TeamFinder: An Application for Building Skill Based Teams

Michael Goff*, Shashank Jha†, Jingjuan Deng‡ and Bhaskar Sinha§
Department of Computer Science, North Carolina State University,
Raleigh, North Carolina 27606
Email: *magoff2@ncsu.edu
†sjha5@ncsu.edu
‡jdeng8@ncsu.edu
§bsinha@ncsu.edu

I. Abstract

Every now and then students from every discipline have to team up for group projects. The success of the projects could depend on various factors such as the skills of the students partaking, social compatibility, matching schedule, interests, past experience, etc. Through our project we have aimed to facilitate the process of team division based on skills or expertise in different facets of software development such as front-end, back-end and database. For the same, we have created three different applications using Google's Firebase and Angular2. We then conducted experiments to compare all three approaches and asked users for feedback for further improvements. In the end, we found that no approach really stood out from the rest as users preferred differnt approaches for different reasons. Some user's preferred to be assigned into a random group while others liked choosing their own members. The best approach moving forward would be to combine the aspects of each application into one.

II. Keywords

Team division, balanced teams, joining team, creating team

III. Introduction

Out of all the factors which contributes to a team successfully working together, we decided to concentrate on skill-set division. After an initial feasibility study, we collected data that indicated a more intelligent way to approach project team creation was desired among graduate students. Our first approach was to allow users to search for other users with a particular skill set which they require in their project. The second approach was intended for the professor, where they already have a list of users and their expertise areas and with one press of button, they can create multiple teams with balanced expertise. The third approach was allowing user to join only those teams which has positions matching their expertise.

In this paper, we discuss the experiments which we did as part of evaluation and the responses of the participants. After that, we analyze the responses to all three approaches and determine improvements which we can make in them based on user feedback. Finally, we conclude which one is the best approach among three and what can be done to take the project further in future.

IV. Experiment

This section will outline the parts of the experiment we conducted with our participants during individual sessions held at Hunt Library.

A. Task Methodology

For each approach, we designed scenarios for users to use the application. Approach 1 and 3 are student based applications, we evaluate them by the user's experience.

Evaluation for approach 1 was pretty straight forward. Users were asked to make accounts on the application and complete their profile by adding their skills. Then the users were asked to go on to the search page of the application and conduct searches on fellow students based on their skills. Thereafter, the evaluators were asked questions in the form of a survey for this approach to evaluate the user experience, learnability, understandability, usability and the human computer aspect of the application.

In order to prove the value of automated group sorting in approach 2 we created a list of 20 randomly generated users with varying expertise in front-end, back-end and database development. We gave each participant a blank sheet of paper and asked them to sort the users into balanced groups based off of the available skills. We intentionally neglected to provide any further direction so we could see what the thought process would be as they wrote the teams on the paper. We also tracked the time it took to complete the formation of teams.

After having the participants manually sort through the table of users we had them log into an account on the team generation web-page and allowed them to generate the teams automatically using the same set of users that were also inserted into the database. We noted the time from logging in to arriving at the generated teams page.

Following the experiment for the second approach we had the participant fill out a Google form with some survey questions to get a feel of their impressions. We asked "Is it clear to you what approach 2 does and its purpose?" in order to confirm that users understood the point of the application. We asked "Is it easy to perform the basic functions of approach 2" and "If you had to use approach 2 again would you need assistance or support?" to make sure that the usability was good for the second approach. We asked if the user felt that

approach 2 made it easier to create teams and to rate the overall experience to make sure that the application was useful to the user. Finally we asked what improvements they suggested for the application. We made this an open ended question so we could get broad feedback about the application.

