Roommate Finder Web Application

Riken Shah
NC State University
Raleigh, North Carolina
Computer Science
rshah9@ncsu.edu

Aaroh Gala
NC State University
Raleigh, North Carolina
Computer Science
agala@ncsu.edu

Himani
NC State University
Raleigh, North Carolina
Computer Science
hhimani@ncsu.edu

Mateenrehan Shaikh
NC State University
Raleigh, North Carolina
Computer Science
mshaikh@ncsu.edu

ABSTRACT

As per the latest facts, around 34000 students join NC state every year. Along with the studies, comes the stress of looking for compatible roommates. In most of the cases, students have to really struggle to get decent roommates, along with finding a sound place to live. We thought of building one stop platform which will suffice all roommate search needs. We realized that social media groups and current platforms lack when it comes to recommendations and search, and they also do not offer personalization. At the same time, we also proposed radically different attributes and features that will help in making the application more dynamic, scalable, feature-rich and user-friendly. We had surveyed and examined the common problems faced by current graduate students when it comes to finding roommates, which helped us in understanding the gray areas. We did a careful study of the existing systems developed and analyzed the code as well as results in great details and proposed an improved solution to solve this problem. We had also demonstrated the inefficiencies of existing solutions and how we can modify those to improve the overall results.

Keywords

Roommate Rapport Scale (RRS), recommendation system, in-built chat, Machine Learning, Graph mining, Indexing

1. INTRODUCTION

The main goal of software engineering is to solve real-world problems using efficient use of technology. There are very few applications that focus on some of the pertinent problems like finding a roommate who is compatible. This paper focuses on the solved problem of finding compatible roommates for incoming non-resident NC State students. Our goal is to derive meaningful results from the survey conducted, browse through relevant journals, and develop a robust, user-friendly, and scalable web application to solve the problem. Our survey strongly emphasized the need for a separate web application for roommate finding. Though there are several social media platforms such as Facebook or WhatsApp that allow people to create 'roommate finder' groups, posts accumulate one after another and reading through all the posts becomes almost impossible. Also, we try to find the pros and cons of the existing application by detailed analysis and research about its code structure and outcomes. Drawing conclusions from the survey results, we infer that poor roommate finding methods necessitate a user-friendly, reliable web application that attests a user as a good roommate through the virtue of recommendations.

In the further sections we dive deeper into working with the previous system and then eventually choosing to create a new one. On the top of it, we added new features that made the application more intuitive and efficient. We will explain various code smells that were present in the existing code base and how we on eliminated those. We created a continuous development and continuous integration pipeline for our project and worked while strictly adhering to agile methodologies. We worked and made sure to provide small deliverables every iteration. To achieve that efficiently we conducted timely meetings to collaborate the work and progress .

The rest of the paper is distributed as follows. We talk about literature in this domain in Section 2, including the background and motivation of such work. Section 3 talks about problems with existing system. Then in Section 4, we talk about various code smells that we discovered in existing code base. Enhancements and Implementation is detailed in Section 5. Section 6 details the architectural components of the project. We propose the use cases of the system in Section 7. We detail the evaluations done on the system in Section 8. Future scope is in section 9 and we conclude in section 10 with references in section 11.

2. LITERATURE REVIEW

Before delving into building the application, we did a careful analysis of existing systems. We read a few research papers and it was evident that interpersonal compatibility, food preferences and sharing the living space with someone with similar personality is a more satisfying experience. A literature review of roommate understanding and preferences depicted that the RRS(Roommate rapport scale) discriminated between those who selected their roommates and those whose did not. Naturally, those who choose their roommates have better rapport. If a student makes good friends with his roommates, it would have a positive impact on his psychological well-being, personal growth, purpose in life, and self-acceptance. Evidence also suggests that women and culturally similar pairs were higher on both trust and intimacy [1].

A student with a roommate conflict due to difficulty communicating with the roommate or advocating for their needs may have a negative impact on their academics [3]. Studies suggest that happiness and depression may be highly contagious across social ties [4]. This means an incompatible roommate could be a potential cause for unhappiness and could be detrimental to the social circle. Most students who live with roommates for the first time often find it difficult to communicate and adjust with their roommates. So, it is essential for the student to pair up with like-minded people as their roommate [6].

Students often do not choose roommates and may experience personality mismatches. In a sample of 31,500 students in a nationwide survey, 50.1% of women and 44.1% of men reported frequent or occasional conflict with roommates or housemates [2]. In a nationwide survey, 5.6% of students reported that roommate difficulties hindered their academic performance. Roommate conflict is a widespread experience among college students.

So our goal was to build a customized application for NC State students that would help overcome the disadvantages of traditional roommate finder applications. In order to ensure optimal matching, our system provides the option of priorities to individual preferences at the users end. Based on the priority matching we enable the users to find a satisfactory match as a roommate. If the priority is high, the preferences must strictly match, conversely the preferences with low priority may or may not match. Another issue with such roommate finder applications is reliability. The user could be a fraudster and may not be someone he claims to be. This calls into question the reliability of the application and compromises the safety of other users interacting with the fraudster.

2.1 BACKGROUND AND MOTIVATION

There are many research papers which show how interpersonal compatibility, food preferences and other attributes contribute towards satisfaction of sharing the living space with someone who has a personality similar to theirs. A literature review of roommate understanding and preferences depicted that the RRS discriminated between those who selected their roommates and those who did not. If a student makes good friends with his roommates, it would have a positive impact on his psychological well-being, personal growth, purpose in life and self-acceptance [1].

From the survey of requirement of an application to find a roommate, we found out none of the users agreed upon on website/source for their roommate search. Hardly any user denied with the necessity of a different application to find roommate. The survey results are presented in figure 1.

Even though mobile devices are widely used and preferred for everything. But people preferred a web application for finding a roommate over mobile devices. This can be for

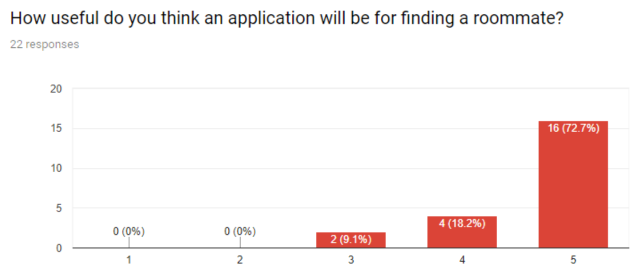


Figure 1: Importance of Application

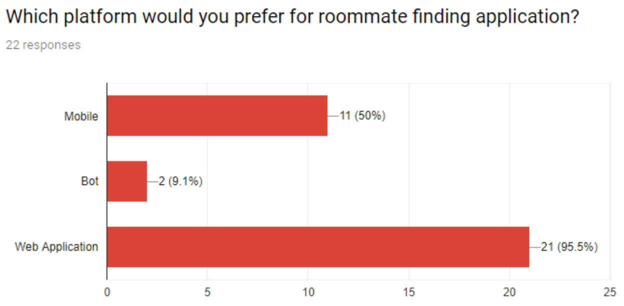


Figure 2: Preferred Platform

many reasons like they can multitasking of looking social media, online search etc simultaneously. The survey results are presented in figure 2.

