),(7, 'Chalk'),(8,
'Ball'),(9, 'Staplers'),(10, 'Toy'),(11, 'Computer'),(12, 'Tea'),(13, 'Scissors'),(14, 'Headsets'),(15, 'Wine'),(16,
'Diaper'),(17, 'Milk'),(18, 'Bread'),(19, 'Apple'),(20, 'Clock'),(21, 'Brush'),(22, 'Camera'),(23, 'Cam.
Film'),(24, '2GB SD'),(25, '4GB RAM'),(26, '16GB RAM'),(27, '32GB RAM'),(28, '32GB USB'),(29, '64GB
USB'),(30, 'Toothpaste');


CREATE TABLE `user_credentials` (
  `username` varchar(25) NOT NULL,
  `password` varchar(25) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;

ALTER TABLE `shopping_list` ADD PRIMARY KEY (`item_id`);
ALTER TABLE `user_credentials` ADD PRIMARY KEY (`username`);
```

**Code 4.1. SQL code (admin DB only)**

## 4.2 Project code (PHP)

The following are the code for the file specified in the file structure shown in Fig 4.1. Note that the code for all the files that are for UI or unrelated to the problem specified in the Midterm project are minified to consume less space within the document. Feel free to use any IDE, such as *Visual Studio Code,* or a *Beautifier* to put the code back to its original formatted form.

***Also, please note that each file maintains MySQL credentials used by PHP's mysqli function. Do change these credentials to make a proper connection with the MySQL database using XAMPP/LAMP.***

### Code for index.php

The following code belongs to *index.php*, which asks user to select their transactional database by means of sign in/sign up authorizations:

```
<?php session_start(); if(isset($_SESSION['logged']) && $_SESSION['logged']){ hea
der("Location: ./shopping.php"); } $title="CS634 - Midterm Project | Sign-
in / Sign-up page";?><!DOCTYPE html><html lang="en"><head> <!-- Metadata --
> <meta charset="UTF-8"> <meta http-equiv="X-UA-
Compatible" content="IE=edge"> <meta name="viewport" content="width=device-
width, initial-
scale=1.0"> <meta name="description" content="CS634 Midterm Project on Frequent I
tem set generation. This project highlight the use of Brute-
force algorithm for frequent itemset generation and improvizes the frequent items
et generation by the use of Apriori frequent itemset generation algorithm. This a
lso showcases the time taken by both versions, where in both algorithm generates
same set of association mining rules after the frequent-
```

```
itemlist generation."> <meta name="keywords" content="cs634, 634, data mining, ja
son wang, midterm project, apriori algorithm, frequent itemset, Brute-
force frequent itemset"> <meta name="author" content="Rahul Gautham Putcha"> <tit
le><?php echo $title;?></title> <!-- Styles: External (opensource) --
> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="an
onymous"> <link rel="preconnect" href="https://fonts.gstatic.com"> <link href="ht
tps://fonts.googleapis.com/css2?family=Montserrat:wght@300&display=swap" rel="sty
lesheet"> <!-- Styles: Rahul Gautham Putcha --
> <link href="./res/styles/style.css" rel="stylesheet" type="text/css"></head><bo
dy id="index"><!-- Sign-in / Signup page --> <header class="navbar-dark bg-
dark"> <nav class="p-2 col-6 col-lg navbar-
brand"><h1>NJIT Mart</h1></nav> </header> <main> <div class="row m-
5"> <div class="col-12"> <h2 class="p-2 text-center text-
white">Welcome to NJIT Mart</h2> </div> </div> <div class="row m-
5"> <div id="signin-form" class="p-2 p-md-5 col-12 offset-md-2 col-md-4 reg-
form"> <form method="post" action="./signin.php"> <fieldset> <legend>Sign in</leg
end> <div> <label for="si_username">Username</label> <input type="text" id="si_us
ername" name="si_username" placeholder="Username..."/> <label for="si_password">P
assword</label> <input type="password" id="si_password" name="si_password" placeh
older="Password..." /> </div> <div class="text-center w-
100"><input type="submit" name="sign_in" value="Sign in" /></div> </fieldset> </f
orm> </div> <div id="signup-form" class="p-2 p-md-3 col-12 offset-md-1 col-md-
4 reg-
form"> <form method="post" action="./signup.php"> <fieldset> <legend>Sign up</leg
end> <div> <label for="su_username">Username</label> <input type="text" id="su_us
ername" name="su_username" placeholder="Username..."/> <label for="su_password">P
assword</label> <input type="password" id="su_password" name="su_password" placeh
older="Password..." /> <label for="su_retyped_password">Retype Password</label> <
input type="password" id="su_retyped_password" name="su_retyped_password" placeho
lder="Retype Password..." /> </div> <div class="text-center w-
100"><input type="submit" id="sign_up" name="sign_up" value="Sign up" /></div> </
fieldset> </form> </div> </div> </main> <footer class='text-
center'><div><h6>&copy; Copyright 2021-
2022 | By Rahul Gautham Putcha<h6></div></footer> <!-
- Scripts: External (opensource) --> <script src="https://code.jquery.com/jquery-
3.6.0.min.js" integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script> <
script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/js/bootstrap.bundle
.min.js" integrity="sha384-
gtEjrD/SeCtmISkJkNUaaKMoLD0//ElJ19smozuHV6z3Iehds+3Ulb9Bn9Plx0x4" crossorigin="an
onymous"></script> <!-- Styles: Rahul Gautham Putcha --
> <script src="./res/script/script.js"></script></body></html>
```

**Code 4.2. Minified /index.php code**

```
<?php session_start(); $_SESSION['logged'] = FALSE; /* -- SQL Config -
- */ $servername = "localhost"; $username = "root"; $password = "root123"; $dbnam
e = "cs634dmadmin"; /* -- Validate Authorization Here -
```

```
- */ /* Create sql connection */ $conn = new mysqli($servername, $username, $pass
word, $dbname); if ($conn-
>connect_error) { echo "Signin Unsucessful!! Cannot connect to MySQL. Please do p
roper setup of XAAMP again.<br/>Redirecting to Home page in 5 seconds."; header("
refresh: 5;url=./index.php"); }else{ /* Validating user credentials in `User_DB`
(or Transaction_DB)*/ $sql = "SELECT username FROM user_credentials WHERE usernam
e='".$_POST['si_username']."' AND password='".$_POST['si_password']."'"; $result
= $conn->query($sql); if ($result->num_rows > 0) { $row=$result-
>fetch_assoc(); $conn-
>close(); /* Creating a Login session*/ $_SESSION['User_DB'] = "cs634DM_".$row['u
sername']."_DB"; $_SESSION['username'] = $row['username']; $_SESSION['logged'] =
TRUE; $_SESSION['cart'] = array(); echo "Login Sucessful!! Redirecting to Shoppin
g Page in 5 seconds."; header("refresh: 5;url=./shopping.php"); } if($_SESSION['l
ogged'] !== TRUE){ echo "Signin Unsucessful!! Invalid username or password<br/>Re
directing to Shopping Page in 5 seconds."; header("refresh: 5;url=./index.php");
} }?>
```

**Code 4.3. Minified /signin.php code**

```
<?php session_start(); /* -- SQL Config -
- */ $servername = "localhost"; $username = "root"; $password = "root123"; $dbnam
e = "cs634dmadmin"; $_SESSION['logged'] = FALSE; /* -
- Create Authorization Here -
- */ /* Create sql connection */ $conn = new mysqli($servername, $username, $pass
word, $dbname); if ($conn-
>connect_error) { echo "Signup Unsucessful!! Cannot connect to MySQL. Please do p
roper setup of XAAMP again.<br/>Redirecting to Home page in 5 seconds."; header("
refresh: 5;url=./index.php"); }else{ /* Creating `User_DB` (or Transaction_DB) wi
th a `Transactions` table within the `User_DB` */ $sql = "INSERT INTO user_creden
tials VALUES ('".$_POST['su_username']."', '".$_POST['su_password']."')"; if ($co
nn-
>query($sql) === TRUE) { echo "Success: In creating the User. <br/>"; $sql = "CRE
ATE DATABASE cs634DM_".$_POST['su_username']."_DB"; if ($conn-
>query($sql) === TRUE) { echo "Success: In creating a Database for the User. <br/
>"; $conn1 = new mysqli($servername, $username, $password, "cs634DM_".$_POST['su_
username']."_DB"); $sql1 = "CREATE TABLE Transactions ( Tid INT(6) UNSIGNED AUTO_
INCREMENT PRIMARY KEY, TransDateTime TIMESTAMP DEFAULT CURRENT_TIMESTAMP ON UPDAT
E CURRENT_TIMESTAMP )"; $sql2 = "CREATE TABLE TransactionDetails ( Tid INT(6) UNS
IGNED, label VARCHAR(25), PRIMARY KEY (Tid,label), FOREIGN KEY (Tid) REFERENCES T
ransactions(Tid) )"; if ($conn1->query($sql1) === TRUE && $conn1-
>query($sql2) === TRUE) { echo "Success: In creating a Tables in User_DB. <br/>";
 } else { echo "Error creating table: " . $conn1-
>error; } /* Creating a Login session */ $_SESSION['User_DB'] = "cs634DM_".$_POST
['su_username']."_DB"; $_SESSION['username'] = $_POST['su_username']; $_SESSION['
logged'] = TRUE; $_SESSION['cart'] = array(); /* Setting up a transaction */ echo
 "Signup Sucessful!! Redirecting to Shopping Page in 5 seconds."; header("refresh
: 5;url=./shopping.php"); $conn1-
>close(); } } if($_SESSION['logged'] !== TRUE){ echo "Error: Username already exi
sts." .$sql . "<br>" . $conn-
>error; echo "Signup Unsucessful!! Redirecting to Home page in 5 seconds."; heade
r("refresh: 5;url=./index.php"); } }?>
```

**Code 4.4. Minified /signup.php code**

## Code for shopping.php

The following code belongs to */shopping.php,* which involves the code needed for making a single transaction request:

```
<?php session_start(); if(!isset($_SESSION['logged']) || !$_SESSION['logged']){ h
eader("Location: ./index.php"); } /* -- SQL Config -
- */ $servername = "localhost"; $username = "root"; $password = "root123"; $dbnam
e = "cs634dmadmin"; $item_array = array(); $conn = new mysqli($servername, $usern
ame, $password, $dbname); if ($conn-
>connect_error) { echo "<hr/><br/>500 - Internal Server Error. <br/> Cannot conne
ct to MySQL. Please do proper setup of XAAMP again.<br/><hr/>"; die("Connection f
ailed: " . $conn-
>connect_error); }else{ $user= $_SESSION["username"]; $title="CS634 - Midterm Pro
ject | Shopping page"; $sql = "SELECT label FROM shopping_list"; $result = $conn-
>query($sql); if ($result->num_rows > 0) { while($row=$result-
>fetch_array(MYSQLI_ASSOC)){ array_push($item_array, $row['label']); } } else { e
cho "<hr/><br/>500 - Internal Server Error. <br/> Cannot connect to MySQL. Please
 do proper setup of XAAMP again.<br/><hr/>"; }?><!DOCTYPE html><html lang="en"><h
ead> <!-- Metadata --> <meta charset="UTF-8"> <meta http-equiv="X-UA-
Compatible" content="IE=edge"> <meta name="viewport" content="width=device-
width, initial-
scale=1.0"> <meta name="description" content="CS634 Midterm Project on Frequent I
tem set generation. This project highlight the use of Brute-
force algorithm for frequent itemset generation and improvizes the frequent items
et generation by the use of Apriori frequent itemset generation algorithm. This a
lso showcases the time taken by both versions, where in both algorithm generates
same set of association mining rules after the frequent-
itemlist generation."> <meta name="keywords" content="cs634, 634, data mining, ja
son wang, midterm project, apriori algorithm, frequent itemset, Brute-
force frequent itemset"> <meta name="author" content="Rahul Gautham Putcha"> <tit
le><?php echo $title;?></title> <!-- Styles: External (opensource) --
> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="an
onymous"> <link rel="preconnect" href="https://fonts.gstatic.com"> <link href="ht
tps://fonts.googleapis.com/css2?family=Montserrat:wght@300&display=swap" rel="sty
lesheet"> <!-- Styles: Rahul Gautham Putcha --
> <link href="./res/styles/style.css" rel="stylesheet" type="text/css"></head><bo
dy><!-- Sign-in / Signup page --> <header class="fixed-top"> <div class="d-
flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 navbar-dark bg-
dark border-bottom box-shadow"> <h1 class="my-0 mr-md-auto font-weight-
normal">NJIT MART</h1> <nav class="my-2 my-md-0 mr-md-3" > <a class="active p-
1 text-light" href="./shopping.php">Shop Item</a> <a class="p-1 text-
light" href="./itemset.php?itemset_method=bruteforce">Bruteforce</a> <a class="p-
```

