Soft-skill Profile Based Group Formation System

Nikki Bouman
Vanisha Jaggi
Job Kanis
Mostafa Khattat
Fianne Stoel

Soft-skill Profile Based Group Formation System

by

Nikki Bouman
Vanisha Jaggi
Job Kanis
Mostafa Khattat
Fianne Stoel

Bachelor Endproject
Faculty of Electrical Engineering, Mathematics and Computer Science Delft University of Technology, June 2020

Presentation:	Thursday July 2, 2020	
Project committee:	C. Lofi,	TU Delft, Coach
	N. Zarin ,	OKademy, Client
	O. W. Visser,	TU Delft, Bachelor Project Coordinator
	H. Wang,	TU Delft, Bachelor Project Coordinator

TU Delft

Preface

This report concludes the TI3806 Bachelorproject of the Bachelor Computer Science at the Delft University of Technology. For ten weeks, we have worked on creating a so-called Soft-skill Profiling Based Group Formation system, a system containing a group formation algorithm. In this report, we will present the final product and the development process.

We would like to thank OKademy for offering us the chance to work on this project. We would like to thank Naqib Zarin in particular for his consultation. Special thanks to our coach, Christoph Lofi, who helped us making decisions and supported us weekly. Despite the current circumstances, they were both able guide us well during the entire project.

Nikki Bouman
Vanisha Jaggi
Job Kanis
Mostafa Khattat
Fianne Stoel
Delft, June 2020

Contents

1 Summary

OKademy is a start-up that wants to improve the healthcare in the Netherlands by improving the process by which graduated medical students are being matched to a hospital team. This is needed since there is a lack of surgery assistants, while graduate students need to wait on average six months to start their work in a hospital. To match quickly and sufficiently, the hard-skills as well as soft-skills of the student need to be known. To achieve this, universities need to provide OKademy with each student's soft-skills. However, this takes an excessive amount of effort for the universities. To let the universities benefit from this idea, OKademy wants to have a system which keeps track of the soft-skills and matches students in groups for assignments, thus being beneficial for universities and hospitals.

To help Okademy achieve its goal we developed a system that can be used by universities to help the students keep track of their soft-skills. The same system can be used to form theoretically well-functioning groups. These groups should be generated using an optimization algorithm that approximates the best groups regarding the soft-skills of students.

In this system, students, referred to as 'members', and instructors, referred to as 'hosts', can register. Members can be assigned into groups and hosts can create or modify groups and courses accordingly. Members can, after registration, fill in their top 10 soft-skills with a corresponding automatized grade. Based on these skills, they will be assigned to groups for a course created by a host. A course will go through multiple phases. When all groups have completed the assignment, a member is able to give feedback to five random soft-skills of their group members and recommend a new one if desired. Based on this feedback, their soft-skill grades will be adjusted.

The possible combinations of groups can be extremely large, therefore, it is not feasible to blindly search for the best group formation. Our algorithm will use the idea of genetic algorithms to explore the search space and approximate the best group formation.

The the final product consists of a web application in which the multi-objective group formation optimization algorithm is implemented. To work in a structured manner, we used the Scrum method. Meetings with our client and coach were held every week. The system has been tested by functionality tests, user tests and unit tests, ensuring a functional system. OKademy will use this system in collaboration with universities and hospitals to solve the problem.

2 Introduction

Due to the lack of surgery assistants, hospitals are forced to cancel planned surgeries. On average, medical graduate students have to wait six months (van den Broek et al., 2018) before they can start working at a hospital. OKademy aims to solve this problem by training medical students to become surgery assistants. After successfully finishing the program, these students will be posted at hospitals to help in the operation theater.

Before a candidate is proposed to a hospital, the hospital wants to know whether this candidate fits in one of their teams. Simply selecting the student with the highest grades does not guarantee success. OKademy's research shows that the right set of soft-skills are equally (if not more) important. In general, hard-skills are easy to prove, while soft-skills are even hard to measure, let alone prove. OKademy wants to keep track of the soft-skills of students during their study in order to create an accurate soft-skill profile of every student. To do this, a system needs to be built that can be used by universities.

To convince universities to use this system, there must be something in the system they can benefit from. During their study, students will participate in a large amount of group assignments. OKademy came up with the idea to expand the system with a group formation optimization algorithm. Soft-skills are one of the key factors in building a successful team. In fact, the absence of soft skills in the process of forming a team is one of the key reasons for the high rate of failure of projects (Agrawal and Thite, 2003). Since the system will keep track of the soft-skills of students, these soft-skills can be used to form theoretically well-functioning groups for the assignments. This provides universities with the option to automatically generate groups with the required soft-skills, while OKademy can keep track of every student's soft-skills and can see what soft-skill requirements for the groups have the best outcomes for certain assignments.

In this paper we represent a web application as a system to help automate the process of building a group with the described approach. At the heart of this web application lays an optimization algorithm used to match the individuals taking into account the requirements specified by the instructor. These requirements are a set of soft-skills.

