



Software Engineering Department
Ort Braude College

Developing system for private tutor matching



Final Project in Software Engineering (Course 61771)

Karmiel – Jan 2021

Authors

Sami Odeh

Or Eliyahu

Supervisors

Mrs. Elena Kramer

Dr. Dan Lemberg

CONTENTS

1	Introduction	1
2	Theory	1
2.1	Distance and Metric Spaces	1
2.1.1	Distance in the plane	1
2.2	Nonlinear Programming	2
2.2.1	Karush–Kuhn–Tucker conditions	2
2.3	Lagrange Multipliers	2
2.4	Quadratic Programming	3
2.5	Summary of The Process	4
3	Preliminary Software Engineering Documents	6
3.1	Use Case	6
3.2	Design (GUI, UML Diagrams)	7
3.2.1	GUI	7
3.2.2	Class diagram	15
3.3	Testing Results	16
4	Results And Conclusions	17
4.1	Results	17
4.2	Troubleshooting	19
4.3	Conclusions	19
	References	19

1 INTRODUCTION

The private tutor plays a key role in the student experience, and for the new generation, the private tutor is not only a teacher with knowledge but also a kind of friend and guide. Therefore, finding a tutor based only on his knowledge is not enough today.

In our project, we have designed and developed a system that will help both student and tutor by finding the best match according to their skills, attitudes, etc. the purpose of the project is to save time and resources for both of them.

As a part of user registration, he needs to answer a set of questions about himself. Every student who searches for a private, the system will consider his answered vector in order to find the most suitable tutor. The system will check based on his answers what his chances of getting a private tutor if those chances are high enough, the system will offer this tutor to the student. The system will show to the student a tutor sorted list by matching percent. From this list, the student will select the tutor. The answers to every student are also saved. Therefore, the system will take into considerations in every future change the right result for every set of answers.

A percentage of a student-tutor match can be different from a tutor-student match. If the selected private tutor accepts the student invitation, it means that both sides need to schedule a lesson, the system will save both matching grade and the set of the answer questions. After some meetings, the student and private tutor needs to summarize the lessons, and give a feedback on each other, then decide if the meetings were successful and if they can continue for more future meetings.

A passing student means that they can continue for more meetings with the selected private tutor, so the system has done its job, and no changes should be made except saving the data for both sides. If, however, the tutor or student decides to stop giving future meetings, the system will have to modify the questions with new weights, so the student with the same answers exactly will not receive the same tutor matching percent.

In the case that will be no more future meetings, the system will use quadratic programming in order to create a new set of questions weight that is close to the old one, and for the recent set of answers, the matching percent will be lower. To maintain the vector of weights, every set of answers will be inserted into the inequality of non-linear programming algorithms.

Ultimately, the goal is to find tutors with the highest match percent for a student. The tutors with the highest matching percent will be the most suitable to continue future meetings.

2 THEORY

2.1 Distance and Metric Spaces

Measuring the distance between two objects is a basic mathematical activity and provides a starting point for geometry [1]. A familiar example of this is finding the (Euclidean) distance between two points in the XY-plane. The set of all points in the plane together with this method of computing the distance between any two points is an example of a metric space, which is a set S of objects together with a metric, a way of computing the distance between any two elements of S . There are many examples of metric spaces other than points in the plane. Metrics can be defined on matrices, functions, sets of points, and other mathematical objects.

2.1.1 Distance in the plane

The (Euclidean) distance $d((x_1, y_1), (x_2, y_2))$ between two points (x_1, y_1) and (x_2, y_2) is the length of the line segment with endpoints (x_1, y_1) and (x_2, y_2) . By drawing a right triangle with vertices at (x_1, y_1) , (x_2, y_2) and (x_2, y_1) , and using the Pythagorean theorem, the length is

$$d((x_1, y_1), (x_2, y_2)) = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

By considering this distance function, we can observe some of its properties and then define a metric or distance function for situations other than points in the plane, based on these properties.

1. The distance between two points $P_1 = (x_1, y_1)$ and $P_2 = (x_2, y_2)$ is always non-negative, $d(P_1, P_2) \geq 0$. Also, the only way $d(P_1, P_2)$ can be zero is if $x_1 = x_2$ and $y_1 = y_2$, meaning that $P_1 = P_2$. So, for any two points P_1 and P_2 , $d(P_1, P_2) \geq 0$ and $d(P_1, P_2) = 0$ if and only if $P_1 = P_2$.
2. The distance is symmetric, meaning that the distance from P_1 to P_2 is the same as the distance from P_2 to P_1 , or $d(P_1, P_2) = d(P_2, P_1)$ for any two points P_1 and P_2 .
3. A less obvious property of the distance function is the triangle inequality: For any points P_1, P_2 and P_3 .

$$d(P_1, P_2) \leq d(P_1, P_3) + d(P_3, P_2).$$

The reason for calling this the triangle inequality comes from the geometric property that if P_1, P_2 and P_3 are the vertices of a triangle. The sum of the lengths of any two sides is larger than the length of the third side. Another way of thinking of the triangle inequality is that if you want to go from P_1 to P_2 , it is never shorter to first go to some point P_3 and then go to P_2 from P_3 .

These three properties are considered to be the most important features of the distance function and provide the definition of a metric space.

Throughout our project, we used the Euclidean distance, and any distance word is related to it.

2.2 Nonlinear Programming

Nonlinear programming problems arise in the mathematical modeling of several problems in real-world applications. A large number of these problems can be formulated as quadratic programming problems (QP) with a quadratic objective function and a linear set of equality or inequality constraints. Primarily, quadratic programming with linear constraints can be viewed as a generalization of the linear programming problem (LP) with a quadratic objective function. Therefore, it encompasses all LP problems, including applications in scheduling, planning, and flow computations.

2.2.1 Karush–Kuhn–Tucker conditions

The Karush–Kuhn–Tucker conditions (a.k.a. KKT conditions or Kuhn–Tucker conditions) are a set of necessary conditions for a solution of a constrained nonlinear program to be optimal [2]. The KKT conditions generalize the method of Lagrange multipliers for nonlinear programs with equality constraints, allowing for both equalities and inequalities.

2.3 Lagrange Multipliers

One of the most common problems in calculus is that of finding maxima or minima (in general, "extrema") of a function, but it is often difficult to find a closed form for the function being extremized. Such difficulties often arise when one wishes to maximize or minimize a function subject to fixed outside conditions or constraints. The method of Lagrange multipliers is a powerful tool for solving this class of problems without the need to explicitly solve the conditions and use them to eliminate extra variables.

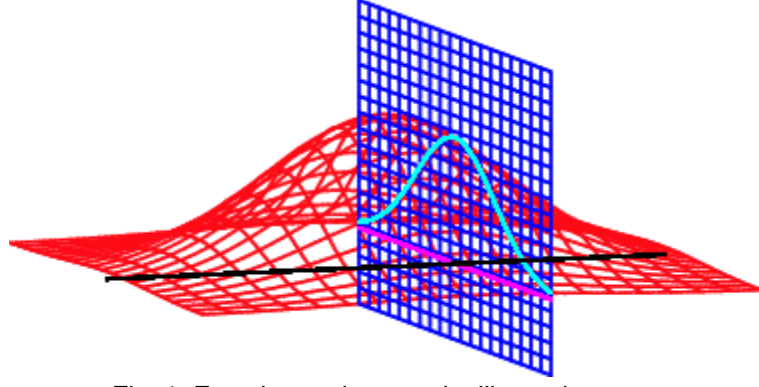


Fig. 1: Function and constraint illustration

Lagrange multipliers, also called Lagrangian multipliers [3], can be used to find the extrema of a multivariate function $f(x_1, x_2, \dots, x_n)$ subject to the constraint $g(x_1, x_2, \dots, x_n) = 0$, where f and g are functions with continuous first partial derivatives on the open set containing the curve $g(x_1, x_2, \dots, x_n) = 0$, and $\nabla g \neq 0$ at any point on the curve (where ∇ is the gradient).

For an extremum of f to exist on g [4], the gradient of f must line up with the gradient of g . In the illustration above, f is shown in red, g in blue, and the intersection of f and g is indicated in light blue. The gradient is a horizontal vector (i.e., it has no z -component) that shows the direction that the function increases. for g it is perpendicular to the curve, which is a straight line in this case. If the two gradients are in the same direction, then one is a multiple ($-\lambda$) of the other, therefore $\nabla f = -\lambda \nabla g$.

The two vectors are equal, so all of their components are as well, giving $\frac{\partial f}{\partial x_k} + \lambda \frac{\partial g}{\partial x_k} = 0$, for all $k = 1, \dots, n$ where the constant λ is called the Lagrange multiplier.

The extremum is then found by solving the $n + 1$ equations in $n + 1$ unknowns, which is done without inverting g , which is why Lagrange multipliers can be so useful.

