MAVROOMIE: AN END-TO-END ARCHITECTURE FOR FINDING COMPATIBLE ROOMMATES BASED ON USER PREFERENCES

by

VIJENDRA KUMAR BHOGADI

Presented to the Faculty of the Graduate School of
The University of Texas at Arlington in Partial Fulfillment
of the Requirements
for the Degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE AND ENGINEERING

THE UNIVERSITY OF TEXAS AT ARLINGTON

December 2016

Copyright © by Vijendra Kumar Bhogadi 2016

All Rights Reserved

ACKNOWLEDGEMENTS

I am extremely grateful to my Advisor, Dr.Gautam Das, for giving me the opportunity to work with him and also for his constant support and guidance throughout my Master's thesis. I feel extremely blessed to be a part of DBXLAB, where I had the opportunity to explore new horizons, which I never knew existed. I am also extremely grateful to Mr.David Levine and Dr. John H Robb for accepting to be on the thesis committee.

I sincerely appreciate the efforts of Mr.Habibur Rahman of DBXLAB for his remarkable guidance and support throughout this project. It was wonderful working with him throughout this program and I am extremely grateful for that.

I would like to thank my Parents and Sister for the moral support and patience. I would like to thank my uncle, Mr. Saibabu Mandali, and his family for their constant support and motivation without which it was impossible to take on this wonderful journey of knowledge and learning. I would like to also sincerely thank my friends for their encouragement and feedback in improving this project.

November 18, 2016

ABSTRACT

MAVROOMIE: AN END-TO-END ARCHITECTURE FOR FINDING COMPATIBLE ROOMMATES BASED ON USER PREFERENCES

Vijendra Kumar Bhogadi, M.S.

The University of Texas at Arlington, 2016

Supervising Professor: Dr. Gautam Das

Team Formation is widely studied in literature as a method for forming teams or groups under certain constraints. However, very few works address the aspect of collaboration while forming groups under certain constraints. Motivated by the collaborative team formation, we try to extend the problem of team formation to a general problem in the real world scenario of finding compatible roommates to share a place. There are numerous applications like ”roommates.com” ,”roomiematch.com” , ”Roomi”, which try to find roommates based on geographical and cost factors and ignore the important human factors which can play a substantial role in finding a potential roommate or roommates.

We introduce ”MavRoomie”, an android application for finding potential roommates by leveraging the techniques of collaborative team formation in order to provide a dedicated platform for finding suitable roommates and apartments. Given a set of users, with detailed profile information, preferences, geographical and budget constraints, our goal is to present an end-to-end system for finding a cohesive group of roommates from the perspective of both the renters and rentee. MavRoomie allows users to give their preferences and budgets which are incorporated into our algorithms in order to provide a meaningful set of roommates. The strategy followed here is similar to the Collaborative Crowdsourcing's strategy of finding a group of workers with maximized affinity and satisfying the cost and skill constraints of a task.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

LIST OF TABLES

CHAPTER 1

INTRODUCTION

"Unity is strength. . . when there is teamwork and collaboration, wonderful things can be achieved." –Mattie Stepanek

1.1 Motivation

Every semester, hundreds of international students move to UTA for pursuing quality education. Every time they are faced with the problem of searching for accomodation and roommates. As being new to the place and country they are limited with the no.of choices for accomodation. They have limited information on the location and also the type of roommates they will be sharing their accomodation. This problem is not just limited to the students but can be extended to all the people who are searching for roommates and accomodation. Hence in order to provide a solution to this problem we have come up with "MavRoomie", an end-to-end system for finding compatible roommates and accomodation.

In this application, to find the group of compatible roommates, we have looked into similar problems and thier solutions. Collaborative Team Formation is a problem that is very similar to ours, and also it has been widely studied and solutions have been formulated. There are many applications of collaborative team formation, which are producing impressive results and hence we are borrowing those methodologies and techniques from these applications.

The importance of team work has been significant in this ever changing fast paced civilization. We can see team work from all species on this planet, working towards one common vision or goal. Team work has been important throughout human civilization and its still has an important role to play in today's technologically fast paced world. Human kind has achieved so much today and will continue to achieve more accomplishments in the future by team work and collaboration. From the above description it is obvious that teams are vital to any organization and the success of these organizations depends on the effective collaboration of the teams. Good collaboration is not easy to obtain from the team and takes in a lot of factors. If collaboration within team cannot be achieved, it can be detrimental to the task and the whole organization. So what can be done in this regard? How to form a team to improve the collaboration from each individual-worker and hence to extract the gains of such a collaboration? Hence Team Formation comes into play and has been a widely researched topic in the scientific community. The importance of Team formation is even more in Crowdsourcing. Crowdsourcing is a process where a task is outsourced to a group of workers. Today crowdsourcing is used in numerous applications and the quality and success of a task depends upon the effective working of the team of workers. Let us look into some examples which will help us understanding the significance of a team and the topic of team formation.

Collaborative Editing or writing[9] is a process where a group of workers or experts work together to produce a work by collaborating and contributing individually. Any collaborative editing or writing task requires that each user or worker is well aware of the other members in the group and must agree and abide to work on their individual writing or editing modules without interfering with other workers. There are situations where the user may try to interfere with other's and thus causing a conflict. This might lead to issues like edit wars where each user may try to undo what others have done leading to less than standard quality of work which might contain misinformation or outright blunders.

Fansubbing[7, 8] is another collaborative team work where a group of fans translate a foreign language film or any foreign language television programme and create subtitles in languages other than the original. The fans who participate in fansubbing are termed fansubbers and are usually non-professionals or volunteers. Fansubbers divide the task of creating subtitles into smaller tasks and are allotted to different group of fansubbers. However similar conflicts to that of edit wars can arise in fansubbing resulting in videos with incorrect subtitles or missing ones. Hence the formation of team plays a significant role in fansubbing.

There are a many examples of collaborative crowdsourcing like online multiplayer games, where a group of users form a team and try to achieve objectives in the game collectively, citizen journalism[9] is another example where citizens or common people playing a major part in accumulating news stories and running an independent analysis of the information. Citizen Science[6] is another arena which uses the cllaborative efforts of distributed community of scientists, software engineers and intellectuals in contributing to new scientific projects with the goal of helping the public to understand the scientific process. Citizen Science Alliance is a wonderful example of Citizen Science, where experts all over the world collaborate to contribute their expertise to the project. Some of the projects by Citizen Science Alliance are Galaxy Zoo, a study of galaxies in the universe, Planet Hunters, a study of planetary data to search for unexplored planets. There are numerous examples where the quality of the work or the success of the task entirely depends on how the team members cooperate with one another.

From the above examples it is evident that Team Formation has an important role in any team task and the results may depend on the how well the team members cooperate with one another. However, from the existing work in organization studies it is understood that human factors have a strong influence on how worker performs in a task. Some of the factors could be financial gains, social pride, or work satisfaction. Existing works have proved that worker-to-worker affinity plays a major role in collaboration between members of a team[1]. Some of the factors that contribute to worker-to-worker affinity measure are simple socio-demographic attributes such as region, age, gender and psychological characteristics. There are several important scientific and theoretical contributions that try to solve this problem of team formation for a particular team task.

In this thesis we extend this problem of team formation to a more general scenario of finding roommates to share a place. Now the problem of finding roommates is a most common problem people face a few times in their lives. Normally the procedure involves either posting an add to the classifieds in newspapers, or posting on online classifieds like Craigslist or resorting to the method of friend-to-friend contact hoping to find a suitable place. Although being old methods, people still resort to these methods for finding a roommate. But in most of the cases the roommate may not be compatible and may lead to frequent personal conflicts.

However in recent times, there have been numerous web and mobile applications promising to find compatible roommates. Although, better than the traditional methods for searching roommates on classifieds, these applications face major drawbacks. Most of these roommate finding applications match a roommate to the other roommate by comparing just the cost, geographical and gender constraints. These applications fail to gather information regarding the user that can actually help find a potential match. These applications fail to help the users in understanding how a roommate is better match other than those mentioned constraints. Some of the examples are Roomi,Roomster,yoroomie, Roommates.com etc. Roomi is an Android/IOS app with a motive of finding the best matched roommates. Although it gathers data regarding the user's lifestyle choices in form of tags, it resorts to matching based on rent affordability and location only. Similar trend is followed by the others as well. There is another tier of web applications as well, which gather a lot of data regarding the user's lifestyle choices and tries to find a best possible match. However the problem is that this match is found by human evaluators who go through the user's information in order to find a match. Although this method produces better match, the process is really slow and costly. So this is not an effective solution either. In this thesis, we introduce MavRoomie an end-to-end architecture for finding compatible roommates based on user-preferences developed in Android.