This paper is structured as follows. In chapter 3 we formulate the problem and in chapter 4 we represent a solution to this problem. In chapter 5 we describe the technical details about components used in our product and how we test our software and maintain the quality of the code. More about the quality of our code can be found in chapter 6. Finally, a discussion and conclusion is provided.

3 The Problem

The problem that this thesis tries to solve is the optimization of a group formation algorithm which is used in a profiling system. This chapter will explain the general problem, after which the desired features the client has specified are sketched and translated to the profiling system.

3.1. General

Each student, referred to as a ‘member’, must be graded on certain soft-skills. However, the phrase "soft-skills" can be vague in the literature. To be consistent in forming a group based on the idea of soft-skills, a global taxonomy of soft-skills has been specified by the client (Appendix E).

Members will participate in courses, for which groups could be created. This creation of groups must be done by an algorithm that takes the soft-skills of the members and the preferences of the course specified by the instructor, referred to as ’host’, into account. These preferences are defined by a group size and an ideal composition of a team. The ideal composition of a team will be specified by soft-skills and stating the minimum and maximum amount of members per group that should possess a soft-skill.

The problem is divided into five phases. The desired features specified by the client, are further elaborated in each phase. These are summarized and ordered in Appendix C.

3.1.1. New user phase

The New user phase is the first phase. Every user will have to go through this phase once. A user, who can be either a member or a host, creates an account to use the system.

A member is asked to fill in ten self-claimed soft-skills, which can be chosen from a list (Appendix E). The member will have to sort the ten soft-skills based on their confidence level. Every soft-skill contains a grade and an amount of votes. These soft-skills will be automatically graded as visualised in Table 3.1. The amount of votes will be zero by default.

Top 10	**Grade**
1	8.0
2	7.8
3	7.6
4	7.4
5	7.2
6	7.0
7	6.8
8	6.6
9	6.4
10	6.2

Table 3.1: The top 10 skills and their corresponding grade at initialization.

3.1.2. Enrollment phase

When a host creates a new course, he is asked to fill in the details and the preferences for the course. These details include an enrollment period, defined by a starting date and an ending date.

During the Enrollment phase, members can enroll for the course. Every member that is enrolled for the course will be taken to the Matching phase.

3.1.3. Matching phase

After the enrollment period, the Matching phase starts. Members can no longer enroll for this course. The system will now need to create the optimal composition for the total set of students enrolled for the course, taking the preferences of the course into account. The following are the criteria for what constitutes a well-functioning group:

- A member possesses a soft-skill if the soft-skill is graded $\geq$ 6.5. When no information about a soft-skill is present, it is considered as if the member does not possess the skill.
- The host has indicated which soft-skills are relevant for the course, containing a minimum and a maximum. Exceeding the maximum and not reaching the minimum is considered as equally bad. The exact number of soft-skills present in a group does not matter as long as it is between the minimum and the maximum.
- The overall availability of soft-skills in groups should be as equal as possible among groups.
- Two different skills not meeting the requirement by one is preferred over one skill not meeting the minimum requirement by two; small deviations from multiple criteria are preferred over one big deviation from one criterion.

These criteria can change, asking for a dynamic approach for approximating the best group formation.

3.1.4. Work phase

The Work phase starts when every member is assigned to a group. The members can start working in their groups. Nothing happens in the system. However, hosts are still able to alter course with corresponding groups if necessary.

3.1.5. Feedback phase

The Feedback phase starts when the feedback start date of the course has arrived and ends after the feedback end date. During this phase, members can give feedback to their group members. When giving feedback to a member, five soft-skills are randomly selected from the highest graded soft-skills of that member, excluding everything past the first twenty. The corresponding grade is shown for every soft-skill. The member can now say for each soft-skill whether this grade is too high, too low or correct. This feedback influences the grade for that soft-skill and increases the amount of votes for that soft-skill by one.

Besides giving feedback to the randomly selected soft-skills, the member can choose to recommend a soft-skill, thus stating that that member possessed that skill during the Work phase. If that soft-skill is not yet in the soft-skill list of the member that is given feedback to, the soft-skill will be graded with a 6.0 and the amount of votes will be set to one. However, it is possible that the member that is given feedback to does already have this soft-skill in his list. In that case, the recommendation does not influence the grade, but does increase the amount of votes for that soft-skill by one.

Lastly, the member can give feedback to the group in general. The user can state whether the group's soft-skills composition was bad, sufficient or good for the collaboration. Currently, the system will save this information and not process it, since the client would like to use this data for further research.

After the feedback phase, the course has ended. The course will remain in the database as data for further research.

3.2. Group formation algorithm

Based on the specifications of the Matching phase, we have defined requirements for the algorithm we are going to use. The algorithm accepts the requirement inputs from the web application as follows:

- A list of members: $\{m_0, m_1, ..., m_n\}$
- A list of all Soft-skills: $\{s_0, s_1, ..., s_m\}$

- The soft-skills that members have. This is a subset of all soft-skills.
  The size of the list of soft-skills of a member depends on the grades of the member, i.e. the list can be empty or as big as the list of all soft-skills. There must be no duplicates in this list.
- The constraints that are provided by the host. These constraints describe for a subset of all soft-skills in what amount these soft-skills need to be present in every group. An example of constraints would be:
  - a minimum of 2 and a maximum of 5 members should possess soft-skill s_0
- The group size. This is the size of each group, which the members are divided into.