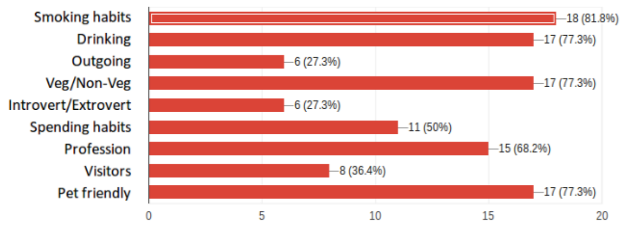


Figure 3: Important Attributes

From Survey result shown in Figure 3, we can see a list of attributes which are important or consider with high importance while looking for roommates. We can also see that there is a similarity or few clusters of attributes considered as important by most of the user.

From the survey, we found out that most of the users look at the social media profile of other person while finding roommates. This can be found in Figure 4.

We asked user how important do they consider friends recommendation while choosing roommates and we found out that most of the user will highly consider a friends recommendation while choosing a roommate. This can be found in Figure 5.

We also asked other question like which attributes do you think apart from the given are more important while choosing roommate and hardly people responded. Because for

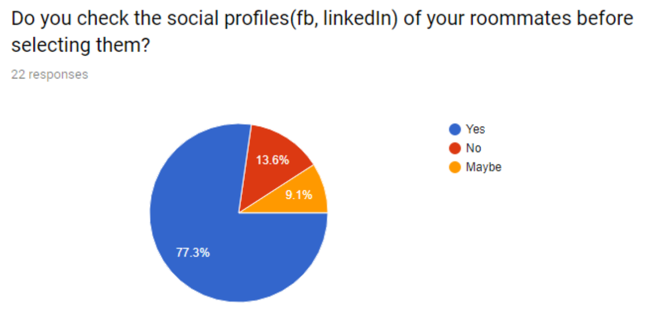


Figure 4: Social Media

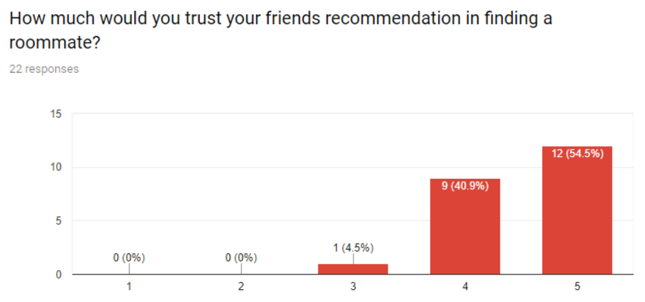


Figure 5: Friend Recommendation

most of the user a common cluster of attributes were more important.

3. PROBLEMS WITH EXISTING SYSTEM

Before getting started, we analyzed the current Roommate Finder web application and we found a couple of problems with the existing system.

1. **Incoherent results -**When a user sets up a profile in the web application, clicking on the profile icon sometimes showed profile of a different user. In the little time that we worked with the existing system, we were not able to figure out the cause of the problem but this was a major bug.

2. **Broken functionalities -**Search was not working for some users, it was giving application errors in case user does not set up their profile information with all the essential details and in an improper format.There were very few user fields search-able.

3. **Email Verification ambiguities -**The system worked on email verification but at times, we were able to login without completing the email verification. The email verification to validate a user was done using a personal email id, so we decided to change that to a more standard and platform specific email id. A standard email id helps establish trust in the application as compared to a personal email id.

4. **Database Design -**There was only one table in the database which stored user information and their profile. All the information was clubbed and stored in one table, even though normalization is good but we intend to work on roommate recommendation and implement back-end algorithms to come up with sound roommate suggestions so we would need to denormalize some data and seriously thinking about database design to meet our goals.

5. **No Modularity -**Modularity was introduced in the project structure but it was not adhered to, most of the code was put into one file.

4. CODE SMELLS

For detecting the code smells we used the paper “Do they really smell bad” where all the parameters were given for evaluation of code smell.[10]

1. **Duplicate code -** In the existing code-base there were quite a few places where there was duplicate code. For example, the model/users.js and model/request.js was exactly the same as app/model/users.js and app/model/request.js.

2. **Data should be private -** In the app.js we found that the mongo URL with the database username and password was available. Anyone can potentially delete the database with just one command. In routes/index.js the Gmail credentials were in clear of the user who authenticated the SMTP server. A good practice would be to have them included in the .env file so that the credentials are not public on GitHub.

3. **Spaghetti Code -** There were many methods without structure. The app/views/ had many files without structure.

4. **Long Methods -** The routes/index.js had some very long methods which could have easily been broken down into small methods. For example, the database update method could have been included in the model/user.js instead of exporting the users.js and then updating it routes files. The routes file should have been way simpler just showing the routes each request will take.

5. **Lazy Class -** There were a few lazy classes as well which were not doing much in the code. The app/views/login had 2 files with just one line of code which could have been easily merged.

6. **Testing -** Testing of the code is not done at all. So we cannot verify the correctness of the search algorithm.

5. ENHANCEMENTS AND IMPLEMENTATION

Because of the reasons stated above, we decided to go ahead with making a new system from scratch, we have a modular code base because it helps not only in development but also debugging and testing application. In modularity, we have followed MVC architecture where we are dividing our project structure in three layers-one for interaction with database, one for the back-end business logic and one for interaction with the user.

As per technical stack, we have built the web application in Node.js because it is scalable, fast and robust, the front end is in HTML/CSS and bootstrap. Also, we have used an existing robust library i.e. passport.js for user authentication and authorization. We have used MongoDB for database design and handling user data. We understand that roommate recommendation is a keynote feature of our application and we have used Graph mining algorithms to come up with relevant recommendations. Along with building a robust application with standard features we have integrated some amazing cool features like in built chat and roommates recommendation, which are discussed in detail in section 7.