MavRoomie is an Android application developed in order to provide a feasible solution to the users searching for roommates. This is an end-to-end system, developed by leveraging the techniques in collaborative team formation, dedicated for finding a cohesive group of roommates and apartments. This application makes use of user-preferences along with cost and location constraints which are incorporated into our algorithms to provide a close group of roommates from the perspective of both roommates and home-owners. This thesis dwells into various scenarios dealt during the search and try to find a solution by employing the methods which are hugely influenced from collaborative team formation problems and solutions. In the next chapters, we look into the details of how each of the challenge has been solved along with theoretical proofs. We also put an emphasis on the User Interface of the android application, which plays a major role in helping the users in their search for potential roommates.

CHAPTER 2

TECHNICAL BACKGROUND

As disscussed earlier, MavRoomie uses user-preferences in calculating the affinity or similarity between one another. In this chapter we will look into all the concepts that we have utillized in solving our problem of finding compatible roommates.

2.1 Distance Measures

In our application we calculate the distances or the similarity between users in order to find a group of closely matched users. We will look into different methods for calculating the distances between users. Before we get into the distance calculation methods, we have to look into the different attribute types.

An attribute is a data field, where it is representing a characteristic or a feature of a data object. For example in our application, some of the attributes that we use are age, gender, location, cooking habbits, sleep cycles and a few more. There are different types of attributes based on what kind of data that can be taken by the attribute. Let us discuss them in detail.

2.1.1 Nominal Attributes

Nominal Attributes[4] are symbols or in particular name of things. Each nominal attribute might take a set of possible values or categories. These values donot have any order or quantitaive value relative to other values or categories for that attribute. Examples include gender, `marital_status` where the attribute gender can take the values as Male, Female, or Other. Whereas `marital_status` can take the values as married, single, divorced. Here nominal attributes can take integers as values but these numbers donot signify any quantitative value but are only for representative purposes. For example `ID`, `room_no`, `house_no` can take integers but cannot be used quantitatively as substracting or adding ID's donot make any sense. For the same reason, these nominal attributes cannot have mean, median values but however can be used for calculating the frequency of a certain value in a particular dataset which in statistical terms is called a mode.

2.1.2 Binary Attributes

A nominal attribute with only two possible values in or categories is a binary attribute[4]. Examples include for attribute smoking are yes or no. These attributes normally assign value 0 if a value is absent or else 1 is assigned if its present. The values true or false are also used to describe the presence of those values. The binary attribute is further divided into two categories based on the weight assigned to each possible value. Assymetric binary attribues is the one where one value carries more weight than the other. Symmetric binary attributes carry equal weights on each of them.

2.1.3 Ordinal Attributes

An ordinal attribute[4] is a categorical attribute just like the nominal attributes, however they have a relative meaningful order. Examples include grades in course which can take any value from the possible set of values A,B,C,D. Here we can see that A is better than B and B is better than C and soon. Attributes explaining organizational hierarchy are also ordinal in the sense that Chairman is superior to the CEO, and CEO is superior to Managers and soon. These attributes may take up numerical values but only as representative symbols and cannot be used quantitatively. Hence mode and median can be calculated for these attributes.

2.1.4 Numerical Attributes

As the name suggests, numerical attributes[4] take on integer or real values and can be used for quantitative calculations. Examples include count attributes like experience, no.of.words in a essay, height , weight , cost etc. These numerical attributes are further divided into two more types called interval-scaled and ratio-scaled. Interval scaled attributes are on a set of values which are equally distributed on a certain scale of units.Examples include distance measurements in miles or dates in a calender. Ratio scaled attributes are the attributes that come with a zero-point. Some examples include count attributes like experience,cost etc.

There are even more ways to order the attributes other than the attribute types discussed above. The most common would be Discrete versus Continuous Attributes where Discrete Attributes take on discrete values for example as smoker which take discrete values of yes or no. Whereas Continuous attributes are numerical attributes that take continuous real numbers.

2.2 Distance Measures for Different Attribute types

2.2.1 Distance Measures for Nominal Attributes

The distance between two objects i and j is computed based on the ratio of mismatches[4]:

$$d(i,j) = \frac{p-m}{p}, \tag{2.1}$$

where 'm' is the no.of matches, p is the total number of features or variable characterizing the objects i and j. To increase the importance of matches, we can assign more wieghts to matches normally when there are large no.of values or states. Similarity can also be computed from the above equation as follows

$$sim(i,j) = 1 - d(i,j) = \frac{m}{p} \tag{2.2}$$

2.2.2 Distance Measures for Binary Attributes

To calculate the distance between two objects i and j with binary attributes we will use the most popular distance metric for binary attributes, Jaccard Index. The equation to find the jaccard index[4] is as follows:

$$J = \frac{M_{11}}{M_{01} + M_{10} + M_{11}} \tag{2.3}$$

here M11 represents the total attributes that have 1(match), M10 and M01 are the attributes that don't match.

2.2.3 Distance Measures for Numerical Attributes

There are many measures[4] for calculating numeric attributes like Manhattan, Euclidean and Minkowski distances. We will look into Euclidean distances here $i = x_{i1}, x_{i2}, ..., x_{in}$ and $j = x_{j1}, x_{j2}, ..., x_{jn}$ will be given by the equation,

$$d(i,j) = \sqrt{(x_{i1} - x_{j1})^2 + (x_{i2} - x_{j2})^2 + ...} \tag{2.4}$$

2.2.4 Distance Measures for Ordinal Attributes

As we know that Ordinal attributes[4] are categorical attributes with an order. So to calculate the dissimilarity between these attributes we can organize these categories into ranks or buckets of values.So if N can be the number of possible categories then there would be ranks from 1.....N for a particular attribute. Then we replace each value with a value in [0.0,1.0] so that each attribute has equal weight. The following equation gives the necessary numerical value for an attribute

$$z_{if} = \frac{r_{if} - 1}{M_f - 1} \tag{2.5}$$

where rif is the rank of that value for that attribute. Now once we have these values of z_{if} for all the attributes we could employ any of the metrics of numerical distance measure to calculate the dissimilarity between i and f.

2.2.5 Distance Measures for Mixed Attributes Types

For calculating the dissimilarity measure for mixed attribute[4] type we calculate the dissimilarity for each individual attribute by using the dissimalrity measures discussed above and then sum them up and normalize them to get a value between [0.0,1.0] This is given by the following equation

$$d(i, j) = \frac{1}{p} \sum_{i=1}^{p} d_{ij}^{(f)} \tag{2.6}$$

2.2.6 Triangle Inequality

Traingle Inequality is a theorem that states that for any triangle , the sum of the distances of any two edges must be greater than or equal to the third side.

$$d(i, j) \leq d(i, h) + d(h, j) \tag{2.7}$$

Most of the metrics that we discuuse above satisfy this inequality however binary and nominal attributes might sometimes violate this. The measure for binary attributes that satisfies this inequality is the Jaccard's Index and we make use of this Inverse Jaccard Index in our computations for calculating the distances or affinity between any two users.

2.3 Different Class of Problems

2.3.1 P class of Problems

The P class[5] consists of all the problems that can be solved in polynomial time. More precisely these problems can be solved in time $O(n^K)$ for some constant k, where n is the size of the input to the problem. Examples of algorithms that can be solved in polynomial time are Djikstra Shortest path, Quicksort, Mergesort, Binary search tree and soon and belong to the P class.

2.3.2 NP Class of Problems

The class NP[5] consists of those problems that are verifiable in polynomial time. Verification involves a certificate of a solution, and the verification of that certificate's correctness in time polynomial in the size of the input to the problem.Examples of problems that belong to this class are Longest simple paths, Hamiltonian cycle problem.

2.3.3 NP Complete Problems

NP-Complete[5] problems are the hardest problems in NP. This can be defined as follows: A language L is NP-Complete if

1. L belongs to NP and
2. $L' \leq L$ for every L' belongs to NP.

If a language satisfies condition 2 but not condition 1, we say that L as NP-Hard. Examples of NP-Complete problems are Circuit Satisfiablity, 3-CNF satisfiability, Clique Problem.

CHAPTER 3

DATA MODEL

In this chapter, we will discuss in depth about the data model that we have used in MavRoomie to solve the problem of finding roommates. Regardless of different possible scenarios or usecases, we use mainly the below mentioned data models.

