

# **WEB - BASED OFFICE MARKET**

A CS 298 Project Report

Presented To

The Faculty of the Department of Computer Science

San Jose State University

In Partial Fulfillment

Of The Requirements for The Degree

Master of Science (Computer Science)

**BY**

**Manodivya Kathiravan**

**MAY 2017**

© 2017

Manodivya Kathiravan

ALL RIGHTS RESERVED

The Designated Project Committee Approves the Master's Project Titled  
Web – based Office Market

by  
Manodivya Kathiravan

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE  
SAN JOSE STATE UNIVERSITY  
MAY 2017

---

Dr. Chris Pollett, Department of Computer Science

Date

---

Dr. Robert Chun, Department of Computer Science

Date

---

Mr. Rajesh Navantheenakrishnan, SonicWALL

Date

APPROVED FOR THE UNIVERSITY

---

Associate Dean Office of Graduate Studies and Research

Date

## **ABSTRACT**

People who work in an office often have different pools of resources that they want to exchange. They want to trade their resources/work(seller) with a person who wants that particular resource(buyer) and in return get another resource the buyer offers. These kind of exchanges are often called Barter-exchanges where an item is traded for another item without the involvement of actual money. An exchange is set to be complete when there is a match between an available item and a desired item. This exchange is called direct exchange. When an item desired by one user is made available through a series of intermediate exchanges this is called as multi-lateral exchange. In this project, I designed an online bartering system with multi-lateral possibilities. The algorithm implemented by the system supports multi-way trades using a graph data structure and then searches the graph for paths to effect the trade. The algorithm also identifies a sequence of trades through at least one intermediate trader that will complete the trades of the two original traders. The system starts with the search for direct exchange for a given scenario. If no such exchange is found, then the system starts looking for an indirect exchange. The system also incorporates a rating mechanism to provide user ratings based on successfully completed trades. A/B testing and stress testing was performed to test the reliability and efficiency of our system.

## **ACKNOWLEDGEMENT**

I would first like to thank my project advisor, Dr. Christopher Pollett for his constant guidance and enduring patience throughout this project. He consistently allowed this paper to be my own work, but steered me in the right the direction whenever he thought I needed it. I would also like to extend my thanks to my committee members, Dr. Robert Chun and Mr. Rajesh Navaneethakrishnan, for their suggestions and time.

I must express my very profound gratitude to my husband Rajesh Perumal for providing me with unfailing support and continuous encouragement throughout my years of study. This accomplishment would not have been possible without him. Finally, I would like to thank my family and friends for their constant motivation.

## **TABLE OF CONTENTS**

1. INTRODUCTION.....	7
2. BACKGROUND.....	10
3. DESIGN OF MULTI-BARTERING SYSTEM.....	15
3.1 Introduction to the algorithm.....	16
3.2 The multi-barter exchange algorithm.....	17
4. IMPLEMENTATION.....	23
4.1 Registration module.....	24
4.2 Add item module.....	24
4.3 Show item module.....	25
4.4 Barter module.....	26
4.5 View trades module.....	27
4.6 Analyzed Scenarios.....	28
4.6.1 Minimum Length Exchange Sequence.....	28
4.6.2 Avoid cycles in the Exchange Sequence.....	30
4.6.3 Satisfy the needs of current exchange pair.....	31
5. EXPERIMENTS.....	33
5.1 Experiments conducted to unit test all the features in the barter system.....	33
5.1.1 Testing a Controller.....	34
5.1.2 Testing an API Call.....	34
5.2 Experiments to stress test the system and find out how it reacts to different sets of data.....	35

5.3 A/B Testing.....	39
6. CONCLUSION .....	42
7. REFERENCES.....	43

## **LIST OF FIGURES**

Figure 3.1. A simple direct barter exchange.....	18
Figure 3.2: Activity diagram for multi barter system .....	21
Figure 3.3: The initial tree as the multi-way trading algorithm begins .....	23
Figure 3.4: Expanded tree .....	23
Figure 3.5: A further expanded tree .....	24
Figure 4.1: MEAN stack flow diagram .....	26
Figure 4.2: bExchange system's User Registration .....	27
Figure 4.3: bExchange system's Add item webpage .....	28
Figure 4.4: bExchange systems' show items webpage.....	28
Figure 4.5: bExchange systems' show items webpage.....	29
Figure 4.6: bExchange system's barter module .....	30
Figure 4.7: bExchange system's view trades webpage .....	31
Figure 4.8: The shortest exchange sequence .....	32
Figure 4.9: A cycle in the exchange sequence.....	33
Figure 4.10: Exchange sequence tree .....	34
Figure 5.1: Testing a Controller.....	37
Figure 5.2: Testing an API Call .....	38
Figure 5.3: Graph plotted for time taken vs number of nodes .....	39
Figure 5.4: Trade scenario with node value 3 .....	40



Figure 5.5: Trade Scenario with node value 5.....	41
Figure 5.6: Trade Scenario with node value 10.....	42
Figure 5.7: Variant B of show items page .....	43
Figure 5.8: Variant A of show items page .....	43

## **LIST OF TABLES**

Table 4.1: A possible exchange pool.....	31
Table 5.1: Number of exchanges and time taken by the algorithm to compute the result.....	37

# **CHAPTER 1**

## **INTRODUCTION**

This chapter starts with the introduction to barter system and expands on the gist of the project. Bartering system has been widely in use even before the invention of money [1]. Traders swapped items and services for other items and services. In ancient times, this system involved people in the same area, however today bartering is global. Bartering has made a comeback using techniques that are more sophisticated to aid in trading; for instance, the Internet. Bartering doesn't involve money, which is one of the advantages. You can buy items by exchanging an item you have but no longer want or need. Generally, trading in this manner is done through Online auctions and swap markets.

In this project we intend to develop a web - based bartering system for office markets which can be used as a portal to trade in the items a user has for other items he desires. There will not be any use of money in the system. The system contains a trade pool or an exchange pool with a set of available and desired trade items. When an available item matches the desired item, an exchange can be made or the trade is possible.

Current online bartering systems such as uExchange, Craigslist perform direct bartering such that a trade is made if the desired item by one user matches the seeking item by another user. In this project we try to come up with a multi-barter system which can look for indirect bartering possibilities for the user if direct bartering is not possible. We call this system for our reference as 'bExchange'.

A class of mathematical problems called matching problems has been of interest in mathematics, operations research, and optimization for some time. In any matching problem there are a group of objects. The idea is to connect the objects in a pair-wise fashion with one another. The quest in mathematics has been to find an algorithm, given a set of objects, that will efficiently compute the maximum number of pairings or matches that can be produced from the set of objects. This is the main requirement behind our multi-barter algorithm. We need to find the maximum pairings from the given set of exchange pool.

An algorithm that uses brute-force approach to find out a match will not be practical. A different approach was proposed by Jack Edmonds et al [2] to find pairs in polynomial time. The main idea behind his approach is to construct a path among two vertices in the graph using an alternating sequence of matched edges and non-matched edges.

The leading question for this project is, how efficiently a possible multi-barter scenario can be predicted with any given set of objects. Kaplan in his paper [3] described an algorithm that uses graph to find multi-way trading possibilities. The algorithm described by him finds at least one midway trader between two original traders who will facilitate the exchange process. Given n-possible barter ways an efficient algorithm must find the minimum length sequence instead of the longest sequence. The system must be able to predict direct barter and multi-barter scenarios. If there are more than one user providing the same offer, then there has to be a way to pick a user based on some kind of ratings. This rating must be based on successfully completed transactions by the user.

Chapter 2 discusses the background research done in the field of multi-barter systems. Chapter 3 discusses the detailed overview of the proposed multi-barter system and the algorithm behind. Chapter 4 gives an overview of the implementation strategy used for the multi-barter system. It also briefs about the mechanism to give ratings to user based on successfully completed transactions. Chapter 5 describes the possible heuristics that are considered to reduce the search space and experiments performed on the system. Finally, chapter 6 discusses areas of improvement and future work.

## **Chapter – 2**

### **Background**

In this chapter we will be discussing about the various background and preliminary study made on bartering or match making system and how it differs from the system that we intent to develop. As described by Sonmez et all [3] in his paper the multi-way bartering or trading is related to matching theory that falls under the important topic in mathematics. The main feature of matching algorithm is to assign the items needed by the user to each other. It also focuses on meeting the expectation of the users and how close the expectation is met.