In approach 3, we created a list of teams, in which there are positions with required skills. Some positions are already filled and not available to other users. After users sign up to the application, they can edit their profiles and skills. When users want to join a team, they go to the teams page, which shows the basic condition of existing teams. Then users can choose from available positions according to their skills set in the profile. They can only apply for positions which are available and fitting for their skills since the applied button will not show up if a user does not meet the requirements. After joining a team, users can go to the team information page to see who their team members are and their contact information. If users change their minds, they go to the teams page, quit from the current team and join another team.

After showing users the basic operations, we asked them to try the application then fill up our questionnaire. The questionnaire evaluates whether the application achieves its purpose, its usability and the overall user experience. Also we compare the three approaches to find their pros and cons, to figure out which approach is the best.

B. Participants

We chose participants who were graduate students at North Carolina State University from the Software Engineering class. All participants came to in-person sessions where we walked them through the applications in order to assess their feedback in person.

Before starting the demos, we wanted to collect background information about the participants. We asked them if they thought skill was an important factor in deciding a team's performance to which 96% of respondents responded yes. The point of this question was to make sure that these participants shared our view that skills were one of the most important factors in developing a strong team since we based our entire applications around this idea.

We also asked the participants if they had the ability to choose team members based off of their skills if it would be helpful to them. We found that 100% of respondents responded yes to this question, proving that skills are certainly an important factor in creating a good team.

V. Results

Approach one received massively positive response in every category of questions with almost all the evaluators endorsing the applications learn-ability, understandability and lauding the user experience. During the design phase one problem we thought about in approach one was that, if users were ready or not to contact the students found on the search page, as in, were they ready to put in such effort. To our surprise, all the evaluators responded to the above question with an affirmative, which strengthens the belief that people are ready to contact and maybe team up with people who they find on the application.

Although the response was mostly positive, a fifth of the evaluators felt that the application should also include a measure of a users expertise for a skill which could be represented by a rating of that skill. The evaluators also, raised the concern that they had no definitive way of knowing how good the person was in the skill which was mentioned by him. Similarly, some evaluators raised the concern that those people who have already teamed up with other people should not be shown the search page so that the users don't undergo the hassle of contacting a person who has already teamed up with someone else.

Other suggestions made for the application by the evaluators were, including the LinkedIn and GitHub profile of the user, adding more information about the people such as their past experiences, interests, schedules, number of hours they can dedicate during a week, and adding an instructions page for the web site. One evaluator also suggested that the search should also be allowed by using a persons name and if an android or iOS app was present that would make the application more handy to use.

In approach 2 the results of the timing activity were pretty much as expected. Out of 25 participants the time to sort the 20 users manually into balanced groups based off of their skills took an average of 3 minutes and 55 seconds. When logging into the application and selecting generate groups the participants took an average of 12.92 seconds to complete the activity. So it is significantly faster to let the computer do the sorting for you to a factor of about 18 times faster.

We were surprised to see that almost all of the manual responses had very similar group setups when the activity was complete. We think that it might have something to do with just going in order down the list filling in each group as you go. Unsurprisingly, this was very similar to the group structure the application took as well so the performance of each was comparable. The time difference is what really stands out in automating team creation.

The results for the survey were positive for the second approach as well. When asked whether the application's purpose was clear, 100% of respondents answered yes. All of the respondents also said yes to the question of whether it was was easy to perform the basic functions of the application. However, only 84% of respondents said they would not need assistance to use the application again. Fortunately, 100% of people agreed that the application made it easier to form balanced teams, which was our intent with the application. Finally, the user experience looks good, on a scale of 1 to 5 for user experience, 84% of users rated it as a 5 and the remainder rated it as a 4 on user experience.