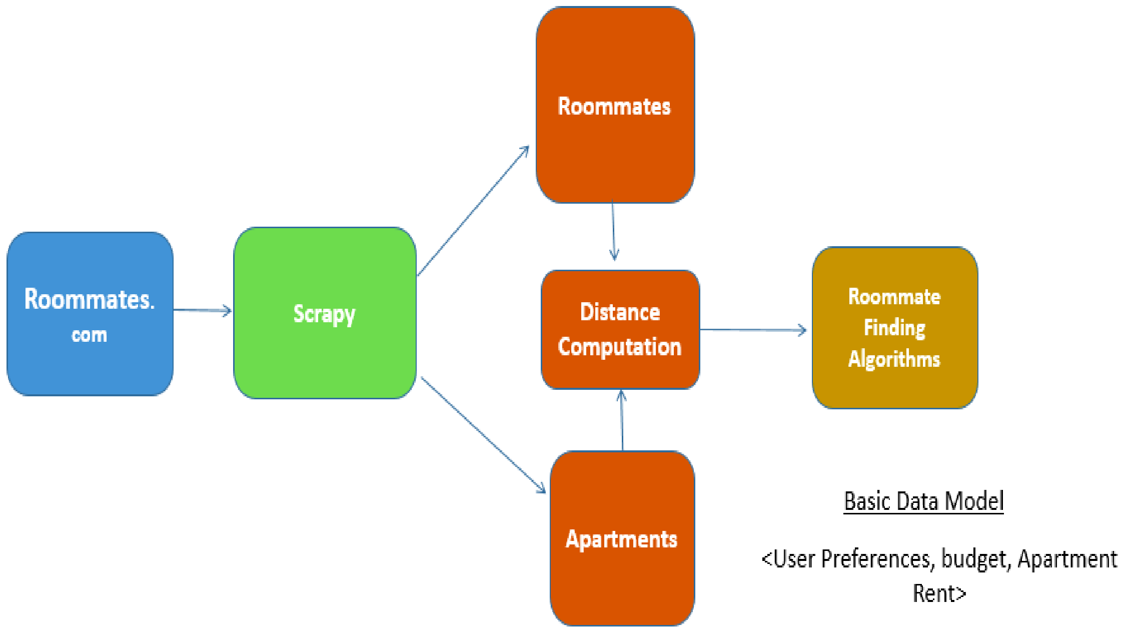


Figure 3.1: Fitting data from Roommates.com to MavRoomie

3.1 User model

We use a set of attributes from our user model for distance or similarity calculations.

Table 3.1: Attributes in User Model

userid	username	Name	email	age	gender	cooking	food preferences	cleanliness	smoking	alcohol	noise	pets	sleep patterns
22	johnd22	John D	`john@gmail.com`	34	Male	5	Vegan	Very Clean	No	No	No	Yes	Early bird
31	linda2275	Linda P Krauss	`linda2277@gmail.com`	27	Female	0	Non-Vegetarian	Very Clean	No	No	No	Yes	Night Owl

3.2 Apartment model

The apartment model is related to the User model such that each apartment is related to one user while a user can be related to many apartments. In the sense, a user can post multiple apartments and search for roommates for each of the apartments. That is each apartment has a one-to-one relationship with the user while a user may have a one-to-many relation.

Table 3.2: Attributes in Apartment Model

username	age range	gender	rent	utilities	vacancy	rooms	movin-date	duration	placetype	location	occupant	amenities
jackre	20-30	Male or Female	500	50	3	4	12-12-2015	More than a Year	Apartment	812 Creek Rd, Austin TX 78703	No	Yes

3.3 Roommate Preferences model

The attributes in this model help us in narrowing down our search list by eliminating the profiles of the users from the querylist based on the hard constraints. Here each Roommate has one associated user profile wheras each user can also have only one roommate preferences instance i.e. the relationship is one-to-one.

Table 3.3: Attributes in Roommate preferences Model

username	age range	gender preference	max rent	movin-date	duration	location
jackre	20-30	Male or Female	500	1-11-2015	More than a Year	812 Creek Rd, Austin TX 78703

To add significance to this thesis we needed to gather real world data of the users searching for roommates and apartments. Although there is a lot of rental and real estate data in the web, the users data with preferences and profile information was extremely rare. The only roommate finding website that comes close to our data model is the data from Roommates.com. Roommates.com is a web application that helps in finding roommates based on location and cost preferences. Below is a table explaing the models that are availble in Roommates.com

3.4 Home Owners looking for Roommates

Table 3.4: Some Attributes in Roommates.com Dataset of Homeowners

payment	term available	Availablility	features	building	features more	location
600	Month to month	Available Now	Electricity, Gas, Water, Trash pickup Private bedroom	House, 3 Bedroom, 2 Bath	Air conditioning Cable	Fort Worth, TX, 76137

3.5 Roommates looking for other Roommates and Apartments

Table 3.5: Some Attributes in Roommates model of Roommates.com

username	age range	gender preference	max rent	movin-date	duration	location
jackre	20-30	Male or Female	500	1-11-2015	More than a Year	Austin TX

As we can see that the data here doesnot have numerous attributes regarding the users preferences however these attributes are sufficient in our calculations of similarity or distance.

CHAPTER 4

Methodology

In this chapter we will discuss in length about how we solve our problem of finding roommates based on the strategies similar to the collaborative crowdsourcing algorithms of finding a group of workers with maximum inter-worker affinity or minimizing the distance between the members in the group[1]. These collaborative crowdsourcing algorithms for forming teams take into consideration both the characteristics of the team and the characteristics of the individual workers of that group. The worker-to-worker affinity is the comfort level between workers in a group who are collaborating on the same task. It has been accepted that the teams with high affinity between their members tend to be more successful when compared to the teams with low affinity between the workers.Let us look into various scenarios of our problem of finding roommates or apartments and how we employed the strategies and algorithms used in team formation for a collaborative crowdsourcing task.

The collaborative crowdsourcing algorithms[1, 2, 3] take into consideration the factors like the skill and cost constraints. Here the skills are used to calculate the affinity between each of the workers whereas cost is the compensation or reward for the user for completing the task. Each collaborative task has a skill and cost requirements that are to be satisfied by the team of workers. Now similarly each MavRoomie user has personal preferences or lifestyle choices along with cost contraints. Now our goal is to suggest a group of people who could be potential roommates for a user searching for roommates by satisfying the cost or the budget constraints. However before we see how we solved this problem, we would need to understand the different scenarios of this problem.

4.1 Home Owner searching for a roommate or roommates:

4.1.1 Home owner is looking for roommates to share his apartment with him:(Home Owner is involved)

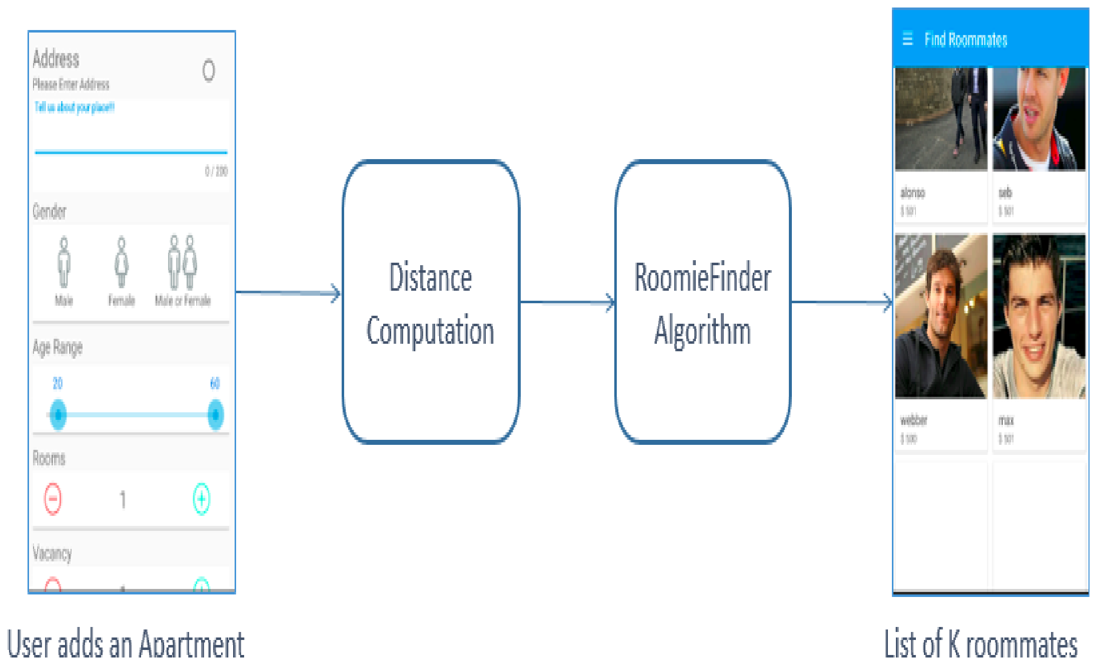
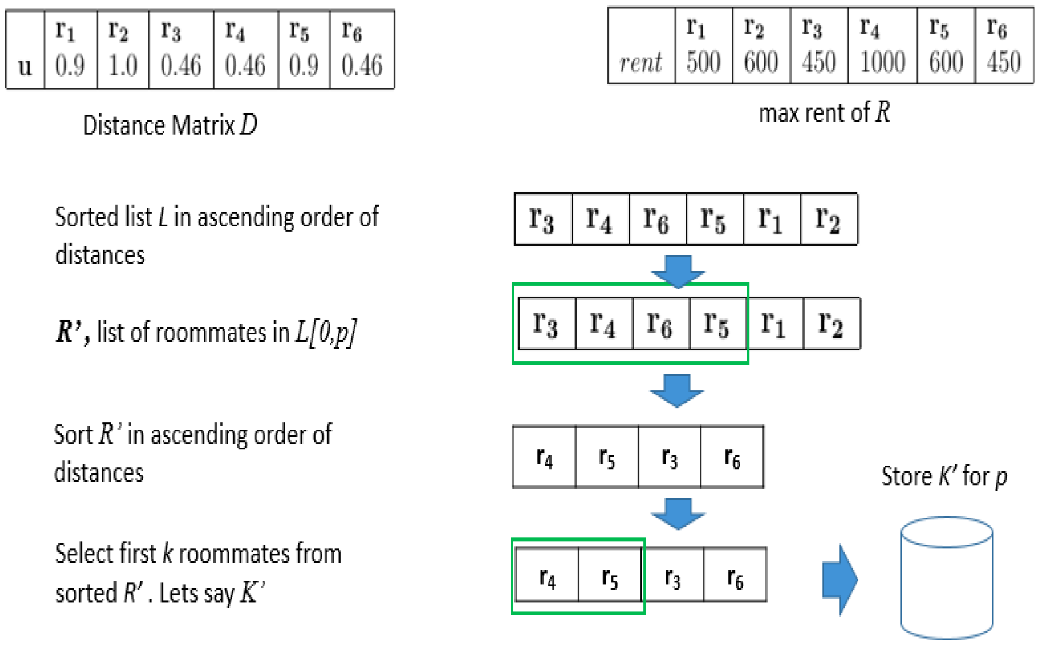