With this input, the algorithm should provide well-functioning groups of the members, using the definitions defined in the Matching phase. However, it might not always be possible to create groups that are all within the given constraints. In this case, it is the task of the algorithm to optimize the group formations. To optimize these group formations, we need to know which properties of the group division need to be optimized and how. After discussing this with the client, it was concluded that the following two properties need to be optimized:

- A group of members can be rated according to the deviation from the constraints given by the host. This rating needs to be optimized for all groups, i.e. the groups need to deviate from the constrains as little as possible.
- A division of groups can be rated according to how similar the ratings of the groups are. This rating is good when the groups' individual ratings are similar. This rating is bad when there is a big difference in the groups' individual ratings. This is meant to make sure all groups have equal chances of working well.

Lastly, since the choice of algorithm depends on the size of the expected input, we asked the client to how large a dataset the algorithm should be optimized. The client defined the following values for which the algorithm should perform best:

- The amount of members. The expected amount of members that the algorithm will be ran on is between 30 and 100.
- The group size. The expected group size that the algorithm will be ran on is between 2 and 5.
- The time that the algorithm has to run. The client has not defined concrete limits for the time the algorithm can take. However, he told us to make sure that the waiting times are reasonable.

3.3. Product design goals

From the requirements of the client, the following goals for the application are defined:

- Intuitive: The product should be intuitive and easy to understand. This means that the structure of the application should be logical and that the provided documentation should be clear.
- Adaptable: This means that the application should be modularized. The algorithm should be able to adapt to a new definition of what constitutes a good group.

4 The Multi-objective Group Formation Optimization Algorithm

Creating an algorithm that meets the requirements is not easily done. To illustrate the difficulty of the problem, we will show a calculation of the amount of possible solutions to consider. The minimal amount of members that the algorithm is expected to run on is 30 and the smallest expected group size is 2. Sorting 30 people into 15 labelled groups is possible in $\binom{30}{2}\binom{28}{2}\cdots\binom{4}{2} = \frac{30!}{(2!)^{15}}$ ways. Since our groups are not labelled, a correction is made by dividing the result by $15!$. This gives the result: $\frac{30!}{(2!)^{15}} \cdot \frac{1}{15!} \approx 6 \cdot 10^{15}$. This means that if we wanted to find the optimal grouping of 30 members into 15 groups by trying out all possible solutions, we would have to compare all $6 \cdot 10^{15}$ possible solutions. It can be seen that this is not feasible using a brute force algorithm, despite the fact that this input gives the smallest amount of possible solutions. This means that other kinds of algorithms had to be considered for this problem.

When considering different types of algorithms, it is important to know the requirements of the problem. The current problem definition is clear in its requirements. However, this was not the case when we started our research at the start of the project. The initial research question defined by the client was "How to efficiently form collaborative learning groups from students". For this we decided to search for papers attempting to effectively form collaborative learning groups based on student profiles. What we found was a broad collection of research-papers addressing the problem in different ways. Some papers attempted to base the group formation by putting together similar students (Jozan and Taghiyareh, 2013), while others tried to maximize the diversity among the students (Sukstrienwong, 2012). From this we concluded that there does not exist a straightforward way to effectively form high performing groups and that this method should be defined by the client. When discussing this with the client, slowly a more specific definition of our problem started to form, resulting in the problem definition being fully defined shortly before the end of the research phase. In the meantime our strategy was to collect methods used in papers addressing comparable problems, so that eventually we could compare the methods and choose the best option that can be used for the problem definition to be proposed by the client. For this we found four algorithms that are commonly used for approximating group formations: Ant Colony Optimization, Clustering Algorithms, Genetic Algorithms and Particle Swarm Optimization (Maqtary et al., 2019). We have made a comparison of these four methods, listing the advantages and disadvantages of using them as well as discussing the accommodations that would have to be made to use these kinds of algorithms for our problem. In retrospect, a few of these algorithms do not fit our current problem and should not have been considered. Nevertheless, since this comparison was eventually used for deciding to go for the genetic algorithm, the comparison is included in Appendix D. Based on this comparison, we have decided to use a genetic algorithm.

4.1. Genetic algorithm

In this section, the aspects of genetic algorithms are described. First, we will describe the general properties of a genetic algorithm. Then, we will go into more detail about some of the advantages and disadvantages of using this algorithm. Lastly, we will explain how our problem can be mapped to this algorithm.

4.1.1. Theory

A genetic algorithm is an evolutionary algorithm that makes use of the principle of evolution. By representing a possible solution as a chromosome and defining a function that can be used to determine how ‘good’ a solution is, the algorithm is able to use previous solutions to create better ones. For this, first, a group of random chromosomes, or solutions, is created. Such a group of chromosomes is called a population. From this population the best solutions are selected. Afterwards, two operators are applied on these selected algorithms to introduce genetic diversity: the crossover and the mutation. The resulting population is a new generation. After a predetermined amount of new generations are produced, the algorithm will stop and the best chromosome can be taken as a result.

