Roommate Finder

Kavi Pandya
Computer Science
North Carolina State University
kdpandya@ncsu.edu

Nilay Kapadia
Computer Science
North Carolina State University
nkapadi@ncsu.edu

Priyance Mandlewala
Computer Science
North Carolina State University
pjmandle@ncsu.edu

Vishrut Patel
Computer Science
North Carolina State University
vnpatel@ncsu.edu

ABSTRACT

One of the major problems faced when moving to a new location when a person is short of money is finding roommates which align with your requirements. Posting on various social media groups and finding the right person is time consuming and cumbersome. In addition to that, sharing your personal information on social media is highly dangerous. Some aspects overlooked in previous application was security of data, better search algorithm, social media integration and lack of connectivity from the web application itself. The solution to all the above problems are integrating chat functionality in the existing platform which can solve the problem of security of data for contacting user, improving the search criteria and making it more elastic by incorporating weighted attribute search and integrating social media like Facebook to scrap data of groups which have the same intention of finding roommates.

1 INTRODUCTION

There are a lot of challenges associated with development a complete application, as mentioned in the abstract. The software developed in the previous iteration creates a solid base to work upon. That said, the user evaluations made by reviewers clearly show a lot of changes need to be incorporated to make this application 'deployment worthy'.

The current iteration of Roommate Finder aims to eliminate the majority of the problems associated with the previous application by removing the bugs, streamlining the functionalities and incorporating new features.

In this iteration, we wish to incorporate many features but the highlight of the application would be Facebook integration. The best place to obtain users interests is Social Media, especially Facebook. Obtaining data from Facebook would help our application to obtain all the information we would need to suggest and matchup potential roommates.

2 PAST WORK

To ensure the correctness and usability of previous application and what new features needs to be added, we conducted a User Survey targeted at NCSU wolf pack community. We obtained following insights from the response of the survey.

Around 70% respondents felt that the login/signup process of the already developed application was quick and easy.

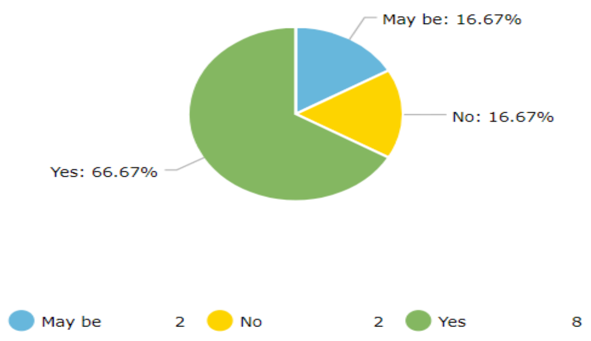


Figure 1: Response visualization of "Was the login/signup process quick and easy"

Around 80% respondents felt that the number of attributes which were used for roommate matching were not sufficient and wanted more attributes.
Around 75% respondents felt that the search process for the roommate was quick but needed improvisation in the search algorithm.

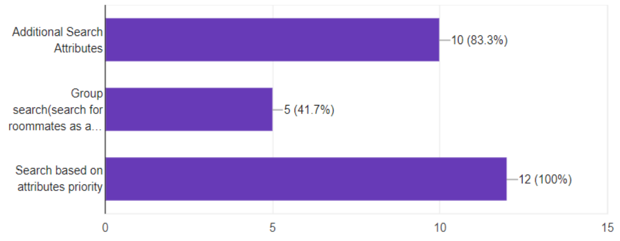


Figure 2: Response visualization of "What would you like us to improve in search process"

Around 84% respondents felt that the search process was not flexible as it was doing one-to-one mapping of attributes for roommate searching and can be made flexible by making searching algorithm elastic with the attributes.

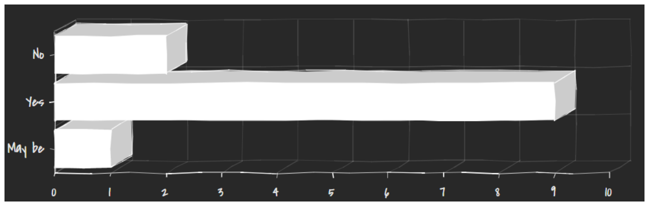


Figure 3: Response visualization of "Was the process to search for a roommate quick?"