An interesting problem in mathematics is the house allocation problem. It shares common background to our bartering system. According to this problem there are a group of houses that needs to be assigned to a group of people. Every person has a personal choice on what house they want. They rate the houses as first choice, second choice etc. The motivation is to allocate the person with the house that is high on the list.

There is a possibility that the top most preferred house by the user can go unavailable. In such a case the person second most preferred house will be taken into consideration. Consider if the person wants to trade a house for every house he gets through the house allocation. This is very similar to our trading problem where a user wants to trade in an item for every item he wants.

Sonmez et all [3] proposed an algorithm to this problem called as serial dictatorship algorithm. A priority is assigned to every person by a procedure based on a function. The person holding the top priority will be allocated the house of their choice.

The next person on the priority list will be allocated the next house. After all the houses have been allocated the algorithm will terminate.

Although this is not specifically applicable to bartering, matching theory is certainly applicable in that it is the same problem as the housing problem – namely the assignment of desired resources to agents that desire them. The concept of trading sequences in multi- way trading comes to play in matching theory when we consider another classic example, the stable marriage problem by Halldorsson et al. [4]. In the stable marriage problem, there are an identical number of men and women. Each has a preference list of whom they wish to marry of the opposite sex.

A stable matching is one in which there is no pair of others who are favored more than their current partners. This problem has many practical applications including the assignment of medical students to hospitals. An important aspect of this problem is to identify a maximal matching such that the highest number of pairs of men and women are matched. An impediment to a stable matching is a blocking pair. A matching can only be called stable if there are no blocking pairs in it. We use the letter  $m$  to denote a male and  $w$  to denote a female.  $M$  represents the set of females and males that can be matched.

According to Halldorsson et al [4], the following conditions need to be met in order for a pair to be a blocking pair.

- (1)  $m$  and  $w$  are not matched together in  $M$ , but are acceptable to each other
- (2)  $m$  is either single in  $M$  or prefers  $w$  to  $M(m)$
- (3)  $w$  is either single in  $M$  or prefers  $m$  to  $M(w)$ .  $M(w)$  refers to the male matched to  $w$  and  $M(m)$  refers to the female matched to  $m$ .

In any list of preferences there may be ties when two or more agents make the same selection. These ties will result in conflicts in the assignments of men to women, and vice versa. Removing the ties yields a possible matching of men and women. An algorithm to accomplish this runs in  $O(N^2)$  time. It operates by scanning all assignments and rearranging the ties in such a way as to create new lists of preferences. If we consider that a trade preference is like a partner preference, then we can see how the algorithm for computing a stable marriage might be used to compute possible trades. It certainly is the case that compatible trades are similar to compatible partner matches. If multi-way trading results in a prioritized list of trades, then this approach would be similar to what would work to solve the stable marriage problem. This is not the case in multi-way trading that we consider here though. The introduction of multiple trades enables more flexibility in the multi-way trading problem but makes the problem different than the stable marriage problem.

Abraham et al. [5] also discusses a similar application of match making for the use in kidney exchange. In this problem there are a set of recipients and donors. Each recipient gets a kidney from the donor. The allocation of recipients to a donor is based completely on genetical background. If a match is not found, then the look for multi-way exchange comes into picture. This problem is also very similar to our core concept of multi way bartering.

The only difference between these two processes involves the criteria used to determine whether a trade can be made. In multi-way trading, the primary criterion is based on whether a trader wants a particular object. In kidney matching, a trade can be made when a donor's kidney is biologically compatible with a recipient. Unver's [6]



recognizes an important aspect of the kidney exchange problem: the pool of available kidneys is constantly changing. The goal of Unver's work is to maximize the matches using the whole pool of recipients and donors. He describes ways to accomplish this while considering the actual matching process as a "black box" process, in that the details of the process are not important. Another class of problems that bears a resemblance to multi-way trading can be found in the work focusing on combinatorial auctions by Vries and Vohra [7].

In a combinatorial auction, bidders bid on packages of items. Objects to be bid on may be complementary or they may be able to be substituted for one another. An interesting aspect of this type of auction is that when a bidder wants to combine goods to bid on, they must submit a bid for every subset of the objects they are interested in. Another problem is how to transmit the bids to the auctioneer. Obviously, depending on the size of the set of items to be bid upon, the number of subsets can be quite large. The only way to resolve these problems is to place restrictions on the way that bidders can combine items and also on the way they can place bids. Unlike the multi-way trading process, the combinatorial process seeks to find combinations of items that are complementary.

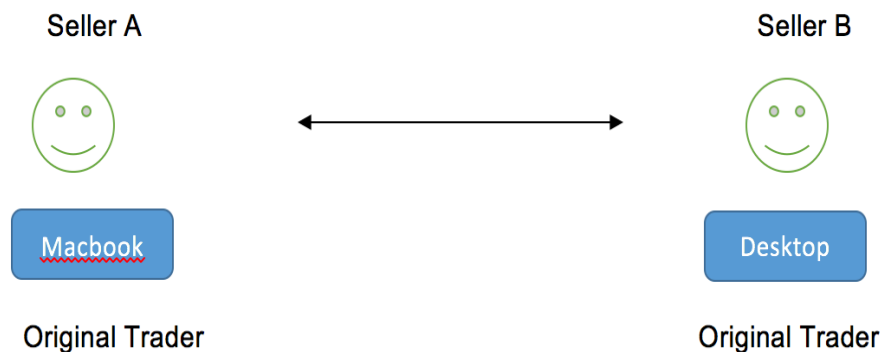
All of the problems and algorithms discussed above are very similar to the problem we are trying to solve which is the match making using multi-bartering exchanges. Current popular online sites such as Craigslist, uExchange perform similar functionalities to that of online direct bartering without involving any actual money. The algorithm they have find direct matches quickly and efficiently whereas they don't have any kind of multi-barter possibilities. Also if too many users in the pool offer the same

exchange then there has to be a way to pick one. This kind of picking is done either on the server side (for example: randomly or based on some priority) or must be left to the user's choice by giving them user ratings for every user. Our system bExchange tries to incorporate both these ideas and bring up a better suggestion mechanism to the end users.

## **CHAPTER 3**

### **DESIGN OF MULTI-BARTERING SYSTEM**

In this chapter we will in detail discuss about the multi barter algorithm used and how the system is designed to accommodate both direct and multi level exchanges. If the only exchanges in the exchange pool consisted of the one that were desirable to complete the exchange, then it would be insignificant. Unfortunately, this will not always be the scenario. The exchange pool will consist of random number of exchanges. If there exists a desired exchange in the pool, in order to find out the correct exchange sequence we need to methodically create exchange sequences of possibly all lengths. We would ideally start with all exchange sequences of length two.



**Figure 3.1. A simple direct barter exchange**

A simple direct exchange trade of length two from the given pool of available trades is illustrated in figure 3.1. Seller A is willing to offer a MacBook and desires a desktop. Seller B is willing to offer a desktop for a MacBook. If no such sequence is found, then all exchange sequences of length three will be considered until a fulfilling exchange sequence is achieved. This will continue through consecutively higher

sequence lengths until an exchange sequence is found or until a threshold is reached on the exchange sequence. The complexity of worst case running time is  $O(n^n)$ . This complexity is unacceptable and we have to find an efficient way to search the possible paths.

### **3.1 Introduction to the Algorithm**

The set of possible trades  $P$  consists of tuples  $p$ , which we call an exchange pair. In the following discussion,  $O$  denotes an object in the trading pool,  $D$  denotes objects in a trader's desired trade, and  $T$  denotes the trading pool. The subscript  $d$  is used to mark an object as a desired object in the trading pool. The subscript  $o$  is used to mark an object in the trading pool as one that is offered or available for trade. Thus  $O_d$  denotes a desired object and  $O_o$  denotes an object available for trade. A trading pair tuple consists of a desired object  $O_d$ , and an offered object  $O_o$ , as in  $\langle O_d, O_o \rangle$ . The trader, who is the person wanting to make the trade, has a desired trade that he wants to have consummated. The desired trade is also a tuple consisting of  $\langle D_d, D_{df} \rangle$ .