```
1 text-
light" href="./itemset.php?itemset_method=apriori">Apriori</a> </nav> <a class="b
tn btn-outline-
primary" href="./logout.php">Logout</a> </div> </header> <main class="p-
3 container" style="margin-top: 125px;"> <div class="text-
center h3"> Hello <?php echo $user; ?>, </div> <div class="header-
text"> <h2>Shopping Items</h2> </div><br/><br/><br/> <div class="m-2 m-md-
0"> <ul class="item-list row text-
center"> <?php $item_count = count($item_array); for($i=0;$i<$item_count;$i++){ e
cho "<li class='item col-12 col-md-5 col-lg-3 m-1'><div></div><div class='p-
3'><span class='label'>".$item_array[$i]."</span><button class='add'>Add</button>
</div></li>"; } ?> </ul> </div> <br/><br/><br/><br/><br/><br/><br/><br/><br/> </m
ain> <footer class='text-center fixed-bottom bg-dark pb-2'> <div id="transaction-
details" class="container-fluid px-md-5 py-md-
2"> <div class="row"> <div class="col-12 col-md-4"> <strong style="font-
size: 20px;">Total Items: <span id="total-
items"></span></strong> </div> <div class="col-12 col-md-4"><button class="clear-
cart"><u>Clear</u></button></div> <div class="col-12 col-md-4"> <button id="make-
transaction">Make a Transaction</button> <form method="post" action="./add_transa
ction.php" style="position:absolute; z-index:-
9999;visibility: hidden;"> <div id="cart-
details"></div> <div><input type="submit" id="add_transaction" value="Send"></div
> </form> </div> </div> </div> <div id="copyright" class="pt-
2"><h6>&copy; Copyright 2021-
2022 | By Rahul Gautham Putcha<h6></div> </footer> <!-
- Scripts: External (opensource) --> <script src="https://code.jquery.com/jquery-
3.6.0.min.js" integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script> <
script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min
.js" integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="an
onymous"></script> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist
/js/bootstrap.min.js" integrity="sha384-
Atwg2Pkwv9vp0ygtn1JAojH0nYbwNJLPhwyoVbhoPwBhjQPR5VtM2+xf0Uwh9KtT" crossorigin="an
onymous"></script> <!-- Styles: Rahul Gautham Putcha --
> <script src="./res/script/script1.js"></script></body></html><?php }?>
```

**Code 4.5. Minified /shopping.php code**

```
<?php session_start(); if(!isset($_SESSION['logged']) || !$_SESSION['logged']){ h
eader("Location: ./index.php"); }else{ /* -- SQL Config -
- */ $servername = "localhost"; $username = "root"; $password = "root123"; $dbnam
e = "cs634dmadmin"; $USER_DB = $_SESSION['User_DB']; $conn = new mysqli($serverna
me, $username, $password, $USER_DB); if ($conn-
>connect_error) { echo "<hr/><br/>500 - Internal Server Error. <br/> Cannot conne
ct to MySQL. Please do proper setup of XAAMP again.<br/><hr/>"; die("Connection f
```

```
ailed: " . $conn->connect_error); }else{ if($conn-
>query("INSERT INTO Transactions VALUES()") && $result = $conn-
>query("SELECT LAST_INSERT_ID() AS last_val")){ echo "Success: in creating a Tran
saction request. (updating Transactions)<br/>"; $items = ""; $Tid = $result-
>fetch_array()['last_val']; for($i=0;$i<(count($_POST['cart'])-
1);$i++){ $items = $items."(".$Tid.",'".$_POST['cart'][$i]."') ,"; }$items = $ite
ms."(".$Tid.",'".$_POST['cart'][$i]."')"; if($conn-
>query("INSERT INTO TransactionDetails VALUES ".$items)){ echo "Success: in creat
ing a Adding items with the Transaction request. (updating TransactionDetails)<br
/>"; echo "Redirecting to shopping page in 5 seconds."; header("refresh: 5;url=./
shopping.php"); }else{ echo "Error: in creating a Adding items with the Transacti
on request. (updating TransactionDetails)<br/>"; echo "Redirecting to shopping pa
ge in 5 seconds."; header("refresh: 5;url=./shopping.php"); } }else{ echo "Error:
 in creating a Transaction request. (updating Transactions)<br/>"; echo "Redirect
ing to shopping page in 5 seconds."; header("refresh: 5;url=./shopping.php"); } }
 }?>
```

**Code 4.6. Minified /add_transaction.php code**

## Code for itemset.php

The following is the code required for *itemset.php*, containing the code for calling the frequent itemset generation process and presenting the output after generation of the association rule:

```
<?php session_start(); if(!isset($_SESSION['logged']) || !$_SESSION['logged']){ h
eader("Location: ./index.php"); } $method = $_GET["itemset_method"]; $title="CS63
4 - Midterm Project | ".(($method=="apriori")?"Apriori":"BruteForce")." Itemset g
eneration";?><!DOCTYPE html><html lang="en"><head> <!-- Metadata --
> <meta charset="UTF-8"> <meta http-equiv="X-UA-
Compatible" content="IE=edge"> <meta name="viewport" content="width=device-
width, initial-
scale=1.0"> <meta name="description" content="CS634 Midterm Project on Frequent I
tem set generation. This project highlight the use of Brute-
force algorithm for frequent itemset generation and improvizes the frequent items
et generation by the use of Apriori frequent itemset generation algorithm. This a
lso showcases the time taken by both versions, where in both algorithm generates
same set of association mining rules after the frequent-
itemlist generation."> <meta name="keywords" content="cs634, 634, data mining, ja
son wang, midterm project, apriori algorithm, frequent itemset, Brute-
force frequent itemset"> <meta name="author" content="Rahul Gautham Putcha"> <tit
le><?php echo $title;?></title> <!-- Styles: External (opensource) --
> <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist/css/bootstrap.min
.css" rel="stylesheet" integrity="sha384-
+0n0xVW2eSR5OomGNYDnhzAbDsOXxcvSN1TPprVMTNDbiYZCxYbOOl7+AMvyTG2x" crossorigin="an
onymous"> <link rel="preconnect" href="https://fonts.gstatic.com"> <link href="ht
tps://fonts.googleapis.com/css2?family=Montserrat:wght@300&display=swap" rel="sty
lesheet"> <!-- Styles: Rahul Gautham Putcha --
> <link href="./res/styles/style.css" rel="stylesheet" type="text/css"></head><bo
dy><!-- Sign-in / Signup page --> <header class="fixed-top"> <div class="d-
flex flex-column flex-md-row align-items-center p-3 px-md-4 mb-3 navbar-dark bg-
dark border-bottom box-shadow"> <h1 class="my-0 mr-md-auto font-weight-
normal">NJIT MART</h1> <nav class="my-2 my-md-0 mr-md-3" > <a class="p-1 text-
```

```
light" href="./shopping.php">Shop Item</a> <a class="<?php echo ($method==="brute
force")?"active":"";?> p-1 text-
light" href="./itemset.php?itemset_method=bruteforce">Bruteforce</a> <a class="<?
php echo ($method==="apriori")?"active":"";?> p-1 text-
light" href="./itemset.php?itemset_method=apriori">Apriori</a> </nav> <a class="b
tn btn-outline-
primary" href="./logout.php">Logout</a> </div> </header> <main class="p-
3 container" style="margin-top: 125px;"> <div class="row"> <form id="itemset-
generation" class="col-
12"> <fieldset> <legend>Generate Itemset:</legend> <label for="min_support">Minim
um Support      </label> <input type="number" id="m
in_support" name="min_support" min="0" max="100" value="50" required/>% <br/><br/
> <label for="min_confidence">Minimum Confidence</label> <input type="number" id=
"min_confidence" name="min_confidence" min="0" max="100" value="50" required />%
<input style="display:none" type="text" id="hidden-
method" name="itemset_method" value="<?php echo $method; ?>" required /> <br/><br
/> <input type="submit" value="Generate" class="px-2 py-
1"/> </fieldset> </form> </div> <br/><br/> <div class="header-
text"> <h2>Input Transactions</h2> </div><br/><br/><br/> <div class="table-
responsive m-2 m-md-0"> <?php /* -- SQL Config -
- */ $servername = "localhost"; $username = "root"; $password = "root123"; $conn
= new mysqli($servername, $username, $password, $_SESSION["User_DB"]); if ($conn-
>connect_error) { echo "<div style='background-
color: white; width:100%;'><hr/><br/>500 - Internal Server Error. <br/> Cannot co
nnect to MySQL. Please do proper setup of XAAMP again.<br/><hr/></div>"; die("Con
nection failed: " . $conn-
>connect_error); }else{ $sql = "SELECT T.Tid, TransDateTime, Detail FROM Transact
ions AS T , (SELECT Tid, GROUP_CONCAT(DISTINCT label) 'Detail' FROM transactionde
tails GROUP BY Tid) AS TD WHERE T.Tid = TD.Tid"; $result = $conn-
>query($sql); if ($result->num_rows > 0) { echo "<table class='table text-
white'>"; echo "<thead><tr><th>Tid</th><th>TransDateTime</th><th>Transaction Deta
ils</th></tr></thead><tbody>"; while($row=$result-
>fetch_array(MYSQLI_ASSOC)){ echo "<tr><td>".$row['Tid']."</td><td>".$row['TransD
ateTime']."</td><td>".$row['Detail']."</td></tr>"; }echo "</tbody></table><br/><b
r/><br/>"; } else { echo "<hr/><br/>No Records to display <br/><hr/>"; } } ?> </d
iv> <div class="header-
text"> <h2>Association Rules</h2> </div><br/><br/><br/> <div id="output" class="m
-2 m-md-0"> <div class="text-center"><p class="h5 text-
white">* Click on <u><strong>Generate</strong></u> to view Association Rules.</p>
</div> </div> <br/><br/><br/><br/> </main> <footer class='text-center fixed-
bottom bg-dark pb-2'> <div id="copyright" class="pt-2"><h6>&copy; Copyright 2021-
2022 | By Rahul Gautham Putcha<h6></div> </footer> <!-
- Scripts: External (opensource) --> <script src="https://code.jquery.com/jquery-
3.6.0.min.js" integrity="sha256-
/xUj+3OJU5yExlq6GSYGSHk7tPXikynS7ogEvDej/m4=" crossorigin="anonymous"></script> <
script src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.2/dist/umd/popper.min
.js" integrity="sha384-
IQsoLXl5PILFhosVNubq5LC7Qb9DXgDA9i+tQ8Zj3iwWAwPtgFTxbJ8NT4GN1R8p" crossorigin="an
onymous"></script> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.1/dist
/js/bootstrap.min.js" integrity="sha384-
Atwg2Pkwv9vp0ygtn1JAojH0nYbwNJLPhwyoVbhoPwBhjQPR5VtM2+xf0Uwh9KtT" crossorigin="an
```