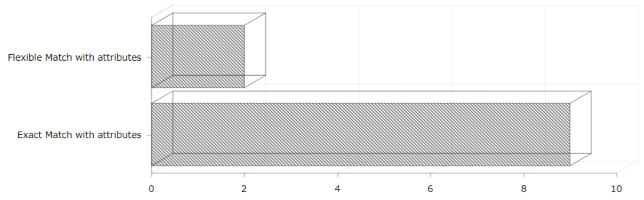


Figure 4: Response visualization of "Was the search doing flexible match or was doing a perfect match with attributes"

100% respondents wanted additional attributes and 84% of respondents wanted search algorithm based on the attributes priority in the new version of the system.

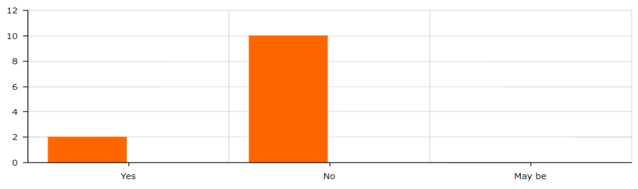


Figure 5: Response visualization of "Were the attributes used for roommate matching sufficient?"

Around 67% of the respondents wanted personal chat between users and 75% respondents wanted to link their social media account to the application, so that they can see the post from other groups related to roommates searching on social media and can contact using new version of application.

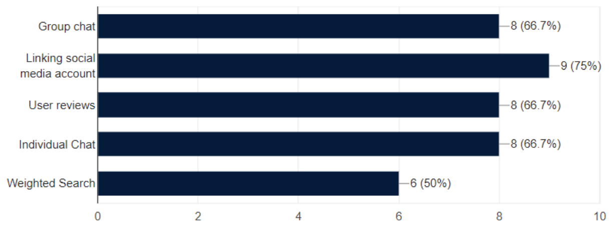


Figure 6: Response visualization of "Which of the following additional features would you want to be incorporated"

In addition to the user survey, we also did some technical survey of the existing system ourselves where we found out that the current application doesn't consider the security aspects of the user's data like email id which were shared with other people to send connect requests. Also, there was no functionality given to users to connect between them without sharing their personal details like email id and phone number which we thought was a major issue with the current system.

With the previous developed system which had a good barebones and some new suggestions from respondents of survey, we thought of moving forward by implementing chat feature within the system, scraping the Facebook media and improving the search functionality by giving more attributes and making already given attributes tertiary from binary.

3. METHODOLOGY

3.1 Project Software Development Model

Agile development was used in developing new version of application. After careful research of each and every software development model available, we thought of using Pair programming as our agile development technique. As all of us were new with the technology stack, there were high chances of causing errors. So, we decided to go with pair programming where we divided functionalities between two groups each of size two.

During all the phases of software development, each developer of the group has to take different role of developer and observer. The task of the developer was to write code and observer helps in finding solution to errors caused or rectify syntactical errors. This way we overcame the problem of making frequent errors and reduced the time for development.

3.2 Architecture

The application has only one type of user who can search for roommates based on his/her priorities of attributes, update its characteristics, change profile and can chat with roommates which he/she finds potential match with. The additional features added can be found within 4 seconds of login to make it easy for users to use.

The additional features were developed based on MVC architecture:

View: The view part of the features were developed using Handlebars.js, HTML, CSS and JavaScript. The view is designed keeping design and user experience principles in mind. The page is requested by the user which is rendered from the server and responses are parsed to JSON to make it more appropriate to display in front-end. Input from users are converted in JSON and dedicated routes in Node.js server handles the server and persist it in database.

Controller: The controller part is developed in Express.js. It parses JSON depending on the type of requests (GET and POST) from the front-end and sends it to the front-end, if it is GET request and persist it to the back-end, if it is a POST request.

Model: The model is developed in MongoDB which is a document-oriented database and has support for JSON-like documents with schemas. Mongoose which is Object Document Modelling layer is used to deal with persisting data to MongoDB. It helps in defining schemas for the data models, so the documents follow some predefined structure and schema.