Since, generally speaking, the bad solutions are removed and the good solutions are kept and slightly changed, the chromosomes in the population will improve every generation. If the algorithm is allowed to run for enough generations, the best solution it produces should be close to the actual optimal solution to the problem.

Motivation behind the genetic algorithm

When choosing the algorithm for solving the described problem, it was important that the algorithm would be able to solve the problem effectively. While, with a genetic algorithm, it is not possible to know whether you found the best possible solution, we believe that, based on its effectiveness in comparable problems (Moreno et al., 2012; Zheng et al., 2018), the found solutions would be good enough. Beside its effectively, the genetic algorithm seems to also fit the Product design goals rather well.

One of the biggest advantages of a genetic algorithm is that it hardly makes assumptions about the problem it is trying to solve (Whitley, 1994). Consequently, when the definition of what constitutes a ‘good’ group changes, the fitness function can be transformed without needing to change other parts of the algorithm. This makes the algorithm easy to adapt.

The genetic algorithm is not only adaptable, it is also intuitive. Since the theory behind genetics is based on the knowledge of genomes found in the field of biology, the theory behind the algorithm may already be known. Also because the algorithm is naturally divided into different parts, the algorithm can be organised in a logical and intuitive way.

Since a genetic algorithm is adaptable, intuitive and able to solve the problem quite effectively, we believe the genetic algorithm is a good choice for solving the optimization problem.

Limitations of a genetic algorithm

A genetic algorithm comes with a few limitations. First, it is impossible for the genetic algorithm to determine whether the found solution is the optimal solution. Since the algorithm is approximating the optimal solution, the result can still be considered as good. The challenge, however, is how to configure the algorithm to have the best chance of reaching the best solution in the least amount of time.

A second limitation of a genetic algorithm is that it is not straightforward to optimize for multiple objectives. While it is possible to have weights turn multiple objectives into a single objective, the challenge how to determine those weights remains. Certain weights will provide solutions which might or might not be wishful as a group division. It will require some effort to find the correct weights

A third limitation of a genetic algorithm is that the fitness function needs to be fast (Whitley, 1994). The fitness of a chromosome has to be calculated for every chromosome found, which means that the speed of the fitness function has great impact on the performance of the algorithm. Since, for the defined problem, a large amount of data has to be compared for calculating the group quality, it remains a challenge to keep the performance of the algorithm high.

4.1.2. Problem encapsulation

In this section, we will explain how it is possible to map all properties of our problem to an input of a genetic algorithm. First, we will explain how chromosomes can be made such that it defines a solution to our problem. Then we will talk about the fitness function that is used to evaluate solutions. Next, we will discuss the operations that can be applied on the chromosomes; selection, crossover and mutation. Lastly, we will talk about populations and generations.

Chromosome

In a genetic algorithm, possible solutions to the problem have to be defined as a chromosome, which means that a possible group formation has to be represented as a chromosome. Similar implementations of group formation problems provide two ways to do this. The first one is to define the genes in the chromosome as members and to use the position of the genes to indicate the group that the members are in (Moreno et al., 2012; de Oliveira da Silva et al., 2009). The second option is to use the genes to indicate the groups and the positions to indicate the members (Zheng et al., 2018; Sukstrienwong, 2012). Since the authors did not include any information about how their method was reflected in the effectiveness of the algorithm, we believe that both methods can be seen as equally effective. For the algorithm, the decision was made to define genes as members and have the positions of the genes in the chromosomes represent the different groups. This will make the encapsulation more intuitive when evaluating the chromosomes and will make it easier to control the group size constraints. We subsequently decided to organise the positions by defining adjacent members in the chromosome to be in a group together. This means that when for example the group size is five, the first five members are in one group, the next five members are in another group, et cetera.

Fitness function

The fitness function is a function that should provide an indication of the overall quality of a solution. For the defined problem, the quality of a solution is determined by two components: the quality of the groups that are formed and the similarity of the quality of these groups. In this section, an explanation is provided on how these components are determined from a solution.

Quality of groups - Groups should be formed by the algorithm such that they deviate from the constraints as little as possible (i.e. the sum of all skills of all members of a group should aim to be within the defined minimum and maximum of the corresponding skills in the constraints). A simple measurement of how well this is done is counting how often and how much the skills of the group deviate from the constraints. We will call these deviations from the constraints "deviation points". These deviation points are an indication of how well the group is formed; the less deviation points a group has, the closer it is to the constraints. The sum of all deviation points of all groups in a group division gives a good indication of the quality of all groups.

Similarity of quality - While the quality of groups of the fitness function gives a good indication of how close groups are to the constraints, it does not account for the client's preference of multiple smaller deviations from the constraints over one big deviation from the constraints. This is why a similarity of quality component is needed in the fitness function.