Figure 4.1: Workflow for user u finding roommates to share his place

Let us say u is the user with a place and is searching for k roommates to fill his place. Generally he would like to share his place with a group of people whose characteristics are similar to his in the sense assuming that the affinity is maximized and who can afford to contribute towards the rent of the apartment. Let us consider roommates $R = r_1, r_2, r_3, ..., r_n$. Each r_i has m attributes that we use to calculate the distances between each of them. Then using the distance measures discussed in the previous chapter we find the distance from u and R. Let us say the d is the distance matrix and it is of the dimensions 1 X n. In the next step we sort the d in ascending order and consider all the r_i 's which fall within the $L/2$th element of the sorted values, where L is the length of the matrix. Now select the top k roomies from this subset and store them. Now this k is an initial solution and is stored and then the algorithm again considers the elements from $[0, L/4]$ of the sorted distance matrix d,$sorted\_d$. Now this procedure continues until the algorithm cannot produce a result with a smaller partition. Then the recently stored result is returned. If during the first run if the algorithm is unable to find a solution in $L/2$ elements, then the search space is increased by $L/4$ and the algorithm is run again. This search space is incremented until a valid solution is found else the algorithm fails to return a result. Below is the psuedocode of this algorithm. This is a 2-approximate solution for finding k roommates, as the distances here follow traingle inequality.



	r_1	r_2	r_3	r_4	r_5	r_6
u	0.9	1.0	0.46	0.46	0.9	0.46

	r_1	r_2	r_3	r_4	r_5	r_6
rent	500	600	450	1000	600	450

Distance Matrix *D*
max rent of *R*
Sorted list *L* in ascending order of distances
r_3 r_4 r_6 r_5 r_1 r_2
R'*, list of roommates in *L[0,p]
r_3 r_4 r_6 r_5 r_1 r_2
Sort *R'* in ascending order of distances
r_4 r_5 r_3 r_6
Store *K'* for *p*
Select first *k* roommates from sorted *R'*. Lets say *K'*
r_4 r_5 r_3 r_6

Figure 4.2: Example for User u to Roommates R

Algorithm 1 Algorithm to find roommates respective to the user **ApproxRoommateFinder()**

```
1: Require: Distance matrix of the roommates to the user $u$ to $R$, $d$.
2: Sort $d$ based on distances in ascending order.We get $sorted\_d$ with length $L$.
3: Let partition size,$p = L/2$ and consider the first part of $sorted\_d$ of size $p$ i.e.$sorted\_d[0, p]$.
4: repeat roommates in this part have distance$\leq sorted\_d[p]$.
5:         sort roommates based on rent affordability in descending order, $R'$.
6:         Select a subset of size $k$, $R''$.
7:         Store the $R''$ for $L/2$th part.
8:         decrease $p$=$l/4$ in binary search manner
9: until the search is complete
10: return $R''$ with the smallest $p$
```

4.1.2 Homeowner wants to rent his place to a group(Home owner not involved)

Let us say u is the user looking for k roommates to fill his apartment from a set of $R = r_1, r_2, r_3, ...r_n$ roommates. Before we can start the algorithm we have to find the pairwise distance matrix for each roomie in R, which will be a nXn matrix. We also have a list of unique sorted distances in ascending order, L. In the first step of *ApprxGrp*[2], we perform binary search on this list L and select the initial value of α. Now the *ApprxGrp* calls the subroutine *GrpDia*(α, C). Inside *GrpDia*, the subroutine for each r_i builds a star graph around r_i such that there is no $edge > \alpha$. This subroutine selects a first k of roommates from R', such that they satisy the cost constraint(rent of the apartment). Now this result R'' is stored along with the α. The algorithm continuous to find a result R'' with smallest α possible. If no further solution is found for a smaller α, then the recently stored result is returned.

Algorithm 2 Approximation Algorithm **ApprxGrp()**

```
1: Require: $R$,prefercences of $R$ and cost constraints C.
2: List $L$ contains all unique distance values in increasing order
3: repeat
4: Perform binary search over $L$ For a given distance $\alpha$ , $R' = GrpDia(\alpha, C)$
5: if $R' \neq \emptyset$ then Store the group $R'$with diameter$d \leq 2\alpha$
6: end if
7: until the search is complete
8: return $R'$ with the smallest $d$
```

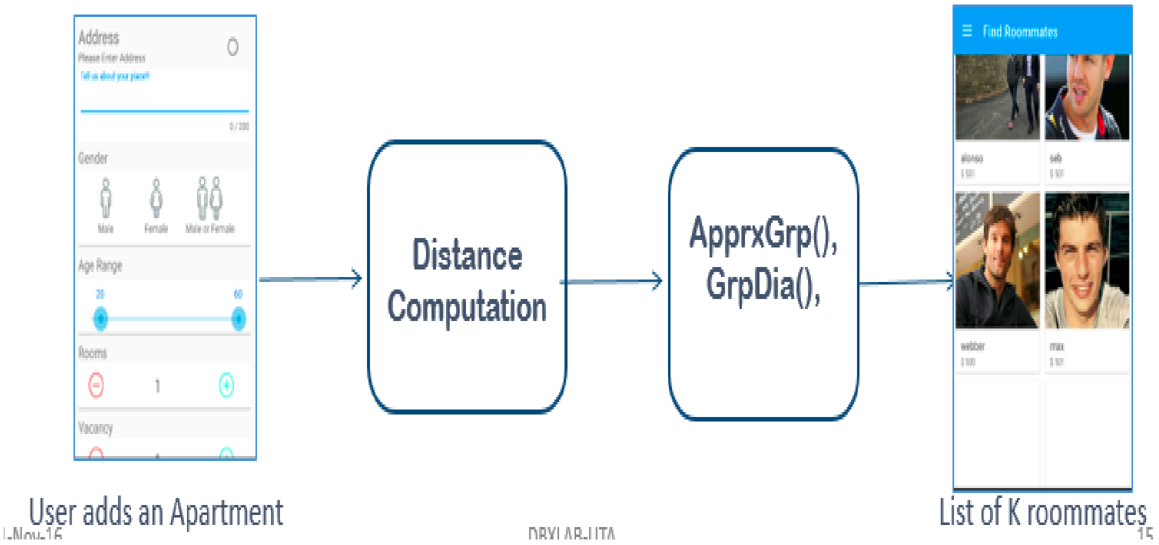
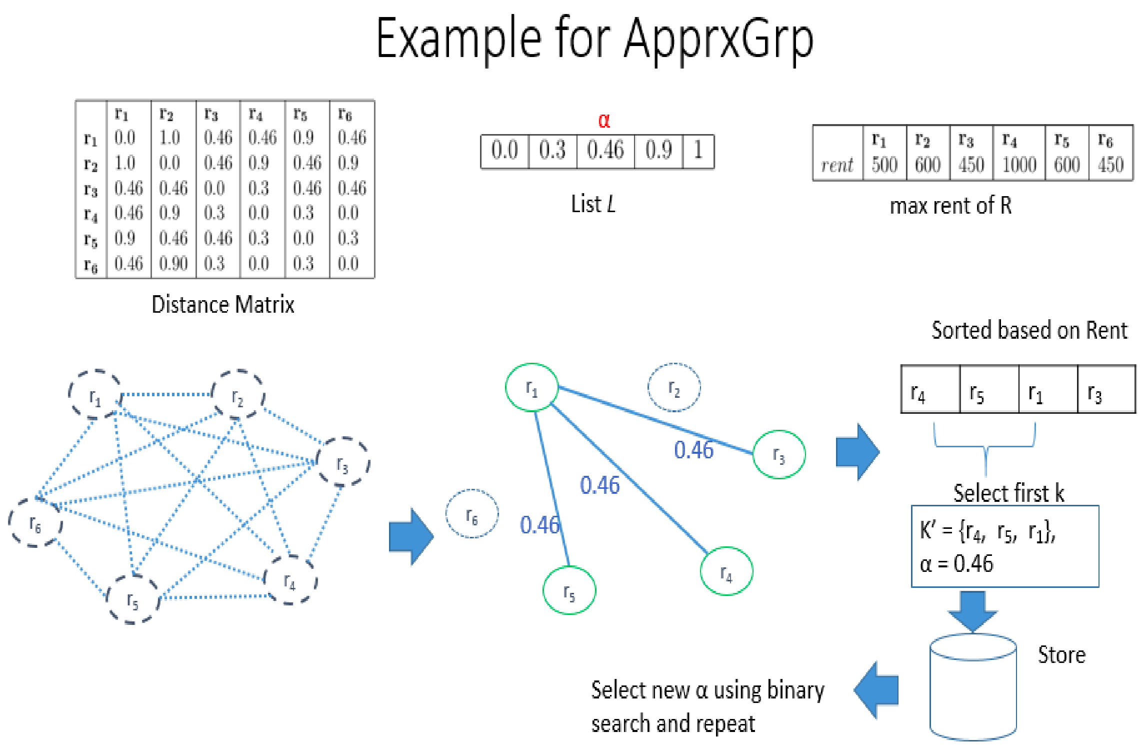