Here is a comparison chart of our system with the existing application.

Table 1: Comparing with the existing system

Metrix	Original System	Proposed System
Recommendation	- -	++
Search	-	++
Authentication	-	++
Security	- -	++
Speed	++	+

5.0.1 *Recommendation*

The current system did not have any recommendation system where the roommates would be recommended based on their preferences. We have used machine learning to recommend the roommates.

5.0.2 *Search*

The current system does the searching by just a database query based on the certain selection. If the search cannot match any one of the parameters given parameters than it will not show any results. Our system has implemented a better search algorithm with more parameters and suggest the users the closest roommates.

5.0.3 *Authentication*

We have implemented google login with only ncsu.edu account thus making sure only NC State students can access the application. The original system had email verification which did not work in very effective way. The email would go but then it was not required for the user to click on link in email to be actually able to login in the system. This whole hassale of email verification is removed by using google login and authentication. We used google OAuth api tokens and PassportJS library to implemented this authentication.

5.0.4 *Security*

Our system has better security as we are using the standard practices like not exposing credentials on public repositories like Github, using standard tested libraries rather than reinventing the wheel which can consist of bugs. The original system did not follow this standard practices and they had the credentials in clear text on Github and they tried to create the authentication system (which had bugs) when there are many libraries available which are doing the same things in a better way.

5.0.5 *Speed*

The speed of our application is slightly slower than the current system as we are using the machine learning to recommend the roommates but since the feature is very useful users would prefer a slightly slower system which can recommend them roommates.

6. ARCHITECTURE

Figure 6 shows the proposed architecture of the application. We take into account various modules that are present in the system and also how they communicate with one another.

The master branch of the GitHub repository is integration with Travis continuous integration pipeline so that any changes that we push to the master branch will be automatically built on Travis and all the test cases will be run. If both the build and test cases passes then the code is pushed on the Heroku server where the NodeJS app resides. The web server will have UI Components, Application Logic, Security and Data unit. The application logic will includes authentication, registration and searching. The database used for the application is MongoDB at MLab Sandbox Database-as-a-Service.

The main application that is hosted on Heroku server has four major modules out of which three depend on individual independent modules. The four major modules are personality api, recommendations using ML, chat integration and search. And apart from search, the other three are independent modules integrated into the system. Here we are using independent modules to make system more modular and flexible to changes.

The chat module is based on sendbird.js API and it supports one-to-one chat as well as many-to-many chat. The recommendations are a very important feature of the system which are based on a custom machine learning algorithm developed using combination of k means clustering and k nearest neighbours algorithm. The goal here was to improve the performance of the system by reducing latency. The personality API gives many additional parameters to actually give better recommendations.

We have managed to follow best practises in version control paradigm of our system as well. We have created issues, and assigned it to specific people. We also created various branches for various features of the system and worked independently as well as collaboratively on those branches.

7. USE CASES

After thorough analysis and conducting a survey on the application, we found out that there are few additional functionality which can be added to enhance the application and make it more user friendly. In this section we will elaborate on each of the four major use cases.

7.1 Chat

Roommate finder helps in finding roommates on the application based on the attributes and the search query. We realized that being able to chat or basically efficiently communicate with a potential roommate in real time is a huge selling point. There can be two ways to connect the user. External and internal. For external communication, we need to use other platform or social media. The user

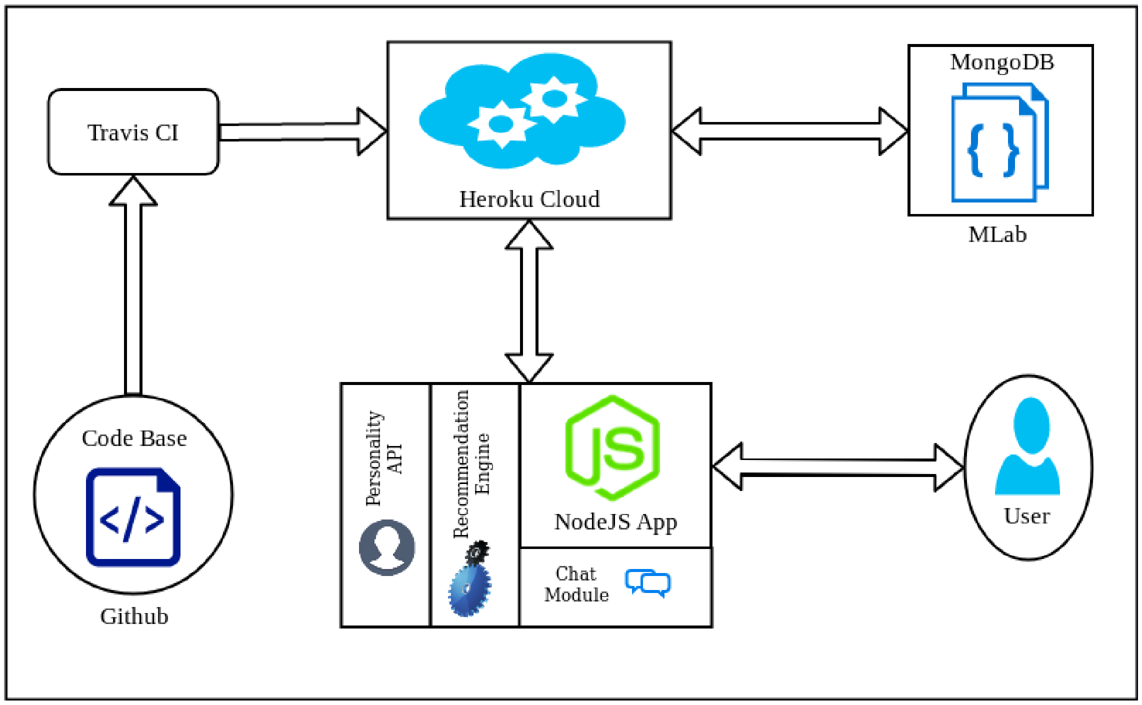
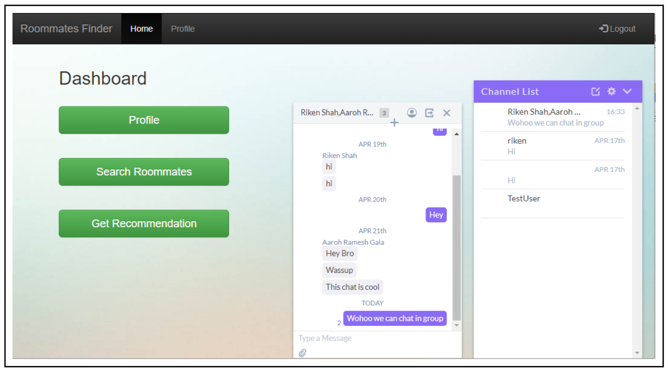


Figure 6: Architecture of the application

is being forced to share personal details like email id, phone number or social media account, which might be uncomfortable for many users. Also, this can lead to privacy issues, where users data can be fetch and used for some other purpose without users concern. In an internal communication, there is an inbuilt chat application, where two users can communicate with each other without sharing any personal information like email id, phone number or social media profile. Further, as this is a roommate finding application two roommates might want to find just one other roommate and this can also be done with the chat provided in our application.