For multiple constraints $g_1 = 0, g_2 = 0, \dots$,

$$\nabla f + \lambda_1 \nabla g_1 + \lambda_2 \nabla g_2 + \dots = 0$$

2.4 Quadratic Programming

Quadratic programming (QP) is non-linear programming. It is the process of solving a special type of mathematical optimization problem—specifically, a (linearly constrained) quadratic optimization problem, that is, the problem of optimizing (minimizing or maximizing) a quadratic function of several variables subject to linear constraints on these variables. In addition, quadratically constrained quadratic programs frequently occur in engineering modeling, design, and control.

The quadratic programming problem with n variables and m constraints can be formulated as follows. Given a real-valued, n -dimensional vector \vec{c} , an $n \times n$ real matrix Q , an $m \times n$ dimensional real matrix A , and an m -dimensional real vector \vec{b} , the objective of quadratic programming is to find an n -dimensional vector \vec{x} , that will minimize $\frac{1}{2} \vec{x}^T Q \vec{x} + \vec{c}^T \vec{x}$ while subject to $A \vec{x} \leq \vec{b}$. Here \vec{x}^T denotes the vector transpose of \vec{x} , and $A \vec{x} \leq \vec{b}$ means that every entry of the vector $A \vec{x}$ is less than or equal to the corresponding entry of the vector \vec{b} [5]. Notice that the function that needs to be minimized will have to take the form of the presented expression, and the variables A, b, Q , and c needs to be defined. We will then define a Lagrange function $L(\vec{x}, \mu) = \frac{1}{2} \vec{x}^T Q \vec{x} + \vec{c}^T \vec{x} + \mu^T A \vec{x}$, and find a solution using Karush-Kuhn-Tucker(KKT) conditions as done for other non-linear problems [6]:

$$\bullet \nabla L(\vec{x}, \mu) = \vec{c}^T + \frac{1}{2} \vec{x}^T Q + \mu^T A = 0$$

- $Ax \leq b$
- $x^T, \mu^T \geq 0$

The algorithm's complexity for positive definite Q , the ellipsoid method, solves the problem in polynomial time [7]. If, on the other hand, Q is indefinite, then the problem is NP-hard. In fact, even if Q has only one negative eigenvalue, the problem is NP-hard [8], which means that some of the most interesting combinatorial optimization problems can be posed in a quadratic programming framework [9].

2.5 Summary of The Process

Because of the similar process for each meeting, we will explain a single non-specific meeting process. Let say a student has n_1 questions and private tutor has n_2 questions. We will represent the weight of those questions as a vector $\vec{m}_1 = (m_{11}, m_{12}, \dots, m_{1n_1})$ and the answers of \vec{m}_1 represented by a new vector $\vec{t}_1 = (t_{11}, t_{12}, \dots, t_{1n_1})$, $t_{1i} \in [0, 1]$, the vectors \vec{m}_2 and \vec{t}_2 respectively.

On registration, the weights \vec{m}_1 vector of the questions will be initialized by the same weight of each question $\frac{1}{n_1}$, assuming there were no specific instructions from the user about the weights of the questions. The student and the tutor must answer the questions about themselves, so, the student will answers n_2 questions about himself and the answers will be saved in \vec{t}_2 , the tutor will answers n_1 questions about himself and the answers will be saved in \vec{t}_1 respectively. This process is similar to the process of giving feedback after the end of the session, which will be explained later.

In case student looking for a private tutor, the system will search the most suitable tutors according to his answered questions \vec{t}_2 and evaluate p , which is the matching grade for each tutor, and the results will contain the private tutors when p is greater than P (P is a minimum constant grade for matching, 60 for example). The student will receive a message when there are no tutors with an evaluation grade of greater than P .

Pseudocode: Find the most suitable tutors

```

1. Input: student as  $S$ , tutors as  $T$ , course as  $C$ , minimal matching grade as  $P$ 
2. Output: most suitable tutors list  $S_T$ 
3.    $L_T \leftarrow T.get\_tutors(course=C)$ 
4.    $S_T \leftarrow []$ 
5.   for  $t \in L_T$  do
6.      $p = t.\vec{m}_2 \cdot S.\vec{t}_2^T$ 
7.     if  $p > P$  then
8.        $S_T.insert(t, p)$ 
9.    $S_T.sort(by\ p)$ 
10.  return  $S_T$ 

```

In case a student has selected a private tutor from the list that the system has offered him, the tutors will receive an invite from the student which includes the matching percentage of tutor-student, which may differ from the percentage of student-tutor (p), from the reason of the way of evaluating $t.\vec{m}_2 \cdot s.\vec{t}_2^T$ and $s.\vec{m}_1 \cdot t.\vec{t}_1^T$.

When a tutor accepts student invitation, the system will save both answered questions and their matching grade, and then they need to schedule a meeting. After several sessions, both of them fill feedback on each other, so the student will create a new \vec{t}_1 and the private tutor create a new \vec{t}_2 , both of the answered vectors will be saved for future evaluation, then they need to decide if they want to continue future meetings.

In case the meetings were successful, and both want to continue for future meetings, the system will just keep the saved data in the database. Since the student selected a suitable private tutor and both have continued for more meetings, then the system has done its job, and there is no need to change any weight vector.

In case one of them decided not to continue the meetings, meaning the system had failed to properly assess the matching and a change in the weight questions are required. The change in the weight of the questions is applied to the one who had decided to stop the meetings. The new vector will be calculated according to the existing saved data, so there will not be a situation where the older matching grade is equal to the older grade with the same answers.

In case both of them decided to stop future meetings, the system will consider this case as the previous case above for each one separately.

Every saved data for each user contains the answered questions and matching grade. Saved matching data will be turned into inequality to calculate the new weights vector. For every rejected user, the multiplication of his answered questions vector with the new weighted questions vector will have to be less than the old matching grade. The multiplication will be represented $\vec{m}_1 \cdot \vec{t}_1^T = \sum_{j=1}^{n_i} m_{1j} \cdot t_{1j}$ and we will demand $\sum_{j=1}^{n_i} m_{1j} \cdot t_{1j} < \text{old matching grade}$ - for the rejected user.

All those inequalities will define a sub-space with endless possibilities for the new vector, but to provide a more reliable and stable system, the closet vector in terms of Euclidean space to the existing weight vector in the defined sub-space will be the best possibility to select.

To find this vector, we will use the process of Quadratic Programming. The function that needs to be minimized is the Euclidean distance between the two points representing the two weight vectors. Notice that taking the square distance instead of the distance itself will not change the value of the new vector.

Let $\vec{x} = (x_1, x_2, \dots, x_n)$ be the new weight vector, then $f(\vec{x}, \vec{m})$ needs to be minimized, where $f(\vec{x}, \vec{m}) = (x_1 - m_1)^2 + (x_2 - m_2)^2 + \dots + (x_n - m_n)^2 = x_1^2 + x_2^2 + \dots + x_n^2 + m_1^2 + m_2^2 + \dots + m_n^2 - 2 \cdot x_1 \cdot m_1 - 2 \cdot x_2 \cdot m_2 - \dots - 2 \cdot x_n \cdot m_n$.

Quadratic programming equation needs to take the form of $\frac{1}{2}x^T Qx + c^T x$ while subject to $Ax \leq b$, for each $x_i \geq 0, 1 \leq i \leq n$ and $\sum_{i=1}^n x_i = 100$. Now the necessary variables for the process must be defined: $Q = 2 \cdot \begin{pmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{pmatrix}$, meaning Q will be twice (to counter the half in the target form) the value of a $n \times n$ matrix saving the coefficient for every multiplication between two values in the vector x , in this case 1 for every $(x_i)^2, 1 \leq i \leq n$, and 0 otherwise. $\vec{c} = -2 \cdot \vec{m}$, meaning \vec{c} will be an n -dimensional vector saving the coefficient for every $x_i, 1 \leq i \leq n$. Let k be the total number of saved data for student or private tutor, G be the average of all feedback's grades, hence the total number of inequalities. Therefore, $A_{k+n \times n}$ will save in row i the coefficients of the vector \vec{x} in inequality $i, 1 \leq i \leq k$, these coefficients are the answers of a specific question, and negative identity matrix $n \times n, k+1 \leq \text{matrix} \leq k+n$. $\vec{b}_{k+n \times 1}$ will save the value that the inequality is less than or equal to, $b_i = G, 1 \leq i \leq n$ and $b_i = 0, k+1 \leq i \leq k+n$. Note that the process only works for inequality that is less than or equal to, so all the bigger than inequalities will have to be multiplied by minus one.

The output of the Quadratic programming process will be a new weight \vec{x} . If the vector is relatively close in Euclidean space to the old \vec{m} , the system will replace it. But there is a possibility where the distance of the new vector is too far from the old one, meaning there is an inconsistent behavior in the user's feedback, which is probably the result of two or more different feedbacks with different estimations. In this situation, we thought about simple method which makes the same process for the

In phrase A, we thought about using claustration for the inconsistent feedbacks problem, but we have found that is complicated work and requires a lot of trial and error, so for the solution we mentioned above makes the software more simplicity within the semester time.