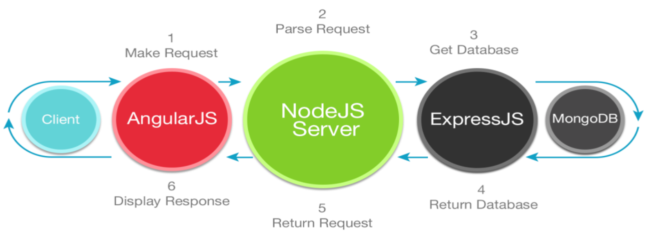


Figure 7: Overall architecture of the development stack[1]

4 IMPLEMENTATION

The application starts with the landing page of the website where the user can see signup and login options. If the user is already registered with the website, the user logs in with the credentials used during signup process and then can see the landing page (post login) with a search button using which a user can search for roommates by selecting various attributes provided.

Based on the attributes selected, the search algorithm search for results from the database and renders it on the page. The search algorithm first takes in the attributes user selected and then finds combination of various queries which are then fired on the database to get various outputs and then those outputs are shuffled which are rendered on the webpage.

The following section elucidates the manner in which we improved the search functionality. To improve search functionality, we incorporated a two-pronged approach. In first we added ‘Flexibility’ option in binary attributes to remove unnecessary rigidness within search parameters and secondarily we moved from previous one-to-one mapping of search result display to percentage based search. In all these two improvements in search helped in increasing the quality and number of search results.

4.1 Adding flexibility to preferences

4.1.1 Limitation of preferences in previous project

The existing Roommate Finder project has laid out a well-defined page for preference settings. In this ‘Set Preferences’ page it takes into consideration monthly budget, location preference, dietary habits covering vegetarian/non-vegetarian option, smoking habits asking whether user is a smoker or non-smoker, alcoholic habits asking whether user is alcoholic or Non-alcoholic, room sharing preferences, earliest moving date and latest moving date.

One of the key issues we found with the existing above-mentioned preference list is that many of the preferences are non-flexible and binary in nature (such as dietary habits, smoking habits and alcoholic habits) which may reduce the scope of potential roommates matching.

Thus, it results in very less number of search results generated.

4.1.2 Changes made by our implementation to overcome binary attributes

Through our survey and in person meeting both during and before taking up this project we found out that at least for some preferences such as dietary habits, smoking habits and alcoholic habits, students (the main user group of this project) as well as people in general are very much flexible.[2]

They sort out some arrangements pertaining the above habits in their own manner. For example, some vegetarian roommates are fine with their other roommate to bring non-vegetarian food to home rather than cooking non-vegetarian food, flexible with smoking outside house rather than smoking in house and similarly having alcohol only on some specific days of week or on certain occasions. Hence, as stated earlier, students are pretty flexible with such requirements.

Thus, we are thinking of providing more options than just binary options such as; ‘Vegetarian’, ‘Non-vegetarian’ and ‘flexible with both’. Providing this option of ‘flexible with both’ would increase the scope of mapping to a great extent and improve the existing system which just works on binary mapping.

We can add other options such as; ‘Vegetarian’, ‘Non-vegetarian’ and ‘Prefer only vegetarian cooking but roommate can bring cooked Non-vegetarian food from outside’ and/or other such similar options. Post final presentation many peers of ours liked this added functionality as it enables their search to be a lot more flexible providing them with variegated search results in comparison to the earlier search results. The added attribute of flexibility option can be seen in image 2.

4.2 Percentage based search

4.2.1 Importance feature of search as developed by previous group

Apart from providing a very trivial search result, which is discussed in below points, the previous group did a commendable job of providing a tag and slidebar based search functionality, having tag and slidebars option to select attributes makes it very easy for the user to select/deselect search attributes and encourages the user to make multiple searches with multiple variations of parameters without filling out some length parameters for or so. Given the above-mentioned usefulness of tags, we maintained the tag based search system.

4.2.2 Limitation of search in previous project

The search of the system is completely depended on the parameters of the system. These parameters are the options the user has to fill up and while searching can specify the parameters the search should take into consideration by selecting the appropriate tags.

