



MENTOR-MENTEE MATCHING SYSTEM

MENTOR-MENTEE MATCHING SYSTEM

FINAL PROPOSAL REPORT

Luisa Rojas-Garcia (100518772)

Alexandar Mihaylov (100536396)

Danilo Carvalho Grael (100662667)

Guilherme Fetter Damasio (100571571)

1.Introduction

The University of Ontario Institute of Technology (UOIT) strives to provide a rich and pleasant learning environment for all students throughout their university career. This is a challenge especially for new students who are transitioning from secondary school into university. One existing remedy available to these students is the mentor-mentee program, which provides support for these students from like-minded mentors throughout the academic year.

The UOIT mentor-mentee program is primarily comprised of new students, mentees, whom are matched with a mentor tasked with providing continual support and guidance to all mentees in their group. Mentors are also responsible for providing recurring logs and reporting any issues they encounter with their mentees, which are then reviewed by faculty staff.

Previously, the student matching and mentor log reviews were done manually by UOIT faculty staff, which was effective given the 125 participants that partook in the mentor-mentee program in previous years. As of recently however, the program has grown to include every first-year student, which brings the participant count to roughly 1,250 students. This influx in participants has raised some problems in the currently existing methods of managing the mentor-mentee program. The matching procedure has become more difficult since all students in a faculty must be considered when matching new students to a particular mentor. This process is also very time consuming, which hinders the quality of the matches, and ultimately doesn't provide the best experience for new students. In addition, mentors' logs and any issues they may bring forth are much more difficult to address since they must all be reviewed manually.

As a result, the current manual process is not sufficient in managing the growing mentor-mentee program at UOIT. This work argues that the key problem in the current process is the considerable amount of manual work involved. To alleviate this, a more autonomous process must be put in place, thus providing the staff with more time to deal with pressing concerns that require human assistance. In this work, we propose a fully equipped software system, tasked with managing the mentor-mentee program in the most automated way possible, whilst maintaining the quality of the participant experience. Additionally, the ease of management allows for fewer issues that could otherwise arise due to human error. The system allows for a fluid and streamline stepwise process which gives room for even further growth of the program.

2. Background Research

One of the biggest challenges the mentor-mentee staff faces lies in the matching process of the students. As such, a large portion of our resources was dedicated to finding an efficient and effective method to automatically conduct the matches. After some preliminary research, we narrowed down the potential methods to one of the following: expert network team formation, genetic algorithms, K-means clustering, Mean-shift clustering, and hierarchical clustering. We will briefly talk about what each method entails, and why it was or was not chosen.

2.1 Expert Network Team Formation

Expert networks are defined as social network of individuals who possess some specific set of skills. These networks are typically represented as graphs, with the experts represented as the nodes, and the relationships between the experts as the edges between the nodes. The team formation problem is defined as follows; given a task, pool of experts, and the relationships between them, how do you find the optimal individuals to perform that task? The first known work to tackle the problem in the presence of social network of individuals was published in 2009 [1]. This work though somewhat related, treated all individuals equally in that it wasn't able to distinguish between what would be the mentee and mentors in the problem at hand.

Upon doing further research, we found work done that tackled the same team formation problem, but with the ability of appointing a leader of the group [2]. This work seemed very parallel to our problem, whereby the leader would be the mentor, and the experts in the team, the mentees. The work utilizes what they refer to as the Leader Distance communication cost function which calculates shortest distances between experts, skills, and the leaders. The communication cost function uses the similarity of expert's skills to match them, which in the mentor/mentee domain would be analogous to the personal qualities that students provide upon registering for the program. This method seemed very promising due to overlapping similarities between the two problems.

When attempting to actually apply the work in [2], we soon realized there is one crucial element lacking from the mentor-mentee problem, thus rendering this approach much more difficult. The team formation problem assumes there to be an existing social network, with previous relationships amongst the individuals. This network is, however, non-existent in the mentor-mentee problem, since no previous relationships exist between new first year students, and their mentors to be. A naive workaround would be to create a fictitious social network, but this did not seem like a practical solution. As such, we decided not to use this method, but to instead, keep it as a potential future work in

dealing with continued engagement of mentors/mentees since they would have existing relationships after participating in the program for a year.

2.2 Genetic Algorithms

Che Ani et al. [3] proposed a method for group formation using genetic algorithms, where the members of each group are selected based on their programming knowledge and skill. This work focuses on the formation of groups with balanced programming capabilities in order to successfully complete software projects.

First, the initial population with which the algorithm will start running needs to be created. Every individual that makes this population is referred to as a chromosome, and it represents a possible solution to the matchmaking problem. After a series of modifications, the expectation is that the overall population is better than the one before it, but also closer to an ideal solution.