3 PRELIMINARY SOFTWARE ENGINEERING DOCUMENTS

We have chosen to implement our application in Angular for client side and Node.js for server side, and the database is stored by MongoDB. We decided these choices since they are the leading technologies for website applications with a lot of advantages. The application compatible for pc and mobile.

3.1 Use Case

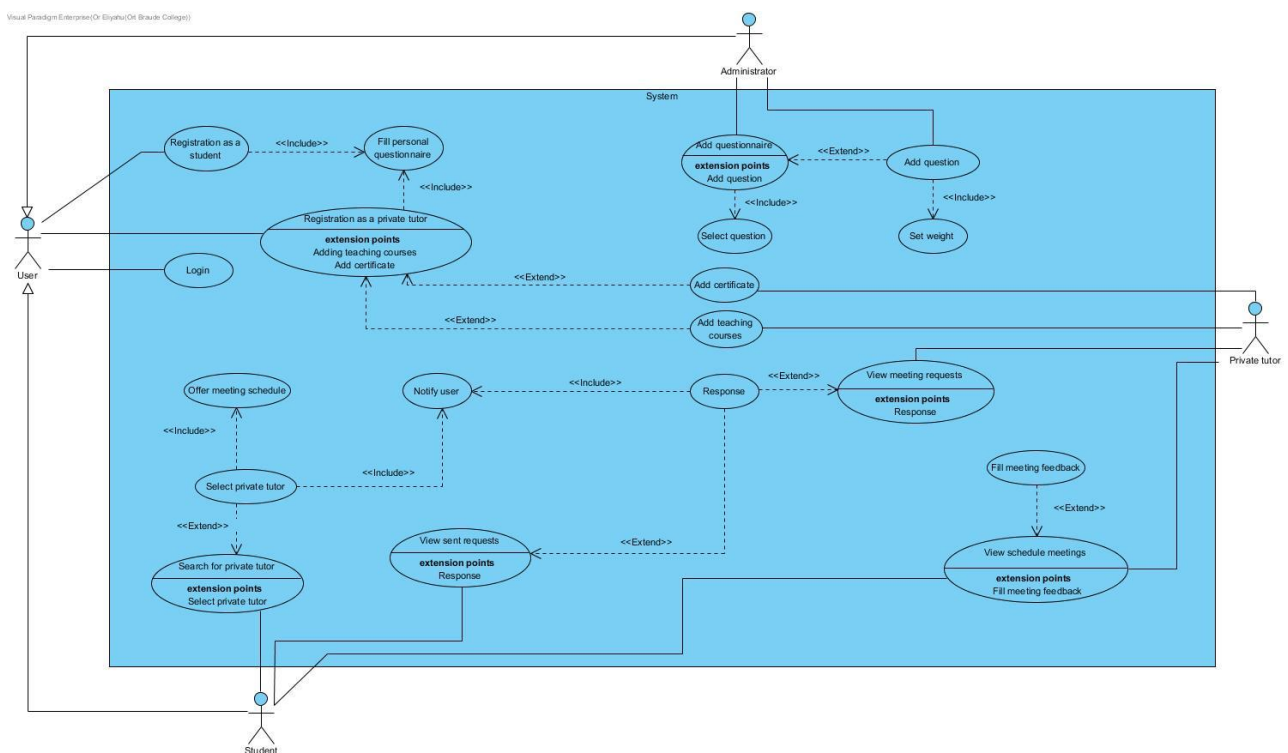


Fig. 2: Use Case Diagram

3.2 Design (GUI, UML Diagrams)


3.2.1 GUI

The screenshot shows the 'Register' page for a tutor. On the left is a dark sidebar menu with the 'PRIVATE TUTOR MATCHING' logo at the top. The menu items are: Home, Login, Register (highlighted with a dropdown arrow), Student, Private Tutor, and FAQ. The main content area has a breadcrumb trail 'Home / Register / Tutor'. The 'Register' section is titled 'Create your account' and contains four input fields: 'First name', 'Last name', 'Email', and 'Password'. Below this is a 'Questionnaire' section with the instruction 'Fill the questions'.

Fig. 3: Tutor registration page, Personal information

This screenshot shows the 'Questionnaire' section of the tutor registration page. It contains the instruction 'Fill the questions' followed by four rating questions, each with a 5-point scale (1 to 5) where '1' is selected. The questions are: 'Does the tutor can explain ideas and concepts clearly?', 'Does the tutor listen to me and try to understand my problems?', 'Does the tutor behaves in a professional manner?', and 'Does the tutor is friendly and eager to help?'. The scales are represented by buttons with numbers 1 through 5, with '1' highlighted in blue.

Fig. 4: Tutor registration page, Questionnaire



MENU

Home

Login

Register

Student

Private Tutor

FAQ

Location

Fill the Location where you live.

Country

City

Street


Knowledge

Let us know more about you

Courses

Please select at least 2 courses

Fig. 5: Tutor registration page, Location



MENU

Home

Login

Register

Student

Private Tutor

FAQ

Knowledge

Let us know more about you

Courses

Please select at least 2 courses

Certificates

Cost per Hour

Create Account

2020-2021 Private Tutor Matching. Sami Odeh & Or Eliyahu | Software Engineering Department | Ort Braude College