```
onymous"></script> <!-- Styles: Rahul Gautham Putcha --
> <script src="./res/script/script2.js"></script></body></html>
```

**Code 4.7. Minified /itemset.php code**

## Code for Frequent-itemset generation

The following snippet holds the code for the Apriori algorithm and Brute-force method:

```
<?php
    session_start();
    $User_DB = $_SESSION['User_DB'];
    $method  = $_POST['itemset_method'];
    $support = $_POST['min_support']/100;
    $confidence = $_POST['min_confidence']/100;

    /* -- SQL Config -- */
    $servername = "localhost";
    $username = "root";
    $password = "root123";
    $dbname = "cs634dmadmin";

    echo "Min-Support: $support<br/>Min-Confidence: $confidence<br/><br/>";
    set_time_limit(900); /* Max PHP Execution time: 15-minutes */
    function getItemset(){
        global $servername, $username, $password, $dbname;
        $conn = new mysqli($servername, $username, $password, $dbname);
        $item_array = array();
        if ($conn->connect_error) {
            return NULL;
        }else{
            $sql = "SELECT label FROM shopping_list";
            $result = $conn->query($sql);
            if ($result->num_rows > 0) {
                while($row=$result->fetch_array(MYSQLI_ASSOC)){
                    array_push($item_array, $row['label']);
                }
            } else {
                return NULL;
            }
        }
        return $item_array;
    }

    function support($X){
        global $servername, $username, $password, $User_DB;
        $conn = new mysqli($servername, $username, $password, $User_DB);

        $X_array = explode(',', $X);
        $sql = "SELECT (count(A.Tid) / (SELECT count(*) FROM Transactions)) AS support FROM Tr
ansactionDetails A WHERE A.label = '".$X_array[0]."'";
        $x=0;
        for($x=1;$x<count($X_array);$x++){
            $sql .= " and A.Tid IN (SELECT B.Tid FROM TransactionDetails B WHERE B.label = '".
$X_array[$x]."')";
        }
        $result = $conn->query($sql);
        if ($result->num_rows > 0) {
            $support_count = $result->fetch_array(MYSQLI_ASSOC)['support'];
            return $support_count;
        }return 0;
    }
```

```
    function confidence($X, $Y){ return support(implode(",", array_merge($X, $Y)))/support(imp
lode(",", $X)); }

    function pruningCandidateSet($c){
        global $support;
        return support($c) >= $support;
    }

    function GenerateList($L){
        $result = array();
        $item_count = count($L);
        for($a=0 ; $a < $item_count ; $a++){
            for($b=$a+1 ; $b < $item_count ; $b++){
                list($p, $q) = [$L[$a], $L[$b]];
                $p_array = explode(',', $p);
                $q_array = explode(',', $q);
                $i = 0;

                for($i=0;$i < count($p_array) - 1;$i++){
                    if($p_array[$i] !== $q_array[$i]) break;
                }if( ($i === count($p_array)-1) && ($p_array[$i] !== $q_array[$i]) ){
                    $p_array_copy = array_merge(array(), $p_array);
                    array_push($p_array_copy, $q_array[$i]);
                    $c = implode(',', $p_array_copy);
                    array_push($result, $c);
                }
            }
        }
        return $result;
    }

    function powerSet($set){
        $powerset = array(array());
        foreach($set as $element){
            foreach($powerset as $combination){
                array_push($powerset, array_merge(array($element), $combination));
            }
        }return $powerset;
    }
    function printAssociationRules($frequent_itemset){
        global $confidence;
        for($i=0;$i<count($frequent_itemset);$i++){
            echo "FREQ-Itemset ".($i+1).": {".$frequent_itemset[$i]."} <br/>";
            $itemset = explode(",", $frequent_itemset[$i]);
            $subsets = powerSet($itemset);

            array_shift($subsets);array_pop($subsets);

            echo "<ul>";
            foreach($subsets as $subset){
                $subset_confidence = confidence($subset,array_diff($itemset, $subset));
                if($subset_confidence >= $confidence){
                    $subset_support = support(implode(",", array_diff($itemset, $subset)));
                    echo "<li>".implode(",", $subset)."&rarr; ".implode(",", array_diff($items
et, $subset))." (Support: $subset_support, Confidence:$subset_confidence)</li>";
                }
            }echo "</ul>";

        }
    }
```

```
    if($method=='apriori'){
        function Apriori(){
            list($L,$C) = [array(), array()];
            $C[0] = getItemset();
            $L[0] = array_values(array_filter($C[0], 'pruningCandidateSet')); /* Frequent Item
set-1 */
            if(!$L[0]) return $L[0];
            $k=0;
            do{ /* Apriori: Choose (k+1)th Candidates from subset of kth Frequent Itemset */
                $C[$k+1] = GenerateList($L[$k]);
                $L[$k+1] = array_values(array_filter($C[$k+1], 'pruningCandidateSet'));
                $k++;
            }while($L[$k]);
            return $L[$k-1];
        }

        /* Generating Association Rules */
        echo "Type: Apriori (Frequent-itemset generation)<br/>";
        echo "Output: <br/>";
        $startTime=microtime(TRUE);
        $frequent_itemset = Apriori();
        printAssociationRules($frequent_itemset);

        echo "Running Time:<br/>";
        $endTime=microtime(TRUE);
        $timeDiff=$endTime-$startTime;
        echo "Start time: ".number_format($startTime, 5, '.', '')." seconds<br/>End time: ".nu
mber_format($endTime, 5, '.', '')." seconds<br/>Time Difference: ".number_format($timeDiff, 5,
 '.', '')." seconds elapsed.<br/>";

    }else if($method=='bruteforce'){
        function BruteForce(){
            list($L,$C) = [array(), array()];
            $C[0] = getItemset();
            $L[0] = array_values(array_filter($C[0], 'pruningCandidateSet')); /* Frequent Item
set-1 */
            if(!$L[0]) return $L[0];
            $k=0;
            do{ /* BruteForce: Choose (k+1)th Candidates from entire Shopping list space */
                $C[$k+1] = GenerateList($C[$k]); /* Choose from Entire Shopping list */
                $L[$k+1] = array_values(array_filter($C[$k+1], 'pruningCandidateSet'));
                $k++;
            }while($L[$k]);
            return $L[$k-1];
        }

        /* Generating Association Rules */
        echo "Type: BruteForce (Frequent-itemset generation)<br/>";
        echo "Output: <br/>";
        $startTime=microtime(TRUE);
        $frequent_itemset = BruteForce();
        printAssociationRules($frequent_itemset);

        echo "Running Time:<br/>";
        $endTime=microtime(TRUE);
        $timeDiff=$endTime-$startTime;
        echo "Start time: ".number_format($startTime, 5, '.', '')." seconds<br/>End time: ".nu
mber_format($endTime, 5, '.', '')." seconds<br/>Time Difference: ".number_format($timeDiff, 5,
 '.', '')." seconds elapsed.<br/>";
    }
```

```
?>
```

**Code 4.8. Formatted code for /generate_itemset.php**

## Miscellaneous Code

```
*{box-sizing:border-box;-moz-box-sizing:border-box;-webkit-box-sizing:border-box;font-family:Montserrat,sans-serif;margin:0;padding:0}body{width:100%;overflow-x:hidden;background-color:#40739e;color:#fff}header{width:100%}h1{padding-left:20px;font-size:20px}#index input,#index label{display:block}#index input{margin:0 0 10px 10px;padding:5px}#index input[type=text],input[type=password]{padding:10px 5%;width:90%}#index input[type=submit]{margin:2% 35%;width:30%}#index legend{text-decoration:underline}nav{margin-left:auto}a{margin:2px 10px;text-decoration:none}nav>a.active{text-decoration:underline}.header-text::before{content:" ";display:block;float:left;width:2vw;height:5.5vh;margin-right:15px;background-color:orange}.header-text{display:inline-block;width:100%;color:orange}.header-text::after{content:" ";display:block;width:90%;background-color:orange;height:4px}.item-list{text-decoration:none;list-style:none}.item{background-color:#fff;color:#000;padding:0}.item>div:first-child{float:left;background-color:#000;height:100%;width:40px}.item .label{margin-left:20px}.add,.delete{padding:8px 30px;margin-left:30px;background:0 0;font-weight:bolder}.add{color:#9acd32;border:2px solid #9acd32}.add:hover{color:#fff;background-color:#9acd32}.delete{color:red;border:2px solid red}.delete:hover{color:#fff;background-color:red}#transaction-details{height:20%;width:100%;background-color:purple}#make-transaction,.clear-cart{padding:8px 15px;background-color:#fff;color:purple;border:none;outline:0;font-weight:700;text-decoration:none;margin:5px}#make-transaction:hover,.clear-cart:hover{background-color:#000;color:#fff}@media (max-width:768px){nav{margin-left:unset}}
```

**Code 4.9. Minified code for /res/styles/style.css**

```
$(document).ready(function(){let t=[];function e(){$("#total-items").text(t.length)}e(),$(".add").on("click",function(l){if($(this).hasClass("delete")){$(this).removeClass("delete").text("Add");var a=t.indexOf($(this).prev().text());a>-1&&t.splice(a,1)}else $(this).addClass("delete").text("Delete"),t.push($(this).prev().text());e(),console.log(t)}),$(".clear-cart").on("click",function(l){t=[],$(".delete").removeClass("delete").text("Add"),e()}),$("#make-transaction").on("click",function(e){for(let e=0;e<t.length;e++)$("#cart-details").html($("#cart-details").html()+"<input type='text' name='cart["+e+"]' value='"+t[e]+"'>"),$("#add_transaction").click()})});
```

**Code 4.10. Minified code for /res/script/script1.js**