The parameters on which the system shows the result of potential roommates are as stated below; monthly budget, location preference, dietary habits, smoking habits, alcoholic habits, room sharing preferences, earliest moving date and latest moving date. And the search method as used by the current application is based on the direct one to one mapping. So, of 8 mapping parameters you only see the result of the potential roommates who match exactly to your 8 parameters. This search logic can be considered as an `select *` statement in sql based query language.

Such kind of direct mapping causes problem because consider a worst-case scenario in which there might be zero search results that maps completely to your 8 parameters but there might be abundant people who might map on maybe 6 or 7 of your parameters.

4.2.3 Changes made by our implementation to improve search logic

To overcome above issue, we implemented percentage based mapping rather than the current one to one mapping. That is, if some user maps to all your parameters than that comes with 100 percent mapping, if they map to 7 of 8 parameters then it results in 88 percent mapping, if they map to 6 of 8 parameters then it results in 75 percent mapping and so on.

Thus, as a part of our recommendation we would be showing the search results in the decrease order of mapping percentage as per the scheme mentioned above.

This logic was developed using npm package of the JavaScript library. Then when a user selects or enters his/her search parameters we develop different combinations of it. We query the database for search result for each of the combination and then display the search results in decreasing order of number of parameters mapped.

To give a clear understanding below is a small example explaining above logic. If user selects three parameters of ‘Housing at Colonial Arms’, ‘Vegetarian’, ‘Non-Alcoholic’ the at first we check for users that map completely to the above three parameters (3/3 i.e. 100 percent mapping) and store them in a buffer. Then we check for users that map to two of the above parameters which results in combinations of ``‘Housing at Colonial Arms’, ‘Vegetarian’`, ``‘Vegetarian’, ‘Non-Alcoholic’` and ``‘Housing at Colonial Arms’, ‘Non-Alcoholic’` (2/3 i.e. 66 percent mapping) and trim the users that are already present in the buffer resulted from above query and at last we check for users that map to one of the above parameters which results in combinations of ``‘Housing at Colonial Arms’`, ``‘Vegetarian’`, ``‘Non-Alcoholic’` (1/3 i.e. 33 percent mapping) and trim the users that are already present in the buffer resulted from above two queries and then display all the result in the above mentioned decreasing order.

This is a logic with which we have modified the existing percentage based search result generation.

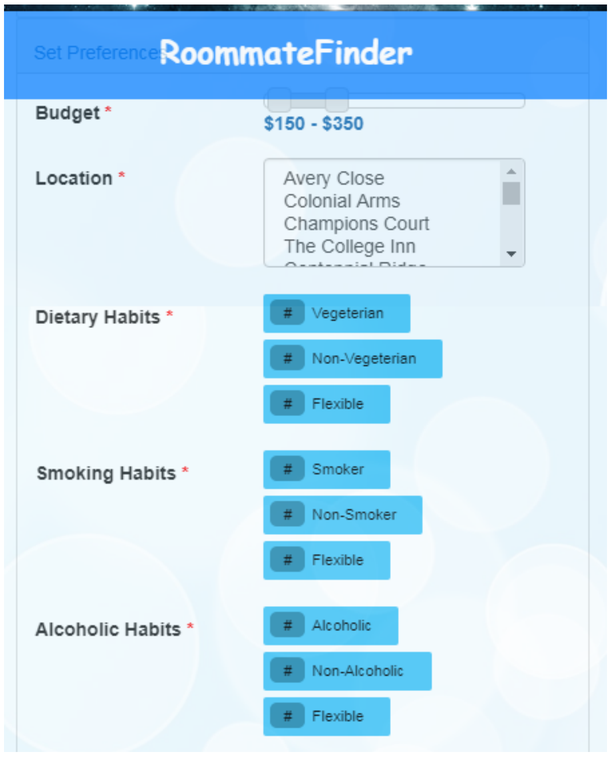
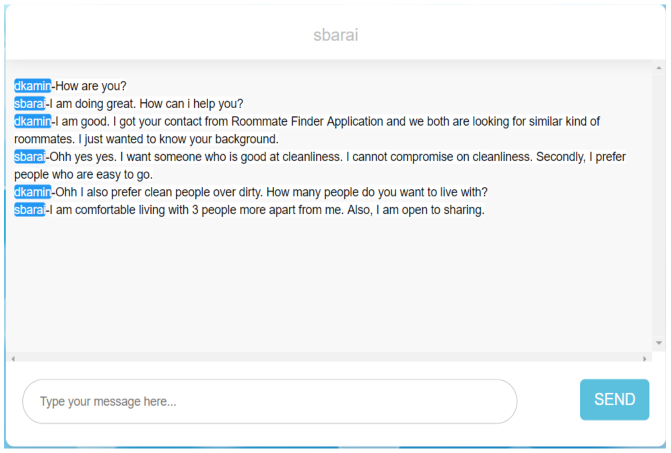