Users' experience of approach 3 is good. All of the participants think this approach achieves the purpose of creating teams. And most of them(88%) think it can create skill-balanced teams. 76% of our users need no help and assistance in the future use(Fig.1); however, there are still 24% users that think this application is hard to use. Among 25 participants, 60% give full marks(5) on user experience, 24% rate it 4 and the left 16% rate it 3.(Fig.2)

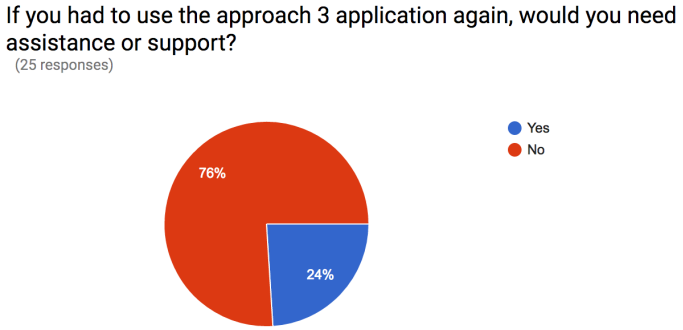

Fig. 1: Need assistance to use approach 3

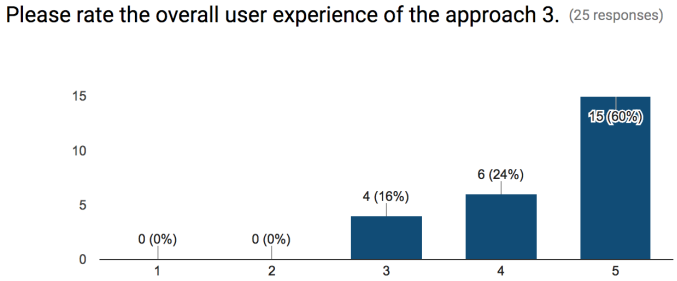

Fig. 2: The overall user experience of approach 3

We asked users questions about which features they liked in approach 3 (multiple choices). The result can be seen in Fig.3. 'Joining teams freely' and 'A glance on teams information' were the two most popular answers. Among all participants, over 80%(20 of 25 participants) of them like these two features. The third popular answer is 'Offering contact information of team members' with 60% users choosing this feature. 'Quitting from a team' and 'Competing for positions in time' are the least two popular features, there are over 50% users who liked these features.

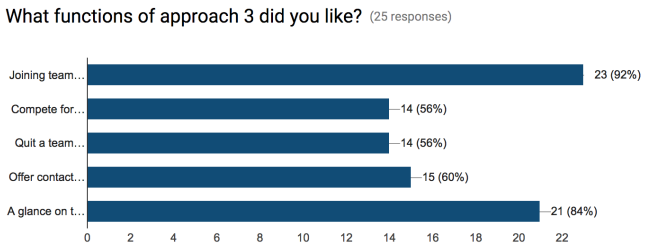

Fig. 3: Good features of approach 3

VI. Discussion

After conducting user evaluations and collecting results we were able to arrive at a plethora of conclusions about our applications. The most obvious being that we have plenty of room for improvement on the layout and flow of our applications. As we were working on each application separately, we all took different design paths which lead to some confusion among participants on how to use the various features. We were already aware of some smaller issues, since we only had a limited development time and a workable prototype was okay, but introducing unbiased users into the system revealed a lot more issues than what was on the surface.

Within this section, we will discuss the key points about each approach especially the strengths and weaknesses of each one. In order to draw these conclusions, we will synthesis the data from the survey with the primary focus on the short answer responses, since that is where the respondents were allowed the most freedom in their opinions.

A. Observations / Limitations

B. Approach 1: Skill based searching

Approach 1 was a student faced approach. Through this approach, the students had to create their profiles which had their skills such as Java, C++, etc. Thereby, other students of the class could search for the students in the database by using skills as keywords. They could make a search for single or multiple skills and the application returns a list of users which match the skills given. The list is represented in a descending fashion with the student who matched the maximum number of skills is at the top.