```
$(document).ready(function(){$("#itemset-
generation").on("submit",function(t){t.preventDefault();var e=$(this).serialize()
;$.ajax({type:"POST",url:"./generate_itemset.php",data:e,success:function(t){$("#
output").html(t)}})})});
```

**Code 4.11. Minified code for /res/script/script2.js**

# Chapter 5. Designing of the Shopping Database

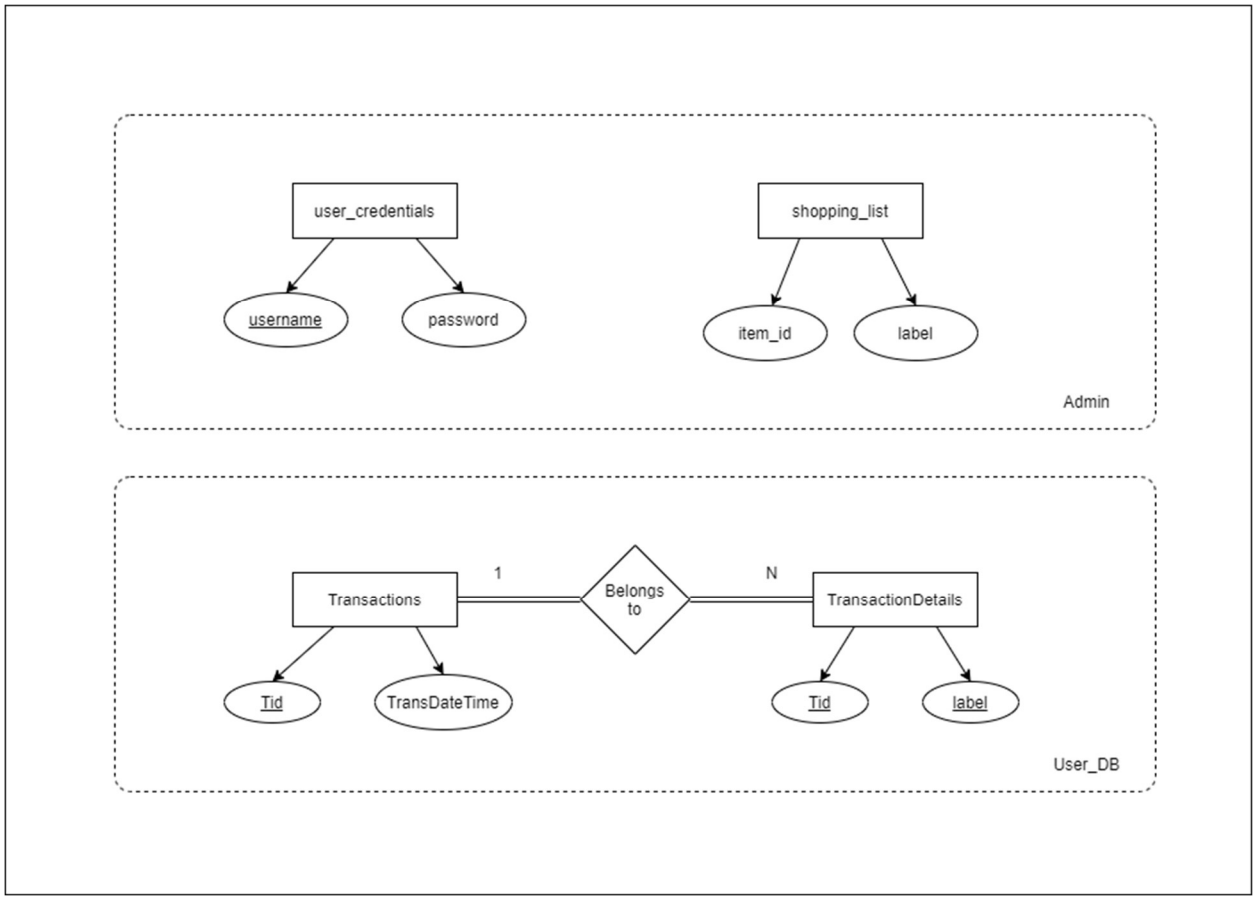



**Fig 5.1. ER-diagram for the admin database and for a single user database**

A database structure is a crucial part of every market basket/shopping application. Every transaction made by a user must be stored and retrieved in a lossless manner. Hence, a proper database design is required. Above ER-Diagram provides the design for an admin and user database. An admin DB stores user's credentials and maintains shopping items, whereas User DB stores transaction details of a particular user. As every user receives their database holding transaction, any computations performed on the transaction are at the user/User DB level.

We have provided a sample list of shopping items residing in the shopping table of the admin DB in the next section.

## 5.1 Take a Peek at Shopping List

| | item_id | label |
|---|---|---|
| Edit Copy Delete | 1 | Jewelry |
| Edit Copy Delete | 2 | Beads |
| Edit Copy Delete | 3 | God Statue |
| Edit Copy Delete | 4 | Skin cream |
| Edit Copy Delete | 5 | Pendant |
| Edit Copy Delete | 6 | Eng. Book |
| Edit Copy Delete | 7 | Chalk |
| Edit Copy Delete | 8 | Ball |
| Edit Copy Delete | 9 | Staplers |
| Edit Copy Delete | 10 | Toy |
| Edit Copy Delete | 11 | Computer |
| Edit Copy Delete | 12 | Tea |
| Edit Copy Delete | 13 | Scissors |
| Edit Copy Delete | 14 | Headsets |
| Edit Copy Delete | 15 | Wine |
| Edit Copy Delete | 16 | Diaper |
| Edit Copy Delete | 17 | Milk |
| Edit Copy Delete | 18 | Bread |
| Edit Copy Delete | 19 | Apple |
| Edit Copy Delete | 20 | Clock |
| Edit Copy Delete | 21 | Brush |
| Edit Copy Delete | 22 | Camera |
| Edit Copy Delete | 23 | Cam. Film |
| Edit Copy Delete | 24 | 2GB SD |
| Edit Copy Delete | 25 | 4GB RAM |
| Edit Copy Delete | 26 | 16GB RAM |
| Edit Copy Delete | 27 | 32GB RAM |
| Edit Copy Delete | 28 | 32GB USB |
| Edit Copy Delete | 29 | 64GB USB |
| Edit Copy Delete | 30 | Toothpaste |

**Fig 5.2. List of available Shopping items**

The above figure shows the list of Shopping items, we have included in our database. These items are dynamically fetched from the database and showcased in the UI when the user visits the */shopping.php* page.

In the next section, we will take the first look at the User Interface. Here, we will give you a guide through the process of using the application for conducting the transaction and determining the association mining rules.

# Chapter 6. The Application User Interface

This chapter gives a walkthrough of the User Interface. If you haven't set up the project files or have a doubt regarding setting up the server, please review the **Setting the Server and Project Files** section.

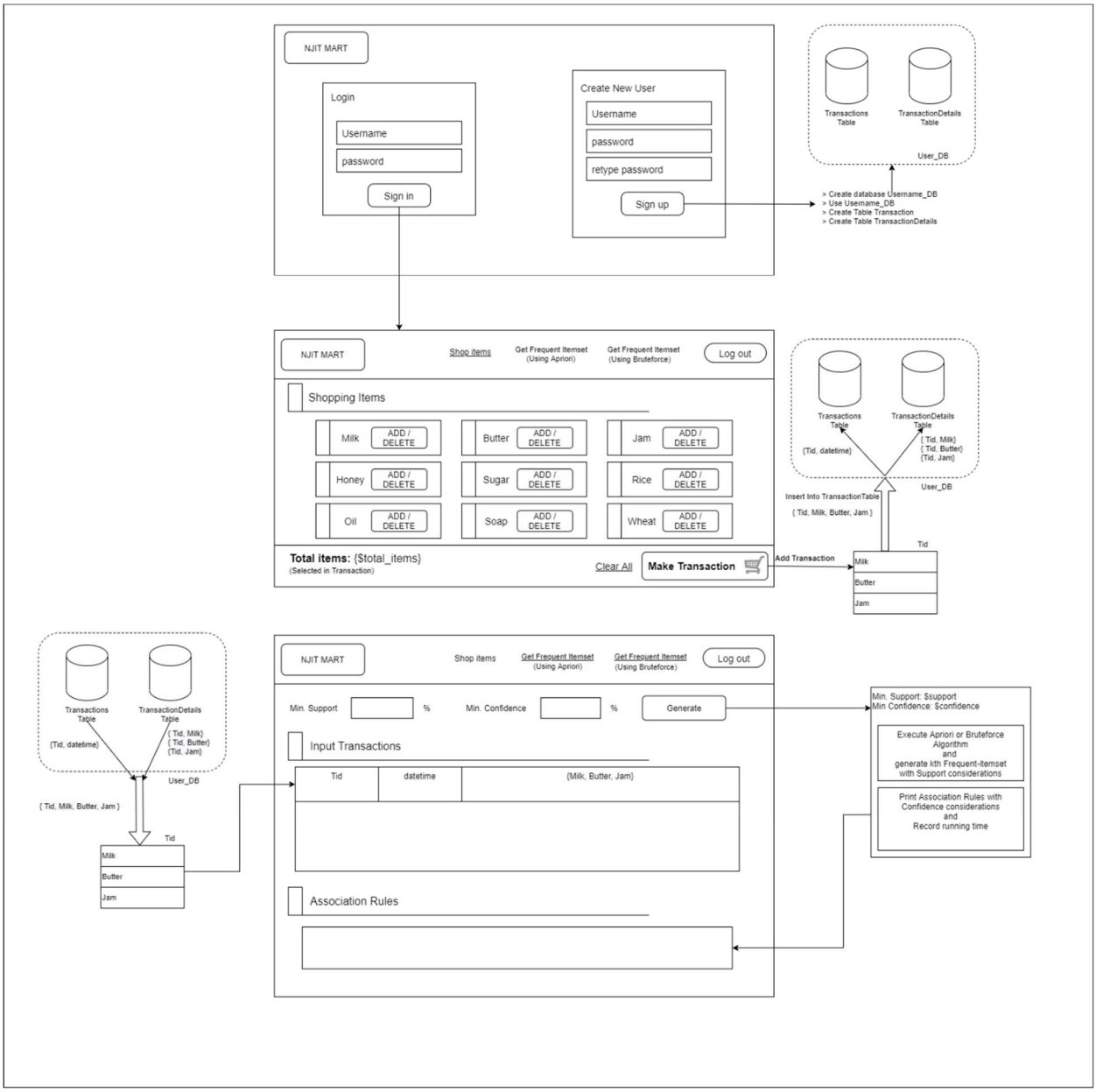



**Fig 6.1. Application Workflow**

The above figure showcases the application workflow in combination with the database. It also describes various activities that are happening within our application.

## 6.1 Setting up the User Database 1

After a proper setup of the project files, every single user of the application will be visiting the Landing/Home page. To make the transaction, A new user must register by completing the *Sign-up form*. After the registration, every new user will be receiving their database, set up in *SQL*, with each database consisting of tables storing transaction details. On successful Sign-up, the user will be redirected to the */shopping.php* page to make further transactions.