The similarity of the quality of the formed groups can be measured by taking the standard deviation of the deviation points of the groups. The smaller the standard deviation of the deviation points, the more similar the deviation points of the groups are. Minimizing this parameter ensures that multiple groups having a small amount of deviation points are favoured over one group having a big amount of deviation points.

Selection

Selection is the process of evaluating all solutions in a population and then discarding some solutions based on that evaluation. There are different methods that can be used for selection. One of the often used methods is called the tournament selection. This method has one variable, the *tournament size*. In this method, first the solutions get evaluated with the fitness function. Afterwards, tournaments take place. In a tournament, a *tournament size* amount of random solutions get chosen. From these random solutions, the solution that had the best evaluation wins the tournament. Such tournaments are executed until the group of tournament winners is the size of the population size. This group will become the new population.

Crossover

Crossover is a genetic operator that can be applied to chromosomes. It takes two chromosomes and combines their genes to make a new chromosome.

Not all methods of crossover can be applied to these chromosomes. Simply combining parts of both parent chromosomes would produce a chromosome with duplicate genes. The solution that would correspond with this chromosome would not be a valid group formation.

A method of crossover that takes this restriction of chromosome composition into account is the partially mapped crossover. This method takes an arbitrary amount of consecutive genes from the first parent chromosome as a base for the child chromosome. The rest of the genes of the child get filled in with the genes of the second parent. This solution, which contains duplicates, gets corrected by replacing the duplicates that were introduced by the genes from the second parent.

Mutation

Mutation is another genetic operator that can be applied to chromosomes. Not all methods of mutation can be applied on our chromosomes because not all chromosomes produce valid solutions.

One method of mutation that can be applied to our chromosome is shuffling the indices of the genes. Swapping genes in a chromosome is the equivalence of swapping members between groups.

Population and generations
In a genetic algorithm, a group of chromosomes is called a population. Within a population, there needs to be some genetic diversity, since this makes it possible for the algorithm to explore a bigger search space. To achieve genetic diversity, a population needs to be big enough. However, what is big enough depends on the problem the algorithm is solving and the input.

While running the algorithm, a specific instance of a population can be called a generation. A new generation is created every iteration of the algorithm. A genetic algorithm needs to run for enough generations to produce a solution that is close to the optimal solution. However, what is enough generations depends on the input and on the problem the algorithm is solving.

Both the population size and the generation amount are variables that can be optimized for a specific problem and input. Since we cannot predict what the optimal values for these variables are, we will have to determine these later.

4.2. Languages and framework

We have chosen Python as the programming language for implementing the algorithm. Python is widely used in different computational intelligence fields (e.g. machine learning, Genetic Algorithms etc) and because of that, it has a large community with a great amount of libraries and frameworks. The language itself is intuitive. As a result we can focus more on the algorithm itself rather than writing code.

We have used the framework DEAP[1] framework to speed up the implementation. DEAP seeks to make the algorithm explicit and its data structures transparent. Moreover, it offers a large variety of operators that can be used in genetic algorithms.

4.3. Implementation

For the implementation of the algorithm, a number of design choices were made and used for designing the structure of the application. First, we will discuss the design choices made and explain the reasoning behind them. After that, the overall design of the application will be discussed.

4.3.1. Design choices

To make the application intuitive and adaptable, a number of design choices are made. First, to increase the adaptability of the application, the algorithm is split up into three components: a data component, a genetic component and a main component. The data component is responsible for all functionality surrounding the input data of the algorithm. The genetic component is responsible for all functionality surrounding the genetic algorithm itself. The main component handles the communication with the web application and stitches the components together. Changing one of the three components can easily be done without needing extensive changes in the other two. Second, external libraries, like the DEAP framework, are used in a single class. This makes it easier to potentially switch to another framework, increasing the adaptability of the algorithm. Third, with the use of classes and packages, the algorithm enforces grouping similar functionalities together. In addition to preventing duplicate code and larger numbers of parameters per function, this increases the understandably and intuitiveness of the application. Fourth, parameter types and return types are always defined, making it easier to understand the functionality of a method and decreases errors due to dynamic typing in Python. To make the type definitions easier and more intuitive, custom types are used. This makes it simple to change data-types and makes methods more readable. Fifth, the web application is able to define the input for the genetic algorithm. To make this possible, configurations can be passed from the web application to the algorithm to be used when running the genetic algorithm. This makes it easy to customize the settings of the algorithm and it adds more configurations to the algorithm.

4.3.2. System overview

The three components of the algorithm are described in Figure 4.1 with their corresponding connections; the data component is connected to the genetic component via the $\mathrm{DataManager}$ and the $\mathrm{FitnessFunction}$, the genetic component has a single connected to the DEAP-framework and the main component connects to the data component and the genetic component by providing required data and a configuration class. To go into more detail, these components will be discussed separately.