So to solve this, we developed an inbuilt chat interface in our application. The users can talk with each other and they can simply add as many users as they want in an ongoing chat thread. The chat is also available on almost all the pages except log in and profile details page so they don't have to go to a separate page to chat. The chat dynamically updates, which means that when a user chats on a dashboard, he can toggle within the application, go to the recommendation result page and still see the same chat in continuation along with the history. We have been super careful to make chat secure by only allowing the users to only view each others user id while chatting and no other information like email or phone-number. The users will be automatically blocked if they use any bad or curse words,as they are restricted by us. Once they are blocked the admin will review them and then if they agree to our terms admin will unblock them otherwise they won't be able to use our chat feature.

7.2 Recommendation



Riken Shah,Aaroh R... 3

APR 19th
Riken Shah
hi
hi
APR 20th
Hey
APR 21th
Aaroh Ramesh Gala
Hey Bro
Wassup
This chat is cool
TODAY
2 Wohoo we can chat in group

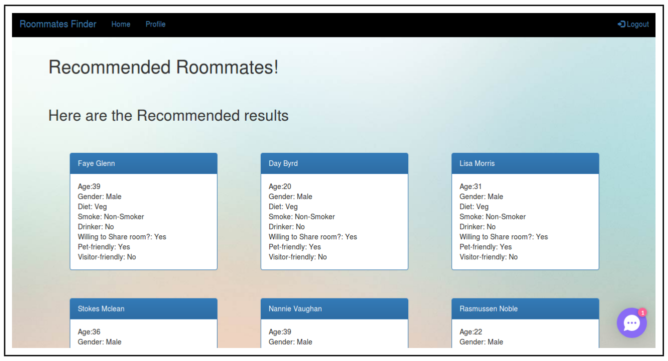
Riken Shah,Aaroh ... 16:33
Wohoo we can chat in group
riken APR 17th
Hi
APR 17th
Hi
TestUser

Figure 7: Chat

7.2.1 *Machine Learning*

The goal of the system is to improve recommendations while not affecting performance. Machine learning is more of recognition the pattern and using that pattern to improve. There are a few application of machine learning, one is Data mining, where huge amount of data is collected and unknown or common pattern and characteristics are found out, self-customizing algorithms that tunes itself according to the user preferences. This self learning mechanism can be used to make better recommendation and suggestions to the user and those are personal to an individual rather than common suggestions for everyone.

Now a days, every search is made more efficient by using machine learning algorithms. Various unsupervised learning algorithms are used to make cluster or groups of people/things which are of similar to each other. Finding



Recommended Roommates!

Here are the Recommended results

Faye Glenn
Age:39
Gender: Male
Diet: Veg
Smoke: Non-Smoker
Drinker: No
Willing to Share room?: Yes
Pet-friendly: Yes
Visitor-friendly: No

Day Byrd
Age:20
Gender: Male
Diet: Veg
Smoke: Non-Smoker
Drinker: No
Willing to Share room?: Yes
Pet-friendly: Yes
Visitor-friendly: No

Lisa Morris
Age:31
Gender: Male
Diet: Veg
Smoke: Non-Smoker
Drinker: No
Willing to Share room?: Yes
Pet-friendly: Yes
Visitor-friendly: No

Stokes Mclean
Age:36
Gender: Male

Nannie Vaughan
Age:39
Gender: Male

Rasmussen Noble
Age:22
Gender: Male

Figure 8: Recommendation

a roommate can also be considered as an search query based on few parameters like preferences, etc. Running a simple query and returning a list of user which can be potential roommates increases the task of the user. The user needs to look at all the profile and see for more similarities between him/her and the list of people. But this can be reduced by using machine learning algorithms. The machine learning algorithms will be trained in such a wait that rather than returning a simple list of users which satisfies the parameters, it can give list of users in an order which are more similar or has more attributes similar to the search user.

In simple terms, machine learning algorithms can return a list of user which satisfy the attributes in such a way that the possibility of two people gelling up together is maximum at top to minimum at the end. We implemented machine learning algorithms to make the search better which will help the users to find roommates which have more similarities and the likeliness to gel up well together. And that was our main reason of using machine learning to make better suggestion, so people find good roommate.

7.2.2 Implementation

Searching roommates based on a list of few attributes might or might not result in finding a perfect roommate. Roommates get along well together if they have similar personalities or something in common. For E.g. if two person like football, then there are more chances of them getting along together with person who like football from two people of where one like football and other does not like football. This could be more better if both the roommates like the same football team. Knowing the personality of a person can help in finding a better roommate then a normal search query which gives list of user which satisfy some criteria.

We have implemented and included a feature which analyzes the user personality based on user description. We have a personality API, which takes in input as text. It analysis the text and gives user characteristics like Openness, Agreeableness etc. It gave a bunch of attributes/characteristics of users. Each attribute has a normalized value from 0 to 1, 0 means the least or none and 1 means high. Personality API gives a lot of parameters about user profile, but we felt that only few of them were significant for identifying roommate's characteristics. So we took only five of the attributes, which were agreeableness, neuroticism, extroversion, conscientiousness and openness.

Once the personality, likes and dislikes are known along with the user input profile attributes. We used machine learning algorithms to make better predication and recommendation of roommate while choosing a roommate. To make better prediction and recommendation, we used K-means clustering algorithms to clusters users based on personality and then we used K - Nearest Neighbour to make better prediction. The top 6 result obtained from K - Means and K - Nearest Neighbour is displayed to the user as recommendation.