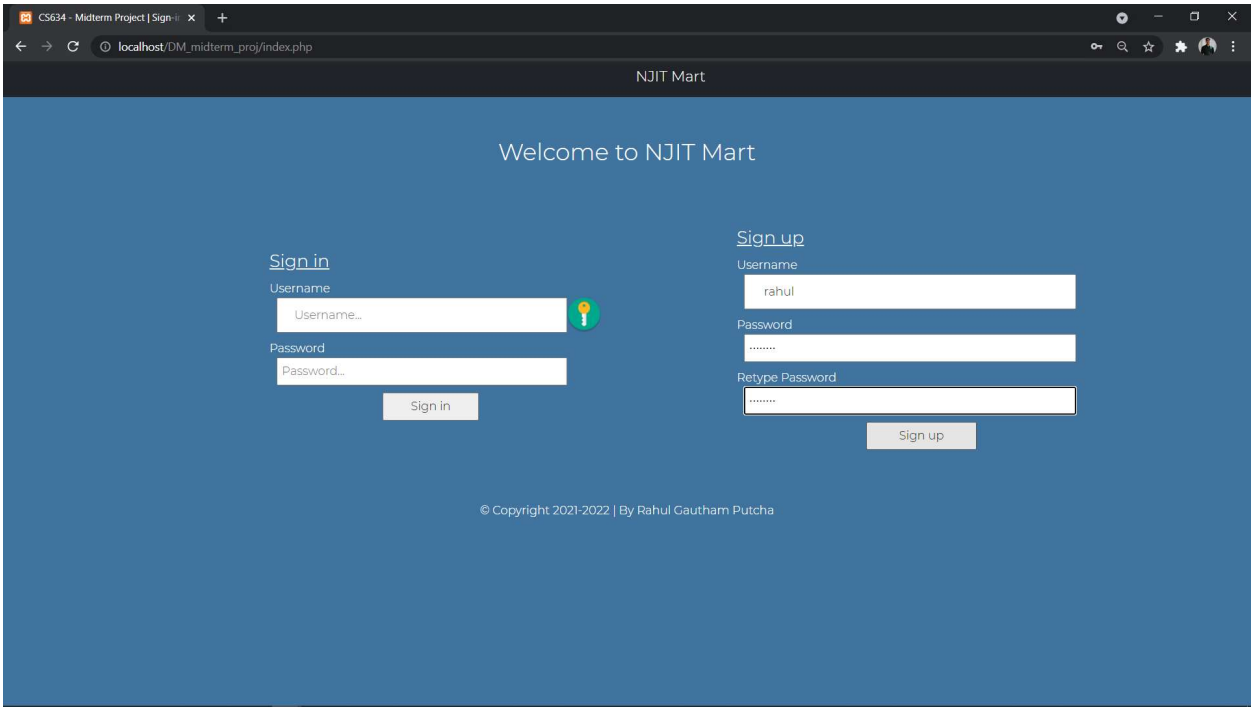



**Fig 6.2. The Home Page**

Likewise, all past users of the application can get to *the /shopping.php* page via completing the *Sign-in form*, by filling up their credential details. By entering the proper credentials, the user will be authorized to enter the */shopping.php* and perform transactions that will only make an effect within the database belonging to the respective user.


**Fig 6.3. Signup (Successful)**

Upon landing into the */shopping.php* page, the user gets options to choose; whether to make transaction or to generate frequent-itemset and association rules, via Brute-force or Apriori algorithm. Below figure show the */shopping.php* page.

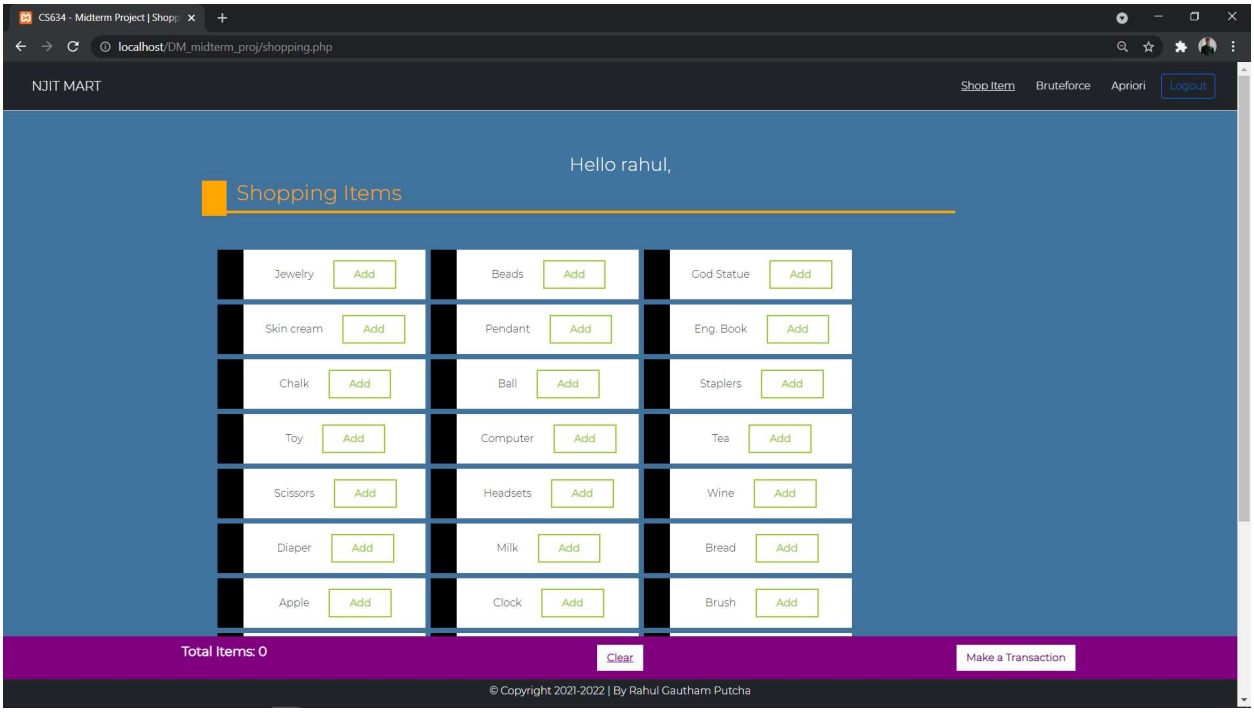



**Fig 6.4. /shopping.php page**

Here, the major portion of the area is occupied by the shopping items for which a user can choose to place their requested item inside a virtual cart by clicking on the *Add* button, located beside every item of their choice. Upon each addition will update the cart and the *Total item* indicator on the bottom left.

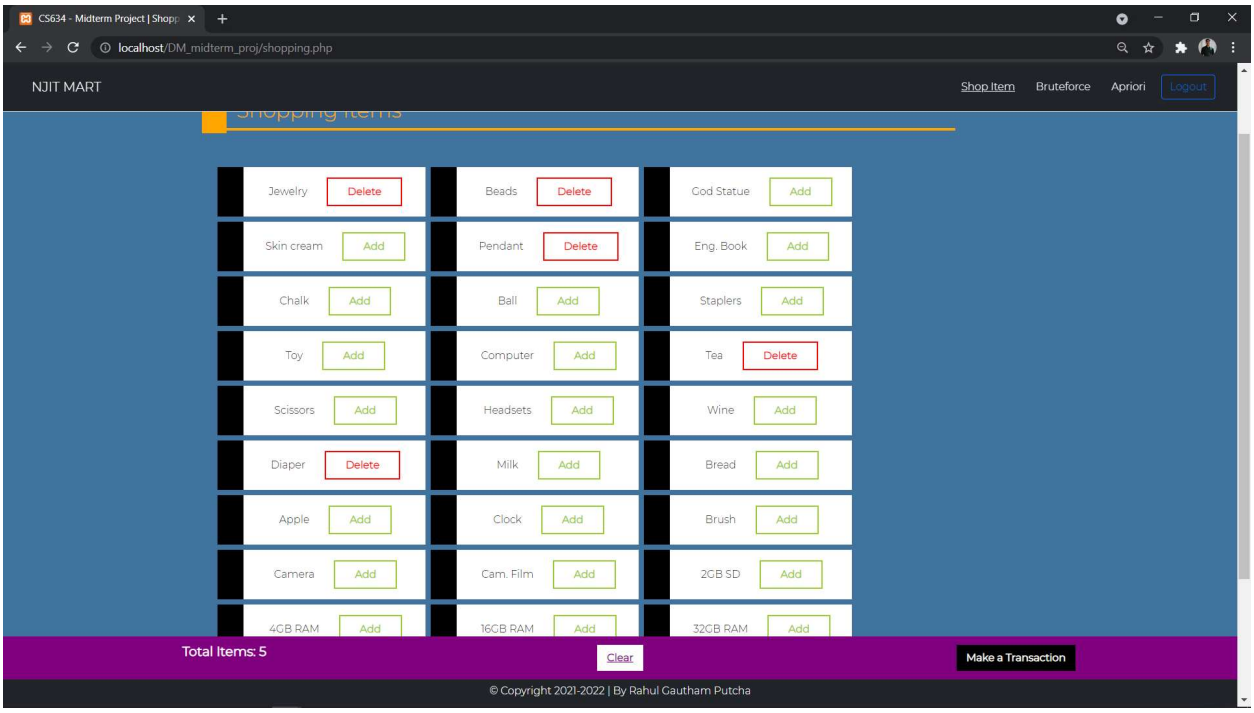



**Fig 6.5. Updating cart-items before making transaction**

For a user to remove items from the cart, the user needs to click the *Delete* button, located beside that cart item. If the cart is loaded will too many items, a user can clear/reset the entire cart by clicking the *Clear* button. After filling the cart with all necessary items, A user can finally make a transaction by clicking the *Make Transaction* button, located in the bottom right corner of the */shopping.php* page.

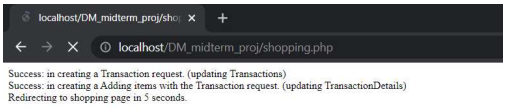



**Fig 6.6. Transaction (Successful)**

By making a transaction, a user will be shown the success message and is automatically redirected back to the */shopping.php* page. After similarly conducting several transactions, say 20 transactions, a user can generate a valid association rule by visiting the page belonging to one of the two methods; namely the Brute-force and the Apriori method.

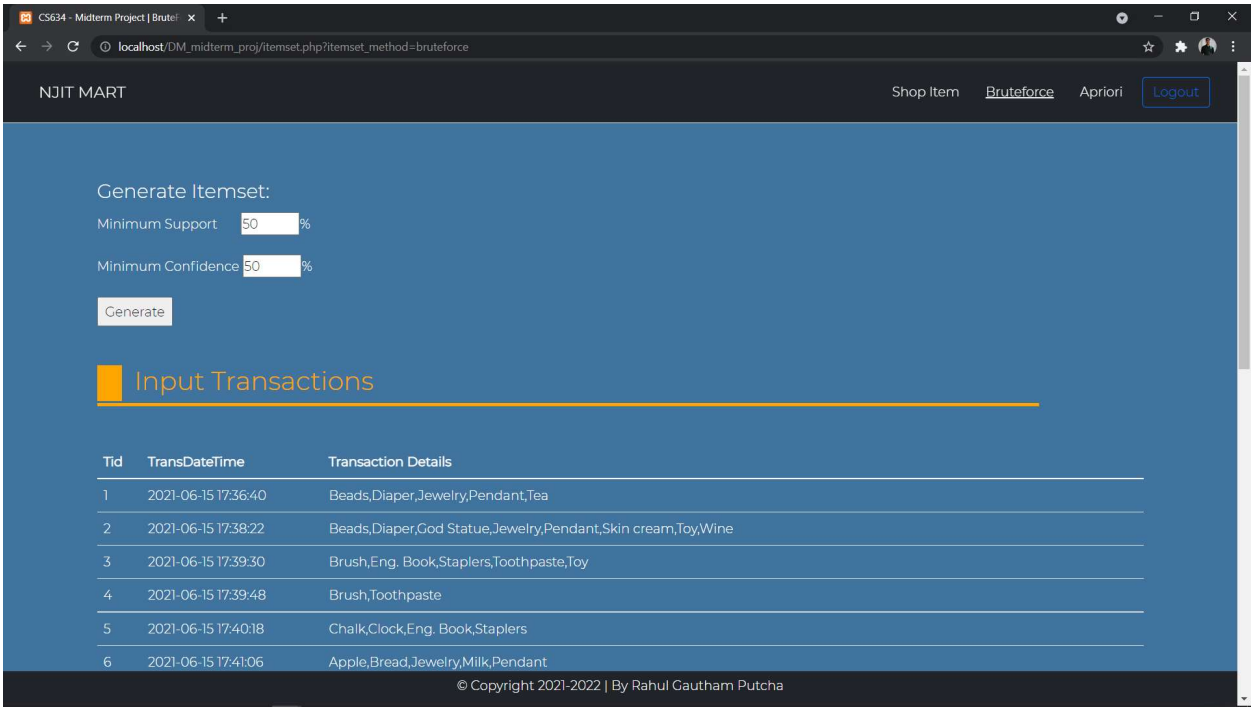



**Fig 6.7. Frequent-Itemset pages (belonging to Brute-force method)**

Each of the methods supports the user with an interface containing, a form taking the frequent-itemset indicators, the minimum support, and minimum confidence, which must be filled by the user and is used for generating the association rules by either method. Along with the frequent-itemset indicator, the interface presents the *Input Transactions*, that showcases all past transaction the user has performed. Fig 6.7. located below shows a sample *Input Transaction* presenting all sample transactions made by an application user.

Every frequent-itemset algorithms supported by our application take in the above details with the shopping item list to determine the frequent-itemset and to generates the respective association rules.

In the next section, we will execute our frequent-itemset algorithm and take a side-by-side comparison of both methods in terms of output presented and time taking to achieve the output for the same frequent-itemset indicators.

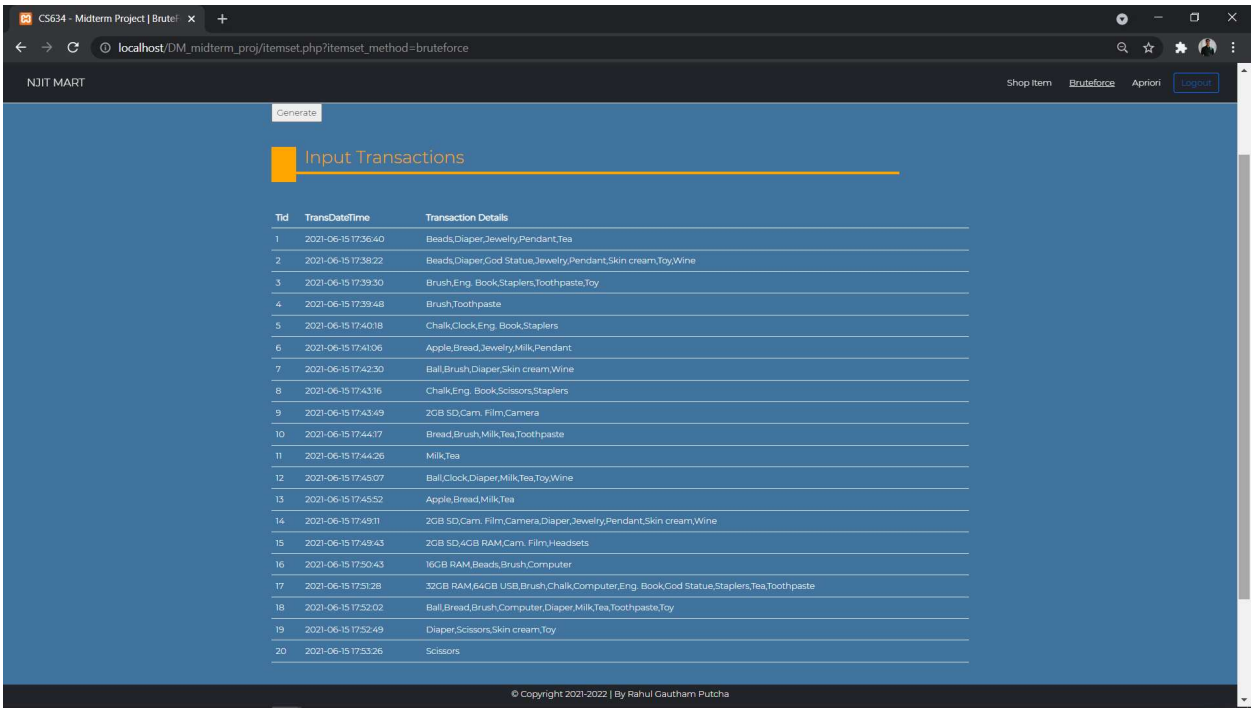



**Fig 6.8. Input Transaction section (in Frequent-Itemset pages)**

## 6.2 Executing the Frequent-Itemset algorithm

Being a user, we are now familiar with the application user interface and the process for conducting a transaction within our database. We will now take a look at the process of generating the frequent-itemset(s) and thereby, the association rules.