Figure 4.3: Workflow for ApprxGrp() and GrpDia()

Algorithm 3 Subroutine **GrpDia()**

```
1: Require: Distance matrix of the roommates set $R$, distance $\alpha$ and cost constraint C.
2: repeat for each roommate $r$
3: form a star graph centered at $r$, such that for each edge $r$,$r_j$, $dist(r, r_j) \leq \alpha$. Let $R'$ be the set of roommates in the star graph.
4: sort $R'$ in descending order of rent affordability.
5: select first 'k' from the sorted list, $R'$.
6: if $R'' \neq \emptyset$ then
7:     return $R''$
8: end if until all n roommates have been fully exhausted return $R'$ =$\emptyset$
```



	r_1	r_2	r_3	r_4	r_5	r_6
r_1	0.0	1.0	0.46	0.46	0.9	0.46
r_2	1.0	0.0	0.46	0.9	0.46	0.9
r_3	0.46	0.46	0.0	0.3	0.46	0.46
r_4	0.46	0.9	0.3	0.0	0.3	0.0
r_5	0.9	0.46	0.46	0.3	0.0	0.3
r_6	0.46	0.90	0.3	0.0	0.3	0.0

0.0	0.3	0.46	0.9	1

	r_1	r_2	r_3	r_4	r_5	r_6
rent	500	600	450	1000	600	450

r_4	r_5	r_1	r_3

Figure 4.4: Example for ApprxGrp() and GrpDia()

4.2 Roommates looking for other roommates to take a new apartment

This problem is similar to the problem of section 4.1. Except that a roomie is looking for other roomies. Let us say r is the user and is searching for k roommates to take a new apartment or a place. Generally he would like to share his place with a group of people whose characteristics are similar to his and who can afford to contribute towards the rent of the apartment. Let us consider roommates R - r = $r_1, r_2, r_3, ..., r_m$. Let us call this set as R_m Each r_i has n attributes that we use to calculate the pairwise distances between each of them. Then using the distance measures discussed in the previous chapter we find the distance from r and R_m. Let us say that d is the distance matrix and it is a 1 X n matrix. In the next step we sort the d in the ascending order and let us say that this sorted distances matrix is *sorted_d* and consider all the r_i's which fall within the $L/2$th element of the sorted values. Now select the k roomies from this subset and store them. Now this k is an initial solution and is stored and then the algorithm again considers the elements from $[0, L/4]$ of the *sorted_d*. Now this procedure continues until the algorithm fails to produce a result. Then the recently stored result is returned. If during the first run if the algorithm is unable to find a solution in $L/2$ elements, then the search space is increased by $L/4$ and the algorithm is run again. This search space is incremented until a valid solution is found else the algorithm fails to return a result. Below is the psuedocode of this algorithm. This is a 2-Approximate solution for finding k roommates.

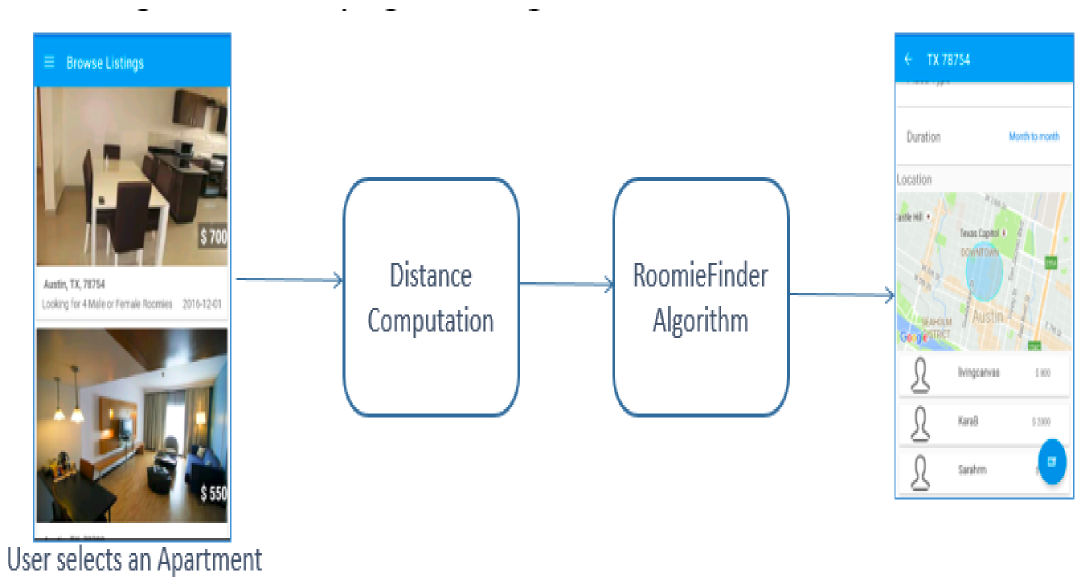


Figure 4.5: Workflow for roommate r to Roommates R

Algorithm 4 Algorithm to find roommates respective to the roommate **Roommate-toRoommateFinder()**

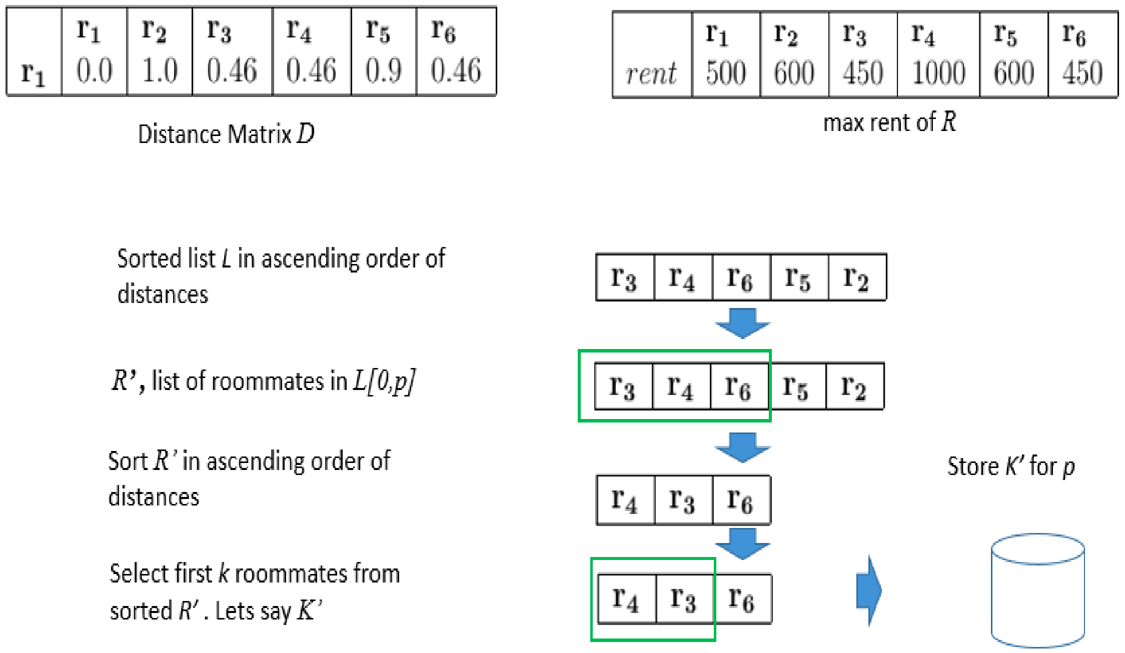
```
1:  Require: Distance matrix of the roommates to the roommate $r_1$ to $R - \{r_1\}$, $d$.
2:  Sort $d$ based on distances in ascending order.We get $sorted\_d$ with length $L$.
3:  Let partition size,$p = L/2$ and consider the first part of $sorted\_d$ of size $p$.
4:  repeat roommates in this part have distance$\leq sorted\_d[p]$.
5:          sort roommates based on rent affordability in descending order, $R'$.
6:          Select a subset of size $k$, $R''$.
7:          Store the $R''$ for $L/2$th part.
8:          decrease $p$=$l/4$ in binary search fashion.
9:  until the search is complete
10: return $R''$ with the smallest $p$
```