Suppose we have set of tuples,  $T$ , representing all of the possible trades that can take place the trading pool:  $T = \{\langle O_{d1}, O_{o1} \rangle; \langle O_{d2}, O_{o2} \rangle; \dots; \langle O_{dn}, O_{on} \rangle\}$  In the trivial case there exists some  $\langle O_{di}, O_{oi} \rangle$  such that  $\langle O_{di}, O_{oi} \rangle \in T$  and  $O_{oi} = D_d$  and  $O_{di} = D_o$ . In the non-trivial case there is no tuple  $\langle O_{di}, O_{oi} \rangle$  that satisfies these criteria, so we need to look for a trading sequence in the multi-way trade that will achieve the same result. If  $S$  is an ordered set of trading pairs that consummates the trade, we are searching for the set  $S$ , as follows:

$$S = \{\langle O_{d1}, O_{o2} \rangle, \langle O_{d2}, O_{o3} \rangle, \langle O_{d3}, O_{o4} \rangle \dots \langle O_{dn-1}, O_{o1} \rangle\}$$

In the above sequence  $\langle O_{d1}, O_{o2} \rangle$  partially satisfies  $\langle O_{d2}, O_{o3} \rangle$  because the object offered in the first pair  $O_{o2}$  is the desired object of the second pair  $O_{d2}$ . Trade 2 is partially satisfied by Trade 1. At the end of the sequence the previous trade satisfies the final trade, and the offered object  $O_{o1}$  of this trade satisfies the originally desired object  $O_{d1}$ . The multi-way trading algorithm identifies the set S of trade tuples that satisfies the originally desired trade.

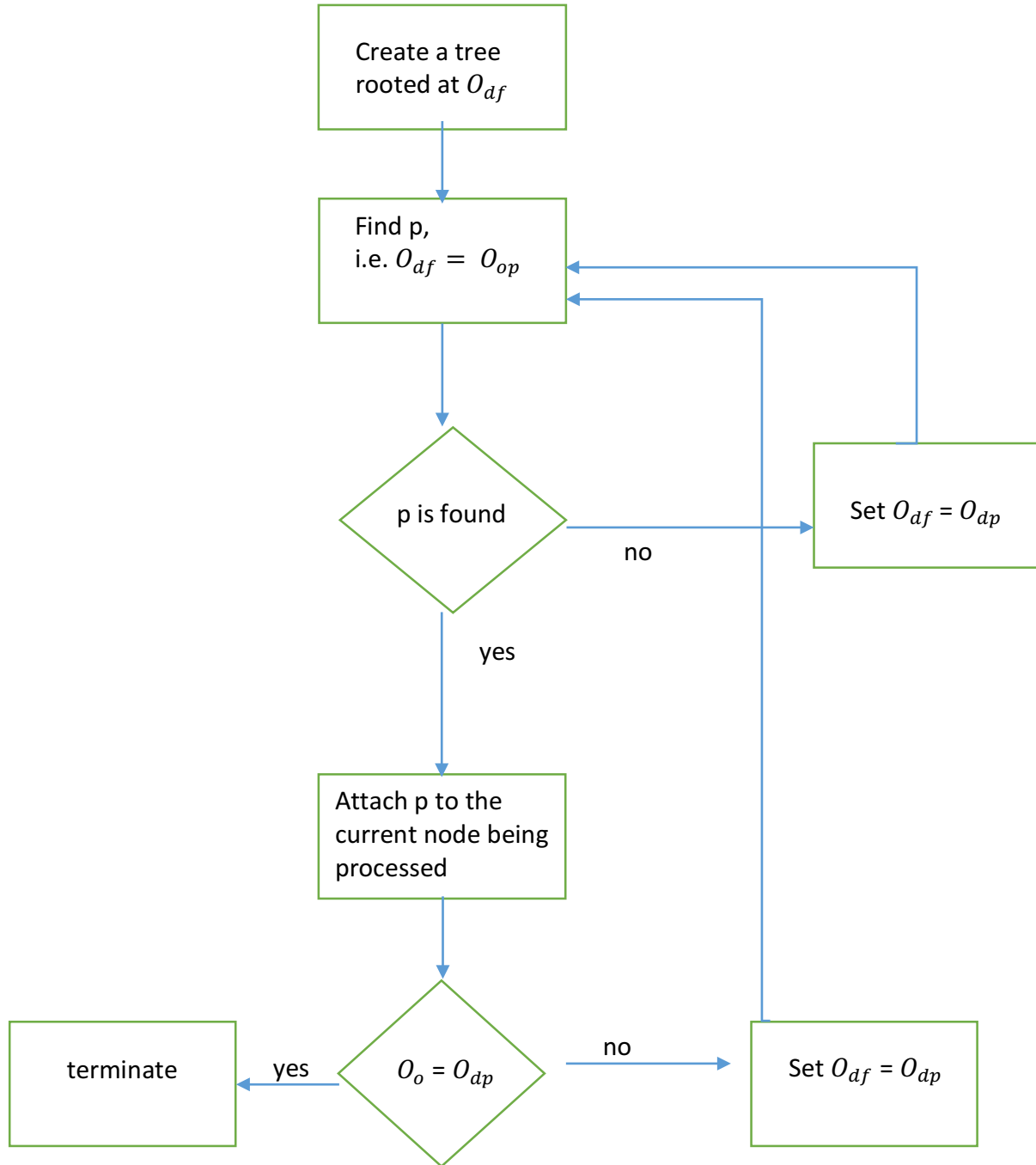
### **3.2 The Multi way trading algorithm**

The multi-way trading problem is one of finding the set S of trading pairs,  
 $S = \{ \langle O_{d1}, O_{o2} \rangle, \langle O_{d2}, O_{o3} \rangle, \langle O_{d3}, O_{o4} \rangle \dots \langle O_{dn-1}, O_{o1} \rangle \}$ . S represents a sequence of trades that will ultimately fulfill a trader's originally specified desired trade,  $\langle O_d, O_o \rangle$ . We need to extract this sequence from a set of arbitrarily many trading pairs. P is the set of all of the trading pairs. To create a tree T whose root is  $O_o$ , the object offered by the trader desiring the object  $O_d$ , we do the following:

(1) We search through the set of possible trades P for all trading pairs such that  $O_i = O_d$ . We attach the trading pairs found, if any, to the current node being expanded. Initially the node that is being expanded is the root of the tree or the original trader's desired object. If there are no objects that meet these criteria, then the path currently being extended by this process ends and the algorithm proceeds to the next available node waiting to be extended. At this point, we continue with the left-most node that has not already been extended.

(2) If  $O_{di} = O_o$ , then this path is finished and the algorithm terminates because we have found the sequence of trades leading to the desired trade.

(3) If  $O_{di} \neq O_d$ , then  $O_d$  is set to  $O_{di}$  and proceed to Step 1.



**Figure 3.2: Activity diagram for multi barter system**

Eventually a path to the desired trade will be completed, and a trade sequence will be found. Otherwise, all paths will have been extended to their limit with no trade sequence found, and the algorithm will terminate with no sequence found. A simple

block diagram depicting the steps to find the desired sequence is explained in figure 3.2.

We start with a desired trading pair  $\langle O_d, O_o \rangle$  a set  $P$  of all possible trades, and we set  $O_{df}$  to  $O_d$ , and proceed with the following steps:

1. Create a tree rooted at  $O_{df}$ .
2. find all  $p$  such that  $O_{df} = O_{op}$ .
3. If there are no  $p$  such that  $O_{df} = O_{op}$ , then the path is complete. Set  $O_{df} = O_{dp}$  of the left-most unprocessed node and proceed to Step 2. If there is no left-most node, then processing is complete and no trading sequence has been found.
4. If there are  $p$  such that  $O_{df} = O_{op}$ , then attach each  $p$  found to the current node being processed such that the child of the node is  $O_{op}$  and the child of the added node is  $O_{dp}$ . If  $O_o = O_{dp}$ , then the trading sequence has been found starting at the root to this node. Since the trading sequence has been found, the algorithm terminates.
5. Set  $O_{df}$  to  $O_{dp}$  for the left-most unprocessed node, and go to Step 2.

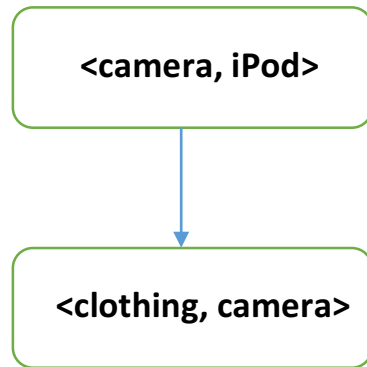
Let's consider a simple exchange example. We define the set of trading pairs  $P$  as follows:

$P = \{\langle \text{camera}, \text{iPod} \rangle, \langle \text{iPod}, \text{shoes} \rangle, \langle \text{shoes}, \text{clothing} \rangle, \langle \text{clothing}, \text{camera} \rangle, \langle \text{shoes}, \text{flashlight} \rangle, \langle \text{flashlight}, \text{radio} \rangle\}$ .