In the previous section, we have seen the form containing the frequent-itemset indicators. We also got to know that this form takes in two values, namely the minimum support and the minimum confidence, both of which are required in the generation of the association rules. Generally speaking, the minimum support is used as a threshold for determining an item (or an itemset) that is frequent. And, the minimum confidence is used as a threshold for determining whether two items (or item sets) are having an association or relation, where the relation indicates confidence for the presence of one item (or item set) to be considered with the other one.

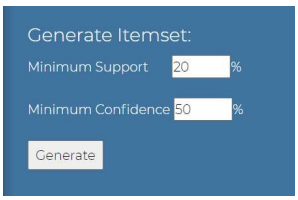



**Fig 6.9. Sample values for frequent-itemset indicators**

After filling the indicators, we generate the rules by submitting the form. Similarly, we can fill up the frequent-itemset indicators for the Apriori algorithm.

## Side by side comparison

We now take a side-by-side comparison of the Apriori-Algorithm and Brute-force method for the same *Input Transaction*. For the example below, we see the percentage time difference as,

- ⇨ $| Time_{Apriori} - Time_{BruteForce} | / [(Time_{Apriori} + Time_{BruteForce}) / 2] \times 100$
- ⇨ $(|2.6-70.8| / [(2.6+70.8) / 2]) \times 100$
- ⇨ $(|-68.2| / [73.4 / 2]) \times 100$
- ⇨ $(68.23/6.7) \times 100$
- ⇨ $1.85831 \times 100$
- ⇨ 185.831%, difference

This tells that Apriori gives a boost of performance by 186% compared to the Brute-force algorithm. This difference is seen even larger for the larger transaction dataset.

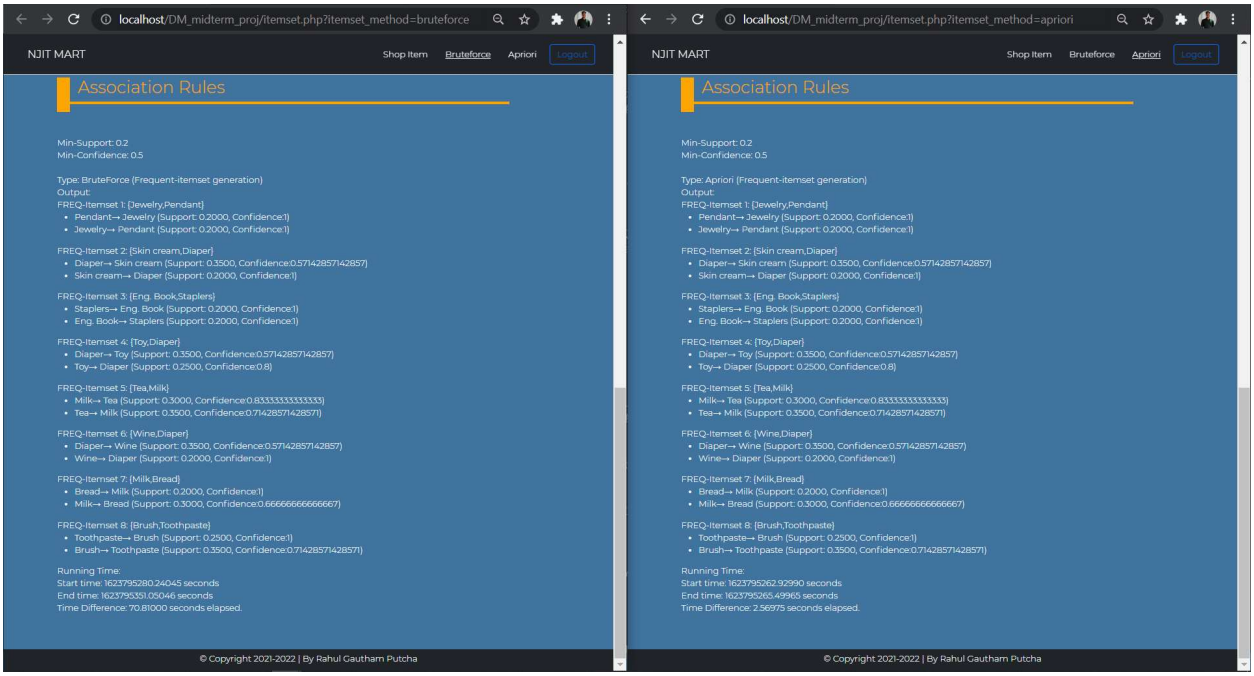



**Fig 6.10. Comparison between Brute-force and Apriori algorithm (Example 1)**

A general note on the output shown in association rules showcases the form indicators, the method used for the frequent-itemset generation, the list of all kth-frequent itemset(s) with each containing the association rules (X→Y) which are shortlisted by the threshold, minimum confidence. Each rule also displays the support ($Support_{(X)}$) and the confidence ($Confidence_{(X \to Y)}$). Finally, we have the information showing the running time.

Another Example: By taking a different min. support and min. confidence values.

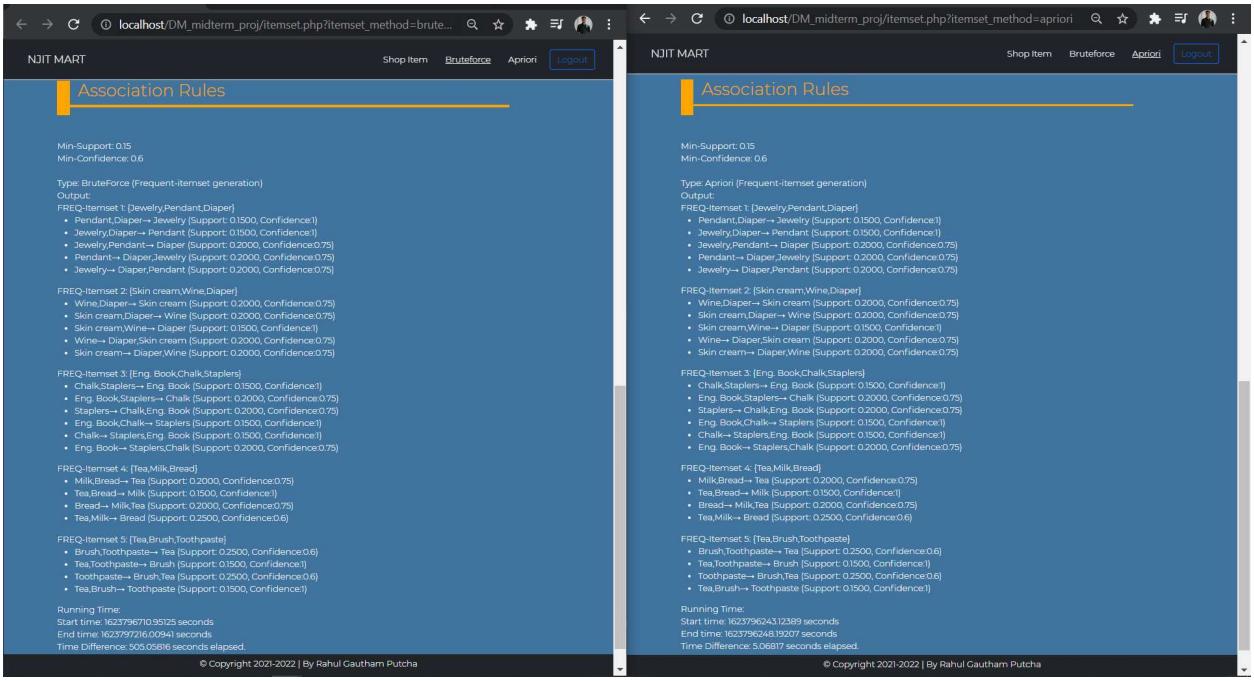



**Fig 6.11. Comparison between Brute-force and Apriori algorithm (Example 2)**

⇨ Shows 196.078% boost by Apriori algorithm in comparison to Brute force method.

We now repeat the same process from Page 25 to Page 30 by creating 5 users, for five databases with each involving a separate record of transactions. In this way, each user will be receiving their own set of association rules based on their past *Input Transactions.*

## 6.3 More Sample Transactions with the application (User 2 to 5)

The following section gives four more sample examples for user transactions. For each of the samples, examples are shown with an image of the user's past input transactions and few images for a side-by-side comparison of the brute-force method and Apriori algorithm.

**User 2:**

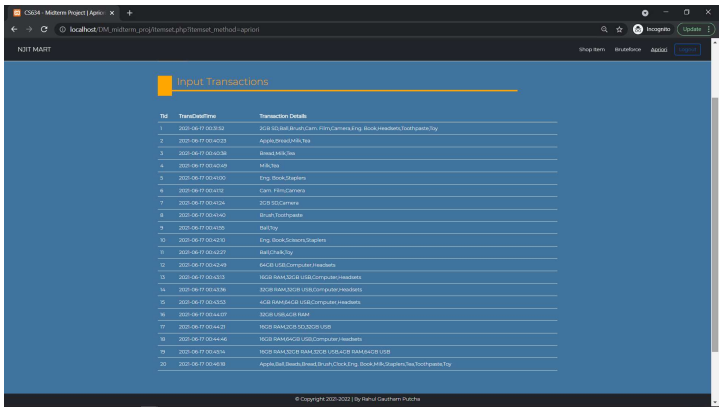



**Fig 6.12. User 2's */itemset.php* transaction**

* I have shirked the above image to make it all fit in a single page. Please zoom in to get clear view of the content in the picture.

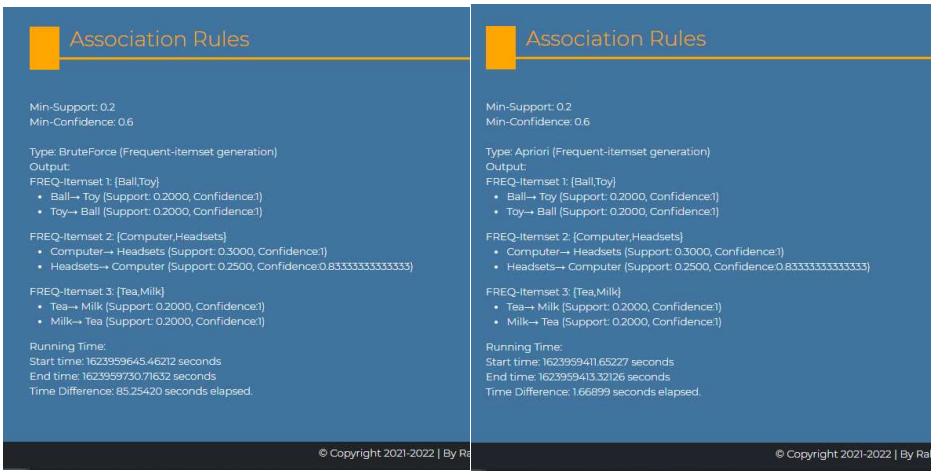



**Fig 6.13. User 2's Brute-force vs Apriori (1)**


**Fig 6.14. User 2's Brute-force vs Apriori (2)**

**User 3:**

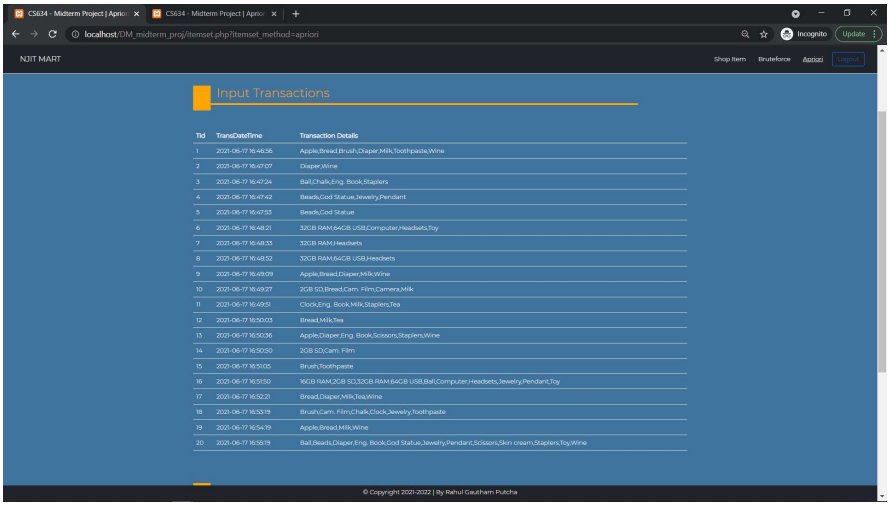



**Fig 6.15. User 3's */itemset.php* transaction**

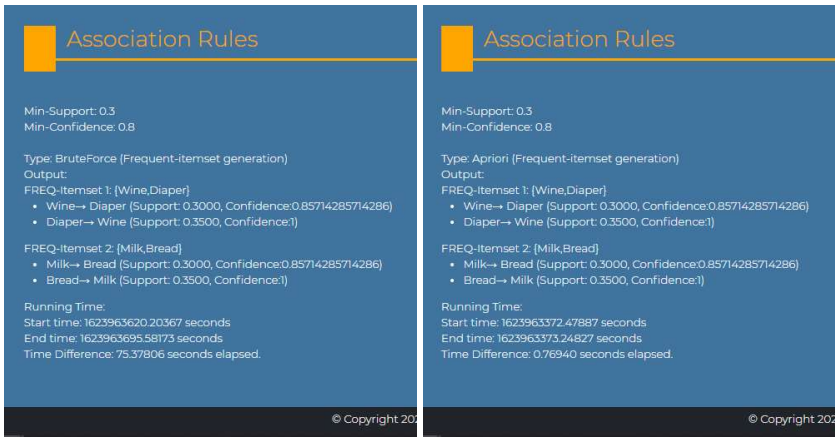