7.2.3 K - Nearest Neighbour

K - Nearest Neighbour is unsupervised machine learning algorithm. It is a very basic pattern recognition algorithm. K - Nearest Neighbour is a non parametric algorithm, meaning, it does not make any underlying assumptions about the distribution of data. It is also a type of lazy learning algorithm, when the function is only approximated locally and all computation is deferred until the last moment i.e. until classification. K - Nearest Neighbour take an input parameter as K which is the number of nearest neighbour the user is expecting as an output.

K - Nearest Neighbour is given prior data as an input, but the prior data is not classified or assigned any category. When a new entry comes, K - Nearest Neighbour takes all the previous data and sorts the data according to similarity. If a data point is most similar to new data entry or same that it will appear on the top, where as if the data points are least similar then it will appear at the end. This similarity is calculated based on some estimation. It can be the distance between two points on the graph, closer the points make it more similar to each other.

For our application, we have used Euclidean Distance as an function to find out similarities between two data points. A small modification was made in Euclidean Distance to suit out project. We had given 90% weight age to the input parameters like smoking, alcohol, pet friendly etc and 10% weight age to the personality attribute. 10% weight age was given, as those attributes were computed by us, and we didn't want to give bad recommendation considering some attributes which were not highly accurate.

This modified Euclidean Distance was used to created a temporary sorted list when a new data point is entered. And the top K results were returned. Choosing K for K - Nearest Neighbour is a difficult task. We wanted to give user lot of recommendation, but we didn't want to give a list of user. We tried choosing different K values for K - Nearest Neighbour, but we did not get much difference in time. So we decided to take K = 6. As it would recommend enough users and not give lot many results at the same time.

7.2.4 K - Means Clustering

K - Means Clustering algorithms is also unsupervised machine learning algorithm. It is popularly used for cluster analysis in data mining. K - Means clustering algorithm aims to partition all the observations or data into k clusters. In which each observation belongs to the cluster with the nearest mean. K - Means clustering has a loose relationship to K - Nearest Neighbours. K - Means algorithms with K = 1 is close to K - Nearest Neighbour with the whole data as one cluster. All the data in a cluster are more similar to each other than data point outside the cluster. A cluster contains data which has high inter similarity.

K - Means Clustering is given prior data as an input, but the prior data is not classified or assigned any category. K - Means clustering algorithms creates clusters of the data. As each cluster has data which are similar. The similarity is calculated based some estimation. It can be the distance between two points on the graph, closer the points make it more similar to each other. We used Euclidean distance to compute similarity between each point. We have modified the Euclidean distance calculation to suit our project. It is similar to the modification made in K - Nearest Neighbour Euclidean formula. 90% weight age is given to basic parameters and 10% weight age is given to personality parameters.

To calculate the clusters, we first initialized K points as means. Then we categorized all the data points in the database to its closet mean and made K clusters. Once the clusters are formed, we updated the mean coordinates which are the averages of all the items in that cluster. Once the mean is updated for every cluster, we then again categorized all the data points in the database to its closet mean and made new K clusters. This update of mean and categorizing data points into cluster will carried till the mean of clusters was not constant. The clusters obtained at the end were considered final clusters with their final means.

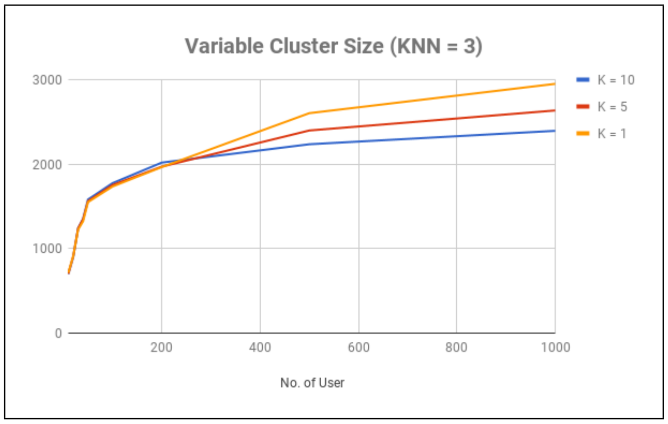


Figure 9: Efficiency with different K values.

Choosing K as the number of cluster becomes critical. This can affect the efficiency of the recommendation and machine learning part. We tried out with different values of K. We noticed that as the K value is increased from 1, the efficiency of the algorithms increased till a point, after which it started degrading.

We kept the K value of K - Nearest Neighbour constant and varied K value for K - Means from 1 to 10 and we obtained this results. From Figure 9 and Figure 10, we

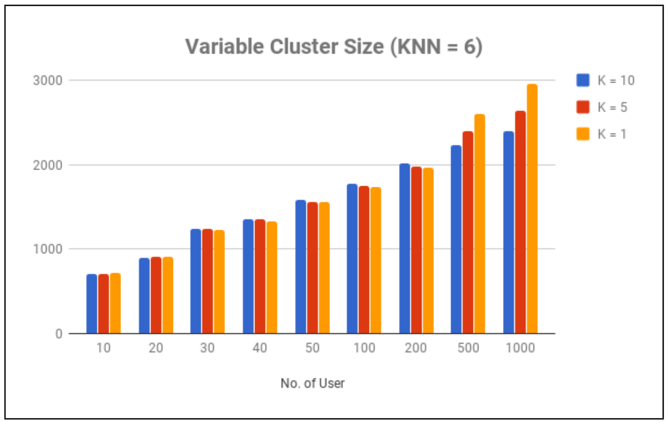


Figure 10: Efficiency with different K values.

can see that as the K value of K - Means clustering is increased, the efficiency of the machine learning algorithm has increased. At K = 10 for K - Means clustering we obtained the most efficiency, so we chose K = 10 for out project.

7.2.5 K-Means and K - Nearest Neighbour

K - Nearest Neighbour algorithm would have been enough for our project to use machine learning and recommendation part. But we were thinking of the application to handle a large number of user. North Carolina State University has thousands of students. If we implemented only K - Nearest Neighbour then the recommendation would have been slowed. Whenever used would asked for a recommendation, the server would have to compute similarity between the user with other thousands of students. This would result into unnecessary heavy computation for each search.

To improve the efficiency of K - Nearest Neighbour, we implemented K - Means clustering along with K - Nearest Neighbour. K- Means Clustering algorithm will create clusters and each cluster is represented by the mean of the cluster. This would help us in increasing efficiency and less computation. The user profile is compared with the clusters mean, and the cluster with the nearest mean was selected. Once the nearest cluster is selected, then we apply K - Nearest Neighbour algorithms among all the users in that cluster to find the recommendation user.