For this example,  $\langle O_d, O_o \rangle = \langle \text{camera}, \text{ipod} \rangle$ . Now, set  $O_{df} = O_d$  with  $O_{df} = \text{camera}$ . We begin by creating a tree with root node  $O_{df}$ . Then, we find all  $p$  such that  $O_{df} = O_{op}$ . Note that we are looking for all  $p$  such that the offered object of the pair is the same as the desired object of the current node (which initially is the root node). From

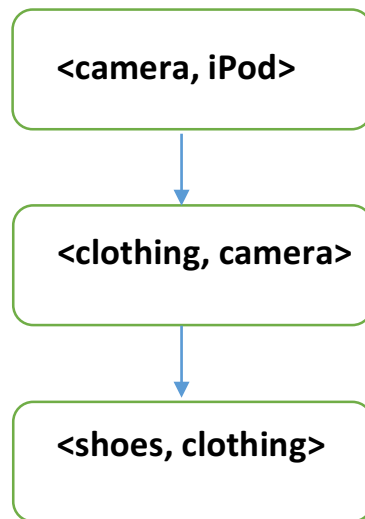
$P = \{ \langle \text{camera}, \text{iPod} \rangle, \langle \text{iPod}, \text{shoes} \rangle, \langle \text{shoes}, \text{clothing} \rangle, \langle \text{clothing}, \text{camera} \rangle, \langle \text{shoes}, \text{flashlight} \rangle, \langle \text{flashlight}, \text{radio} \rangle \}$

we can see that there is  $\langle \text{clothing}, \text{camera} \rangle$ .



**Figure 3.3: The initial tree as the multi-way trading algorithm begins**

In this case, there are  $p$  such that  $O_{df} = O_{op}$ . The algorithm specifies that all of the pairs found should be attached to the current node being processed. In this case, that is the root node. The resulting tree is shown in Fig. 3.4.



**Figure 3.4: Expanded tree**

We proceed as follows. Now  $O_{df}$  is set to clothing from the trading pair  $\langle \text{clothing}, \text{camera} \rangle$ . Next, we find all  $p$  such that  $O_{df} = O_{op}$ . We note that



$$P = \{<iPod, shoes>, <shoes, clothing>, <shoes, flashlight>, <flashlight, radio>\}.$$

This is represented in figure 3.3. That trading pair that meets this condition is <shoes, clothing> as in figure 3.4.

We next set  $O_{df}$  to shoes from the trading pair, <shoes, clothing>. Again we find all  $p$  such that  $O_{df}=O_{op}$ . After this, the set of possible trades is  $P = \{<iPod, shoes>, <shoes, flashlight>, <flashlight, radio>\}$ . The resulting tree is shown in Fig. 3.5 now. At this point  $O_o = O_{dp}$ , where  $O_o = iPod$  and also  $O_{dp}=iPod$ , then a path has been found that offers the originally desired object.

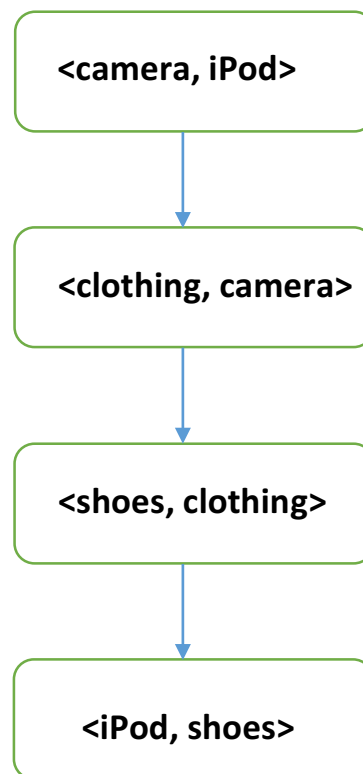


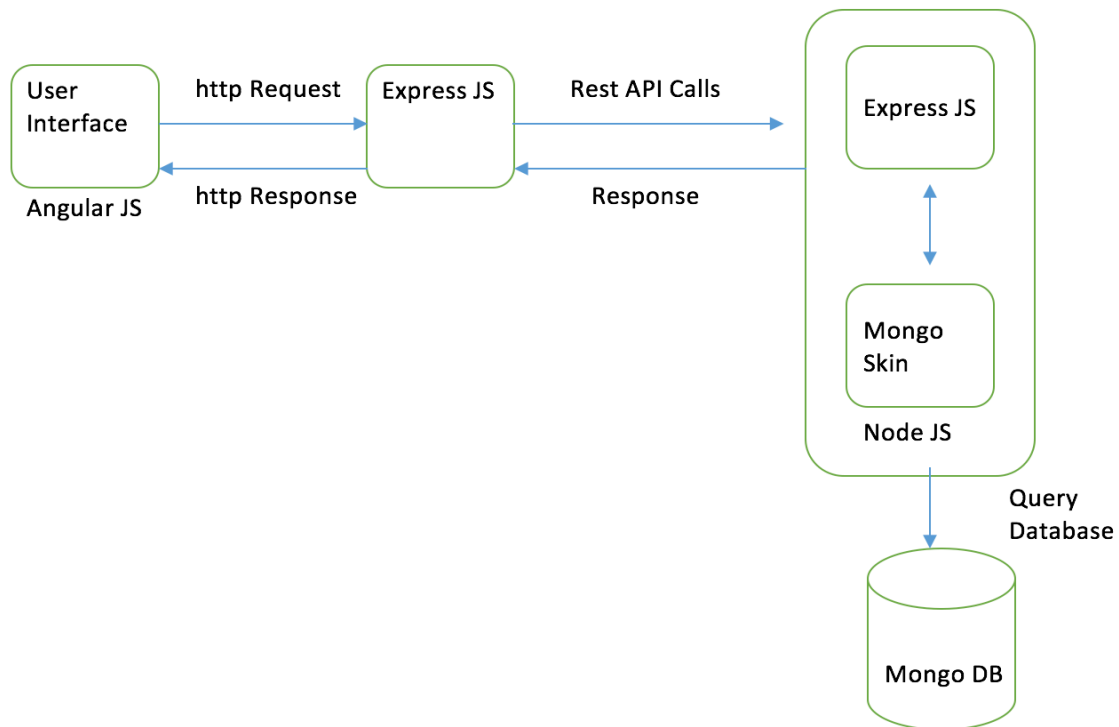
Figure 3.5: A further expanded tree

The above example explains how a possible sequence is found from a given set of exchanges. The system is designed to find such possible sequences starting from length two. We will further see about the implementation of this design in the next chapter.

## **CHAPTER – 4**

### **IMPLEMENTATION**

In this chapter we will in detail discuss about the implementation part of the system, the technologies used, various core modules of the system and analysis of the algorithm on few corner cases. The application is implemented using MEAN stack. MEAN acronym expands to (Mongo DB, Express JS, Angular JS, Node JS). Angular JS is used on the client side/ front-end of the MEAN stack. Node JS is used on the server side. Express JS is used as a node JS web application framework. Mongo DB is a no SQL database that is scalable and uses JSON like documents. A block diagram depicting the flow is illustrated in figure 4.1.

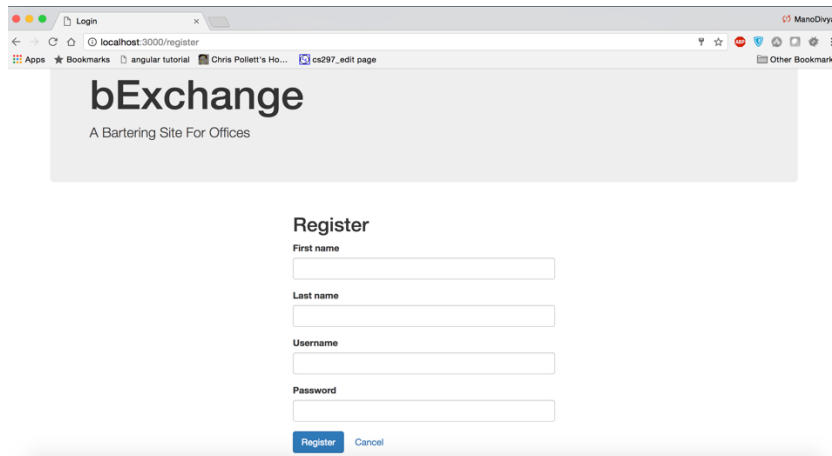


**Figure 4.1: MEAN stack flow diagram**

The application contains many core modules. Each of the core module is explained in detail below.