**Fig 6.16. User 3's Brute-force vs Apriori (1)**

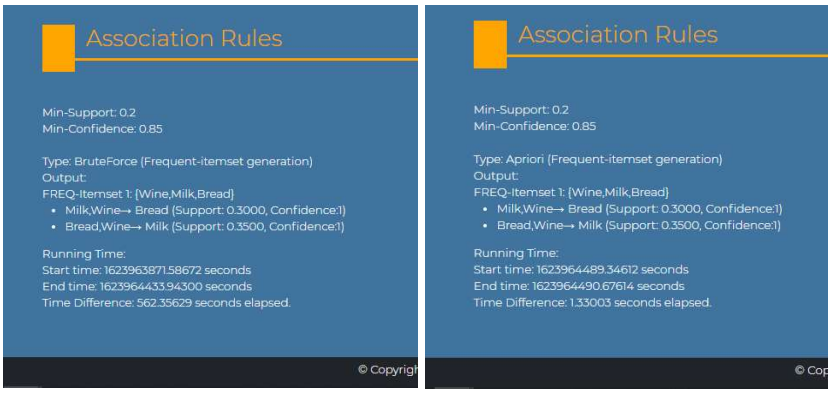



**Fig 6.17. User 3's Brute-force vs Apriori (2)**

**User 4:**

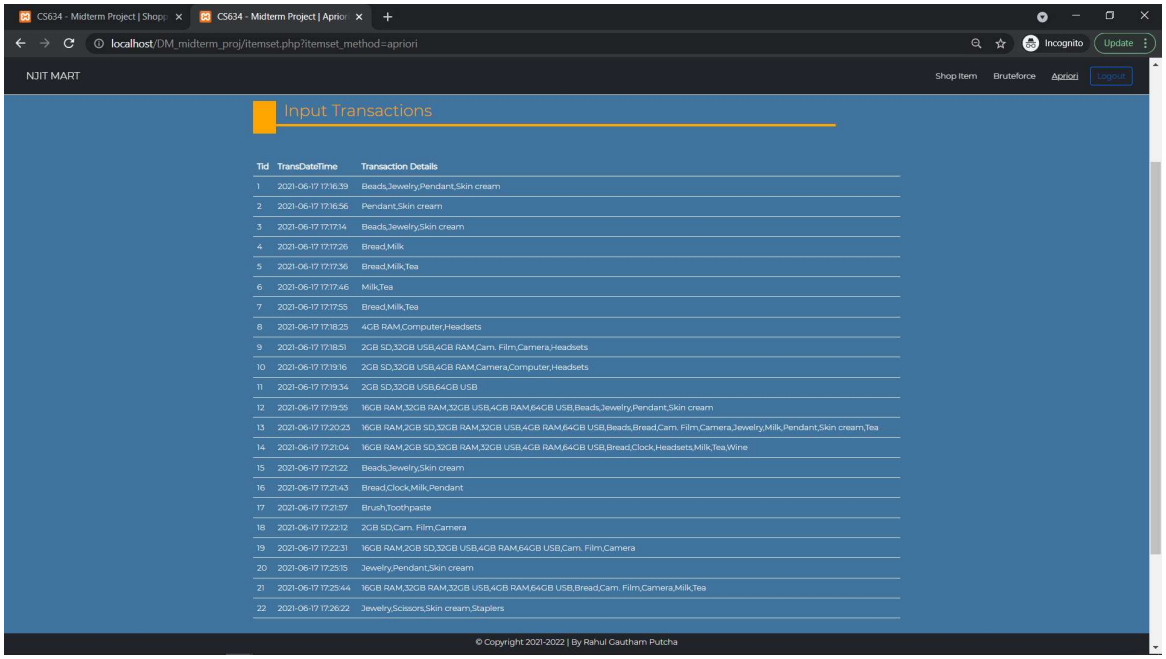



**Fig 6.18. User 4's */itemset.php* transaction**

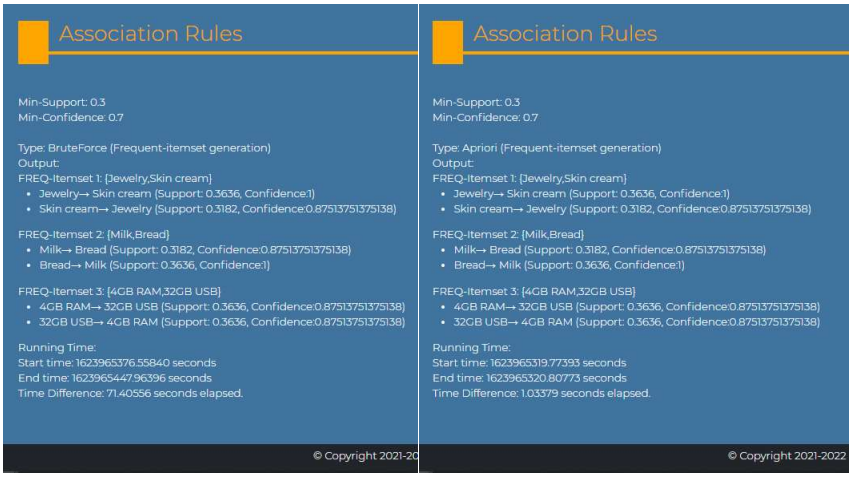



**Fig 6.19. User 4's Brute-force vs Apriori (1)**

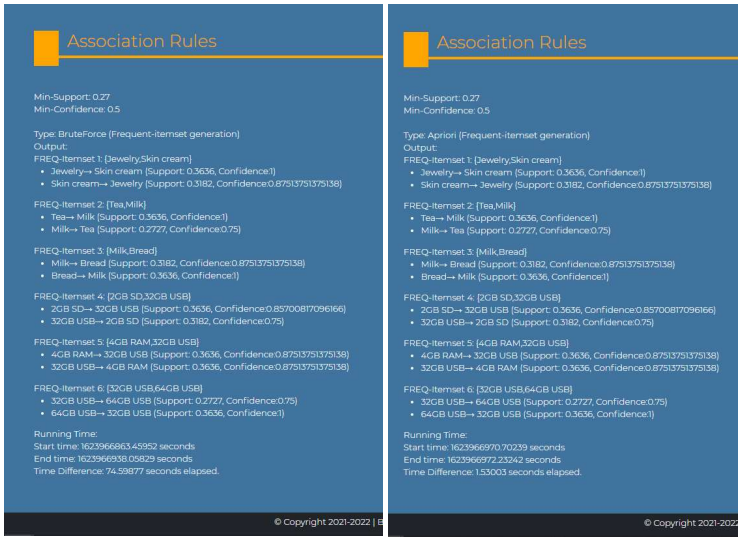



**Fig 6.20. User 4's Brute-force vs Apriori (2)**

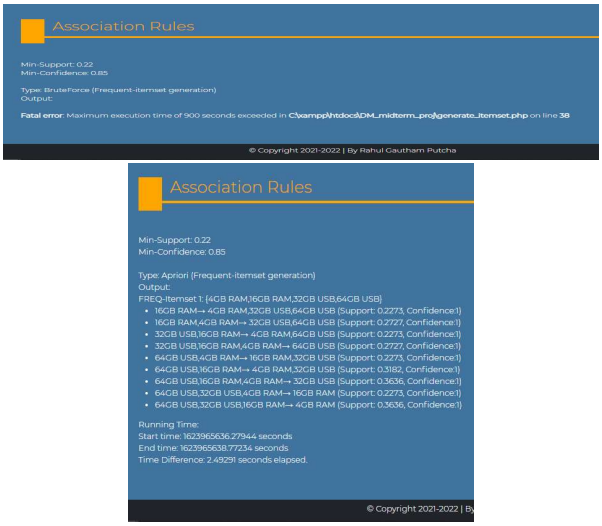



**Fig 6.21. User 4: Drawbacks of Brute-force**

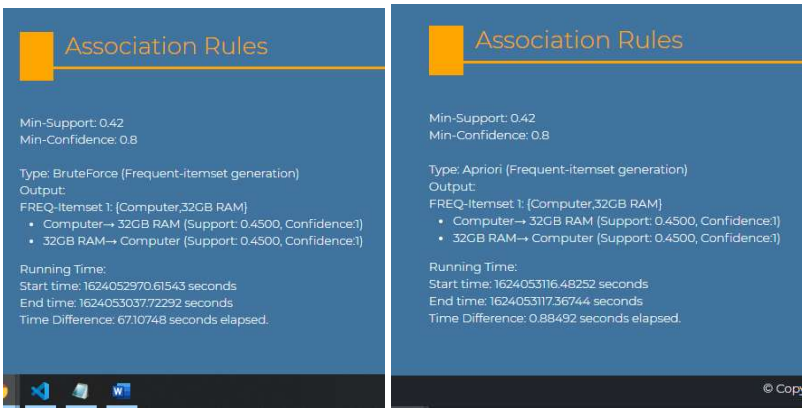
**(In above image, Brute force exceeds the PHP's max recommended execution time.)**

**User 5:**

Input Transactions

| Tid | TransDateTime | Transaction Details |
|---|---|---|
| 1 | 2021-06-18 17:22:10 | Bread,Milk |
| 2 | 2021-06-18 17:22:24 | Bread,Milk,Wine |
| 3 | 2021-06-18 17:22:42 | Bread,Milk,Tea |
| 4 | 2021-06-18 17:22:51 | Milk,Tea |
| 5 | 2021-06-18 17:23:12 | 32GB RAM,64GB USB,Clock,Computer,Headsets |
| 6 | 2021-06-18 17:23:26 | 32GB RAM,Computer |
| 7 | 2021-06-18 17:23:47 | 32GB RAM,64GB USB,Computer,Headsets,Milk,Tea |
| 8 | 2021-06-18 17:24:05 | Apple,Bread,Milk,Tea,Wine |
| 9 | 2021-06-18 17:24:20 | 2GB SD,Cam. Film,Camera |
| 10 | 2021-06-18 17:24:44 | 2GB SD,32GB RAM,64GB USB,Cam. Film,Camera,Computer,Headsets,Milk,Tea |
| 11 | 2021-06-18 17:25:19 | 16GB RAM,2GB SD,32GB RAM,32GB USB,4GB RAM,64GB USB,Cam. Film,Camera,Computer,Headsets |
| 12 | 2021-06-18 17:25:48 | 32GB RAM,64GB USB,Beads,Bread,Computer,Headsets,Milk,Tea |
| 13 | 2021-06-18 17:26:04 | Beads,Jewelry,Pendant,Skin cream |
| 14 | 2021-06-18 17:26:32 | 32GB RAM,64GB USB,Ball,Beads,Computer,Headsets,Jewelry,Pendant,Skin cream,Toy |
| 15 | 2021-06-18 17:27:03 | 32GB RAM,64GB USB,Ball,Beads,Bread,Computer,Headsets,Jewelry,Milk,Pendant,Skin cream,Tea,Toy,Wine |
| 16 | 2021-06-18 17:27:40 | 32GB USB,64GB USB |
| 17 | 2021-06-18 17:28:03 | Apple,Clock |
| 18 | 2021-06-18 17:28:23 | Apple,Bread,Clock,Jewelry,Milk,Skin cream,Tea |
| 19 | 2021-06-18 17:28:43 | Apple,Bread,Clock,Milk |
| 20 | 2021-06-18 17:29:12 | 32GB RAM,64GB USB,Apple,Cam. Film,Camera,Clock,Computer,Headsets |

**Fig 6.22. User 5's */itemset.php* transaction**



Association Rules

Min-Support: 0.42
Min-Confidence: 0.8

Type: BruteForce (Frequent-itemset generation)
Output:
FREQ-Itemset 1: {Computer,32GB RAM}
- Computer→ 32GB RAM (Support: 0.4500, Confidence:1)
- 32GB RAM→ Computer (Support: 0.4500, Confidence:1)

Running Time:
Start time: 1624052970.61543 seconds
End time: 1624053037.72292 seconds
Time Difference: 67.10748 seconds elapsed.

Association Rules

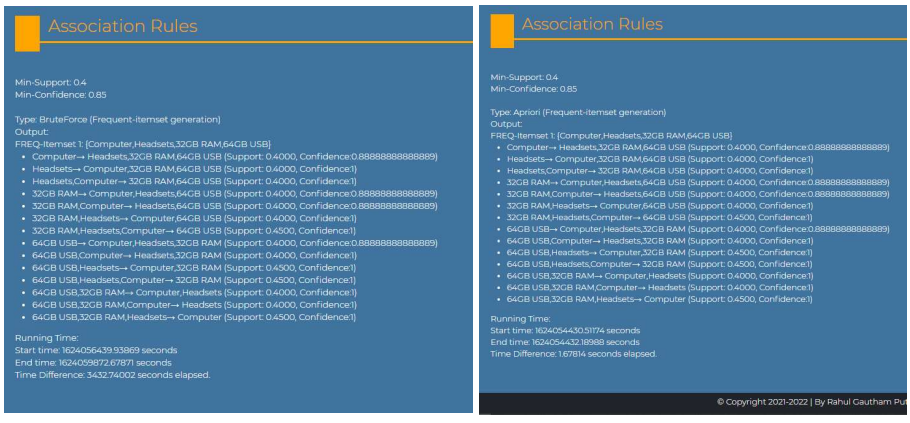
Min-Support: 0.42
Min-Confidence: 0.8

Type: Apriori (Frequent-itemset generation)
Output:
FREQ-Itemset 1: {Computer,32GB RAM}
- Computer→ 32GB RAM (Support: 0.4500, Confidence:1)
- 32GB RAM→ Computer (Support: 0.4500, Confidence:1)

Running Time:
Start time: 1624053116.48252 seconds
End time: 1624053117.36744 seconds
Time Difference: 0.88492 seconds elapsed.

**Fig 6.23. User 5's Brute-force vs Apriori (1)**

**(After adjusting the max PHP execution time from 15 minutes to 100 minutes following are the results. This will make us see that the running time of the brute-force method is 60 min approx. for the above problem transaction set.)**



Association Rules

Min-Support: 0.4
Min-Confidence: 0.85

Type: BruteForce (Frequent-itemset generation)
Output:
FREQ-Itemset 1: {Computer,Headsets,32GB RAM,64GB USB}
- Computer→ Headsets,32GB RAM,64GB USB (Support: 0.4000, Confidence:0.8888888888888889)
- Headsets→ Computer,32GB RAM,64GB USB (Support: 0.4000, Confidence:1)