Each combination's closeness to the optimal solution is calculated using a fitness function, which takes into account each of the students' programming skills. Their programming knowledge is based on their final exam score for a programming course, which can fall under one of three classes: good, average, or poor.

Next, a series of alterations are done to the initial population in order to search for a solution. In order to do this, first, two chromosomes are selected from the current population and their information is mixed using the "one-point" method in order to produce new offspring; this operation is called crossover. Additionally, and in order to maintain diversity and avoid a local minimum within the population, the offspring undergoes a mutation process as well. During this step, a randomness element is introduced by changing an arbitrary bit in the chromosome's sequence.

This process, including fitness calculation, crossover, and mutation, is repeated until the optimal solution is found, or until the set stopping condition is met. The results showed that a genetic algorithm approach is a good optimizing method for the group formation problem, since the method used was capable of producing different balanced groups. However, this potential solution was also quite slow, making it very difficult to scale. For this reason, we did not proceed with this method for the matching problem at hand.

2.3 Clustering

Since the main goal is grouping, the best group of algorithms to solve this is clustering. Clustering is an unsupervised machine learning algorithm that takes feature

data and splits this data into groups (clusters) following some kind of criteria. For that, we considered three main clustering algorithms in our initial research:

2.3.1 K-Means Clustering

The k-means clustering algorithm [4] is the simplest one; it groups the samples at hand into clusters, such that the sum of squares within the cluster is minimized. In other words, such that the Euclidean distance between each sample in the cluster the minimum possible; in fact, it is called k-means because it is based on the mean for each cluster. First, it defines the centroid¹ for each cluster, then groups the samples into their closest cluster. Then, it calculates the metric and re-evaluates. This process is repeated until a reasonable minimum is achieved. After some a series of tests, it was concluded that this approach may be the most suitable for our problem (see section 4.2.1.1. *Algorithm*).

2.3.2 Mean-Shift Clustering

The mean shift clustering algorithm [5] is based on finding blobs. The main difference between this algorithm and k-means is that mean-shift clustering uses a mathematical method to estimate the centroids for the algorithm. It is an iterative algorithm that updates the centroids at every iteration until it reaches the final result. The operation used to update the centroid values is what is actually called the mean shift. While this approach also seemed promising, it was found that k-means better fit our problem (see section 3.2 *Algorithm*).

2.3.3 Hierarchical Clustering

The hierarchical clustering method [6] is one of the classical clustering methods. This algorithm works by merging and breaking clusters until the final result is reached. First, each sample is their own cluster, then, the clusters are merged until a defined metric is satisfied. This was a potential solution for our mentor-mentee problem; however, one of its downsides is its high dependence on the method chosen for joining samples. Moreover, after testing, it was found that k-means is a more appropriate approach for the given problem (see section 3.2 *Algorithm*).

3. Proposed Solution

As a solution to the aforementioned problems, we propose the Mentor Mentee Matching (MMM) System; a web-based application designed to automate and streamline the mentor-mentee program management process. Users will have the ability to automatically match mentors with mentees on per-faculty basis. Additionally, users will

¹ Namely, the mean of all the nodes within a cluster.

have access to student engagement information in the form of visualised data, as well as the ability to download mentor-mentee match results for documentation.

The system is packaged as a web application, thus alleviating need for external installation of updates from the user. As a result of being built with the most up to date back-end and front-end frameworks, the MMM System can easily adapt to the needs of the staff and requires little to no maintenance from the developers.

Our matching system is comprised of two main parts; a front-end, which handles user interactions, and a backend, which handles the matching and both of which can be visualized in detail in *Appendix A*. The integration of all its parts will be done by the use of Docker², an open platform software that provides the virtualization of applications through “containers”. Each container is a packaged application along with its dependencies, helping in the flexibility and portability of the application.

Since each part of the system is containerized in its own environment, they will need a method of communicating amongst one another. The Docker-Compose³ tool is designed with the sole purpose of running multi-container applications and was used in MMM to help bridge the communication gap between containers. All the containers communicate via an internal network created by Docker-Compose that is in essence invisible to all outside traffic. This guarantees that in the event of an adversary, all sensitive information will remain hidden in the internal network thus maintaining the privacy of students.

3.1 Data Cleaning

The main source of information to perform the grouping is the participants’ answers to a series of questions regarding interests, hobbies, and personality traits. Ideally, the staff using the system is only required to upload an Excel file⁴ containing all of participants’, which may be in yes/no format, or rating format (e.g. agree, partially agree, neutral, partially disagree and disagree). Once the matching is done, the system will store this information for later use, allowing for group editing, e-mailing, and downloading of the generated matches.

As a means to shape the data into a format that the computer can easily process, we converted all answers into numerical values. For example, the yes/no questions were converted into 0/1s and the rating questions into numbers from 0 to 4. This way, our data mass is actually a matrix of features, where each line represents a student, and each column represents a feature of such student.