## **4.1 Registration Module**

The first or one of the initial module is the registration module. We have split the login and registration pages out from the angular application in order to secure access to the angular client files, so all front end angular files (including JavaScript, CSS, images etc.) are only available to authenticated users. The main benefit of securing the client files is just to prevent any accidental leak of any secure information. Figure 4.2 is the webpage for the user registration module of the system. Once a user registers successfully the page redirects to the login page. The user is expected to login using the information given during registration. Upon successful login the user can see the home page.

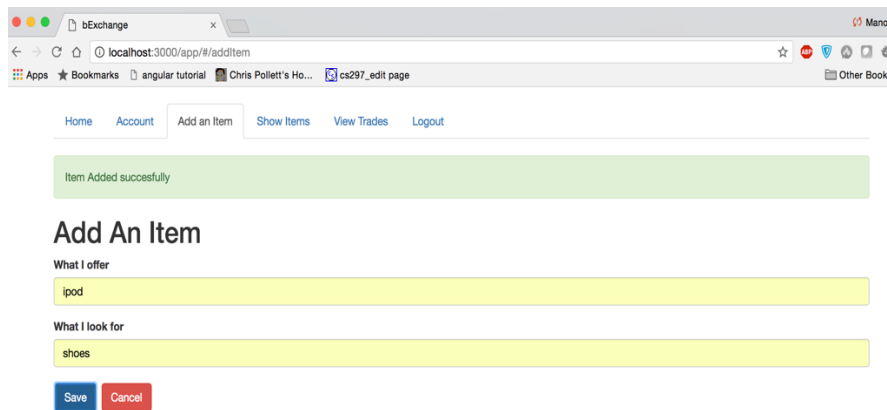
A screenshot of a web browser displaying the registration page for 'bExchange'. The browser's address bar shows 'localhost:3000/register'. The page has a light gray header with the 'bExchange' logo and the tagline 'A Bartering Site For Offices'. Below the header, there is a 'Register' section with four input fields: 'First name', 'Last name', 'Username', and 'Password'. At the bottom of this section are two buttons: a blue 'Register' button and a gray 'Cancel' button. The browser's bookmark bar is visible at the top, showing links to 'angular tutorial', 'Chris Pollett's Ho...', and 'cs297\_edit page'.

**Figure 4.2: bExchange system's User Registration**

## **4.2 Add Item Module**

The add item module will add an item to the exchange pool. The user can enter the item they are willing to offer and the item they are looking for. The exchange

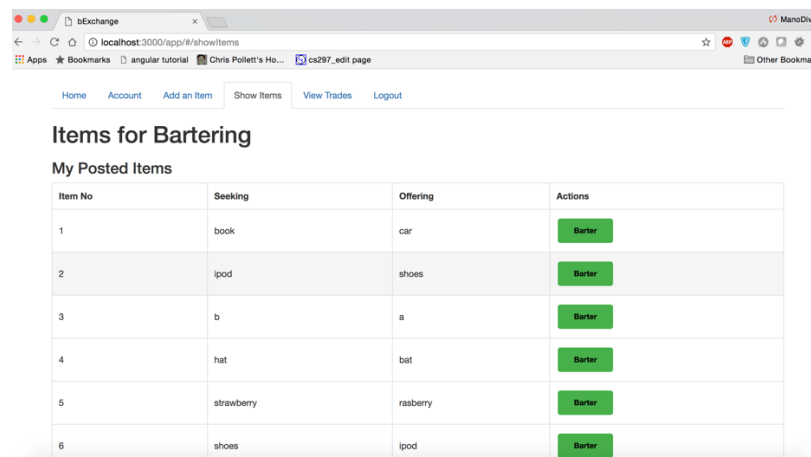
information is stored in a JSON object and a POST API request is made to the server. The server then sends the information to the mongo DB where an entry with the exchange and user information is made to the database.



**Figure 4.3: bExchange system's Add item webpage**

### **4.3 Show Items Module**

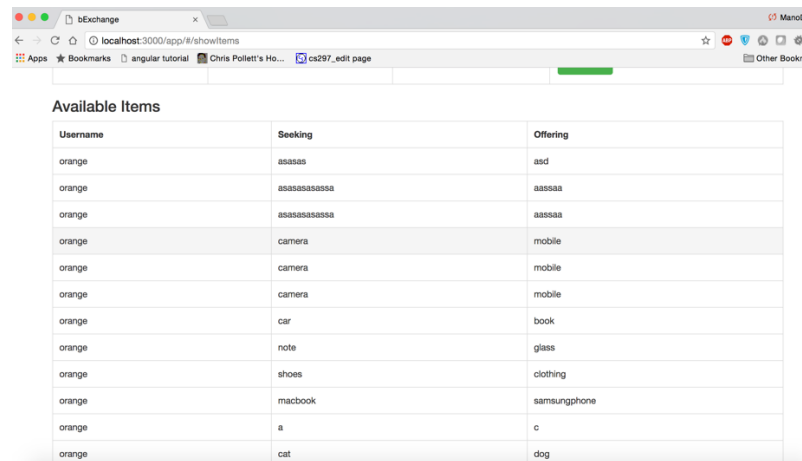
The show items module displays all the available exchanges from the exchange pool. It displays a table with items posted by the user for bartering. Followed by that it displays all other available items posted for bartering by other users. A GET request is made to the server to fetch all the necessary information.



Item No	Seeking	Offering	Actions
1	book	car	Barter
2	ipod	shoes	Barter
3	b	a	Barter
4	hat	bat	Barter
5	strawberry	rasberry	Barter
6	shoes	ipod	Barter

**Figure 4.4: bExchange systems' show items webpage**

As shown in the figure 4.4 first the item posted by the user for bartering is listed under my posted items. Followed by that a table containing the available list of items for bartering posted by other users is listed as shown in figure 4.5.

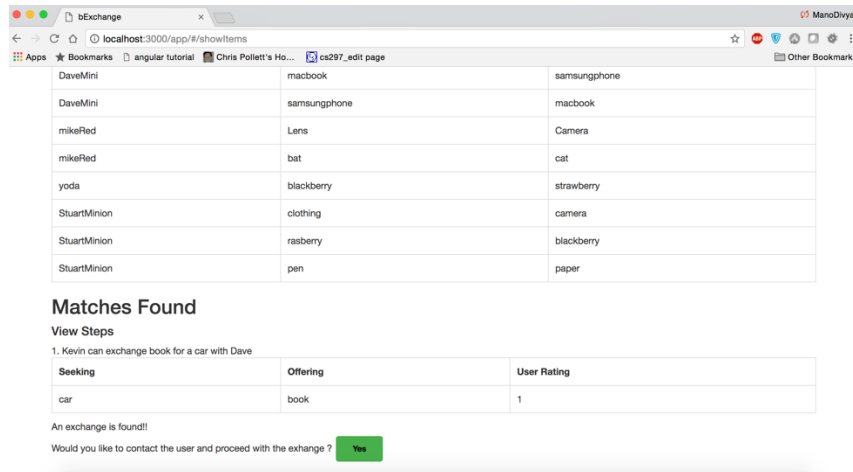


Username	Seeking	Offering
orange	asasas	asd
orange	asasasasassa	aassaa
orange	asasasasassa	aassaa
orange	camera	mobile
orange	camera	mobile
orange	camera	mobile
orange	car	book
orange	note	glass
orange	shoes	clothing
orange	macbook	samsungphone
orange	a	c
orange	cat	dog

**Figure 4.5: bExchange systems' show items webpage**

## **4.4 Barter Module**

The barter module is where the action of finding a match or an exchange comes into picture. The barter module posts the item the user is looking to barter as an JSON object to the server. The server processes the request and tries to look for a direct match in the pool of exchanges. If such a match is found the server returns the ideal match and displays it to the user with the option of proceed the trade. Incase if there are no direct match the server tries to look for a multi-bartering exchange possibilities. If such an indirect match is found the server returns the possible exchange in a JSON object. This is then displayed to the user as a list of steps on how to go with the trade. There is a user rating information displayed which lets the user to pick an exchanger when there are many users offering the same item.