Suggestions made by users which are explained in the Results section are mostly achieve-able and would help the further growth of the application. The ratings for each skill can easily be included in the system. If included, these ratings for each skill will be provided by the user, but even after adding this the question of authenticity of this information arises. There is no definitive way of knowing if the information provided by the user is true or not. Apart from the possibility that the user puts in false information in the system deliberately, it can also happen that the user is unaware of their level in a particular skill and unknowingly puts in false information. A possible solution to this problem could be to include the LinkedIn profile of the user to the application can evaluate how many people have endorsed a particular skill of the user. But then again, even that is not a definitive way of examining the level of a person in a certain skill. Perhaps, it is still better to add the skill rating portion on the website as the user searching for a team mate can then choose between multiple people who claim to know a certain skill based on the skill rating they have provided.

Another suggestion made was to remove people from the search page as soon as they have teamed up with people. This can be easily included in the current code base and caters a very eminent need as if you look from the perspective of a user searching for a team mate, if they find a potential candidate only to know that they have already teamed up, would be a futile effort. By including this scenario, only people who are still looking for team mates would be shown and as soon as they find out team mates they would be removed from the search list.

Other functionality was suggested such as the number of hours a person can dedicate in the project per week, previous projects, GitHub link, LinkedIn link, interests, etc. can easily be included in the software. If the user has more information, they can make a better choice. This could certainly be done.

In future an Android and iOS app can also be made for this application, but we still feel that the web application would be a better platform for this application. Consider this use case, a user makes a search for a potential team mate on the application and now wants to see the the Github or Linkedin profile of the user. As soon as the user clicks on the respective link he is guided to their respective URL's and now he might have to log in to LinkeIn which adds an extra overhead (if it was a laptop, people are generally already logged into their accounts such as linkedin and github), moreover if has opened the github page and wants to open a repository, it would be very difficult for him to check out the code on a mobile phone. Furthermore, a user would use this type of application only twice or thrice in a semester so it does not make any sense that the user would keep this application downloaded as it is just using space and not being used, hence web platform would be better and what can be done is that made sure that the application is mobile and tablet friendly just in case if the user still wants to use such platforms.

Overall, a third of the evaluators were in favor of using approach 1 as an application which could facilitate team making as it was pretty straight forward to use and it gave more details about a persons skills unlike approach 2 and 3 which gave an high level perception of a persons skill. Also, the evaluators who voted approach 1 as the best one said that because they felt that knowing precisely what skills a person processes will help them create and balanced team as they would know beforehand which team member is bringing in what skill to the table. So, for people who feel the need of adding a team member to their team by gauging the skill set of that person, approach one is the way to go.

C. Approach 2: Automated Sorting

Approach 2 was the most hands-off approach for the user since the machine does most of the work on the user's behalf. This made it more difficult to give a traditional tour of the application since there was pretty much only one button for the user to press when acting as the role of the professor. However, the results were exactly what was expected. The application successfully created balanced teams at the push of a button that were on par with the teams that were formed manually by survey participants and in a fraction of the time.

When we asked the users about what improvements could be made to the second approach we got a lot of great responses back. We were pleased to find out that most of the responses were compliments including that it was perfect and that it was well developed, both of which are great signs for the usability of the application. In addition to praise, a lot of the responses were smaller nitpicks about the application itself, like to add the names to the sorted teams page, a small detail we forgot to add in implementation. Other suggestions included, adding a scale of proficiency for the skill categories or even the introduction of a Testing category for required skills in a team. Both of these suggestions are great recommendations to improve the quality of the application that were not even considered at all during development. When you introduce weighted proficiencies in multiple skills you can allow the system to more intelligently place you into a group under a skill that you are stronger with rather than one that you have just learned. It also avoids the binary approach to either knowing something or not since that can vary based on the subject's own opinion of their skills. The Testing category is another great addition since it is another broad category and does not fit well underneath any of the other existing skills.

A lot of participants suggested to break the categories down into a more granular approach of skills similar to approach 1. While having more information about each user is generally a good thing we think that keep the categories broad is more appropriate for a project environment. It would be much harder to sort groups based off of a wide skill pool since it would be nearly impossible to create balanced groups. At least within each broad skill category the thought process would translate easier across arbitrary languages. Since the languages are not defined for projects the idea of transferable skills were more preferred. School always preaches that you shouldn't learn the language but rather the concept.