	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$	$\mathbf{r_5}$	$\mathbf{r_6}$
$\mathbf{r_1}$	0.0	1.0	0.46	0.46	0.9	0.46
$\mathbf{r_2}$	1.0	0.0	0.46	0.9	0.46	0.9
$\mathbf{r_3}$	0.46	0.46	0.0	0.3	0.46	0.46
$\mathbf{r_4}$	0.46	0.9	0.3	0.0	0.3	0.0
$\mathbf{r_5}$	0.9	0.46	0.46	0.3	0.0	0.3
$\mathbf{r_6}$	0.46	0.90	0.3	0.0	0.3	0.0

Table 4.1: Roommates Pairwise Distance Matrix

	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$	$\mathbf{r_5}$	$\mathbf{r_6}$
$\mathbf{u}$	0.9	1.0	0.46	0.46	0.9	0.46

Table 4.2: Roommates to user u Pairwise Distance Matrix



	r_1	r_2	r_3	r_4	r_5	r_6
r_1	0.0	1.0	0.46	0.46	0.9	0.46

Distance Matrix *D*

	r_1	r_2	r_3	r_4	r_5	r_6
rent	500	600	450	1000	600	450

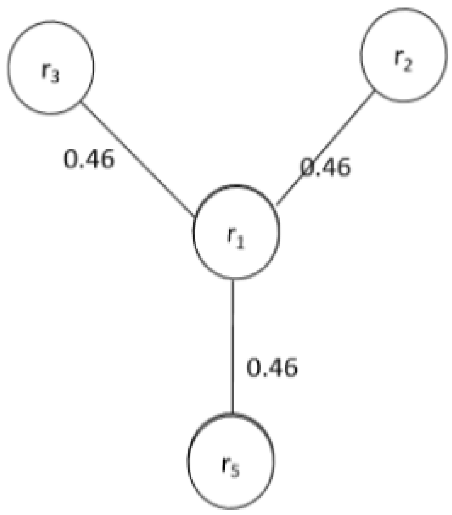
max rent of *R*

Sorted list *L* in ascending order of distances
r_3 r_4 r_6 r_5 r_2
R', list of roommates in *L[0,p]*
r_3 r_4 r_6 r_5 r_2
Sort *R'* in ascending order of distances
r_4 r_3 r_6
Select first *k* roommates from sorted *R'*. Lets say *K'*
r_4 r_3 r_6
Store *K'* for *p*

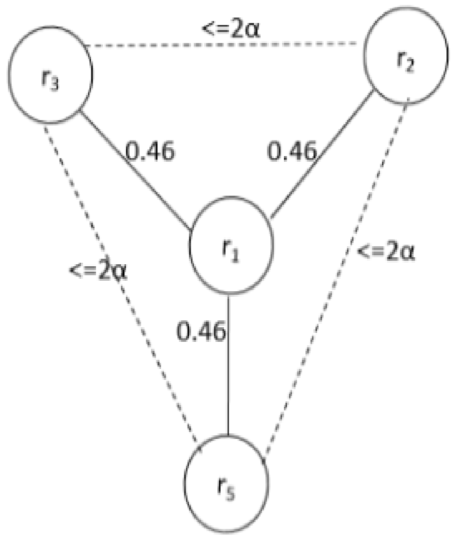
Figure 4.6: Example for roomate r to roommates R

	$\mathbf{r_1}$	$\mathbf{r_2}$	$\mathbf{r_3}$	$\mathbf{r_4}$	$\mathbf{r_5}$	$\mathbf{r_6}$
rent	500	600	450	1000	600	450

Table 4.3: Roommates Max budget details



(a) Star Graph around r_1



(b) Complete Graph around r_1

Figure 4.7: Proof for 2-Approximation

4.3 Proof for 2-Approximation Factor

We have seen that the ApprxGrp() algorithm has a factor of 2-Approximaton in the earlier sections. Now we will prove how it is a 2-Approximate approach.Algorithm ApprxGrp[3] as we know works as described.In step 1 it sorts the distance values in ascending fashion to create a list L and performs a binary search. It picks a value α and call $GrpDia(\alpha)$. For e.g consider the table 4.3 and if we perform a binary search over it the first value of α will be 0.46. Now the star graph for $\alpha = 0.46$ will be as shown in the figure 4.7b. If this graph is completed by adding the missing edges then we get a graph as in the figure 4.7. From this graph we can see that no edge will be $\geq 2\alpha$ according to triangle inequality,as we mentioned earlier that the distances satisy the triangle inequality.Therfore any of the roommates returned will be atmost 2-times worse than the optimal solution.

CHAPTER 5

Architecture

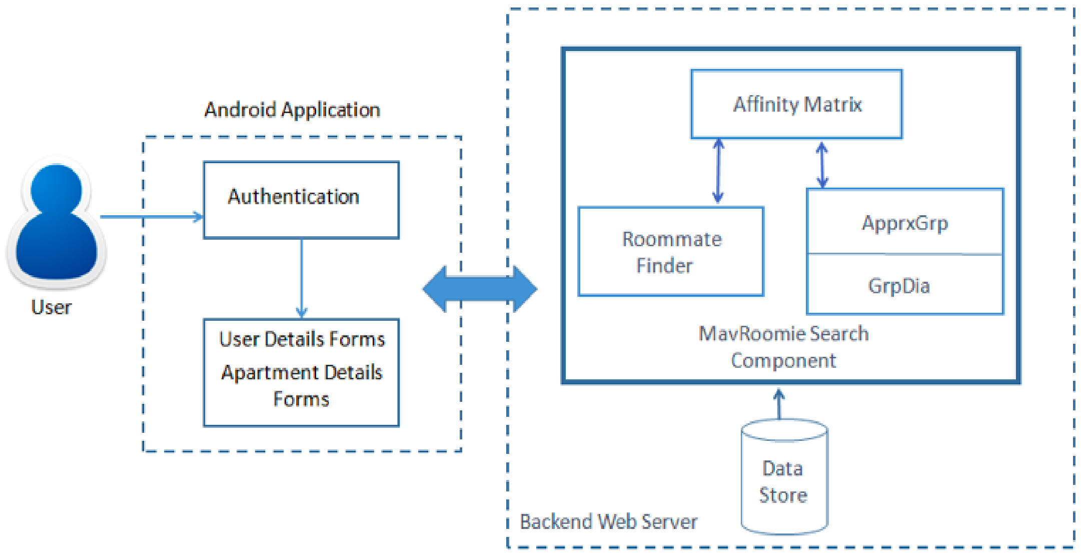


Figure 5.1: System architecture

MavRoomie is a complete system for enabling users to find potential roommates or apartments by satisfying the budget and geographical constraints. This system is built for enabling users to enter their information like their preferences, cost and location requirements and then presenting a number of search results which satisfy the hard or fixed constraints and also which are closer in terms of affinity. The architecture of MavRoomie is as shown in the figure 5.1. It consists of an Android Application in the front end, which lets the user to input various information using a responsive User Interface.