**Figure 4.6: bExchange system's barter module**

Once the user clicks on the barter button for a particular exchange the system will start looking for a matching exchange from the pool. Incase if it finds one it displays the result in the matches found section with list of steps as shown in figure 4.6.

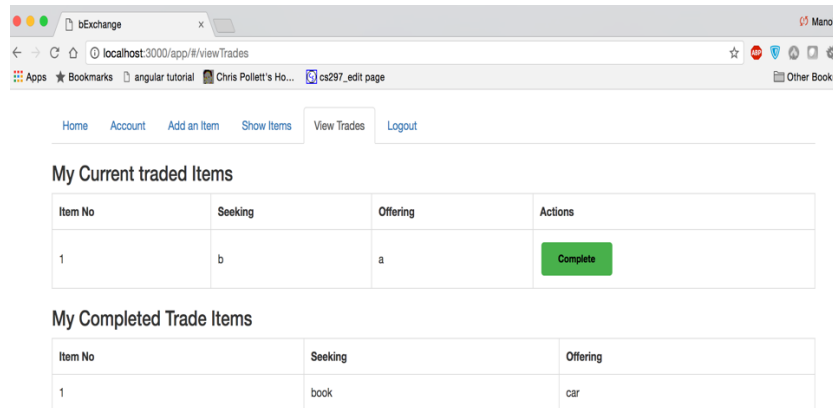
## 4.5 View Trades Module

The view trades module focuses on listing the pending trades of the user. That is if the user agrees to proceed with a particular exchange that will be the listed by this module until the exchange has been agreed by both the buyer and seller. It also shows a list of completed trades by the user. After every successful completion of the exchange within the specified time frame the user gets a user rating count incremented by one. Incase of an unsuccessful or incomplete exchange there is no change to the user rating. The user rating is calculated as

$$\text{User Rating} = \frac{\text{Total number of Transactions}}{\text{Total Number of successful transactions}}$$

Once the user decides to go ahead with the exchange and click on the proceed button then the view trades tab display that exchange under the current traded items list

as shown in figure 4.7. As soon as the user clicks on complete button the particular exchange is completed and then the exchange is listed under my completed trade items.



**Figure 4.7: bExchange system's view trades webpage**

The above described modules are the core modules/controllers of the application. While implementing these modules the following three scenarios were analyzed and made sure the algorithm works efficiently.

- (1) The algorithm must pick a minimum length exchange sequence.
- (2) The algorithm must avoid any cycles in the exchange sequence; and
- (3) The algorithm must take into consideration the exchanges that satisfy the needs of the current exchange pair.

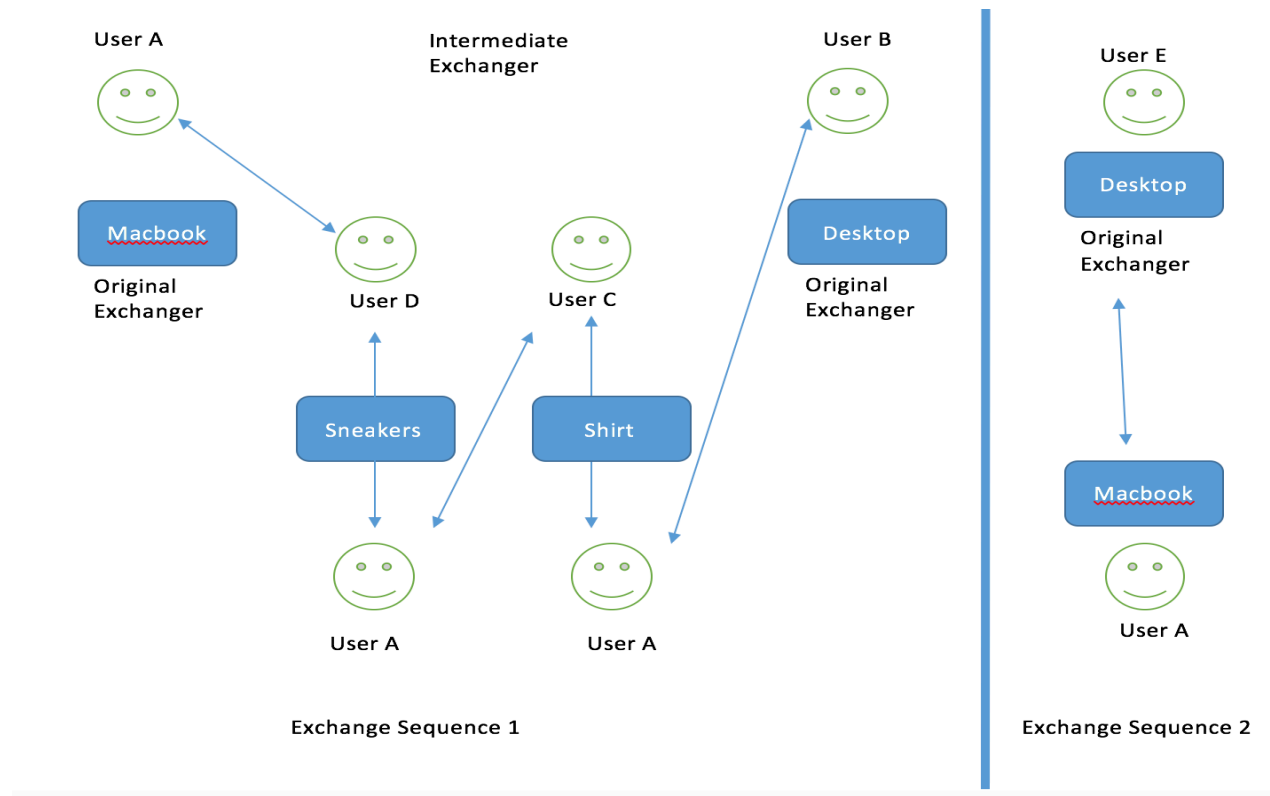
## **4.6 Analyzed Scenarios**

### **4.6.1 Minimum Length Exchange Sequence**

The algorithm must find the desired exchange with the most minimum possible length. If there are more than one way to facilitate the necessary exchange the



algorithm must be efficient enough to pick the shortest path in the exchange instead of the longest path.



**Figure 4.8: The shortest exchange sequence**

In figure 4.8 there are two exchange sequence possible. User A wants to trade MacBook for a desktop. In the exchange sequence 1 user A exchanges the MacBook for sneakers with user D and then trades the sneakers for shirt with user C and finally completes the exchange with User B for the desktop. In the other exchange sequence 2 there is a direct trade possible where user E is willing to trade the desktop for a MacBook. User A can directly trade with user E and thus complete the exchange with the minimum length. The algorithm will first find the exchange sequence two prior to exchange sequence one. It will look for the minimum length or the shortest path available.

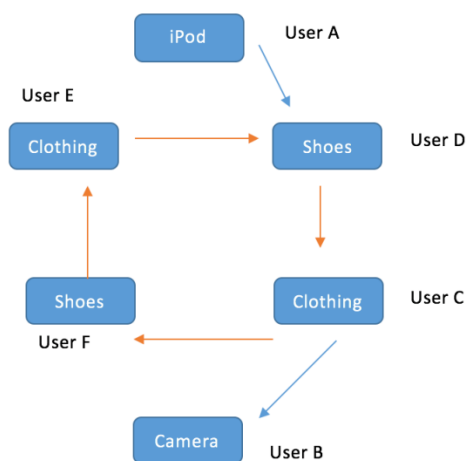
### 4.6.2 Avoid Cycles in the Exchange Sequence

Cycles must be avoided when looking for the exchange sequence. When we use brute-force methodology to find the desired multi way exchange there is a possibility for the exchanges to go into a cycle. This is because of the available exchange pool.

User	Desires	Willing to Offer
A	Camera	iPod
B	Clothes	Camera
C	Shoes	Clothes
D	iPod	Shoes
E	Shoes	Clothes
F	Clothes	Shoes

**Table 4.1: A possible exchange pool**

Consider the trading pool data in table 1, an exchange sequence can be constructed as in figure 4.9 from the available pool of exchanges in the table.