² Available at www.docker.com

³ See <https://docs.docker.com/compose>

⁴ As far as we know, the raw data from the questionnaires is, by default, stored in this format.

3.2 Algorithm

After careful consideration and exhaustive brainstorming, we concluded that a clustering method would be the most appropriate given the properties of the mentor-mentee matching problem. Within this area, three methods were considered: K-Means, Mean-shift and Hierarchical clustering (see section 3.3. Clustering); these provide a parameter for customizing the number of clusters wanted, which is needed, since we have a limited number of mentors.

While all three algorithms lead to satisfactory results, we tested and compared them in order to determine the best fit. For this purpose, we used the Cluster Silhouette Score [7] and the Calinski Harabaz Score [8], which showed that the best algorithm for our data was K-Means clustering.

After determining the method to use, we encountered a number of complications in terms of its implementation. Firstly, K-Means is highly available in multiple machine learning frameworks; however, they only allow the developer to define a number of clusters, and based on that, it generates the seeds for the cluster centroids. Moreover, the algorithms do not allow for the definition of a maximum number of elements for each cluster, which could potentially cause a great imbalance between mentors, for example, causing some to be in charge of 60 students, while having others be in charge of 2. Lastly, given the moving cluster centroids, there is no way of guaranteeing the presence of a mentor per group.

In an attempt to solve these issues, we decided to use a modified version of the K-Means clustering algorithm and applied it to the data in a per-faculty basis. First, we use each mentor as a cluster “centroid”, which will never be re-calculated. Then, we compute the Euclidean distance between each first-year student and each centroid. Afterwards, we do a round robin of each centroid and assign the mentee that has the smallest Euclidean distance to it. The algorithm is carefully outlined in *Appendix B*.

3.3 Front-end

The front-end is comprised of all the elements that a user would interact with directly. This involves the overall design; layout, color, button placement, and any other interactive elements in the application. This, though not directly involved in the matching process, is a very important aspect of the user experience. The design is typically neglected when creating task-specific tools, but we want to stress its importance so that crucial elements are not hidden or hard to find, and ultimately improve overall user experience. The main goal when considering the layout of the system is for the user to require the least number of clicks for accessing the critical elements of the application, such as matching and communication pieces.

3.3.1 New Match

The matching process is a two-step operation, accessible through the side menu under the “New Match” tab. Step 1 (*Appendix C Figure 1*) involves the uploading of mentee and mentor raw data files, while step 2 (*Appendix C Figure 2*) allows for the manipulation of the previously uploaded data and then performs the matching itself. In this step, first, the user is shown a list of the identified questions (number 1 in *Appendix C Figure 2*) with a default weight, which can be customized (number 2 in *Appendix C Figure 2*). Once the user is satisfied with the selected weights, the matching process begins upon the clicking of the “Match” button (number 3 in *Appendix C Figure 2*).

Once this process is completed, the user will be able perform the following actions: E-mail all mentors with a list of their assigned mentees (number 4 in *Appendix C Figure 2*), view the matches through manual assignation of mentors/mentees (number 5 in *Appendix C Figure 2*), or download the file containing all the groups that have just been generated (number 6 in *Appendix C Figure 2*).

3.3.2 Last Match

The other primary feature of the Mentor-Mentee Matching System is the ability to view and manage the most recent match performed. This can be accomplished through the “Last Match” tab, under “Recent”, which is displayed in *Appendix D Figure 1*.

The most aesthetically dominant and notable feature in this section is the complete list of students that were matched into groups, where mentors are coloured in grey and mentees in white. This data can be easily narrowed down through the search bar (number 1 in *Appendix D Figure 1*) and the filtering feature (number 3 in *Appendix D Figure 1*), allowing for more accurate results and easier lookups. Once a person is selected from this section, their corresponding group will be shown in the panel to the right (number 4 in *Appendix D Figure 1*), as well as the assigned mentor and the group’s engagement metrics (number 5 in *Appendix D Figure 1*). For clarity, the selected student is highlighted in red, both in the main panel and the group panel.

Moreover, the email (*Appendix D Figure 2*) and manual assignation (*Appendix D Figure 3*) features can be found in the “More Options” menu (see number 2 in *Appendix D Figure 1*). To achieve this, the user must select the mentors and mentees to email, and then select the corresponding option from the menu. In a similar fashion, if the user wishes to manually re-assign mentees to different groups, they must select the students they wish to edit, click on the menu, and select the manual assignation option.

Lastly, a download button at the bottom right of the page (number 6 in *Appendix D Figure 1*) allows the user to download either the current groups stored, or the summary of engagement tracking for each of them. Once clicked, they may select the format that best suits their needs.