Figure 8: Improved Search Features.

4.3 Chat Application

The application starts with the landing page of the website where the user can see signup and login options. If the user is already registered with the website, the user logs in with the credentials used during signup process and then can see the landing page (post login) with a search button using which a user can search for roommates by selecting various attributes provided. Based on the attributes selected, the search algorithm search for results from the database and renders it on the page. The user can then click on user’s profile which he/she thinks is a potential match and can decide to chat without sharing his/her personal details.



dkamin-How are you?
sbarai-I am doing great. How can i help you?
dkamin-I am good. I got your contact from Roommate Finder Application and we both are looking for similar kind of roommates. I just wanted to know your background.
sbarai-Ohh yes yes. I want someone who is good at cleanliness. I cannot compromise on cleanliness. Secondly, I prefer people who are easy to go.
dkamin-Ohh I also prefer clean people over dirty. How many people do you want to live with?
sbarai-I am comfortable living with 3 people more apart from me. Also, I am open to sharing.

Figure 9: New chat feature.

4.4 Installation Instructions

This section contains information regarding how to set up and run this project. Please follow following steps to setup the project.

1. Install NodeJS
2. Install latest version of NodeJS (8.10.0). You can get installable for your host operating system on this link.
3. Install MongoDB
4. Install latest version of MongoDB enterprise edition (3.6). You can get documentation and resources required for MongoDB installation here. This guide contains information regarding MongoDB installation on all platforms.
5. Clone Repo
6. Clone or fork our project repo on your local machine.
7. NPM Install
8. Run npm install command from the directory where your clones our application. This will install all the required modules.
9. Start MongoDB Service
10. Start MongoDB service on your local machine or on a hosted environment. Steps to start MongoDB service is given in above installation guide.
11. Change MongoDB Connection Details
12. In app.js file change the mongodb connection to connect to your mongodb service instance.
13. Change other parameters.
14. In app.js file change parameters, such as PORT parameter to start application on your desired port. In /routes/index.jsfilr change SMTP server connection details with your credentials.
15. Start Web Application.
16. Run node app.js command to start the web application.

5 USER EVALUATIONS

As discussed above, we started evaluation with the aim to test the functionalities on two aspects

1) The functionality does what it is meant for
2) Time a functionality takes to respond with correct output.

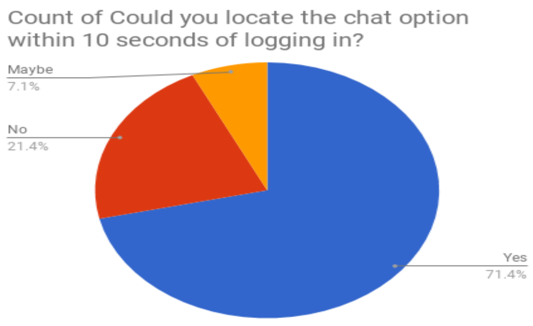


Figure 10: Ease of locating chat option functionality

The question posed in figure was meant for us to know that whether the newly added functionality was easier for the users to find or not, based on response 2 things could have been done

1) Provision of user guide
2) Make it easy to locate

As the we can see from the results in Figure 10 about 72% of the people could locate it in 10 seconds of logging in to the system, about 7.1 % evaluators told that they might be able to locate given the context of searching for the function.

This results indirectly informed us that the users were easily able to locate the functionality and hence we didn't need to software assisted provide user guide for it.