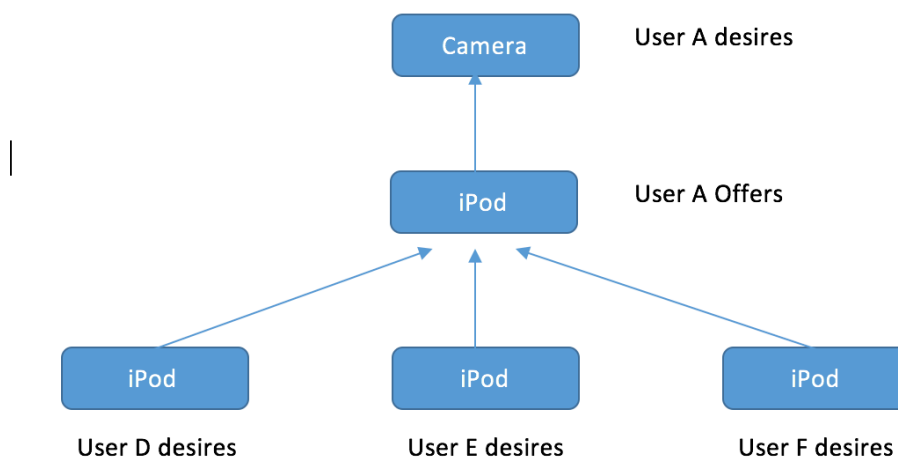


**Figure 4.9: A cycle in the exchange sequence**

The cycle in the exchange sequence is denoted in orange. The available exchanges in the pool allow for a repetitive sequences of exchange. The trade is completed by User B but because of the available exchanges in the pool the sequence will continue through user F and User E making it go through a pool. These kind of repetitive exchanges are avoided by the algorithm else the length of the exchange sequence would repeat until all the exchange pairs in the exchange pool are used.

### **4.6.3 Satisfy the needs of Current Exchange Pair**

The most efficient way to determine the desired exchange is to categorize the exchanges in the pool so that only appropriate exchanges are taken into account. In the given exchange scenario if there is no direct exchange possibility then multi way barter is needed. As the exchange sequence is created, the exchanges that are matching the current exchange will be considered.



**Figure 4.10: Exchange sequence tree**

If there are more than one user desiring for the same item, then the algorithm will consider the user that closely satisfy the needs for the current exchange pair. In the

figure 4.10 User A offers iPod that is desired by three other users. The algorithm will pick the desired user based on what they have to trade and whether that item can complete the exchange sequence.

## **CHAPTER – 5**

### **EXPERIMENTS**

This chapter discusses the various experiments and testing performed on the bExchange system. The experiments carried out can be divided into three parts:

1. Experiments conducted to unit test all the features in the barter system.
2. Experiments to stress test the system and find out how it reacts to different load of data.
3. A/B testing the system.

#### **5.1 Experiments conducted to unit test all the features in the barter system**

The experiments conducted to unit test the features in the barter system gives us an idea on how well the system behaves under given conditions. Angular's built in dependency injection makes components testing easier as you can pass in the dependencies of the components and mock them as we want.

Karma and Jasmine are tools that are used to test Angular JS applications easily. In our application we have used both karma and jasmine to write various unit test cases to test our controllers, API calls and directives.

##### **5.1.1 Testing a Controller**

Angular JS clearly separates the view layer from the logic layer, it maintains controllers very easy to test. Usually controllers scope is not global. We will have to use *angular.mock.inject* to test our controller using injection. Angular-mocks provide a module function that is used as the first step in testing. This will load the module

followed by this we pass the module into a jasmine function *beforeEach* that helps us to run the code before every test. We can then use the inject to get access to the \$controller which is the service responsible for controller's instantiation as shown in the figure 5.1 below.

```
describe('PasswordController', function() {
  beforeEach(module('app'));

  var $controller;

  beforeEach(inject(function(_$controller_){
    // The injector unwraps the underscores (_) from around the parameter names when matching
    $controller = _$controller_;
  }));

  describe('$scope.grade', function() {
    it('sets the strength to "strong" if the password length is >8 chars', function() {
      var $scope = {};
      var controller = $controller('PasswordController', { $scope: $scope });
      $scope.password = 'longerthaneightchars';
      $scope.grade();
      expect($scope.strength).toEqual('strong');
    });
  });
});
```

**Figure 5.1: Testing a Controller**

### **5.1.2 Testing an API Call**

When performing unit tests we ideally want our tests to quickly run without having any external dependencies such as JSONP or XHR requests to actual server. We will just have to verify if a particular request is sent out to the server or not.

\$http service is usually used when our Angular JS application requires data from the real server. \$http service uses \$httpBackend service to send these requests. For testing API calls or service, we can mock this \$httpBackend using dependency injection and check whether requests are fired and response are got back from the server without actually sending any kind of requests to the real server using some test data.

Two ways are used to indicate what data must be returned as responses by the mocked backend when http requests are made by the code. They are listed below:

- \$httpBackend.expect – This indicates what a http request expects

- `$httpBackend.when` – This indicates what definition is needed by the backend

Figure 5.2 contains a simple API call test for the authentication module. It uses `$httpBackend.expect` to specify what the request expectation must be. `$httpBackend.flush` is another method used to explicitly flush any kind of pending requests to the server. This conserves the asynchronous API of the backend and also allows the tests to finish synchronously.

```
it('should fail authentication', function() {  
    authRequestHandler.respond(401, '');  
  
    $httpBackend.expectGET('/login');  
    var controller = createController();  
    $httpBackend.flush();  
    expect($rootScope.status).toBe('Failed...');  
});
```

Figure 5.2: Testing an API Call

## 5.2 Experiments to stress test the system and find out how it reacts to different sets of data

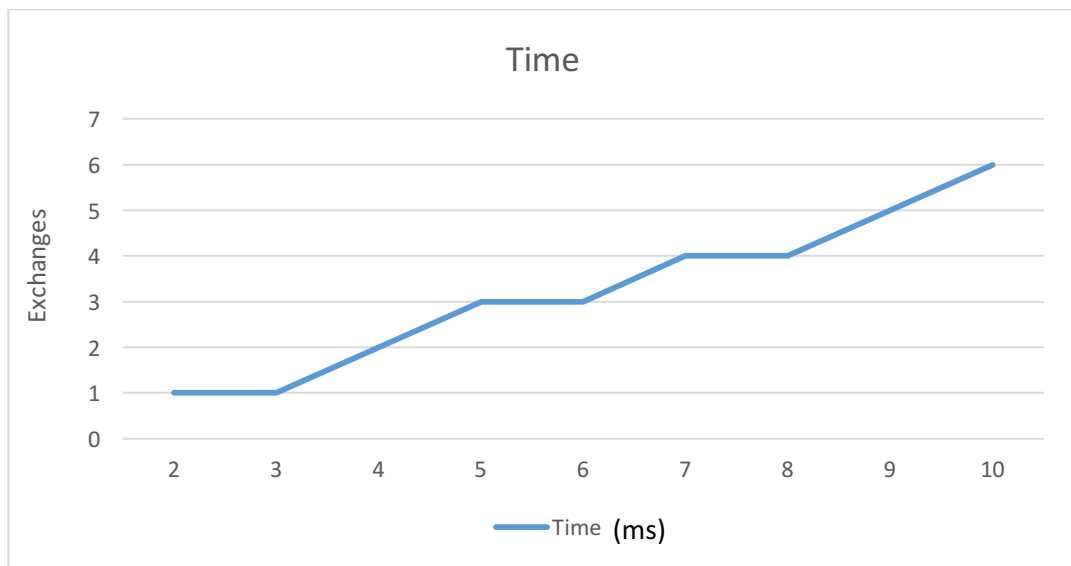
The bExchange system can perform both direct and multi- lateral bartering. In case if there is no direct exchange possible the system will look for indirect exchange. Stress testing also called torture testing is to test the system beyond operational capability and look for results. In this part of the experiment we start with 500 exchanges made available in the exchange pool and start with direct barter scenario i.e. exchanges = 2 and we go about increasing the node size thereby making the system to look for multi-barter scenarios and test the system for stability

and correctness of the result. Table 5.1 compares the time taken by number of nodes to compute a desired exchange under such a scenario.

Number of exchange	2	3	4	5	6	7	8	9	10
Time taken (ms)	1	1	2	3	3	4	4	5	6

**Table 5.1: Number of exchanges and time taken by the algorithm to compute the result**

From the table we can clearly see the system is able to compute multi barter exchanges successfully for a larger number of exchanges. A graph plotting the table is shown in figure 5.3.