3.4 Backend

The backend is comprised of all the services of the system that run “under the hood”. These are back-end components that the users will not interact with directly, but that are crucial in the running of the system. These components include the matching server, matching system and the authentication system. Each of these components are further described below:

3.4.1 Matching Server

The matching system service is responsible for taking the raw data files supplied by the client, which include the personality questionnaire answers for both the mentors and mentees. The raw data is read in by the matching system service, cleaned, and then used to perform the matching process using the following described in *Appendix B*. All outside routes are re-routed through the matching server to the rest of the back-end components. This is done with the intention of having a single and secure entry point, so that the mission critical part of the system is completely sealed off from any potential adversaries.

3.4.2 Matching System

The matching system is composed to the matching database and the RESTful API as displayed in *Appendix A*. After the matching process in the matching server, the data is generated and sent to the matching database for persistent storage. This ensures that users of the system can conveniently view their last match without having to do the matching process all over again.

Given the graph nature of our data, where mentors are connected to mentees in a group, we decided to use a graph database. This type of data storage is composed of nodes and relationships between them. In this case, the nodes would represent mentors, mentees, and questions. Moreover, relationships between mentors and mentees could be drawn with an attribute representing their similarity, while another could be drawn between any participant and a question, with their answer as a relationship attribute. Altogether, a cluster of linked mentor/mentee nodes would represent a matched group. We chose the Neo4J⁵ graph database management system for this project, since it is optimized to map and store connected data, and it provides all the features we need to manipulate our own data.

The second component, the RESTful API, is responsible for querying the matching database for each specific API call. For example, these calls could request a list of all the mentors for a given faculty, or the group of a student. This component is strictly concerned

⁵ Available at neo4j.com.

with sending queries to the matching database and is only visible from within the server, since it handles sensitive data.

3.4.3 Authentication System

The authentication system is similarly comprised of a RESTful API and an authentication database. The database is a PostgreSQL relational database in which all authorized user information will be stored. Information like email, password, and role will be kept, as well as their generated tokens for each log in session. Tokens will be used as signatures with an expiration time that allows the owner of the token to have access to the server while the token is not expired. The RESTful API is responsible for querying information from the authentication database and performs token generation and checking to ensure that all requests are made from authorized users. This is a very important part of the system since it adds another layer of security to protect the student data.

3.5 Testing

Despite being a crucial element in software development, testing is a process that is often overlooked. The system's backend can be entirely tested with automatic test cases that would be run at every iteration of the development lifecycle. These test cases ensure that new features don't regress the overall system and provide a level of confidence to both developers and users in the advancement of the software. The coverage of the test cases should be large enough to protect the security and integrity of the user sensitive data. The test cases should assume an adversary and simulate potential breaches of the system in order to ensure it is well protected.

4. Current Status

The system thus far has been implemented with very careful picking and choosing of requirements. Given the limited time of developing the system we wanted to ensure that only the most essential aspects are actually implemented. We are happy to say that we have met all of our goals since the interim mid-way mark and are in a comfortable position to demonstrate the ability of the system to handle the task at hand. The two main features that we strived to implement from the start were the New Match (*Appendix C*) and the Last Match (*Appendix D*). These are the two features we have focused on implementing which remain not fully completed. We wish to be as explicit as possible in regard to what features are functional and which are not.

The New Match (*Appendix C*) menu provides the ability to upload the raw mentor/mentee files and also the matching process. In addition to custom question

weights, the matching algorithm, as well as the ability to save information to a database after the match are fully functional. The emailing of mentors after the matching process remains as a semi-implemented feature, where the UI is fully implemented, but the backend has not been connected to the front-end. This part of the system is very close to being fully implemented if given the time.

The Last Match (*Appendix D*) part of the system is responsible for viewing the previously matched mentors/mentees and gives the user the ability to modify groups as required. Filtering by faculty, searching, and managing the visible columns are fully functional features in addition to the student table on the right-hand side of the page (number 4 in *Appendix D Figure 1*). The engagement visual chart at the top (number 5 in *Appendix D Figure 1*) only shows randomly generated data for now due to the lack of engagement tracking data from the client. As such the “Engagement” download button is disabled until the feature is implemented. The ability to download the matched data works as intended, similar to the “New Match” part of the system. Emailing selected students similarly has all the data required to send the email but lacks the back-end function to send that email to the students. The manual assignment accurately depicts the potential mentors for each selected person’s faculty, but does not yet update the database, and remains a semi-implemented feature. Lastly, the coloured bar at the bottom works as intended and accurately depicts the distribution of students in each faculty.

5. Future Work

The following features are components that are either under construction or require full implementation. The containerized architecture makes the system incredibly malleable, allowing for the effortless addition of new containers, and the system has been designed with these missing features in mind so that adding them wouldn’t require a large amount of time and resources. The specifics of the future works in this system are more clearly outlined in Appendix F and will be further discussed here.