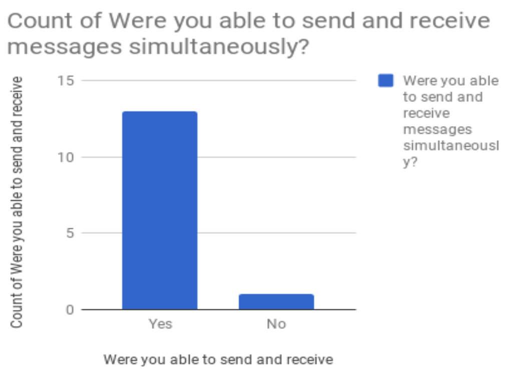


Figure 11: Working of Core functionality of feature

The question posed in Figure 11 targeted the core functioning of the functionality i.e. were we able to provide what we suggested that is sending and receiving messages in real-time.

The result would allow us to infer 2 things

1) Functionality was working as required
2) If it wasn't working then we would have to find out why and act appropriately.

As the result in Figure 11 show that almost 13 out of 14 evaluators were able to chat (i.e. send and receive messages) through the application.

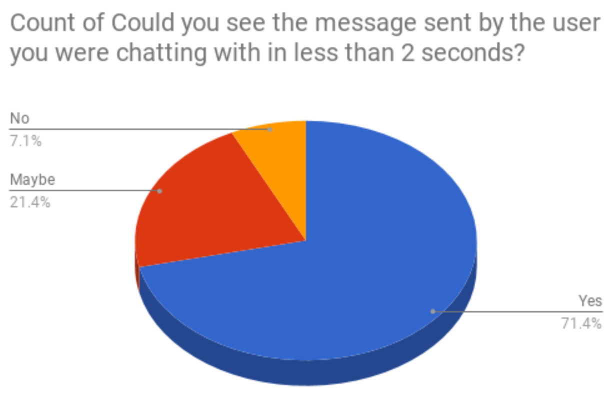


Figure 12: Message Delivery time logging stress test

The question posed in Figure 12 was intended to find that the message delivery was efficient. Maybe option here meant that they could see the messages sent but not within two seconds but also not after a long time, it could have been better given certain conditions changed.

As the results in Figure 12 suggest almost 72% of the people were able to see the delivered message on their chat box with 2 seconds of sending, about 22% suggested the message was delivered in short time but not within 2 seconds.

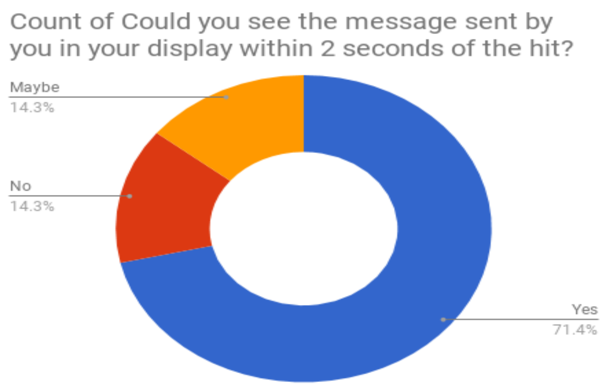


Figure 13: Message receiving stress test through time logging

The question posed in Figure 13 was intended to find that the message display after receiving was efficient.

Maybe option here meant that they could see the messages received but not within two seconds but also not after a long time, if required it could have been improved by not updating the entire chat box rather just updating a section of it and appending a new message.

As the results in Figure 13 suggest almost 71% of the people were able to see the delivered message on their chat box with 2 seconds of sending, about 15% suggested the message was received in short time but not within 2 seconds and for about 15% of the evaluators the message receipt took a lot more time than expected. The results suggest that the message rendering could be improved by enhancing the way messages are handled in User Interface or by notifying specific user threads about message update so that rendering can start early.