Fig. 6: Tutor registration page, Location

8

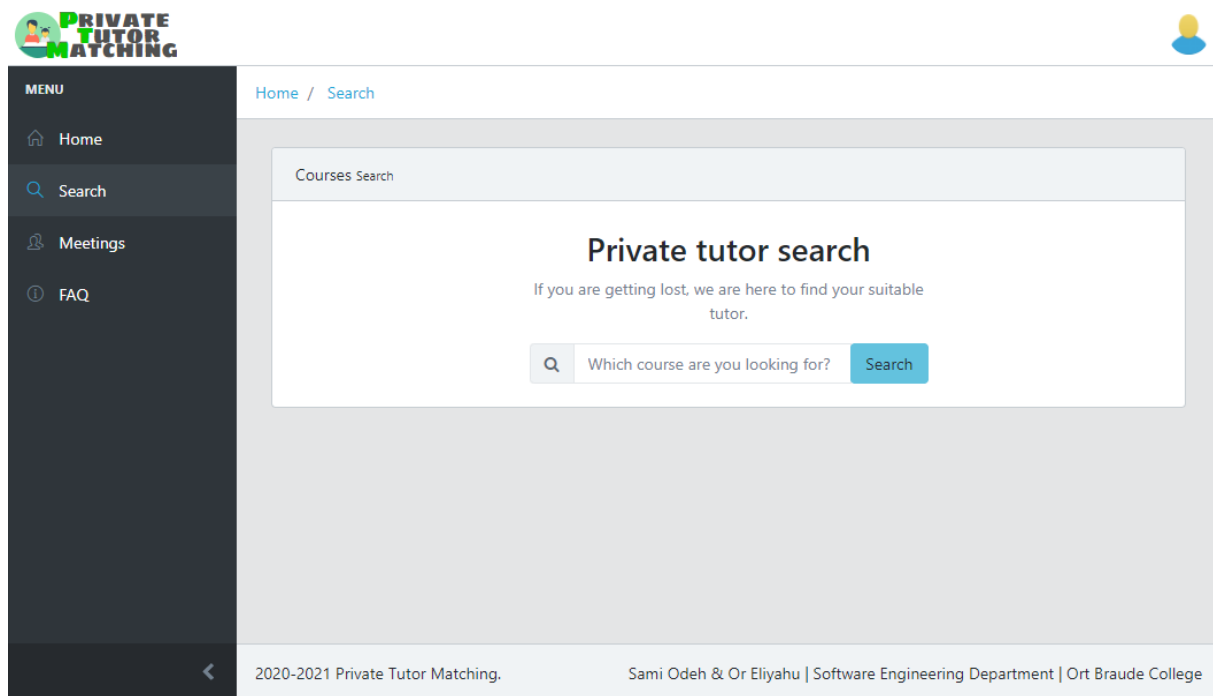


Fig. 7: Student – Search tutor page for a course

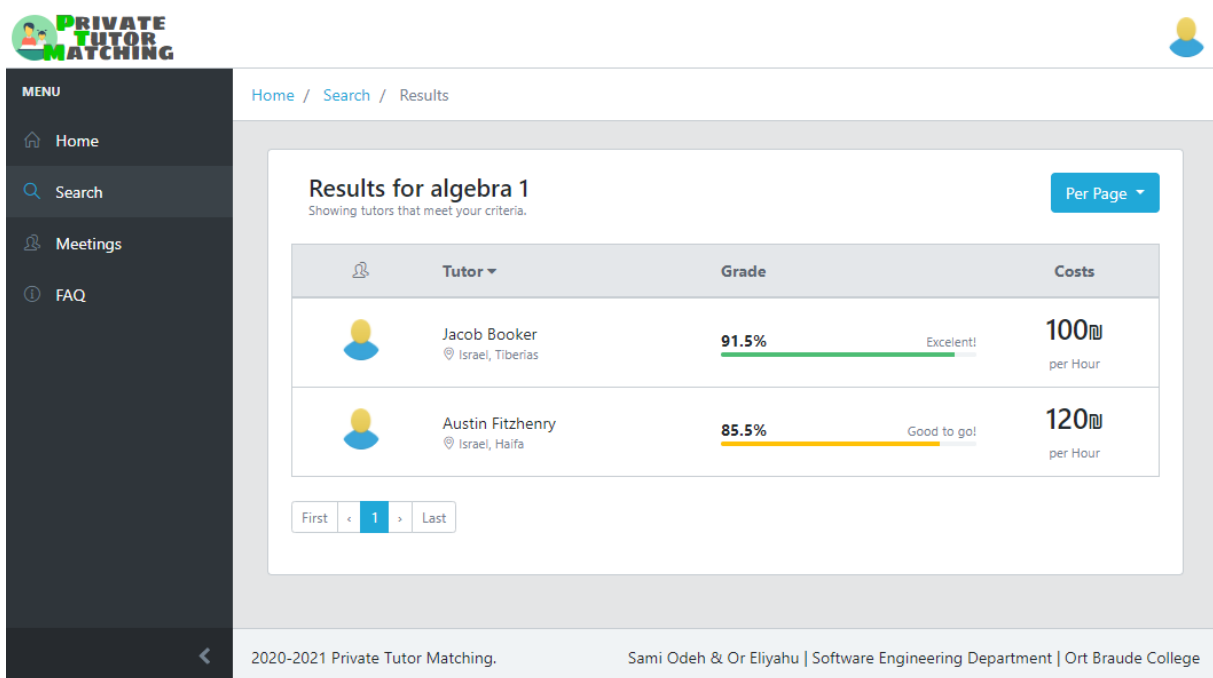


Fig. 8: Student - The suitable private tutors for algebra 1 course

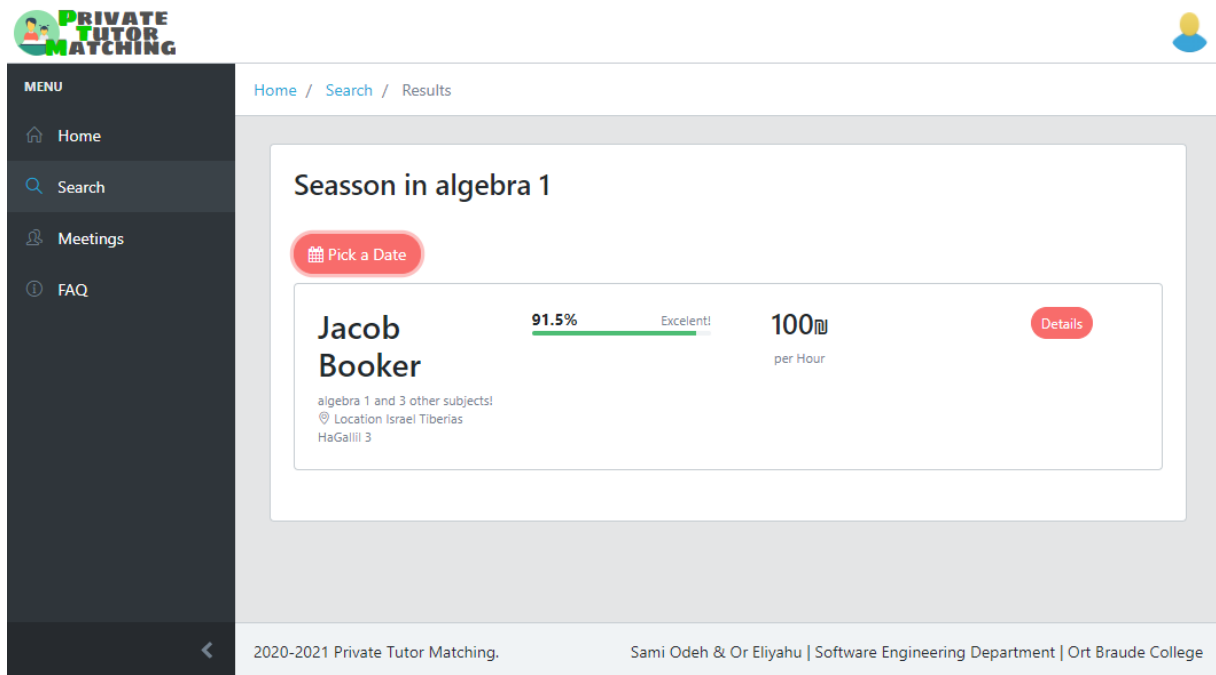


Fig. 9: Student – Request a meeting page

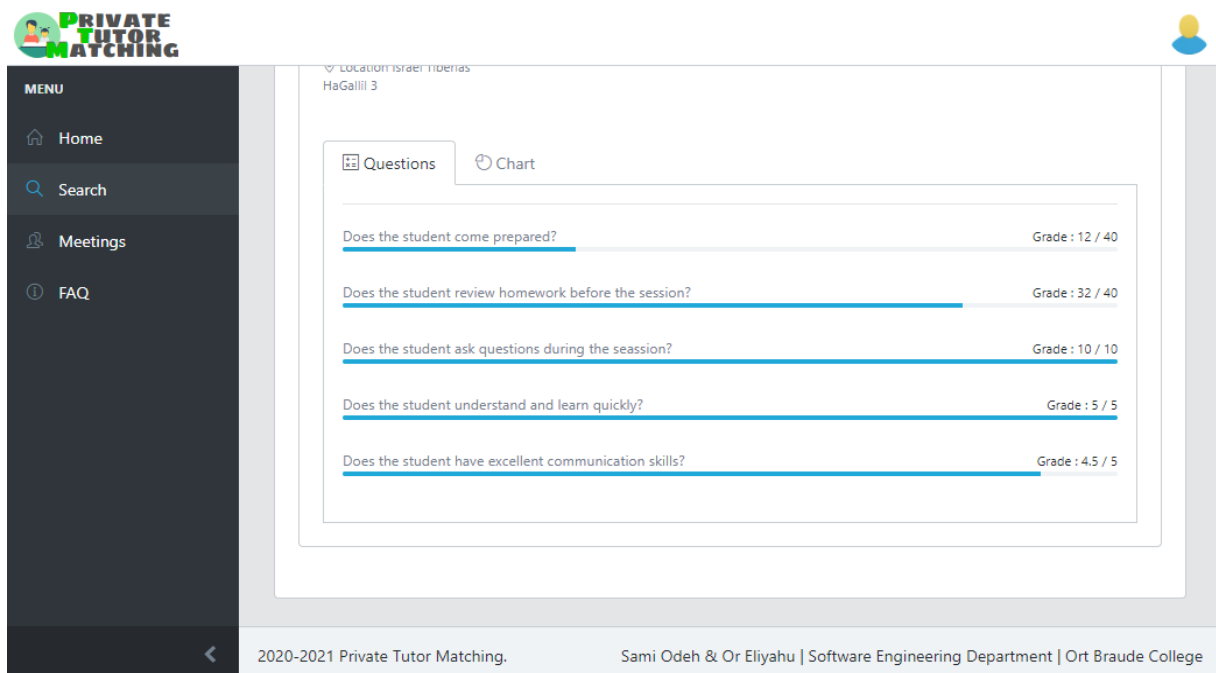