One feature near completion is the ability to send emails, both after the matching process to all mentors, and in the last match step by specifically selecting students to email. All the functionality is in place, the only issue is acquiring a mail server to actually send the emails. This is something that was prototyped but proved to be more complicated than expected.

There are also three additional pages that must be designed: Mentor Logs page, Feedback page, and Help Page. Following the design stage, the front-end will have to be created, along with the implementation of all functionalities for each. The biggest challenge would be the mentor log page since it has the most requirements to be met, whereas Feedback and Help pages are fairly static.

The last thing that the system will require is from an authentication standpoint. The process of creating accounts, logging in, and authentication must all be handled by the Authentication API and the Authentication Database. In addition, any time a query is sent to any of the databases, sensitive information is being handled, so token authentication would be required to ensure that the request is coming from a legitimate user.

We believe that with the aforementioned features implemented, a fully operational and efficient system would be entirely able to replace the existing process of running the mentor-mentee program at UOIT.

6. Resources

The system thus far has proven to be more than just a proof of concept, but a fully capable base for the final system. Despite the functionality thus far, several resources would be required in order to fully implement the whole system.

We roughly estimate that given 5 developers working 10 hours a week, it would require close to 4 months or 800 hours in order to complete the remainder of the requirements. This goes under the assumption that the developers are able to closely work together and meet on a weekly basis.

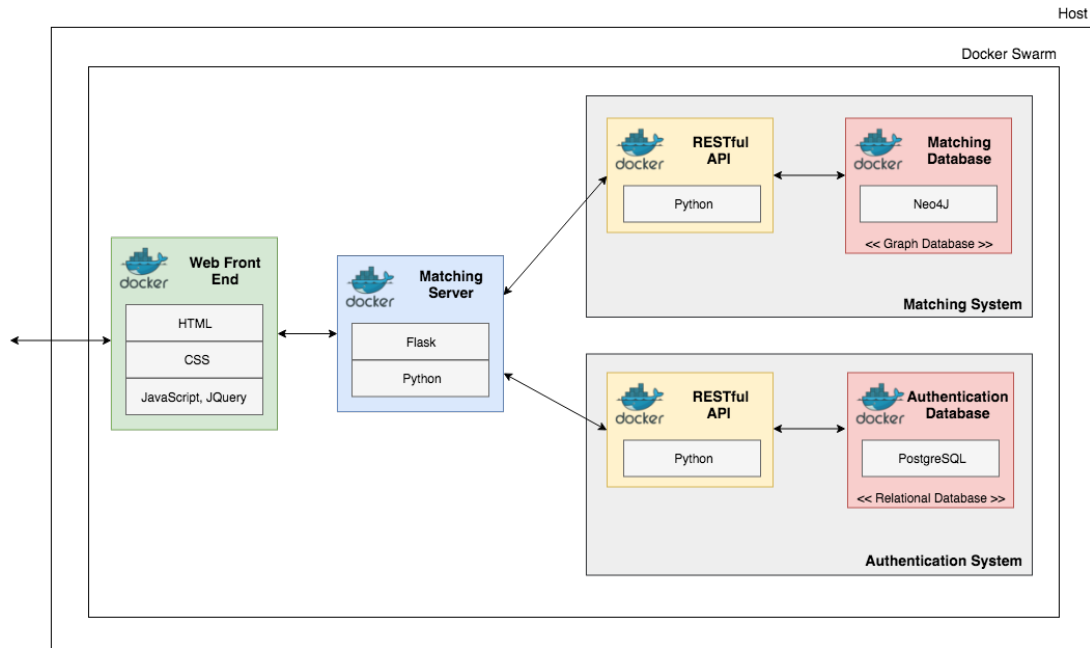
The second resource required would be the server on which everything is hosted, as well as a possible domain. Currently it is hosted on a server that costs \$15/month for development purposes, but should the users require a smoother and faster matching experience, this would naturally increase. If the system were to ever grow past the 5 concurrent user mark, this would also require further expansion in the hardware, and thus the cost. In addition, we would require full licenses for any pieces of software that are used in the development process.