One final preference was to add the ability for the professor to tweak the results. We thought this was a great idea because the professor will always know the class better than the computer does and will already have some idea of how well certain students may work together. In order to implement this, we should let each row in the team table be draggable which will allow for a professor to slide a student from one group to another however they want.

Overall, the second approach was pretty popular among survey respondents, there were a number of great suggestions that were not previously considered which is great insight for future development on these applications. When we asked users for their reasoning behind picking their favorite application, the respondents had a lot of good insight about approach 2. The most popular opinions were that it was fast and random which allowed busy students to quickly find teams that were automatically balanced. So when students don't mind meeting new people and quickly getting into a team without any extra work on their part, the automated team creation application definitely shines.

D. Approach 3: Joining existing teams

Approach 3 is similar to the course selection process in college. Users must satisfy the skill requirement of specific position. The positions in a team is limited, so if one team is full, users have to choose other teams.

Since the skills and positions in each team are fixed, this approach is designed to guarantee creating skill-balanced teams, which is the primary concern of our project. But few of the users think it is not easy to create skill-balanced teams. We want to figure out the reason. For approach 3 evaluation, we focused on the user experience, how forming a team affects user experience, and what changes we can make to improve it.

According to the survey, we found that for the users who choose approach 3, most of them give two main reasons. One is this approach offers information about the overall teams condition and team mates' contact information. In order to create user-satisfied teams, users need information about the current teams formation, how many positions are left, what positions they can compete for, and members' details in each team. This information helps them to make better decisions when joining or quitting a team.

Although we offer useful information about the team formation and position availabilities before users select their teams, the approach lacks information about the details of team members in each team. Some users think it's better to have this information when making team joining decisions. This is one drawback of approach 3, while approach 1 provides detailed individual information. A better way to solve this is combining the advantages of approach 1 and 3.

Another advantage of approach 3 is that they can join a team or quit from a team whenever they want. Compared to approach 2, approach 3 gives users freedom when selecting teams or changing their minds. However, some users think this freedom might cause some problems. Imagine that if users keep changing their minds, teams will never settle down. One improvement on this problem is setting some constraints on changing team actions. For an example, adding time limit on joining and quiting teams. In the time limit, user can compare teams and choose which team they fit the most. When it exceeds time limit, users can not change their team status anymore, which keeps the teams formation more stable.

Due to trading off between time and implementation, approach 3 has some drawbacks. Several users asked for improvements on UI in approach 3. One question in our questionnaire is "If you had to use the approach 3 application again, would you need assistance or support?"(Fig.1). About 24% of participants required assistance in the future use. We realized that non-functional requirements like interface design and usability are also important factors affecting user experience.

	pros	cons
1	clear and concise	UI unfriendly
2	overall team information	lack of detailed information
3	join or quit freely	team breaks easily

TABLE I: Pros and cons of approach 3

In all, approach 3 gets good feedback in evaluation because it offers convenience, free choices and some useful information for users to form teams. About one of third users prefer to use approach 3 to solve the team formation problem. The primary drawbacks which affect users' experience are UI, constraints around quitting a team, and the lack of teammates' information before selecting a team.

E. Best Approach

To understand which approach users liked the most, we asked two questions. In first question, we asked which idea do they like the most. Here, searching for team members came out at top with 44%. Randomly assigned team was least favorite with only 24% users choosing that. These were what we did expect.