**Figure 5.3: Graph plotted for time taken vs number of exchanges**



When we stress test the system with 500 exchanges in the exchange pool and look for exchanges starting with node length 2 to node length 10 the system functions as expected giving us correct results.

**Example 1:** Consider user Kevin wanted to barter a raspberry he has for a strawberry. The system originally looks for a direct barter, once it is not found it starts looking for the indirect exchange. Another user Dave is looking for a raspberry and willing to give a blackberry whereas Stuart the third user is willing to give his strawberry for a blackberry. The system finds this pattern and notifies Kevin with a list of steps for the desired exchange. Figure 5.4 is the actual result from the system. The steps show the list of exchanges needed for satisfying the original exchange.

### Matches Found

#### View Steps

1. Kevin can exchange raspberry for a blackberry with Dave
2. Kevin can exchange blackberry for a strawberry with Stuart

Seeking	Offering	User Rating
blackberry	strawberry	1
rasberry	blackberry	1

An exchange is found!!

Would you like to contact the user and proceed with the exchange ?

Yes

**Figure 5.4: Trade scenario with 3 exchanges**

**Example 2:** Consider user A who wants to trade his iPod for a camera. The system finds a possible exchange with node value 5. A can exchange his iPod for sneakers with user E then exchanges sneakers for shoes with user D. Further user A exchanges shoes for clothes with User C and then for camera with user B and completes the

desired original exchange. Figure 5.5 is the actual result given by the system. The steps show the list of exchanges needed for satisfying the original exchange.

## Matches Found

### View Steps

1. A can exchange Ipod for Sneakers with E
2. A can exchange Sneakers for Shoes with D
3. A can exchange Shoes for clothes with C
4. A can exchange clothes for a Camera with B

Seeking	Offering	User Rating
Ipod	Sneakers	1
Sneakers	Shoes	1
Shoes	Clothes	1
Clothing	Camera	1

An exchange is found!!

Would you like to contact the user and proceed with the exchange ?

Yes

**Figure 5.5: Trade Scenario with 5 exchanges**

**Example 3:** This example is an extension of the above example. The actual exchange is found at node value 10. After user A exchanges sneaker for socks with user E the exchange pattern continuous as user A exchanges socks for t-shirt with user F and then t-shirt for shirt from user G. This is followed by pant from user H and book from user I and finally iPod for book from user K. Thus user A gets the desired exchange after a long pattern. Figure 5.6 shows the actual result from the system.

## Matches Found

### View Steps

1. A can exchange Ipod for book with K
2. A can exchange book for pant with I
3. A can exchange pant for shirt with H
4. A can exchange tshirt for shirt with G
5. A can exchange tshirt for socks with F
6. A can exchange socks for Sneakers with E
7. A can exchange Sneakers for Shoes with D
8. A can exchange Shoes for clothes with C
9. A can exchange clothes for a Camera with B

Seeking	Offering	User Rating
Ipod	book	1
book	pant	1
pant	shirt	1
shirt	tshirt	1
tshirt	socks	1
socks	sneakers	1
Sneakers	Shoes	1

Shoes	Clothes	1
Clothing	Camera	1

An exchange is found!!

Would you like to contact the user and proceed with the exchange ?

Yes

**Figure 5.6: Trade Scenario with 10 exchanges**

## 5.3 A/B Testing

A/B testing which is also called as split testing, is used to compare the two versions of the web page to users and find out which one has better performance. We start the testing by designing two different variation of the same webpage and show them to users at same time and rate them. The one with better user rating and experience is the winner.

This testing was performed by comparing two different versions of the show items webpage. The difference in both the variation is that view steps. In variant B the view steps part was removed and only the table with exchange data was displayed. In the variant A both the table and the list of steps to achieve the necessary exchange was

shown. Figure 5.7 shows the variant B that was used for testing purposes. Figure 5.8 shows the variant A of the show items page.

### Matches Found

Seeking	Offering	User Rating
blackberry	strawberry	1
rasberry	blackberry	1

An exchange is found!!

Would you like to contact the user and proceed with the exchange ?

Yes

**Figure 5.7: Variant B of show items page**

### Matches Found

#### View Steps

1. Kevin can exchange raspberry for a blackberry with Dave
2. Kevin can exchange blackberry for a strawberry with Stuart

Seeking	Offering	User Rating
blackberry	strawberry	1
rasberry	blackberry	1

An exchange is found!!

Would you like to contact the user and proceed with the exchange ?

Yes

**Figure 5.8: Variant A of show items page**

The variants were shown to three different users and their responses are listed below.

**User 1:** User 1 found it difficult to understand what the results were to convey. Once the user was explained how to read the result displayed by the system the user 1 found variant B to be precise and easy to understand.

**User 2:** User 2 considered the variant A to be easy and simple to use. Variant A listed the necessary steps to achieve the desired exchange. This user found variant B little

confusing as the table alone listed only the items and no other information on the users or information on which transaction to start with.

**User 3:** User 3 found the variant B to contain little information regarding the exchange and felt the system to be less usable from user perspective. The user felt that more information displayed about the exchange to the end user the more the user will likely start using the system. The user also felt displaying the exchange information with steps explicitly does not need any explanation on how the system works as the user will be able to figure that out.

Taking all of the suggestions from the users we can conclude that although variant B displays short and precise information about the exchange variant A is more preferred by users as it offers clear steps about the exchange. Thus the result of the A/B testing is variant A and hence we go about using the variant A.

## **CHAPTER - 6**

### **CONCLUSION**

In this project, we designed an online barter exchange system similar to existing systems such as Craigslist and uExchange. Our system has the feature to perform multi-lateral or indirect bartering exchanges. The mechanism that we have designed can be described as enabling traders, through a series of trades, to consummate a trade to obtain a desired object or resource. The main advantage of our system is that it starts with the search for direct exchange for any given exchange scenario. When the system cannot find a direct exchange it starts looking for multi-bartering possibilities from the same exchange pool. If a valid exchange is found, the system displays the exchange pattern. Once the user agrees to proceed with the exchange it initiates the exchange process.

The system rates every user based on their successfully completed exchanges. The rating will be very useful when there are more than one users offering the same exchange. The system works efficiently to find the minimum length sequence for any given exchange. It also avoids any kinds of cycles in exchange sequence.

Currently, the system works for multi-barter scenarios without any quantity specified. A future improvement to the project could be to add a module that can handle quantities of items. Given any quantity of an item the system must be able to find a direct matching exchange or substantially reduce the quantity of the item and look for bartering possibilities. This reduction value can either be determined by the user or we can use systems such as auctioning systems and set a time and lowest value the user is ready to offer.

## **CHAPTER – 7**

### **REFERENCES**

- [1] Barter system (2015) Retrieved from <https://en.wikipedia.org/wiki/Barter>
- [2] Edmonds, J. Paths, trees, and flowers.1965. Canadian Journal of Mathematics, 449–467.
- [3] Sonmez, T., and Unver, M. U. 2009. Matching, allocation, and exchange of discrete resources., Boston, MA.
- [4] Halldorsson, M. M., Iwama, K., Miyazaki, S., and Yanagishi. 2007. H. Improved approximation results for the stable marriage problem. In proceedings on the ACM Transactions on Algorithms.
- [5] Abraham, D. J., Blum, A., and Sandholm, T. Clearing algorithms for barter exchange markets: Enabling nationwide kidney exchanges. In Proceedings of the 8th ACM Conference on Electronic Commerce, San Diego, CA, ACM Press, New York, NY, 2007.
- [6] Unver.2010. M. U. Dynamic kidney exchange. Review of Economic Studies, 372–414.
- [7] Vries, S. D., and Vohra, R. 2003. Combinatorial auctions: a survey. INFORMS Journal on Computing, 284–309.
- [8] Vijay Krishna .2010. Auction Theory. Academic Press.
- [9] Randy.M. Kaplan. 2011.An improved algorithm for multi-way trading for exchange and barter. Electronic Commerce Research and Applications Volume 10, Issue 1, Pages 67–74

[10] How Barter system works retrieved from

<https://www.mint.com/barter-system-history-the-past-and-present>

[11] Sebastien Mathieu. 2006. Match-Making in Bartering Scenarios. Master's thesis.

University of new brunswick, Canada