By looking at the cluster only, we are already narrowing down the comparison heavily. We only apply K - Nearest Neighbour algorithm to hundreds of user of a particular cluster than hundred thousands of user. As number of comparisons are drastically reduced, we are improving the efficiency.

We compared the time taken by only K - Nearest Neighbour and K - Means Clustering and plotted the graph. From Figure 11 we can see that as the number of users increases, the time taken by combination of K - Means Clustering algorithm along with K - Nearest Neighbour was lesser than only K - Nearest Neighbour. At the start when the number of users were less than 200 users. K - Nearest Neighbour was more efficient than the

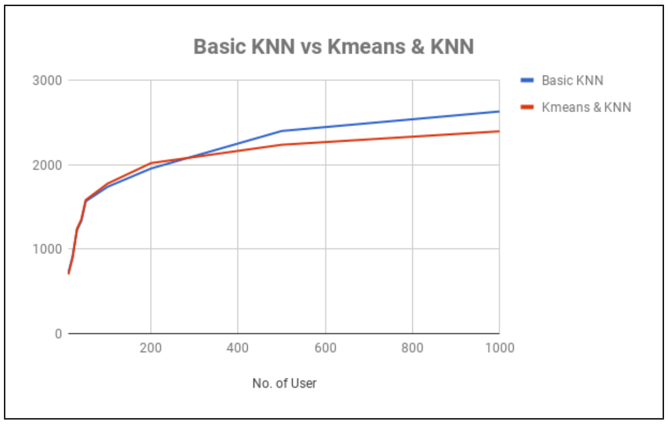


Figure 11: Comparison of K -Means with KNN and KNN.

combination of K - Means Clustering and K - Nearest Neighbour. This was because the time taken to compute the cluster and then apply K - Nearest Neighbour was more.

The combination was implemented with the intention of large number of user. We found a saddle point of around 200 users from the graph. If number of users are less than or around 200 then only K - Nearest Neighbour is more efficient, but it number of users are more than 200 then the combination of K - Means Clustering and K - Nearest Neighbour would be more efficient.

After many experiments and measuring performance for different combination values of K for K - Means Clustering algorithms and K - Nearest Neighbour. We came up with the appropriate value of K to be used based on the number of users which will be efficient. The table is as follows:

Number of Users	K (Clustering)	K (KNN)
<200	1	3
500	5	3
1000	10	5
10000	20	6

Table 2: Machine Learning Tuning Parameters

This table depicts how for less number of users it is not very efficient to use combination algorithm but as the number of users increase, the performance of the combination algorithm surpasses that of base algorithm.

7.3 Personality API

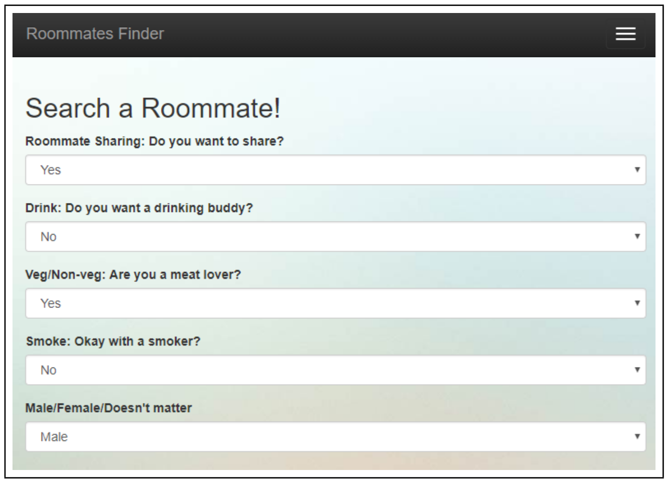
The parameters that are taken as input from the user are limited and can not always take into account subtle nature of humans and their personality. we have used the magicsauce API which takes in the summary section of User profile and analyzes it to give various personality parameters like Agreeableness, Neuroticism, Extraversion, Conscientiousness and Openness. The values of this attributes are in the range 0-1 which can be easily used in the recommendation engine.

Now, the API is also capable of using social media accounts of the person to give additional parameters. But according to our survey, many users were still hesitant to share their social media profiles. Hence, we decided not to include that in the system. However, with the text analysis of the summary section of user profile, the additional parameters that we obtain improves the recommendations considerably.

To make the system more modular, we have developed a separate application and developed the personality module using Python and flask. Hence, the NodeJS application hosted on heroku gets the personality parameters using the REST application that is separately hosted.

7.4 Search

The previous system did a good job of implementing the search. We noticed that previous system was search-able on few user fields, so we made more user parameters search-able. Now a user can search over a budget range, or within an age range. He can search across all user parameters like drinking habits, smoking habits, pet friendly, visitor friendly etc. We also made it optional to skip few fields and fill some. We also took care of the situation where the user does not search over any field.We made the search query faster by adding indexes on all user fields(profile information), we also added text based search on user summary. Then we used logical And and OR filter operators of mongodb to create dynamic query and to check over multiple fields of user.


Roommates Finder

Search a Roommate!

Roommate Sharing: Do you want to share? Yes

Drink: Do you want a drinking buddy? No

Veg/Non-veg: Are you a meat lover? Yes

Smoke: Okay with a smoker? No

Male/Female/Doesn't matter Male

Figure 12: Search

8. EVALUATION

We had planned to test the effectiveness of our system by performing real-time evaluation among NC state students. As per the groundwork, we had deployed RoommateFinder web application on Heroku web server and asked interested students to sign up the evaluation sheet. We asked them to fill a short survey to rate the application across several parameters-Predictability, Availability and Adaptability. Apart from evaluating around the non functional requirements, we also covered task-based evaluation, time based evaluation and A/B testing. In time and task based evalu-

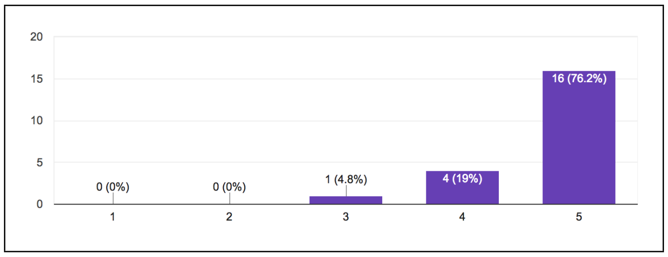


Figure 13: Relevance of Recommendation System

ation we urged the evaluators to perform some particular tasks within a tight time bound constraint.