MavRoomie lets the user to maintain a profile which has all the user information, provides the user with means for creating new Apartments with vacancy, and also lets them store images of themselves or their apartments. This application also lets the user to contact other users through inbuilt chat messenger. The back-end server has a component which handles all the requests from the users and is also responsible for sending responses. This component based on the user request must be able to direct it to appropriate algorithmic component. Here there are two algorithmic components which handles the requests based on whether the user requested for roommates or apartments. The component 2-Approx Roommate Finder handles the requests from the user looking to fill vacancy in his apartment(Home owner). Whereas the 2-Approx Apartment Finder returns the respose to the request of a roommate who is looking for a place to live along with the potential list of roommates for that place. There is a data store for storing the information of the users and the apartments. The component for pairwise distance calculation is used by both of the algorithmic components.

The rest of the chapter explains in detail the technological components and frameworks used to build this complete system of MavRoomie. The following image 5.2 gives the complete picture of the complete system design.

The biggest challenge in this project was the system design needed to achieve the complete Android application. Our goal was to develop an application which was responsive, user friendly and comes with the necessary and required functionalities which help the user in searching for roommates or apartments.

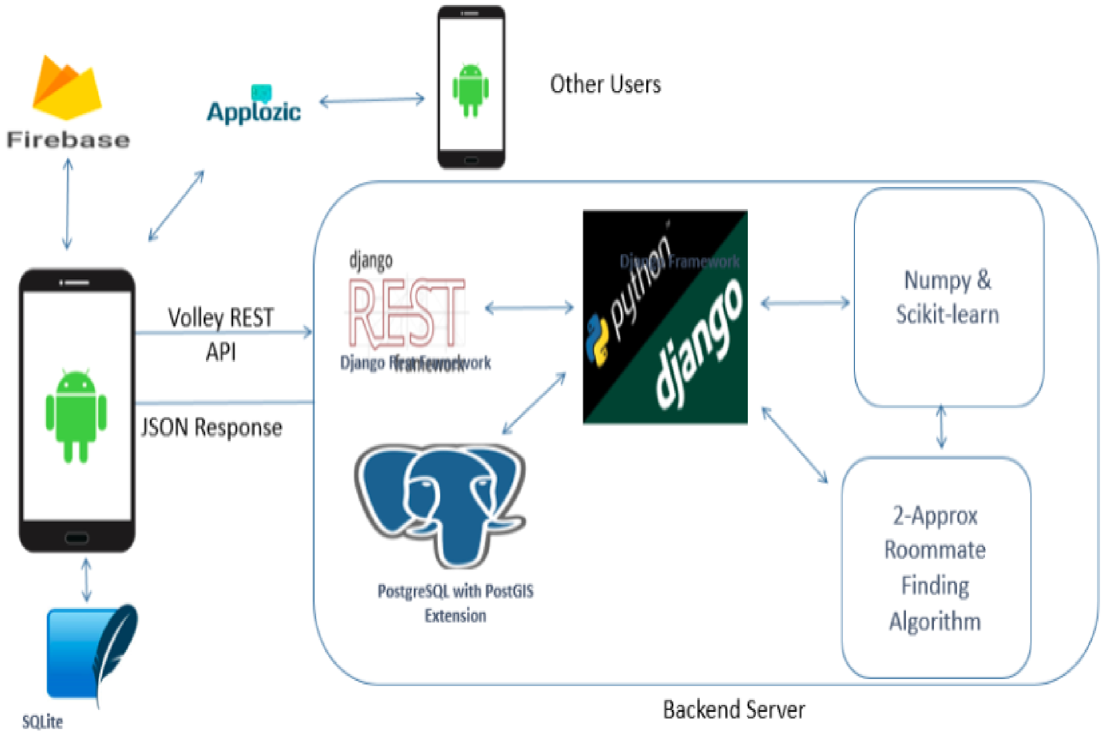

Firebase
Applozic
Other Users
Volley REST API
JSON Response
django REST Django REST Framework
python Django Framework django
Numpy & Scikit-learn
2-Approx Roommate Finding Algorithm
PostgreSQL with PostGIS Extension
SQLite
Backend Server

Figure 5.2: System Design

Let us start with the backend architecture. The backend is completely managed by the Django Web Framework. Django Framework is a MVC framework used for fast development of web applications making use of the Python programming language. As the Android application or the front end is API driven we needed a framework to develop rest services. DjangoRestFramework is rest framework supporting Django appication development framework. This framework plays a vital role in incoming HTTP requests. It listens for all incoming requests at different URLs and the based on the URL serves that request with an appropriate response. Django Rest Framework has serializers which help Django in converting the responses to JSON or XML responses to be sent to the android application user. For storing the data, we have used PostgreSQL with PostGIS extension. PostGIS extension was required for geographical queries like narrowing the users in a specific location within a specified radius. Django helps in creating the realtional tables in the database or the data store by using the models defined by the programmer. We have three main models in the design of MavRoomie like userprofile, roomie and apartment. Files like images are stored in a local filesystem and their links are stored in the database.

The algorithms have been developed using Python 2.7 with the support of packages like numpy, scipy and Scikitlearn's. Django on each request from a user passes the lists of users and their details to the algorithms which the compute diatance matrices and returns the list of users that satisfy the cost, geographical constraints and are close in terms of affinity. This list is then converted to JSON data format and sent as a HTTP response.

In the Android Application, the user authentication is handled by the Django server with token authentication. Each time the android application interacts with the API endpoints, it has to pass a valid authentication token generated by the server. This token is part of every request from the application and only expires on logout from the application. If the token is not valid, HTTP 404 BAD REQUEST is generated and the request will not be forwarded. There is also the provision for messaging other users using the chat functionality implemented using a third party API. The application also has Firebase error reporting enabled for detecting and notifying errors or system crashes. SQLite has been used as a local datastore to store some fo the essential information locally.

CHAPTER 6

User Interface Description

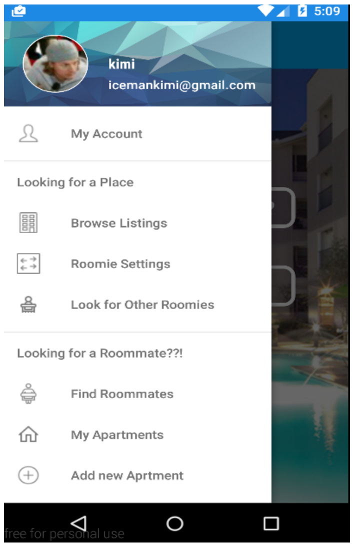
As MavRoomie is an Android application developed with the goal of helping users to find roommates and apartments/places, there was a lot of emphasis on how user friendly and unambiguous the user interface should be designed. So to build a responsive and user-friendly UI, we have used Google Material Design for User interface Design. In this chapter we will go over the flow of the application also discussing about the functionality of the User interface in different scenarios.

6.1 Home Screen

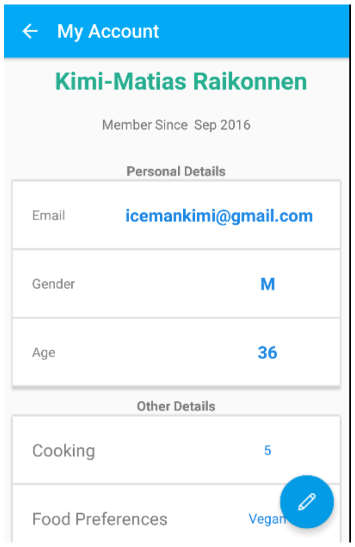
MavRoomie's home screen is as shown in the figure6.1a below. It has a navigation bar for navigation through different scenarios as seen in the image.

6.2 User Profile Screen

On selecting the "My Account" option, a new screen comes up as shown in the second image 6.1b. This screen shows all the profile information of the user. The screen also shows the preferences 6.1c or the lifestyle choices of the user like cooking, food preferences, sleep patterns ,social,noise, alcohol and soon. This information can be edited by clicking on the edit button as seen in the image. We use the preferences provided here for our affinity or distance calculations.