Fig. 10: Student – Request a meeting page, Matching details

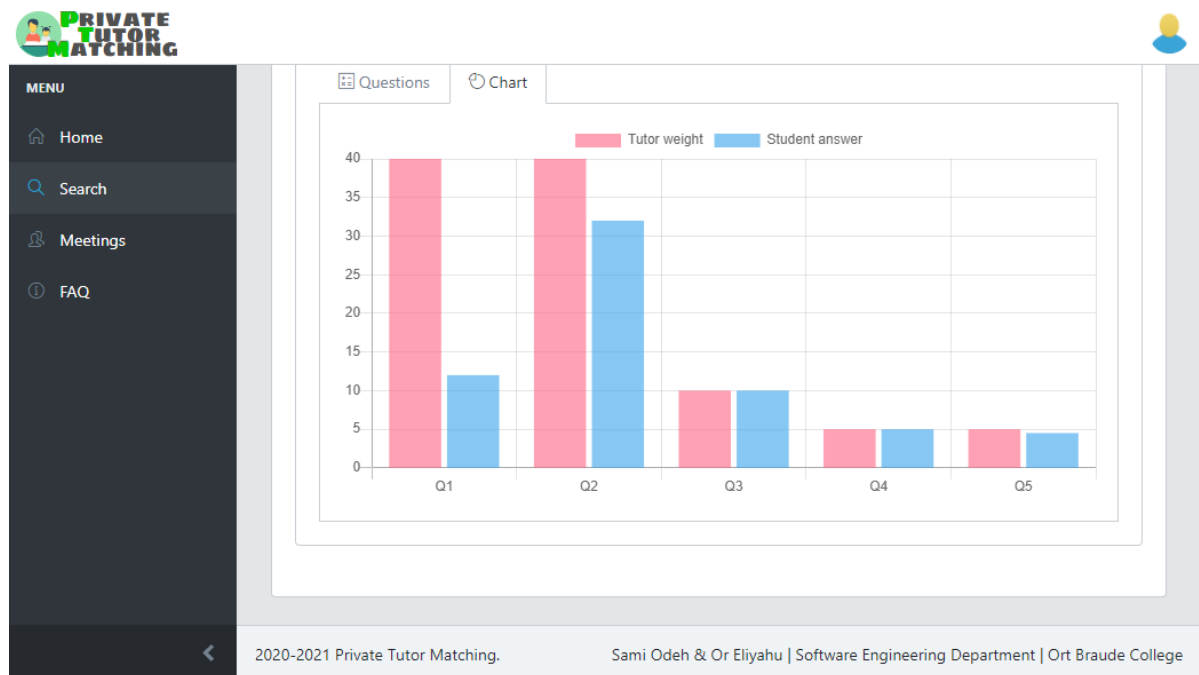


Fig. 11: Student – Request a meeting page, Matching details

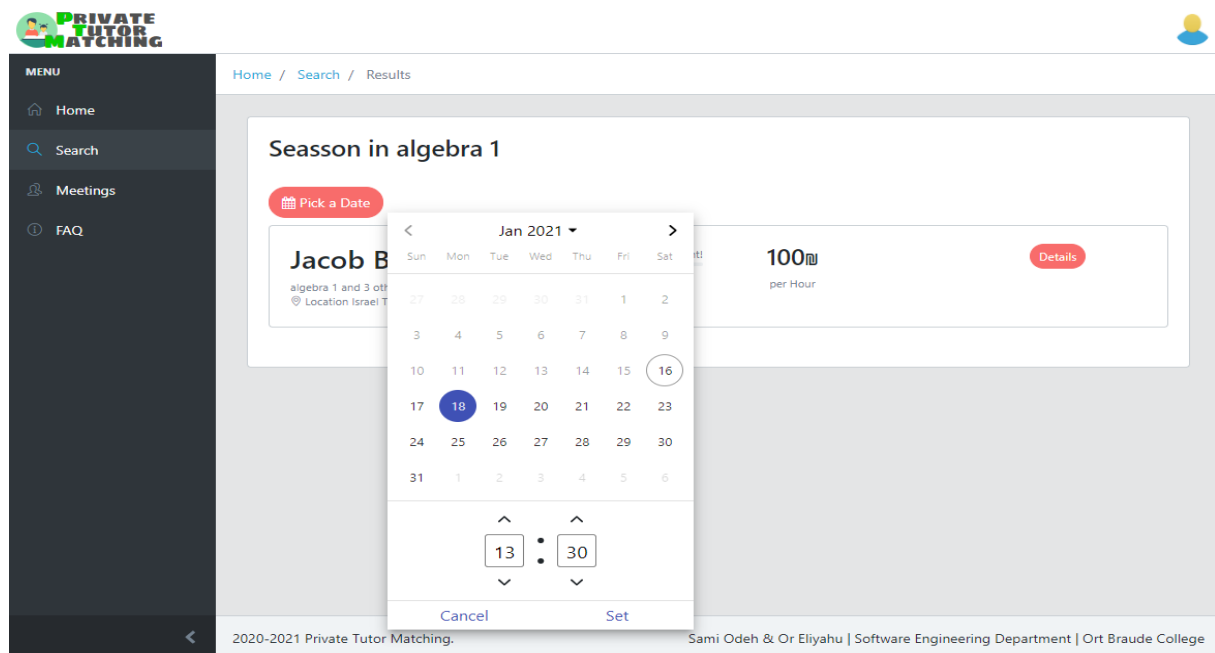


Fig. 12: Student – Request a meeting page, Pick a date

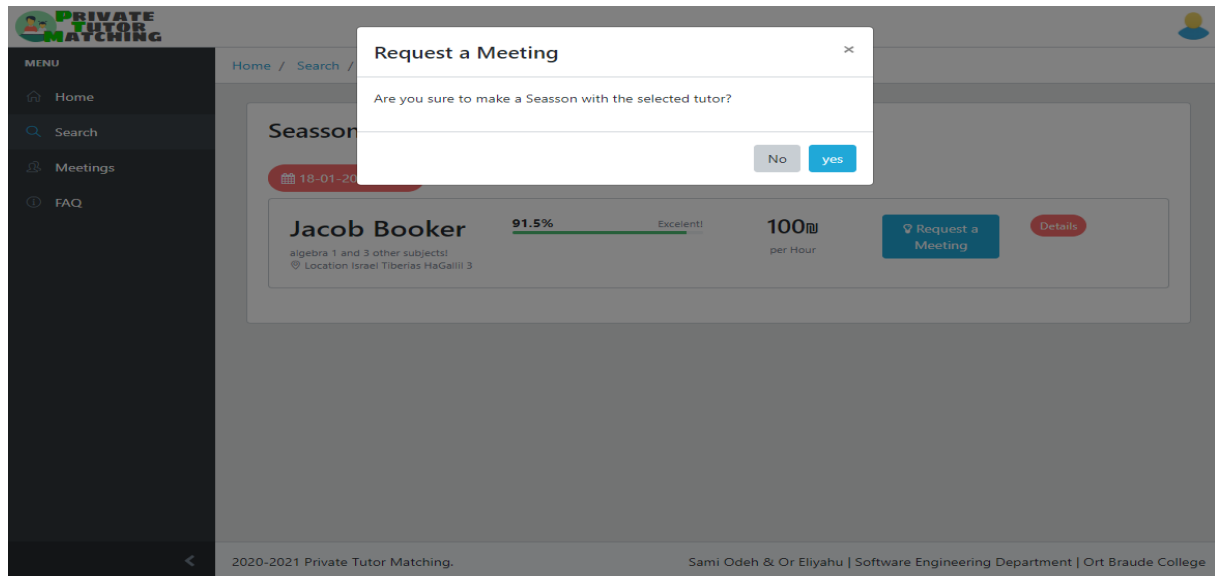


Fig. 13: Student – Request a meeting page

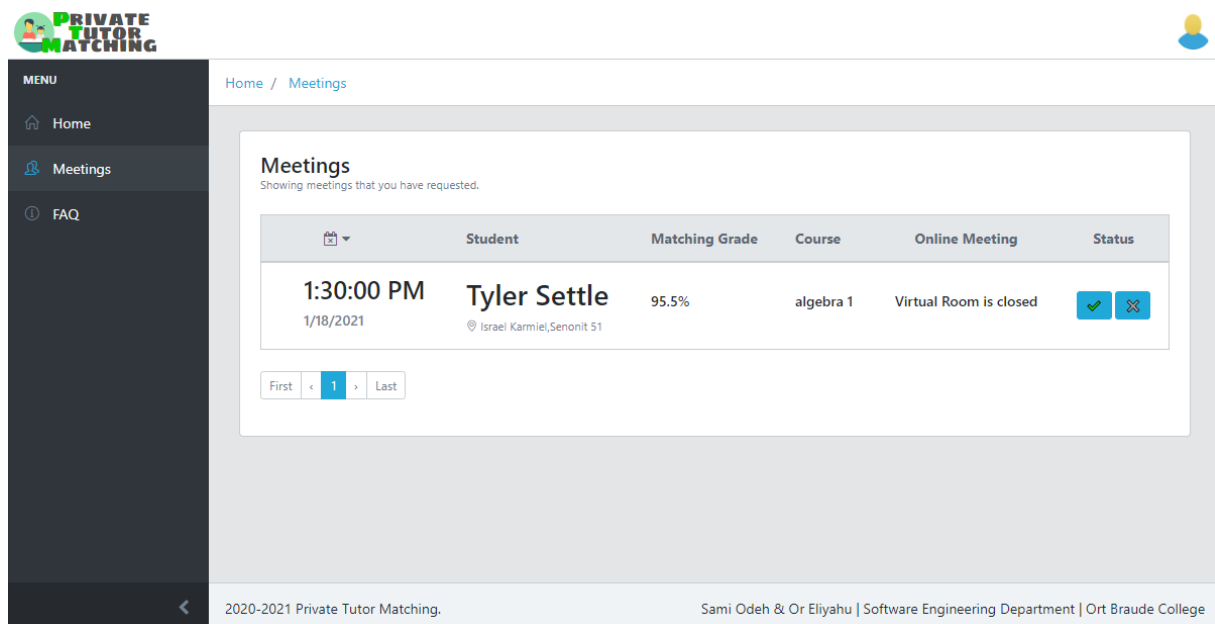
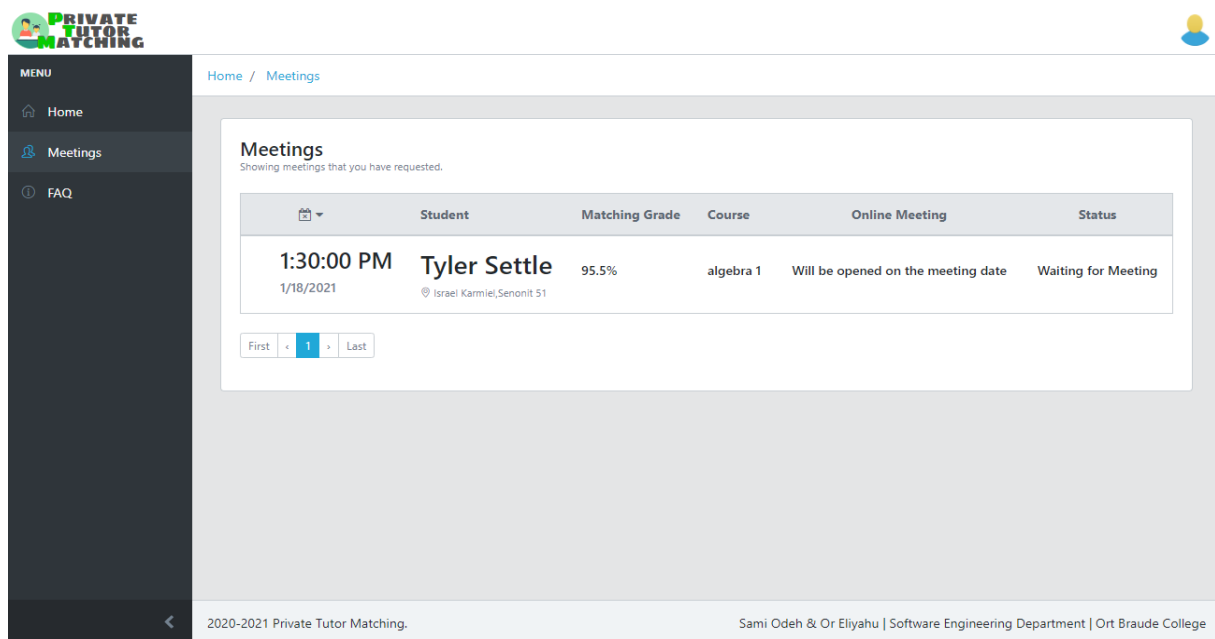