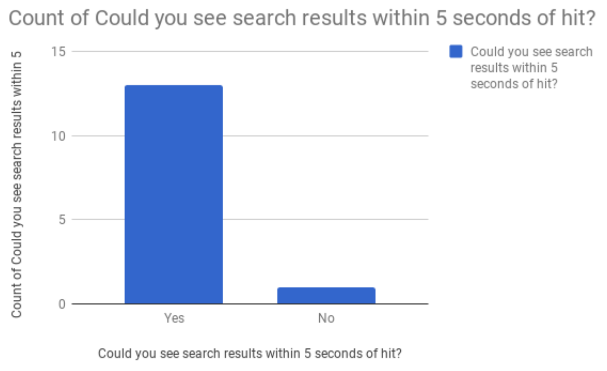


Figure 14: Search feature response time test

The question posed to users in Figure 14 was intended to find whether the search result display was efficient and the amount of time it took was bearable or not.

The result could have help us analyze how usable the current search functionality is and what we need to improve upon in terms of efficiency.

The results in Figure 14 show that 13 of 14 evaluators were able to get the search results within 5 seconds of hit, which implies it was efficient enough for current use.

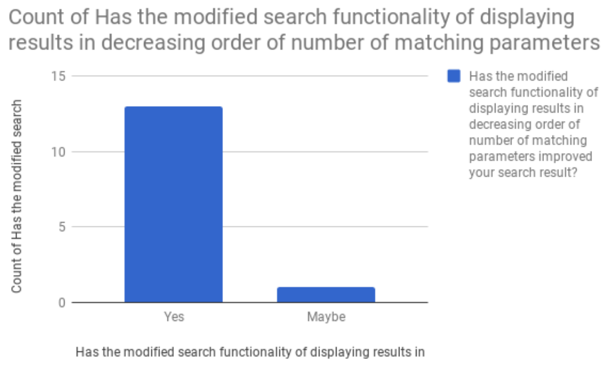


Figure 15: Search result usability test

The question in Figure 15 was posed to evaluators to ensure that the functionality that we were aiming to import in existing system what able to do what was intended to do i.e. check the correctness of the system.

The result in Figure 15 indicate that the functionality was correctly implemented which was reflected by almost 90% of evaluators suggesting it, 1 of evaluators responded with maybe which on further inquiry we found that the search parameters the users had selected didn't need have any data which could be displayed through decreasing parameter matching.

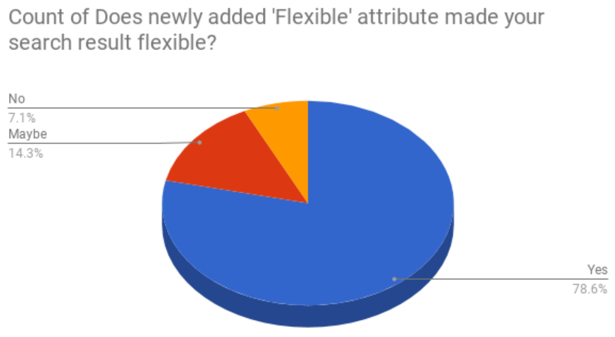


Figure 16: Functionality operation Check

The question in Figure 16 was posed to evaluators to ensure that the functionality that we were aiming to import in existing system what able to do what was intended to do i.e. check the correctness of the system.

The results in Figure 16 indicate that almost 80% of the evaluators were happy with what the result of flexibility gave i.e. the search was indeed flexible enough a displayed result appropriately. To conclude based on user evaluation the overall system developed was quite efficient and did what it was supposed to do. Thus, our aim of developing a usable system was achieved.

6 CONCLUSIONS

Based on the user survey, it can be inferred that the current application requires more functionalities like integration of social media account, weighted search for flexibility with attributes, chat feature and user reviews which helps users make a decision whether to stay with the person or not. Thus, from the user survey of previous application by 12 students, this additional feature can add a big enhancement to the current version of the system and can attract many users. Thus, after adding this features to the current version will able to help NCSU community find best roommates of whom they can be proud of.

REFERENCES

[1] https://www.dealfuel.com/seller/mean-stack-tutorial/
[2] https://www.niu.edu/mptss/_pdf/roommate-workshop.pdf

Chit Numbers:

1. QSY
2. YTO
3. DGV
4. JFA
5. SBF
6. YIA
7. UNE
8. PKA
9. SUH
10. VEU
11. RFZ
12. IQU
13. SNX
14. KSY