[1]Distributed Evolutionary Algorithms in Python, https://github.com/DEAP/deap

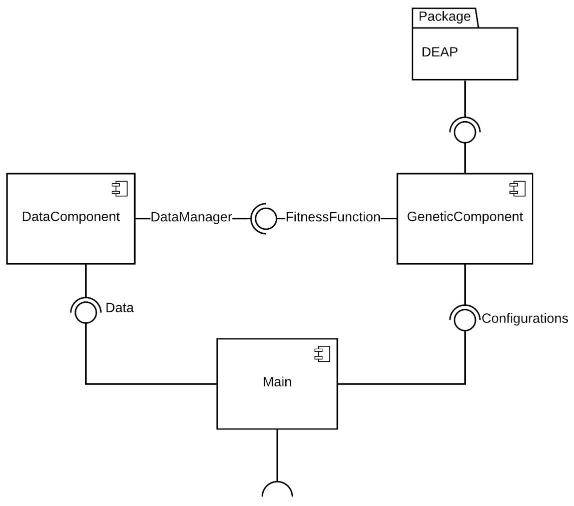


Figure 4.1: Algorithm Component Diagram

Genetic component

The genetic component is responsible for running the genetic algorithm and, as visualised in orange in Figure 4.2, consists of the following classes:

- The GeneticAlgorithm class, containing the connection to the DEAP-framework and being responsible for running the entire genetic algorithm.
- The GeneticAlgorithmHelper class, responsible for managing the run status of the GeneticAlgorithm.
- The FitnessFunction class, which provides methods to calculate the fitness of chromosomes.
- The Configurations class, containing configurations used for running the genetic algorithm.

To run the algorithm, the GeneticAlgorithm class is provided a connection to the data component and a Configurations class. The connection of the data-component is passed to the FitnessFunction class and the configurations are used for specifying the behaviour of the genetic algorithm. By using the mutation, crossover and selection methods provided by the DEAP-framework, the algorithm is run, passing the result to the web application.

Data component

The data component is responsible for managing all data necessary for running the genetic algorithm. It provides an interface for the FitnessFunction to retrieve information needed for calculating the fitness of a chromosome, for which it keeps track of the gene representation of groups and members. As visualised in blue in Figure 4.2, the data component consists mostly of the data package and containing the following classes:

- The Member class, containing data of a member enrolled for the course.
- The SkillRequirement class, containing data of one the soft-skill requirements that the host has defined for creating groups.
- The DataManager class, containing both a list of members and a list of SkillRequirements. This class holds the gene-data mapping and provides information to the FitnessFunction necessary for calculating the fitness of a chromosome.
- The DataParser class, able to parse retrieved data from the application to a DataManager or Configurations class.
- The DataGenerator class, able to generate a DataManager with random data for testing purposes.

The DataManager lays at the heart of the data component and can be parsed or randomly generated by the DataParser or the DataGenerator respectively. It holds all members and SkillRequirements and provides an interface for the FitnessFunction for obtaining the data necessary for calculating the fitness. To keep the FitnessFunction fast and efficient, the DataManager is optimized to provide this data as efficient as possible.

Main component

The main component consists of a single class, which is visualised in green in Figure 4.2. The algorithm accepts data as a file location or a string containing data in a list of members, a list of constraints, the desired group size and an optional $\mathrm{Configurations}$ class. It also provides the option to randomly generate the data. From this data, the genetic algorithm is executed and the best results will be returned.