Fig. 14: Private tutor – Meeting requests page



PRIVATE TUTOR MATCHING

Home / Meetings

Meetings

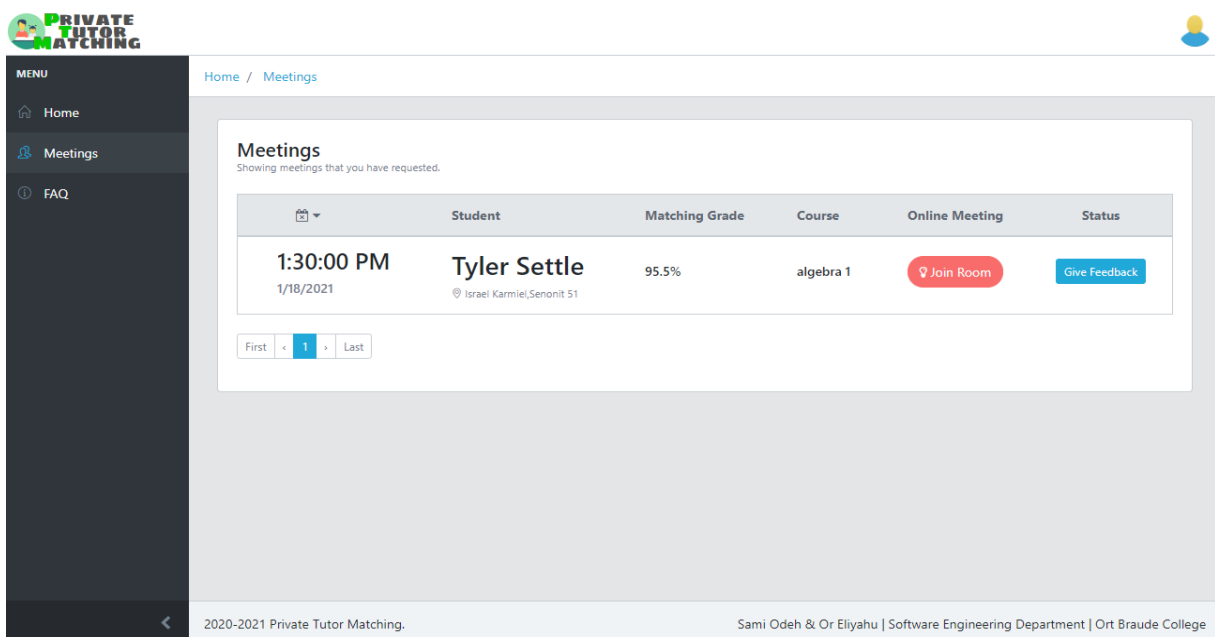
Showing meetings that you have requested.

	Student	Matching Grade	Course	Online Meeting	Status
1:30:00 PM 1/18/2021	Tyler Settle <small>Israel Karmiel, Senonit 51</small>	95.5%	algebra 1	Will be opened on the meeting date	Waiting for Meeting

First < 1 > Last

2020-2021 Private Tutor Matching. Sami Odeh & Or Eliyahu | Software Engineering Department | Ort Braude College

Fig. 15: Private tutor – Meeting requests page, Once the meeting was approved



PRIVATE TUTOR MATCHING

Home / Meetings

Meetings

Showing meetings that you have requested.

	Student	Matching Grade	Course	Online Meeting	Status
1:30:00 PM 1/18/2021	Tyler Settle <small>Israel Karmiel, Senonit 51</small>	95.5%	algebra 1	Join Room	Give Feedback

First < 1 > Last

2020-2021 Private Tutor Matching. Sami Odeh & Or Eliyahu | Software Engineering Department | Ort Braude College