7. References

- [1] Lappas, Theodoros, Kun Liu, and Evimaria Terzi. "Finding a team of experts in social networks." *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2009.
- [2] Kargar, Mehdi, and Aijun An. "Discovering top-k teams of experts with/without a leader in social networks." *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011.
- [3] Zhamri, Che & Che Ani, Zhamri & Yasin, Azman & Husin, Mohd Zabidin & Hamid, Zauridah. (2010). A Method for Group Formation Using Genetic Algorithm. *International Journal on Computer Science and Engineering*. 02. 3060-3064.
- [4] David Arthur and Sergei Vassilvitskii. 2007. k-means++: the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (SODA '07)*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1027-1035.
- [5] D. Comaniciu and P. Meer, "Mean shift: a robust approach toward feature space analysis," in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 24, no. 5, pp. 603-619, May 2002
- [6] R. Sibson; SLINK: An optimally efficient algorithm for the single-link cluster method, *The Computer Journal*, Volume 16, Issue 1, 1 January 1973, Pages 30–34
- [7] P. J. Rousseeuw, "Silhouettes: A graphical aid to the interpretation and validation of cluster analysis," *Journal of Computational and Applied Mathematics*, Volume 20, 1987, Pages 53-65
- [8] T. Calinski and J. Harabasz, "A dendrite method for cluster analysis," *Communications in Statistics*, vol. 3, no. 1, pp. 1-27, 1974.

Appendix A

Proposed system architecture



Appendix B

Mentor-mentee matching algorithm breakdown

Algorithm 1 Matching Algorithm

```
1: groups  $\leftarrow \{\}$ 
2: for each Faculty do
3:   candidates  $\leftarrow \{\}$ 
4:   max_size  $\leftarrow \text{ceil}(\text{num\_mentees} / \text{num\_mentors})$ 
5:   for each mentor do
6:     for each mentee do
7:       score  $\leftarrow \text{euclidean}(\text{mentor}, \text{mentee})$ 
8:       candidates.add((mentor, mentee), score)
9:   candidates.sortby(score, "descending")
10:  target_mentor  $\leftarrow 0$ 
11:  unmatched_mentees  $\leftarrow \text{mentees}$ 
12:  while unmatched_mentees.length > 0 do
13:    for each pair(mentor, mentee) in candidates do
14:      if (mentor is target_mentor)
15:        and (mentee is unmatched)
16:        and (mentor.group.size < max_size) then
17:        mentor.group.add(mentee)
18:        unmatched_mentees.remove(mentee)
19:    target_mentor  $\leftarrow \text{target\_mentor} + 1$ 
20:    target_mentor  $\leftarrow \text{target\_mentor} \% \text{mentors.length}$ 
21:  for each mentor in Faculty do
22:    groups.add(mentor.group)
23: return groups
```

Appendix C

User interface for the matching section of the proposed system

MMM
MENTOR-MENTEE MATCHING SYSTEM

NEW MATCH

RECENT

FEEDBACK

STEP 1: UPLOAD

Please upload the excel sheets containing the information for mentors and mentees. They must be in two separate files, and should adhere to the format below. Please make sure both files contain the same questions in the same order, as this could greatly affect the accuracy of the matches. Supported question answers: Yes, Not Applicable, No, Strongly Agree, Agree, Neutral, Disagree, Strongly Disagree.

STUDENT ID	FIRST NAME	LAST NAME	E-MAIL	FACULTY	PROGRAM	Q1	Q2	Q3	...	QN
100123456	Andrew	Goods	andrew@email.com	Science	Computer Science	No	Not Applicable	Strongly Agree	..	Neutral
..
100123457	Jessica	Ronalds	itsjess@email.com	Business and IT	Commerce	Not Applicable	Yes	Neutral	..	Strongly Agree

UPLOAD MENTOR FILENo file selected.

UPLOAD STUDENT FILENo file selected.

Go to STEP 2

Figure 1. Step 1: Upload of raw mentor and mentee data collected.

MMM
MENTOR-MENTEE MATCHING SYSTEM

NEW MATCH

RECENT

FEEDBACK

STEP 2: MATCH

Below, please select the level of relevance you consider for each of the questions identified. Once you're done, click the **MATCH** button to begin the matching process.

☐ Merge with most recent matches stored.

QUESTION	RELEVANCY
Mentor in same faculty	Medium
Commuters	Medium
Off-campus	Medium
Transfer Students	Medium
Mature Students	Medium
International Students	Medium
On Campus	Medium
Leadership	Medium
First Generation Students	Medium
Hard worker	Medium
Thoughtful	Medium
Nature/Environment	Medium
Helps Others	Medium
Click to Trust	Medium

MATCH

Matching 1049 students..
Successfully generated 109 groups.

EMAIL MENTORSVIEW MATCHES

DOWNLOAD MATCHES

Figure 2. Step 2: Question weight customization and matching.

EMAIL MENTORS

109 Mentors will be sent a list of the mentees that have been assigned to them. You can choose to include a personalized message in said e-mails, which can be redacted below.

You may also use the following tags to personalize your email subject and content:

- [FNAME] = Mentor's first name.
- [LNAME] = Mentor's last name.
- [MLIST] = List of mentees assigned to the mentor.

Subject:

Mentee Assigination

Content:

Hello [FNAME] [LNAME],

Please see the mentees that have been assigned to you below.