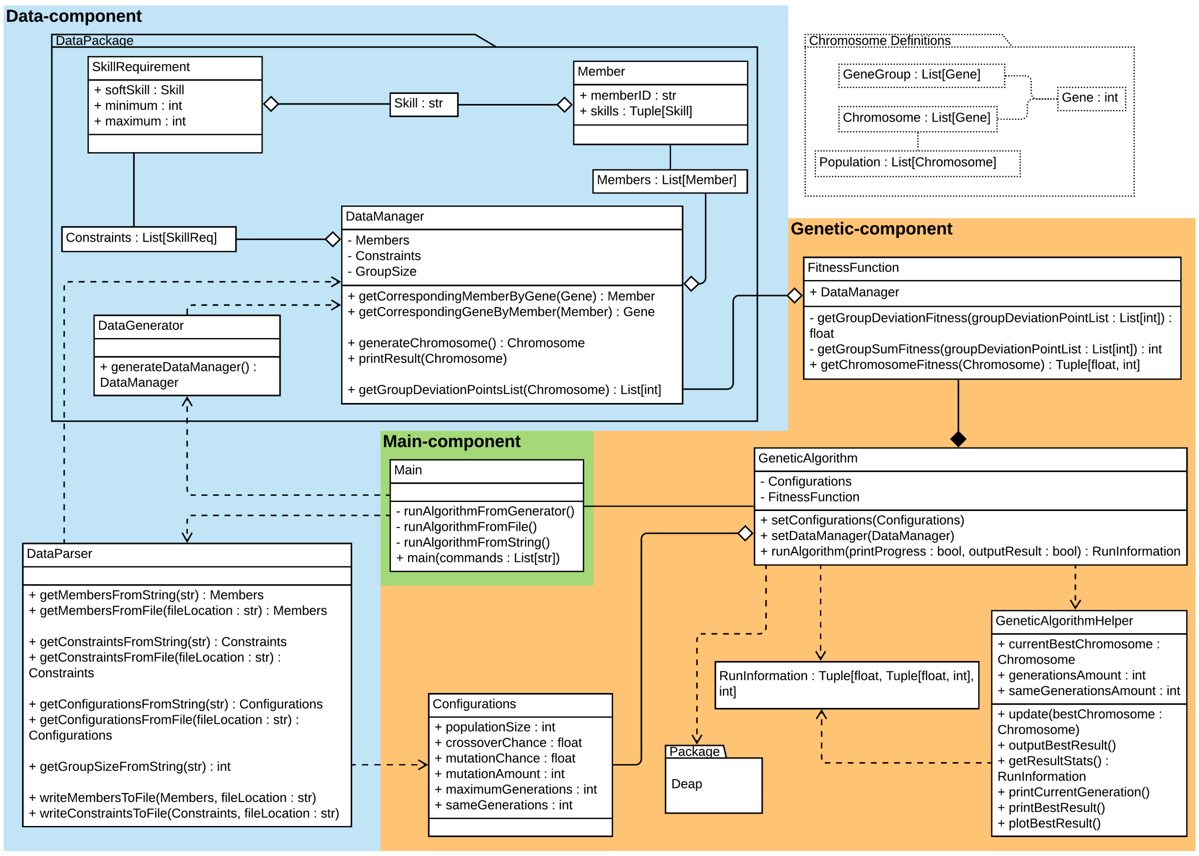

Data-component
DataPackage
SkillRequirement
+ softSkill : Skill
+ minimum : int
+ maximum : int
Skill : str
Member
+ memberID : str
+ skills : Tuple[Skill]
Members : List[Member]
Constraints : List[SkillReq]
DataManager
- Members
- Constraints
- GroupSize
+ getCorrespondingMemberByGene(Gene) : Member
+ getCorrespondingGeneByMember(Member) : Gene
+ generateChromosome() : Chromosome
+ printResult(Chromosome)
+ getGroupDeviationPointsList(Chromosome) : List[int]
DataGenerator
+ generateDataManager() : DataManager
Chromosome Definitions
GeneGroup : List[Gene]
Gene : int
Chromosome : List[Gene]
Population : List[Chromosome]
Genetic-component
FitnessFunction
+ DataManager
- getGroupDeviationFitness(groupDeviationPointList : List[int]) : float
- getGroupSumFitness(groupDeviationPointList : List[int]) : int
+ getChromosomeFitness(Chromosome) : Tuple[float, int]
Main-component
Main
- runAlgorithmFromGenerator()
- runAlgorithmFromFile()
- runAlgorithmFromString()
+ main(commands : List[str])
GeneticAlgorithm
- Configurations
- FitnessFunction
+ setConfigurations(Configurations)
+ setDataManager(DataManager)
+ runAlgorithm(printProgress : bool, outputResult : bool) : RunInformation
DataParser
+ getMembersFromString(str) : Members
+ getMembersFromFile(fileLocation : str) : Members
+ getConstraintsFromString(str) : Constraints
+ getConstraintsFromFile(fileLocation : str) : Constraints
+ getConfigurationsFromString(str) : Configurations
+ getConfigurationsFromFile(fileLocation : str) : Configurations
+ getGroupSizeFromString(str) : int
+ writeMembersToFile(Members, fileLocation : str)
+ writeConstraintsToFile(Constraints, fileLocation : str)
Configurations
+ populationSize : int
+ crossoverChance : float
+ mutationChance : float
+ mutationAmount : int
+ maximumGenerations : int
+ sameGenerations : int
RunInformation : Tuple[float, Tuple[float, int], int]
Package
Deap
GeneticAlgorithmHelper
+ currentBestChromosome : Chromosome
+ generationsAmount : int
+ sameGenerationsAmount : int
+ update(bestChromosome : Chromosome)
+ outputBestResult()
+ getResultStats() : RunInformation
+ printCurrentGeneration()
+ printBestResult()
+ plotBestResult()