Fig. 16: Private tutor – Meeting requests page, On the meeting day

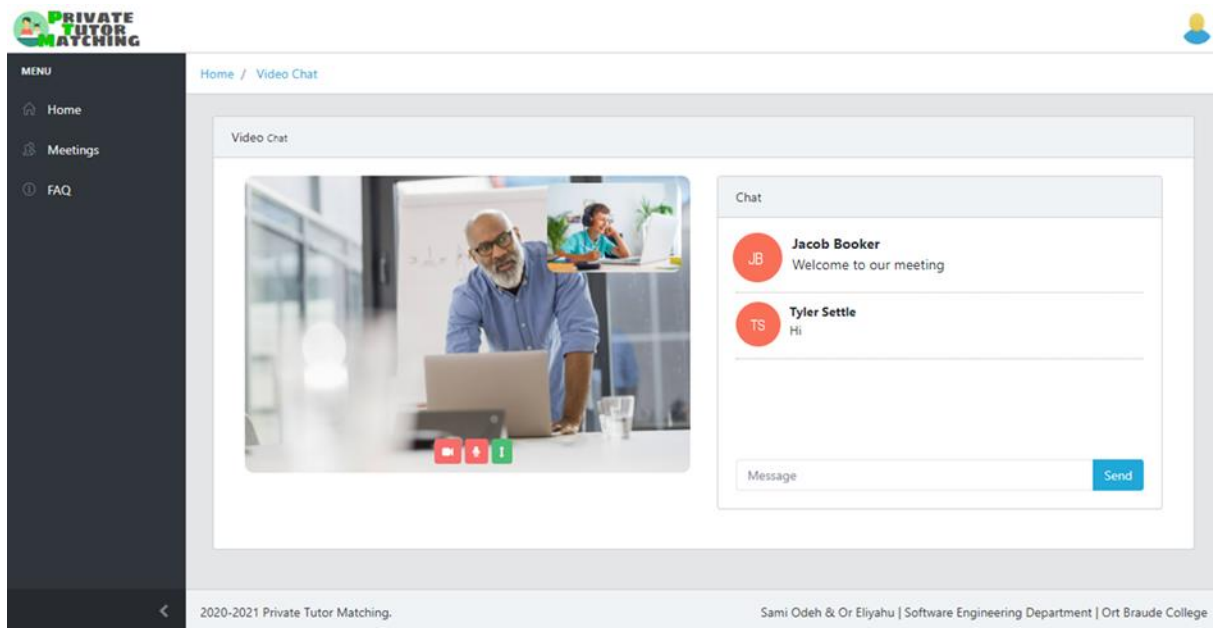


Fig. 17: Private tutor – Virtual room for the meeting

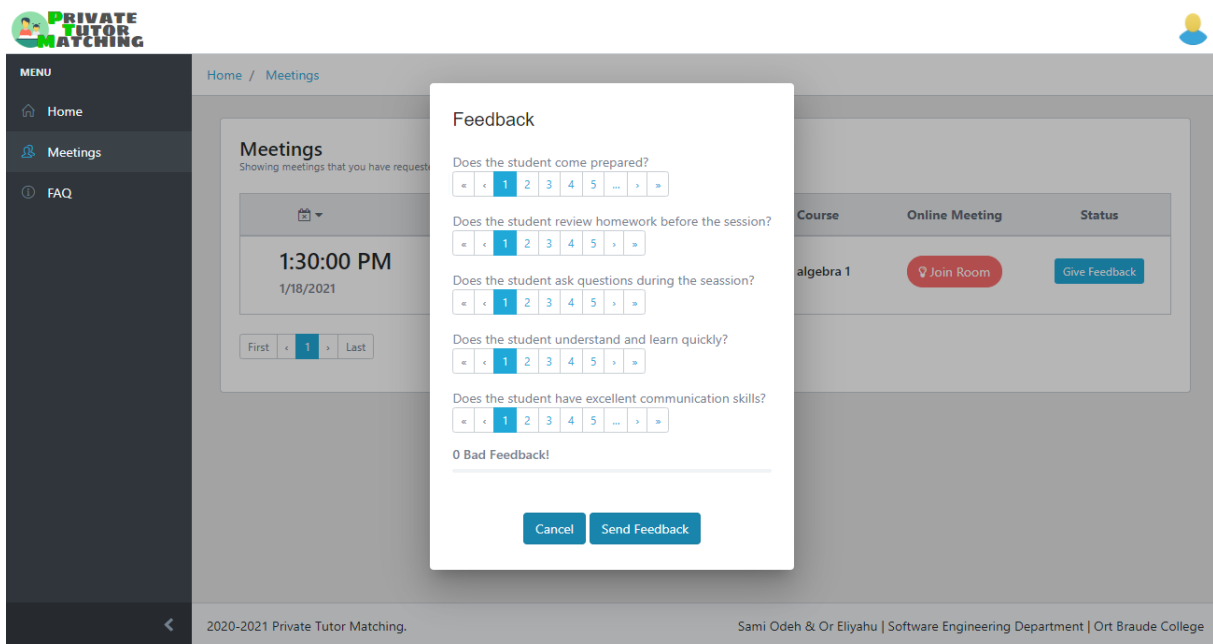


Fig. 18: Private tutor – Meeting requests page, Giving a feedback

3.2.2 Class diagram

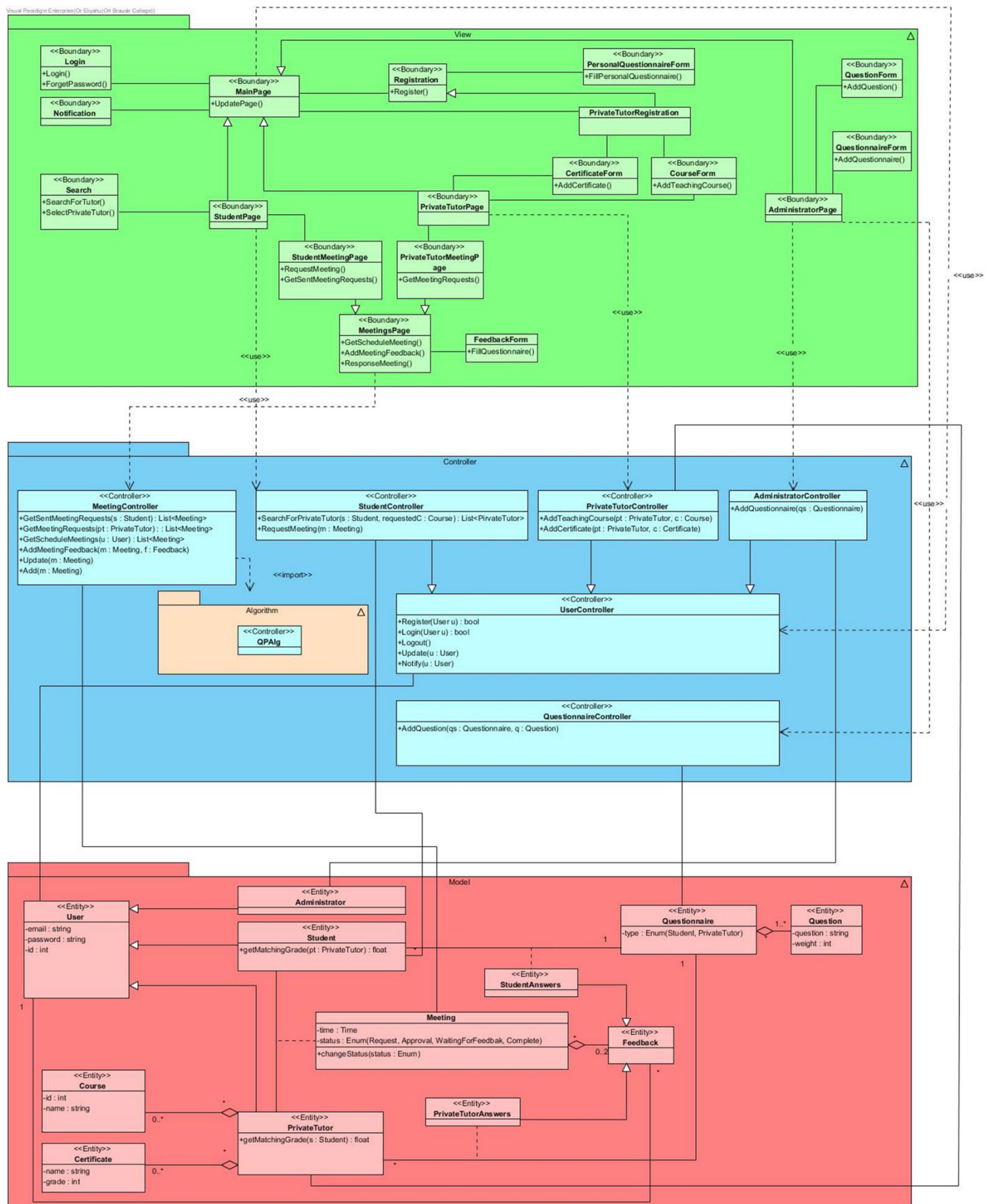


Fig. 19: Class Diagram

3.3 Testing Results

To check out the system performance we ran the program on some significant input:

Test ID	Description	Expected results	Actual Results
WrongUserName or WrongPassword	Entering invalid username and/or password.	Shows a message: "Wrong Username or Password".	Pass
UpdateWeightsVector	Once user response a feedback, obtain a new weights vector corresponding to the feedback.	Vector v: The best approximation weights vector which meets all the conditions of the weights.	Pass
ScheduleMeeting	According to the test weights, calculating the matching grade, if passed the minimum constant value (P), then he can send meeting request to the private tutor.	Private tutor received a meeting request for the scheduled time.	Pass
SearchSuitablePrivateTutors	After searching for a course, the system offers according to his properties the top suitable private tutors.	The search results page is ordered by the suitable private tutors matching grade.	Pass
ReceiveMeetingRequest	Once student send a meeting request, the request will be sent to the selected private tutor.	The meeting request has been added to the private tutor meeting page.	Pass
AddTeachingCourse	When the private tutor registers or he want to add more information about himself, he can add teaching courses to increase his options of teaching.	When searching for the newly added course, the system will calculate the matching grade of this private tutor.	Pass
AddCertificate	When the private tutor registers or he want to add more information about himself, he can add certificate to increase his options of teaching.	When searching for some course with the new certificate, the system shows the new certificate for the private tutor.	Pass

Table 1: Testing Results

4 RESULTS AND CONCLUSIONS

4.1 Results

Let us see an example of two students with the same answer vector that was looking for a private tutor, they made a session with the same private tutor and we can see how the weights and their answers vector changed according to the feedback.

Step 1: Student are looking for a private tutor (Tutor1), first Student 1, second Student 2.

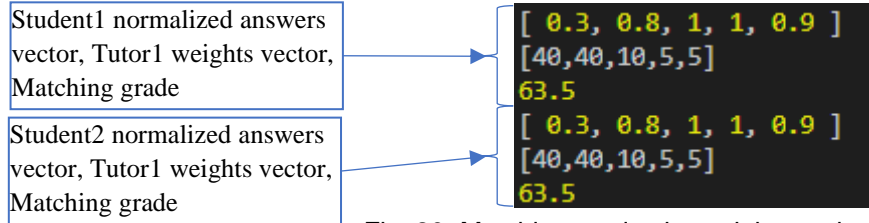


Fig. 20: Matching grades by weights and answers vector

Step 2: Tutor1 gave a bad feedback to Student2.