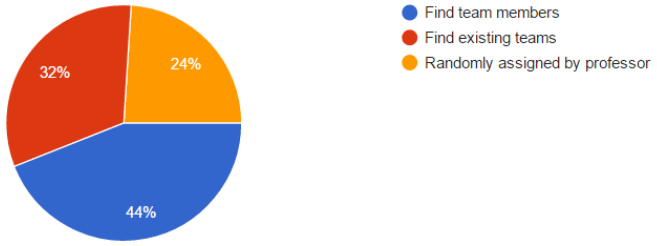

Fig. 4: Which method would you prefer to build a team?

However, when we asked which approach did they like the most, result were evenly divided. Since we had three different and independent approaches, we were expecting one to be better than the others. Also, as our evaluators were fellow Software Engineering students, we expected approach 1 or approach 3 to be the winner. Both these approaches were student based and approach 2 was professor based. We thought that approach 2 would be favored only by professors who like dividing the team randomly. Surprisingly, all three approaches got similar amount of votes as seen in fig. 5. This further reiterates our belief that if skills are divided evenly among teams, students are willing to work with new people as well.

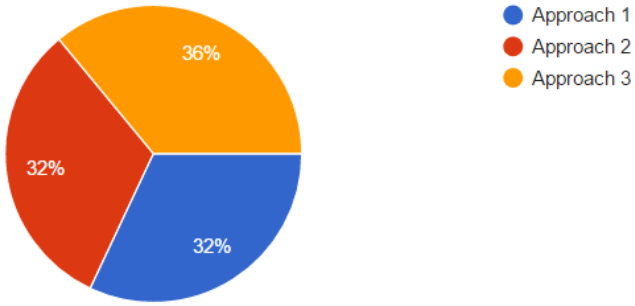

Fig. 5: Which approach do you think was the best?

VII. Future Work

Since we did not have any clear winner among three approaches, we believe that best way forward would be to combine all three approaches into single application. All three approaches already have a common starting point, i.e., signing up and adding your skill sets. We could also ask users to provide their LinkedIn profile. The next step would be to ask if the user wants to add new project or check existing ones. If he decides to create new project, he can list that as open or closed. If it is a closed project, then he needs to contact other people for forming the team (approach 1). An open project would mean that other users can see that project and apply if they have desired skills (approach 3). This would be same as the user selecting 'check existing projects' and then joining team. For approach 2, the user needs to join a list shared by the professor. The professor can then divide the list into further teams after a deadline (approach 2). Having roles defined will

make sure that only the professor is able to generate teams. Currently, all the users have the ability to generate teams. So one single application can cover all the approaches.

Though our application is targeted for students and professors, it can easily be extended to cover professional employees and their managers as well. Approach 3 would be ideal for crowd sourcing. The user would create a project and others would be able to join the project if they have the desired skills. Since approach 3 does not give the project owner any control of who joins the team, it would be best suited for crowd sourcing.

One of the things which we learned during the evaluation process was that our presence in the same room while evaluators wrote responses made them lean towards more positive feedback than we would have otherwise gotten. The mostly positive feedback we received on our applications was reassuring, but we were deprived of honest feedback which we could have used for improving our product. Hence, in future, we will work more on getting honest feedback as well.

VIII. Conclusion

All three of our approaches were different and each of them assisted users to form better teams on their own. While approach 1 allowed the group lead to select their own members, approach 3 gave the users an option to join an existing team. Approach 2 would be ideal for professors who want students to have the experience of working with random people. Though students are not appreciative of this idea, this is what happens in post-school world or in offices. Since our target was to build a working and presentable prototype, we did achieve that target. But there is still lot of room left for improvement.

Chits

bjksauio
blhhieii
cfmqoaee
cgdteiai
chmyaoao
cjjpouue
ckmfuoii
clmbaaoe
cmmdoiue
dcbviuai
dclxuauo
ddmwauii
dfhcuoea
dgjdeaiu
dkjpeiao
dmbwiiie
dmhcieao
fbkbeiou
fcjcoeao
fdffoaia
ffhxoeou
fljnueii
fmhzoaio
gbkdoiee
gcgliauu

Total: 25