We received an amazing and overwhelming response from 21 evaluators. Students in general really liked our application and complemented us on how smoothly everything was working. One of the major achievements were the success of Roommate Recommendation system. We asked people for relevancy of recommendation system and 76.2% people found it very relevant and rated it five and 19% rated it four on a scale of 5.

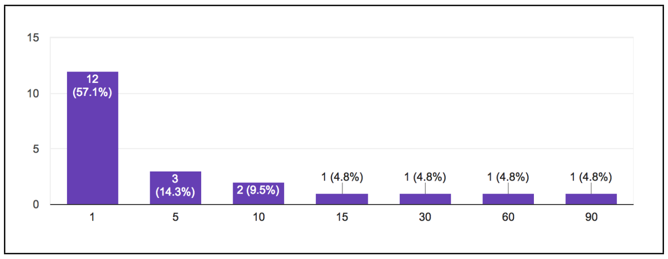


Figure 14: Rendering of Results in Recommendation System

To test the effectiveness of Recommendation System, 57.1% people were successfully able to fetch results within 1 seconds and 23.8% people got results with 10 seconds. This is a good time considering we are using KNN for recommendation and we had populated our database for we thousands of users. And as of now we are not using any caching or MapReduce to speed up our queries or optimize the querying process.

Now that we have improved search and added indexes and logical operators in the query in operators to accommodate the filtering of user entered data. This really enhanced our search results, so we asked users to enter the time it took for them in rendering the search results. And as per the evaluation results, around 78% of the users were able to render the results in less than 30 s.

We also compared the efficiency of our search system with the previous system, 95.2% agreed that our search was faster than the previous system. Not only ours was faster, our system could handle scenarios where user did not select any filter. Adding indexes really made it faster, and I think it can further be improved by adding elastic search and adding compound indexes.

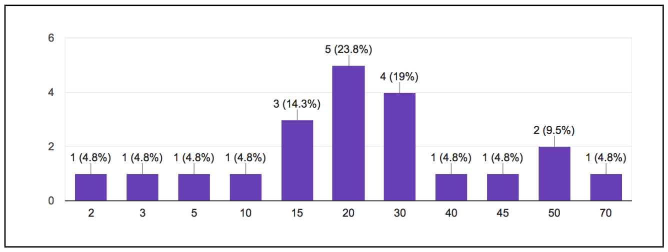


Figure 15: Rendering of Search Results

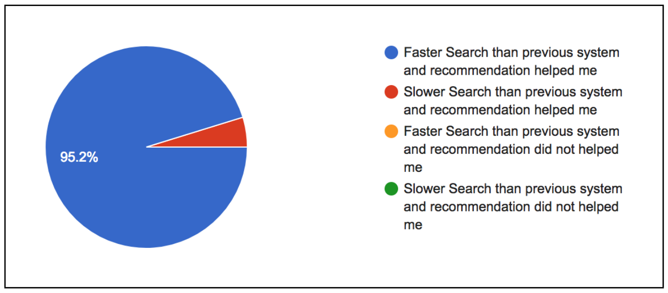


Figure 16: Search Comparision.

To measure predictability and adaptability of our applications, we asked users to perform strict time bound tasks and later questioned them if they were able successfully perform them or not. So users were successfully able to figure out chat, recommendation and search within a minute of using the application.

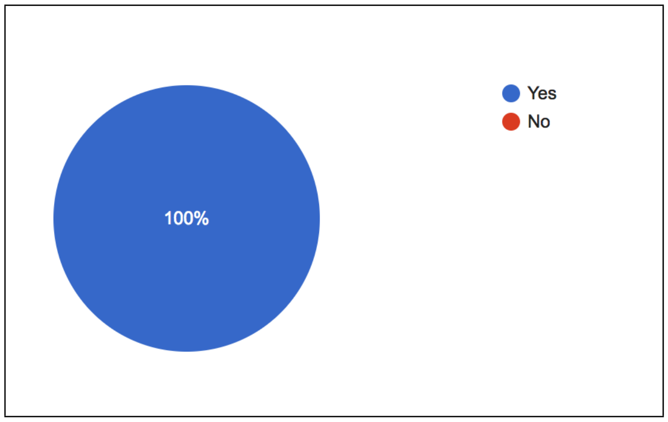
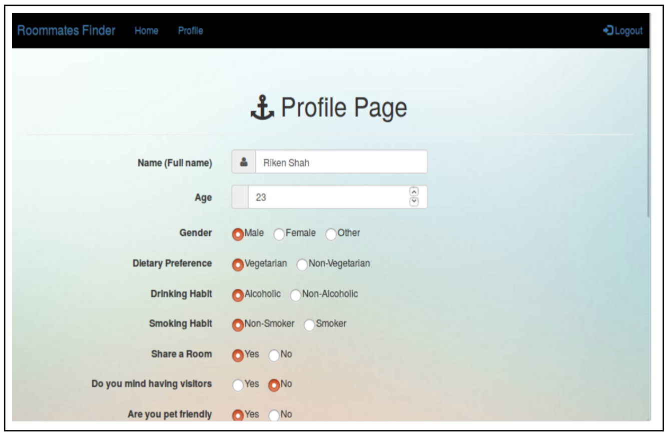


Figure 17: Adaptability and Predictability.

Apart from these parameters we also covered Likeability, Future scope, and Usability. We have also explained a little bit in detail about platform compatibility and maintenance. As per platform compatibility of the application, our UI is responsive and designed such that the application renders without distortion on mobile interfaces too. We have also tested that the chat works fine in the mobile browser, even though that was not a part of Evaluation.

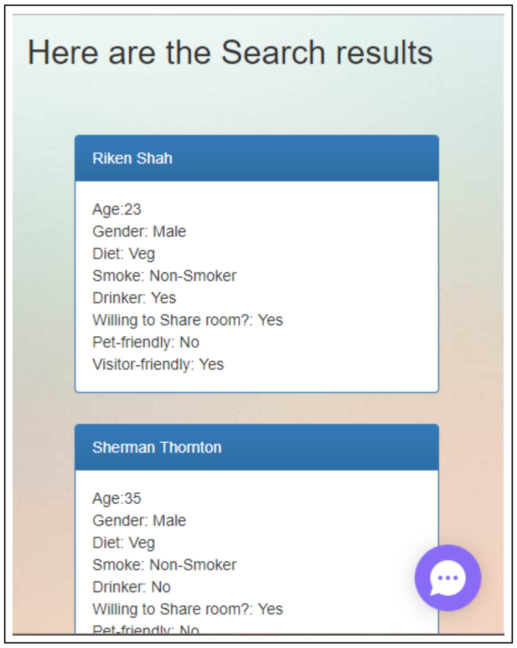
We believe that aesthetics of a web application play a very important role in the usability of the web application. To attain that we added a bootstrap interactive component.