[MLIST]

Regards,
UOIT - Student Experience

SEND EMAIL

Figure 3. *Step 2: Once matching is completed, an email can be sent to all mentors, listing the mentees assigned to them.*

Appendix D

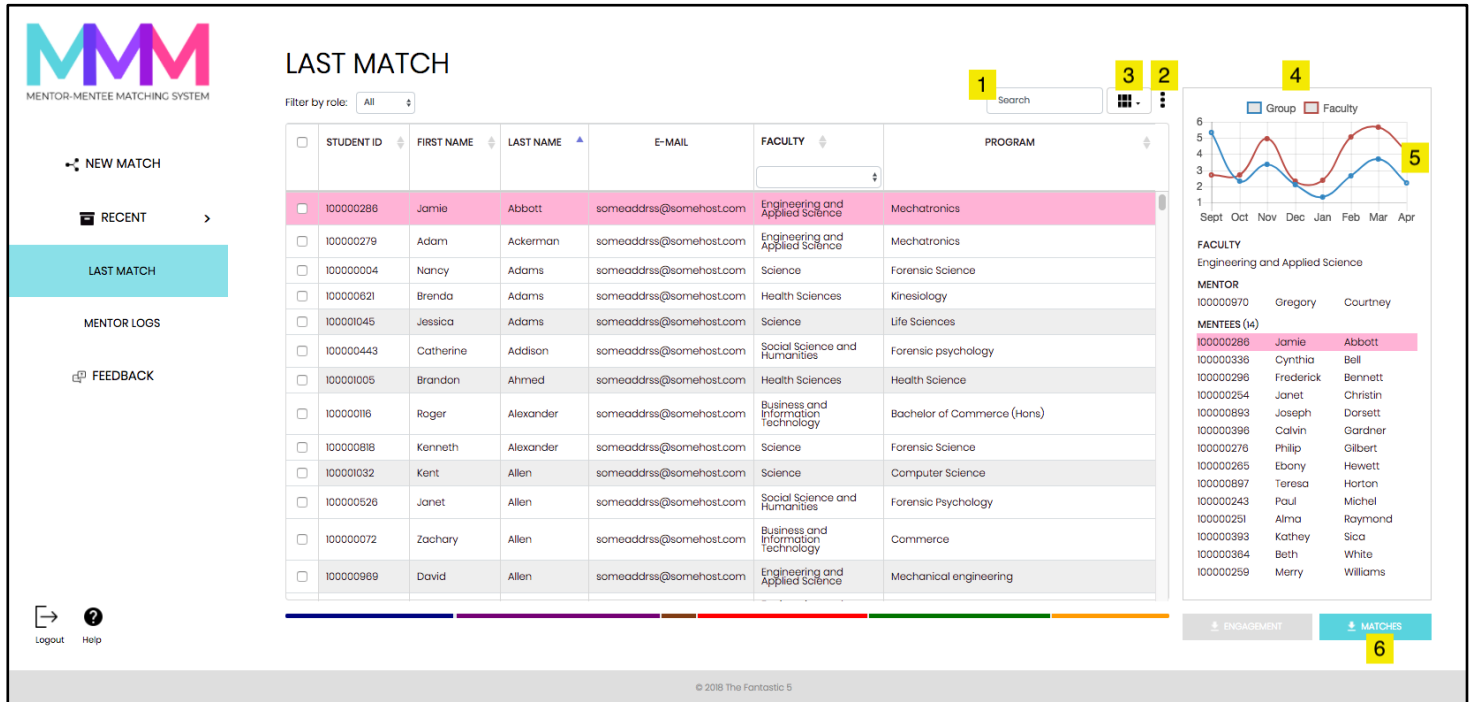


Figure 1. User interface for the administrator view of corresponding to the most recent match performed.

EMAIL STUDENTS

The selected students will be sent an e-mail, where the content can be specified below.

You may also use the following tags to further personalize your email subject and content:

- [FNAME] = Student's first name.
- [LNAME] = Student's last name.
- [MLIST] = List of mentees in the group.

Subject:

Content:

☒ Also e-mail all members in the same group the students selected.


 SEND EMAIL

Figure 2. *User Interface for emailing selected students from table*

MANUAL ASSIGNATION

Change the current mentor in the dropdown box and click the **SAVE** button below. If you wish to cancel, just close this window; no changes will be saved.

The current mentor for each mentee is marked with a star (★) beside their name.

FACULTY	MENTEE	MENTOR
Business and Information Technology	Edgar Massey	Carrol May (13) ★ ▾
Business and Information Technology	John Rich	Carrol May (13) ★ ▾

☒ E-mail mentors with updated mentee list.

SAVE

Figure 3. *User interface for the manual assignation feature*

Appendix E

Graph database samples.