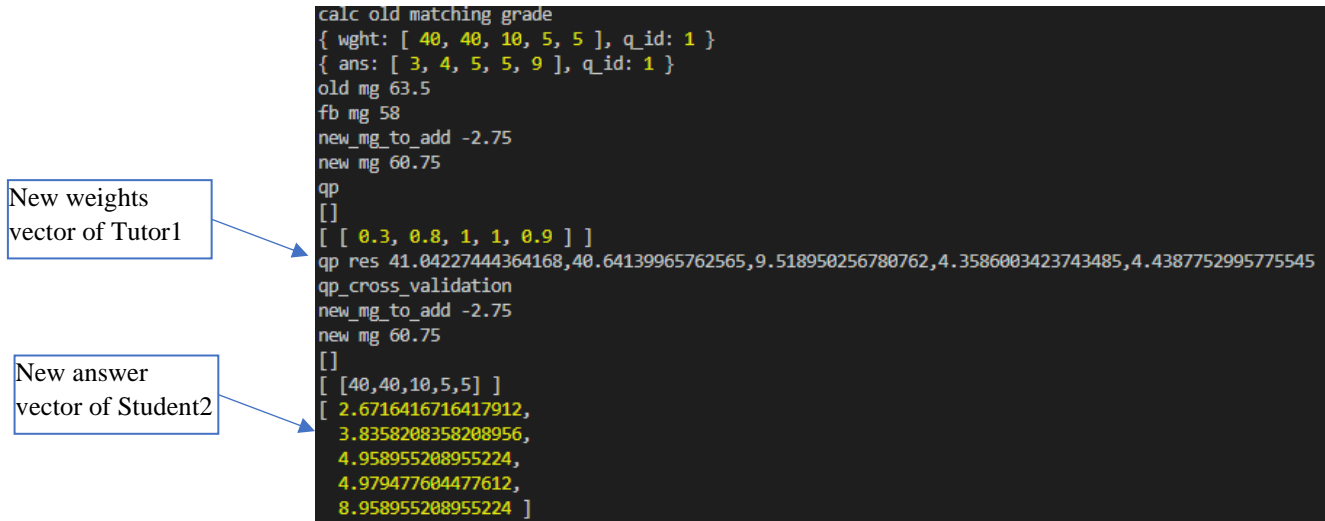


Fig. 21: Updated weights and answers vector

Step 3: Student are looking for a private tutor (Tutor1), first Student 1, second Student 2.

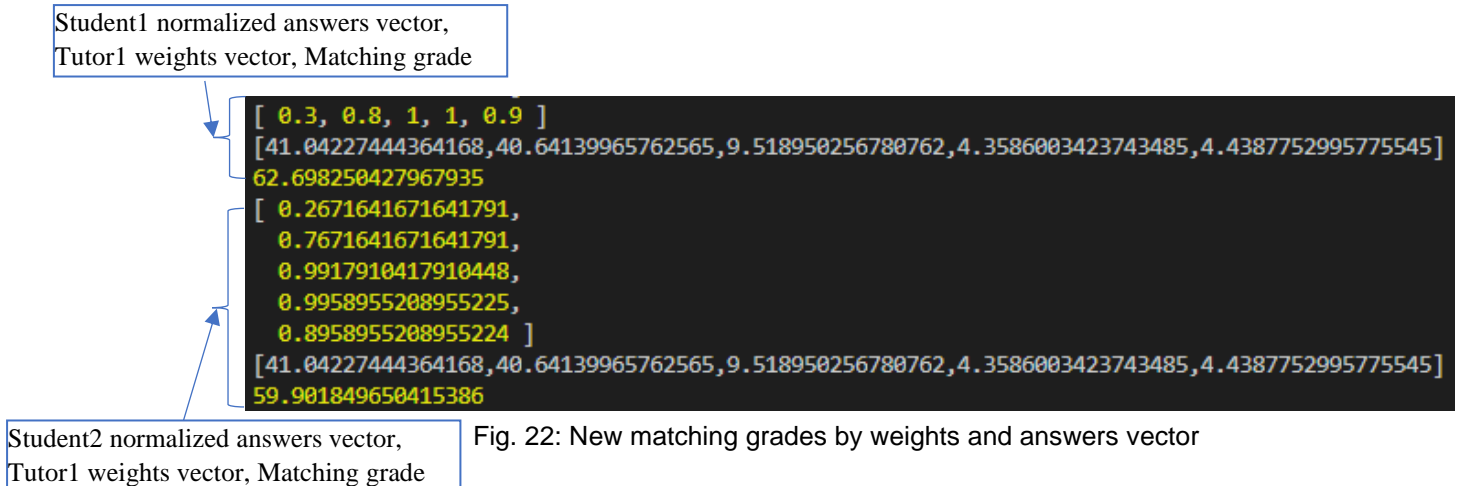


Fig. 22: New matching grades by weights and answers vector

Step 4: Tutor1 gave a good feedback to Student1.

```
calc old matching grade
{ wght:
  [ 41.04227444364168,
    40.64139965762565,
    9.518950256780762,
    4.3586003423743485,
    4.4387752995775545 ],
  q_id: 1 }
{ ans: [ 3, 4, 5, 5, 9 ], q_id: 1 }
old mg 62.698250427967935
fb mg 79.14431492296579
new mg to add 5.482021498332618
new mg 68.18027192630055
qp
[ [ 0.3, 0.8, 1, 1, 0.9 ] ]
[]
qp res 39.08573532842363,39.43737558672223,10.421968309958325,5.5626244132777645,5.492296361618044
qp_cross_validation
new mg to add 5.482021498332618
new mg 68.18027192630055
[ [41.04227444364168,40.64139965762565,9.518950256780762,4.3586003423743485,4.4387752995775545] ]
[]
[ 3.6492411755637844, 4.321449900717701, 5, 5, 9.070216276574012 ]
```

New weights
vector of Tutor1

New answer
vector of Student1

Fig. 23: Updated weights and answers vector

Step 5: Student are looking for a private tutor (Tutor1), first Student 1, second Student 2.

Student1 normalized answers vector,
Tutor1 weights vector, Matching grade

```
[ 0.36492411755637844,
  0.8642899801435402,
  1,
  1,
  0.9070216276574012 ]
[39.08573532842363,39.43737558672223,10.421968309958325,5.5626244132777645,5.492296361618044]
69.31488034525637
[ 0.2671641671641791,
  0.7671641671641791,
  0.9917910417910448,
  0.9958955208955225,
  0.8958955208955224 ]
[39.08573532842363,39.43737558672223,10.421968309958325,5.5626244132777645,5.492296361618044]
61.49398057920513
```

Student2 normalized answers vector,
Tutor1 weights vector, Matching grade

Fig. 24: New matching grades by weights and answers vector

We see that the matching grades are changing according to the Tutor1 feedbacks. In our example, at the end of the steps, the Student1 received good feedback have a higher matching grade than Student2 that received bad feedback.

4.2 Troubleshooting

Our project was very challenging, we faced several issues that require us to think of creative solutions during the development. In the following section we will discuss about them.

Reliability of the users – Once the private tutor or the student gave feedback, in fact, he triggered a change on his weights without thinking about the answer vector of the second side, therefore, for example if the student rated himself at the maximum rank for each question the matching grade for all private tutors would be always 100. So, we made the same process in order to reduce his answer vector.

Target matching grade – In phase A of our project we thought about a constant matching grade in order change the weights, but non-linear programming does not consider the feedback grade and it's reduced or increased the weights in a way that the new matching grade is very close to the constant matching grade. Therefore, for each calculation the system calculates a dynamic matching grade that consider all the previous feedbacks grade.

4.3 Conclusions

Over the years, technology has revolutionized our world and daily lives, but finding a suitable tutor still takes time and resources for students. Today's technology can help us to minimize time and resources.

In our project, we have designed and developed an application for tutor-student matching, which integrates a learning system that always improves itself by filtering most of the private tutors using a set of questions with changeable weights. The most suitable tutors will be selected with the highest-grade result and be sent to the student. To improve matches, the system will learn by an answered set of questions that the tutor and the student had done. It uses nonlinear-programing for that. This work aims to help students find a suited private tutor according to their learning skills and attitudes.

According to the results section, the weights and the answers vector changes according to the feedback and effects the matching grades for all the users, but non-linear equations is not giving us always an answer and that may be a problem in some cases, we have decided to avoid non-results but in the long term we must think about another solution for that.

For this application, specific tests are not enough to evaluate its efficiency and for that, we need to recruit real users to use the application for the long term and track the calculations after a period of time to make sure that we can achieve the project target.

The application can be used for various uses that are not profitable such as an internal website for a college to create connections between students for helping each other through private lessons.

REFERENCES

- [1] D. Hertenstine, "Distance and Metric Spaces", *Math Circle*, March, 2004.
- [2] D.P. Bertsekas, "Lagrange Multiplier Theory", *Nonlinear Programming (2nd ed.)*: 309, 1999.
- [3] G. Arfken, "Lagrange Multipliers", *Mathematical Methods for Physicists (3rd ed.)*: 945-950, 1985.
- [4] D. Zwillinger, "Lagrange Multipliers", *CRC Standard Mathematical Tables and Formulae*: 389-390, 2003.
- [5] J. Nocedal and S.J. Wright, *Numerical Optimization (2nd ed.)*, Berlin, New York: Springer-Verlag: 449, 2006.
- [6] P. Bradley, C. Hax, and L. Magnanti, "Solving Linear Programs", *Applied Mathematical Programming*: 40-421, 2015.
- [7] M.K. Kozlov, S.P. Tarasov, and L.G. Khachiyan, *Polynomial solvability of convex quadratic programming*, 1979.

- [8] M. Pardalos, and A. Vavasis, “Quadratic programming with one negative eigenvalue is NP-hard”, *Journal of Global Optimization*, 1991.
- [9] A. Floudas, and V. Visweswaran, “Quadratic Optimization”, *Handbook of Global Optimization*: 217, 1995.