Profile Page

Name (Full name)	Riken Shah
Age	23
Gender	[✓ Male] Female Other
Dietary Preference	[✓ Vegetarian] Non-Vegetarian
Drinking Habit	[✓ Alcoholic] Non-Alcoholic
Smoking Habit	[✓ Non-Smoker] Smoker
Share a Room	[✓ Yes] No
Do you mind having visitors	Yes [✓ No]
Are you pet friendly	[✓ Yes] No

Figure 18: Better Aesthetics for more usability.



Here are the Search results

Riken Shah

Age:23
Gender: Male
Diet: Veg
Smoke: Non-Smoker
Drinker: Yes
Willing to Share room?: Yes
Pet-friendly: No
Visitor-friendly: Yes

Sherman Thornton

Age:35
Gender: Male
Diet: Veg
Smoke: Non-Smoker
Drinker: No
Willing to Share room?: Yes
Pet-friendly: No

Figure 19: Responsive UI.

Please see figure 18, which shows a user profile page. Also our core UI design has many out of the box components like panels and jumbotrons that could be used for further enhancements in the future scope.

We also had an optional column where evaluators could report any bugs that they found in the system, there were no major bugs reported, our application was highly available and consistent throughtout the evaluation period. One bug that was reported, was that move in date was added in the search filtering but not implemented successfully. We agree that this is a bug in the application but it does not break functionality, we can view this as an enhancement as a part of future scope.

Finally based on the survey results and exhaustive evaluation analysis, we have established overall effectiveness of Roommate Finder to improve and enrich Rommmate Search.

9. FUTURE SCOPE

Our aim was to build a system that would suffice all major requirements of a roommate finder system. We have successfully implemented the 4 major use cases of search, recommendation, chat and personality API, along with standard authentication via Gmail and basic CRUD operations on the user profile. Here we enumerate some enhancements and new additions that will add the depth as well as breadth of roommate finding experience.

1. **Integration with Social Media-**Integration with social media like Facebook messenger can really improve the usability of the web application. Imagine getting to know the list of mutual friends(if any) with a potential roommate. Also, implementing something similar to LinkedIn first, second and nth connection can be really helpful for establishing the credibility of the application.

2. **Roommate Suggestion-**One very cool feature that can really improve the roommate search experience can be roommate suggestion. Once the social media is integrated, and if a user happens to come across out a good roommate profile for his friend who is also searching for a roommate, then he can suggest this user's profile to his/her friend. Roommate suggestions will add credibility to the application and provide a much more personalized experience.

3. **Mobile Support-**From our survey analysis, we found that there were many users who wanted the application on their mobile devices like android and IOS. So, a mobile application will help in building a wider audience and attract more people to engage in our system.

4. **Search Housing-**The next big step after successfully finding a roommate is finding a house together. So we can add a feature where two or more users can sign-up together as roommates and look for listing of houses together.

5. **Review Roommates-** A user can review his roommate on a web application. This review can be anonymous or identified. Such reviews can be useful for other users to make better-informed decisions.

10. CONCLUSION

After having established that sharing living space with a person with similar interests and preferences is more satisfactory and healthy in general, our application had set out to incorporate crude elementary details and preferences which make roommates compatible. And we had also gained sufficient data from pre-evaluation surveys and online resources supporting the fact that, there was a lot of scope of improvement in the roommate search and there was a dire need of an effective application which delivers to all roommate search needs.

Post-Evaluation surveys suggest that we have built a robust Recommendation system using Machine learning algorithm. We were successfully able to to mine relevant data-sets for our application from the widespread use of social media. Having relevant data-set from the users helped us better understand if our recommendation results offer good roommate matching and it is persistent to the actual experience of our target audience. Also, we added an online chat platform that will help the user in connecting more efficiently while choosing a roommate.

Also our application delivers faster search and wider scope of filtering options, and users agreed that search of our system is faster than the previous system. We have been able to offer good latency in fetching search and recommendation results even though we have thousands of users in the system. It offers high availability and platform compatibility. We also added gmail authentication to enhance user experience and prevent unnecessary sign-up verifications. It is interesting to note that our system reported no bugs during the evaluation period, only some basic tweaks and enhancements, which we have covered in detail in the future scope. Our four major use cases are fulfilling the expectations of a typical roommate finder application.

11. REFERENCES

[1] Erb, S.E.,Renshaw, K.D.,Short, J. L., Pollard, J. W. (2014), The importance of college roommate relationships: A review and systemic conceptualization, Journal of Student Affairs Research and Practice.

[2] M. Shekhawat, S. Deshmukh, G. Monroy, A. Tiwari, X. He, H. Shin, Y. Hong and H. Lu, Usability Test of Personality Type within a Roommate Matching. Journal of International Technology and Information Management, Volume 25 | Issue 1

[3] Ashley M. Payne, The challenges of living with a roommate: the impact on students with disabilities' residential experience, Rowan University

[4] Ezra Golberstein, Janis L. Whitlock, and Marilyn F. Downs, Social contagion of mental health: Evidence from college roommates, Health Econ. 2013 Aug; 22(8): 965-986

[5] https://yocket.in

[6] Kimberly Halpin, Roommate Rants: Understanding Roommate Conflicts among MSU Students, Journal of Undergraduate Research at Minnesota State University, Mankato. Volume 9 | Article 3

[7] https://github.com/googlemaps/google-maps-services-js

[8] https://www.mongodb.com/blog/post/the-modern-application-stack-part-2-using-mongodb-with-nodejs

[9] Itti Hooda, Rajendra Singh Chhillar,"Software Test Process, Testing Types and Techniques", International Journal of Computer Applications (0975 - 8887) Volume 111 - No 13, February 2015.

[10] Fabio Palomba, Gabriele Bavota, Massimiliano Di Penta, Rocco Oliveto, Andrea De Lucia (2014), Do they Really Smell Bad? A Study on Developer's Perception of Bad Code Smells

12. CHITS

- HZI
- JXL
- URA
- LQK
- MOY
- PNZ
- WHL
- KVS
- PAU
- XXA
- NFZ
- ZYZ
- DOI
- WIA
- NXQ
- KKC
- RZB
- RJJ
- ZNU
- ZVW