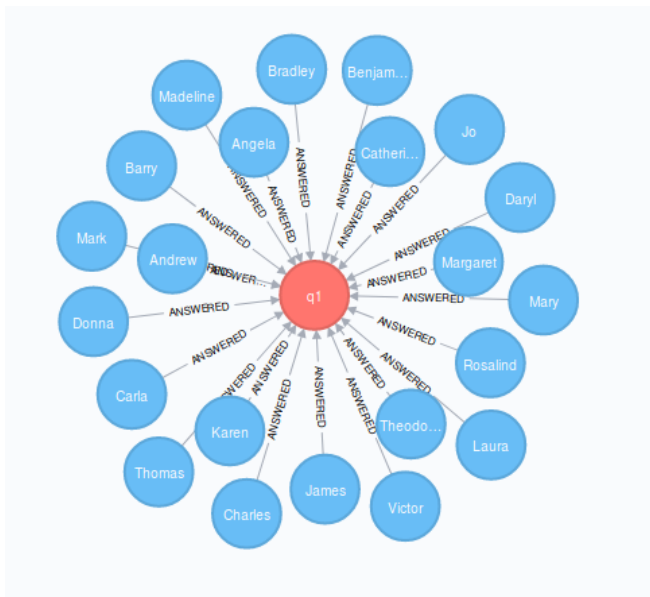


Figure 1. Matched group, consisting of multiple mentees and one mentor.

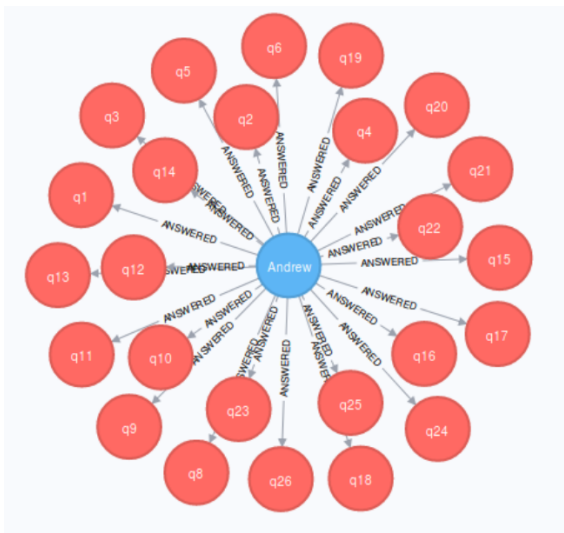


Figure 2 An answered question, as well as every mentor and mentee that answered

Figure 1. Matched group, consisting of multiple mentees and one mentor.

Appendix F

Current status and Future Work for the system.

TASK	STATUS
Background research on matchmaking.	COMPLETED
Design matching algorithm.	COMPLETED
Implement matching algorithm.	COMPLETED
Propose solution for communication between administrators, mentors, and mentees.	COMPLETED
Propose design solution for manual assignation of mentees to mentors.	COMPLETED
Propose design solution for custom weighted questions.	COMPLETED
Propose design solution for submission of students and mentors in batches.	COMPLETED
Propose design solution for the tracking of engagement and its visualization. These are calculated using the mentor logs.	COMPLETED
Implement simple prototype for front-end system.	COMPLETED
Implement simple prototype for back-end system.	COMPLETED
Implement simple prototype for databases.	COMPLETED
Propose design solution for manual mentor assignation	COMPLETED
Improve the group distribution of the matching algorithm.	COMPLETED
Introduce question weight within the algorithm.	COMPLETED
Connect Matching Database to backend.	COMPLETED
Create front-end for “New Match” section (Appendix C).	COMPLETED
Create front-end for “Last Match” section (Appendix D).	COMPLETED
Implement functionality for “New Match” section (Appendix C).	COMPLETED
Implement functionality for “Last Match” section (Appendix D).	COMPLETED
Create front-end for the manual assignation of mentors and mentees.	COMPLETED
Create front-end for emailing all mentors after match	COMPLETED
Create front-end for emailing selected students in Last Match	COMPLETED
Create front-end for the login page.	COMPLETED
Improve the styling of the Excel file downloaded, containing the recently matched groups.	COMPLETED
Implement functionality for emailing all mentors after match	FUTURE WORK
Implement functionality for emailing selected students in Last Match	FUTURE WORK

Propose design solution for mentor log completion.	FUTURE WORK
Propose design solution for Feedback section	FUTURE WORK
Propose design solution for Help Page	FUTURE WORK
Create front-end for mentor log completion	FUTURE WORK
Create front-end for Feedback section	FUTURE WORK
Create front-end for Help Page	FUTURE WORK
Implement backend for mentor log completion	FUTURE WORK
Implement Authentication RESTful API	FUTURE WORK
Implement Authentication Database	FUTURE WORK
Implement token authorization for requests	FUTURE WORK