- Headsets,Computer→ 32GB RAM,64GB USB (Support: 0.4000, Confidence:1)
- 32GB RAM→ Computer,Headsets,64GB USB (Support: 0.4000, Confidence:0.8888888888888889)
- 32GB RAM,Computer→ Headsets,64GB USB (Support: 0.4000, Confidence:0.8888888888888889)
- 32GB RAM,Headsets→ Computer,64GB USB (Support: 0.4000, Confidence:1)
- 32GB RAM,Headsets,Computer→ 64GB USB (Support: 0.4500, Confidence:1)
- 64GB USB→ Computer,Headsets,32GB RAM (Support: 0.4000, Confidence:0.8888888888888889)
- 64GB USB,Computer→ Headsets,32GB RAM (Support: 0.4000, Confidence:1)
- 64GB USB,Headsets→ Computer,32GB RAM (Support: 0.4500, Confidence:1)
- 64GB USB,Headsets,Computer→ 32GB RAM (Support: 0.4500, Confidence:1)
- 64GB USB,32GB RAM→ Computer,Headsets (Support: 0.4000, Confidence:1)
- 64GB USB,32GB RAM,Computer→ Headsets (Support: 0.4000, Confidence:1)
- 64GB USB,32GB RAM,Headsets→ Computer (Support: 0.4500, Confidence:1)

Running Time:
Start time: 1624056439.93869 seconds
End time: 1624059872.67871 seconds
Time Difference: 3432.74002 seconds elapsed.

Association Rules

Min-Support: 0.4
Min-Confidence: 0.85

Type: Apriori (Frequent-itemset generation)
Output:
FREQ-Itemset 1: {Computer,Headsets,32GB RAM,64GB USB}
- Computer→ Headsets,32GB RAM,64GB USB (Support: 0.4000, Confidence:0.8888888888888889)
- Headsets→ Computer,32GB RAM,64GB USB (Support: 0.4000, Confidence:1)
- Headsets,Computer→ 32GB RAM,64GB USB (Support: 0.4000, Confidence:1)
- 32GB RAM→ Computer,Headsets,64GB USB (Support: 0.4000, Confidence:0.8888888888888889)
- 32GB RAM,Computer→ Headsets,64GB USB (Support: 0.4000, Confidence:0.8888888888888889)
- 32GB RAM,Headsets→ Computer,64GB USB (Support: 0.4000, Confidence:1)
- 32GB RAM,Headsets,Computer→ 64GB USB (Support: 0.4500, Confidence:1)
- 64GB USB→ Computer,Headsets,32GB RAM (Support: 0.4000, Confidence:0.8888888888888889)
- 64GB USB,Computer→ Headsets,32GB RAM (Support: 0.4000, Confidence:1)
- 64GB USB,Headsets→ Computer,32GB RAM (Support: 0.4500, Confidence:1)
- 64GB USB,Headsets,Computer→ 32GB RAM (Support: 0.4500, Confidence:1)
- 64GB USB,32GB RAM→ Computer,Headsets (Support: 0.4000, Confidence:1)
- 64GB USB,32GB RAM,Computer→ Headsets (Support: 0.4000, Confidence:1)
- 64GB USB,32GB RAM,Headsets→ Computer (Support: 0.4500, Confidence:1)

Running Time:
Start time: 1624054430.51174 seconds
End time: 1624054432.18988 seconds
Time Difference: 1.67814 seconds elapsed.

© Copyright 2021-2022 | By Rahul Gautham Put

**Fig 6.24. User 5's Brute-force vs Apriori (2)**

# Chapter 7. Conclusion

We have finally reached the conclusion for this project document. We now know how to generate frequent-item sets and the association rules utilizing the Brute-force method and Apriori algorithm.

We learned about the Brute-force method of creating frequent itemset generation which is the fundamental way in which we as humans normally approach the problem and gave a demo algorithm and code for this method. We have also made a comparison of both algorithms and found out that the Apriori algorithm gives us a significant boost in performance by reducing the search space by using Apriori Principle mentioned in Chapter 2.

Therefore, Apriori algorithms is a powerful and fast algorithm for frequent-itemset generation and is widely used in many areas, especially market basket analysis. We may now conclude the midterm project.

# References

[1] https://en.wikipedia.org/wiki/Apriori_algorithm
[2] http://www.cse.msu.edu/~cse960/Papers/MiningAssoc-AgrawalAS-VLDB94.pdf
[3] Data Mining: Concepts and Techniques, Han et al., Elsevier, 2011, ISBN 978-0-12-381479-1
[4] Introduction to Data Mining, Tan et al., Pearson, 2019, ISBN-13: 978-0-13-312890-1