(a) Home Screen



My Account

Kimi-Matias Raikonnen

Member Since Sep 2016

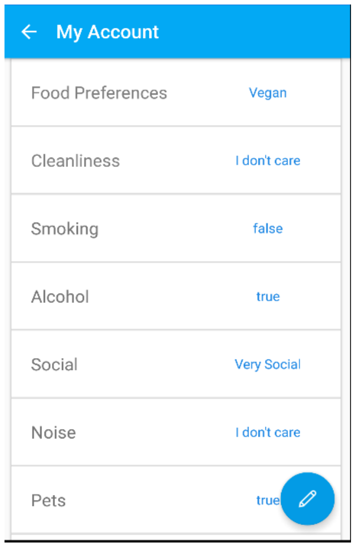
Personal Details

Email	icemankimi@gmail.com
Gender	M
Age	36

Other Details

Cooking	5
Food Preferences	Vegan

(b) Profile Information



My Account

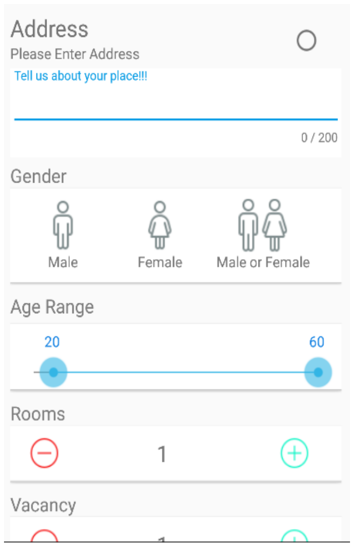
Food Preferences	Vegan
Cleanliness	I don't care
Smoking	false
Alcohol	true
Social	Very Social
Noise	I don't care
Pets	true

(c) Preferences

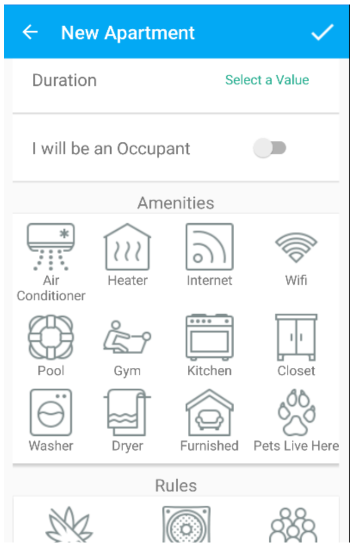
Figure 6.1: Home Screen and Profile

6.3 Scenario : User Looking for roommates to fill into his place

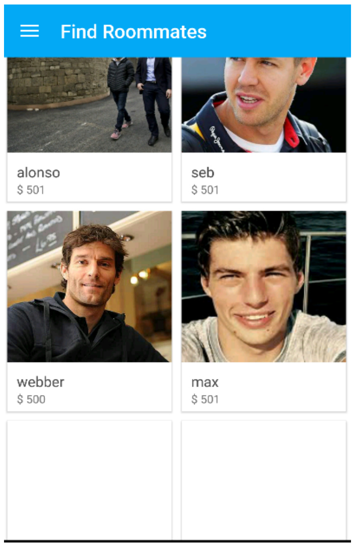
Before the user can look for the users, he has to add an apartment. For that, the user can tap the "Add New Apartment" option in the navigation bar and a new screen appears as shown in the image. The user can then enter the address of the apartment, prefered gender of the roommates, age-range, rooms, vacancy, amenities,rent,etc along with the images of the apartment. On Successfully uploading the apartment, the user can find the apartment in the apartments list in the screen "My Apartments". The user can check for the information by clicking on it or edit the information. To find the roommates for each particular apartment, the user has to long press on a apartment and a new screen appears with the list of roommates who satisfy the cost constraints and are close in affinity to the user(Home owner) and between each of them. If the user while adding the apartment has not selected the option of "I will be an Occupant", then the algorithm ApproxGrp is run and the resulting group is the one with the roommates who have close affinity to one another and satisfy the cost constraints.



(a)



(b)

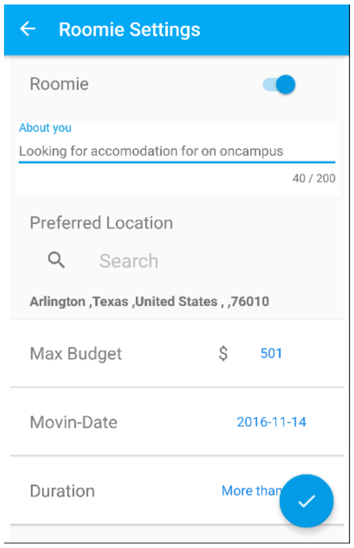


(c)

Figure 6.2: Adding a new Apartment

6.4 Scenario:Roommate looking for other roommates to rent an apartment.

Although the user can directly search for all the apartments in a particular location, he has to provide other essential information using the Roomie Settings option in the navigation bar of the home screen. The Roomie Settings screen is as shown in the image. It asssists the user in provinding the information regarding the preferred location, and maximum afforadable cost along with other details. Once the user has provided the details, he can go back to the navigation bar and select Browse Listings. On selecting any of the apartment from the list a new screen is opened which contains all the details of that apartment. At the bottom of this Apartment details screen is shown a list of roommates who are the potential roommates with close affinity towards you and also among themselves and who satisfy the cost constraints of that particular apartment. To view the profile of those roomies , the user has to tap them and a new screen opens with their profile information. The user can also send a message to the roomie if he wants to get more information.


Roomie Settings

Roomie

About you
Looking for accomodation for on oncampus
40 / 200

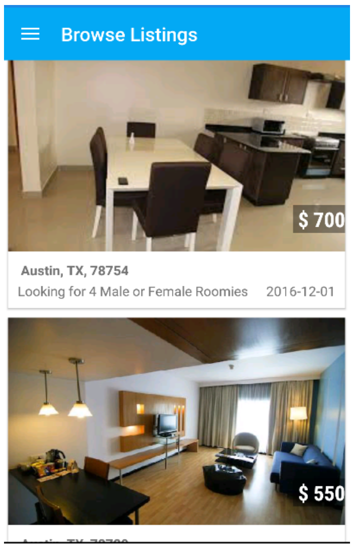
Preferred Location
Search
Arlington ,Texas ,United States , ,76010

Max Budget $ 501

Movin-Date 2016-11-14

Duration More than

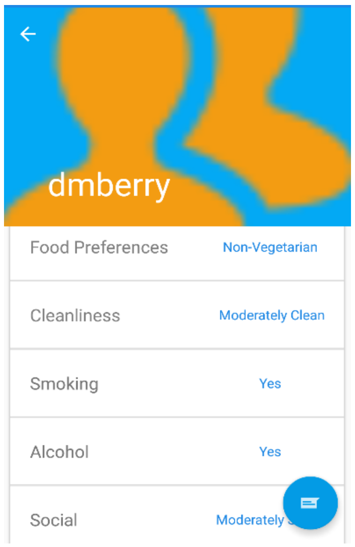
(a)


Browse Listings

$ 700
Austin, TX, 78754
Looking for 4 Male or Female Roomies 2016-12-01

$ 550

(b)


dmberry

Food Preferences	Non-Vegetarian
Cleanliness	Moderately Clean
Smoking	Yes
Alcohol	Yes
Social	Moderately

(c)

Figure 6.3: Searching for Apartments and Roommates

Apart from the main functionalities, MavRoomie comes with a robust token authentication system for user authentication. To logout from the system the user can tap on the Logout option from the Home Screens navigation bar.

CHAPTER 7

Conclusion

In this thesis, we have extended the problem of Team Formation to a more genaral problem of finding compatible roommates. We have built a end-to-end system that assists the user in searching for potential roommates, searching for apartments, posting apartments, mainting profile information. We have also explained the different scenarios and how each of them is solved. We have also provided with the algorithms that we have used in finding the potential roomates along with the proof of 2-Approximation factor used because of the triangle inequality. We have also explained why the existing applications fail to leverage the information provided by the user and why our solution is better. We have explained how the strategies followed in collaborative team formation environments can be used for a more general problem.

7.1 Future Work

Future plans for MavRoomie involves deploying the android application in the Google Play store, so that it can be downloaded by the Android users and also help them by providing a sytem for finding compatible roommates. We are also planning to integrate a notification system, so that users can get notifications regarding new apartments that match thier requirements. Also we are planning to integrate maps, which will help the users to choose or navigate over the available apartments in a location.

As MavRoomie is heavily influenced by the idea of team formation, this solution can be extended to other team formation problems. For e.g. this application can be extended to create a framework for collaborative editing, fansubbing or even citizen journalism by providing dedicated platforms for each of the problems.

REFERENCES

[1] Habibur Rahman, Saravanan Thirumuruganathan, Senjuti Basu Roy, Sihem Amer-Yahia, Gautam Das. Worker Skill Estimation in Team-Based Tasks. In PVLDB 2015.

[2] S. B. Roy et al. Task assignment optimization in knowledge- intensive crowd-sourcing.VLDB Journal, pages 1-25, 2015

[3] Habibur Rahman, Saravanan Thirumuruganathan, Senjuti Basu Roy, Sihem Amer-Yahia, Gautam Das. Task Assignment Optimization in Collaborative Crowdsourcing

[4] Jiawei Han, Micheline Kamber, Jian Pei. Data Mining Concepts and Techniques.Morgan and Kaufman.

[5] Charles E. Leiserson, Clifford Stein, Ronald Rivest, and Thomas H.Cormen.Introduction to Algorithms.MIT Press.

[6] https://www.citizensciencealliance.org/.Citizen Science Alliance.

[7] https://fanlore.org/wiki/Fansub

[8] https://en.wikipedia.org/wiki/Fansub

[9] https://www.litterareport.com/about/