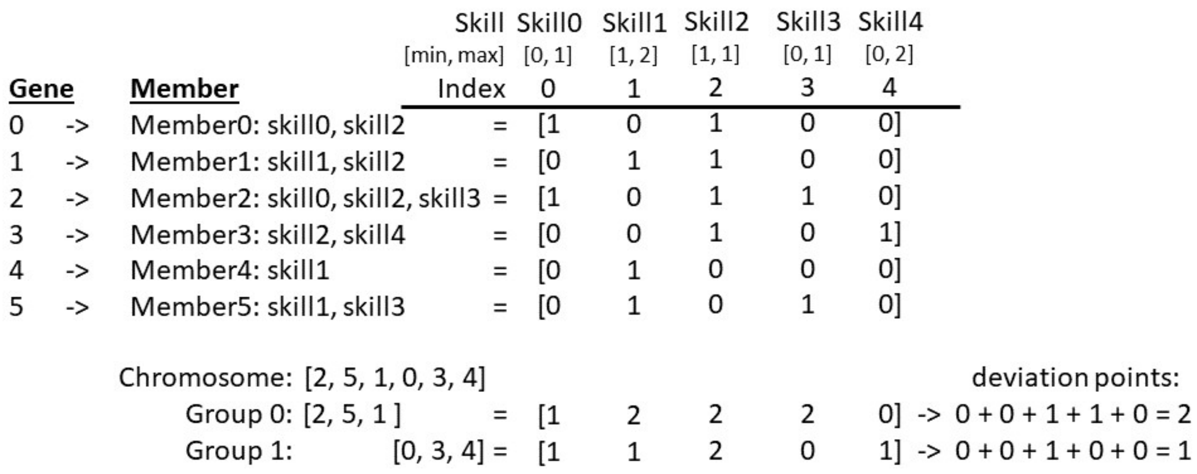
Figure 4.2: Algorithm Class Diagram

4.3.3. Challenges

In this section we will discuss some of the complications that we encountered during the implementations of the algorithm and explain how we have resolved these.

Speed of the fitness function

One of the challenges for the implementation of the genetic algorithm is that the fitness function needs to be fast. The large amounts of data that is used for determining the fitness of a proposed group formation raises issues. To optimize the fitness function, a chromosome-data mapping is created, which is shown in Figure 4.3. This mapping is accessed by the fitness function via an interface that takes a chromosome as input and returns a list of deviation points, each deviation point representing the deviation of a single group from the requirements. To calculate these deviation points, the $\mathrm{DataManager}$ holds a chromosome-data mapping with the genes being the index of the members found in the list in $\mathrm{DataManager}$. This allows for accessing the data of a member fast. By mapping the $\mathrm{SkillRequirements}$ to a list of integers corresponding to its index, the application can effectively pre-process the data of a member to a list of zeros and ones. A zero represents the member not possessing the corresponding skill and a one represents the member possessing the corresponding skill. By summing the entries of a column of all members in a group and comparing it to the minimum and maximum of the defined group requirements, the deviation points of a single group can be calculated and passed to the fitness function. This results in a fast and effective way of calculating the fitness of a chromosome, resulting in a faster algorithm.



Gene		Member	Skill [min, max] Index	Skill0 [0, 1] 0	Skill1 [1, 2] 1	Skill2 [1, 1] 2	Skill3 [0, 1] 3	Skill4 [0, 2] 4
0	->	Member0: skill0, skill2	=	[1	0	1	0	0]
1	->	Member1: skill1, skill2	=	[0	1	1	0	0]
2	->	Member2: skill0, skill2, skill3	=	[1	0	1	1	0]
3	->	Member3: skill2, skill4	=	[0	0	1	0	1]
4	->	Member4: skill1	=	[0	1	0	0	0]
5	->	Member5: skill1, skill3	=	[0	1	0	1	0]

```
Chromosome: [2, 5, 1, 0, 3, 4]                                    deviation points:
     Group 0: [2, 5, 1 ]           =  [1   2   2   2   0]  ->  0 + 0 + 1 + 1 + 0 = 2
     Group 1:           [0, 3, 4]  =  [1   1   2   0   1]  ->  0 + 0 + 1 + 0 + 0 = 1
```

Figure 4.3: Algorithm Chromosome-Data Mapping

Optimization in DEAP

The usage of the DEAP-framework has allowed us to easily construct an instance of a genetic algorithm by using its built-in functionalities. It has enabled us to implement a fitness function that is used to optimize two objectives: the Quality of groups and the Similarity of quality, which we will call the sum-fitness and the std-fitness respectively. When defining the fitness attribute of a chromosome, it is possible to define weights for the individual objectives to assign relative importance to them. The two objectives were given the same weight as we believed them to be equally important when optimizing a solution. However, a problem occurs when the solutions of a population have reached a point where the maximum difference in deviation points of two random groups is one. An example of this is if the list of deviation points of the groups would look like this: [2,2,2,2,3,3,3]. If the std-fitness still gets optimized at this point, the list of deviation points could for instance be changed from [2,2,2,2,3,3,3,3] to [2,2,2,3,3,3,3,3], which would make the sum-fitness worse, but the std-fitness better. The algorithm might consider this sort of change to be good, since the std-fitness has been improved. However, we prefer the original solution, since we prefer groups that deviate less from the constraints. Minimizing the std-fitness at this point does not produce desirable results.

Initially, this problem was solved by adding a separate case for when maximum difference of deviation points of groups is equal to one, which would set the std-fitness to be zero. This ensures that only the sum-fitness will be optimized.

However, this introduces a new problem: when a solution has reached the point where its std-fitness is zero, there is a good chance that the deviation points of the groups cannot be improved anymore. For example, this can happen with a solution with the following deviation points: [3,3,3,3,3,3,4,4]. This solution has an std-fitness of zero. If a mutation produces a solution with deviation points [2,3,3,3,3,3,4,4], the std-fitness is no longer zero, which means that this is not seen as an improvement. However, this step might have been needed to achieve a solution with the deviation points [2,2,3,3,3,3,3,3].

More undesired behaviour was found when looking at the implementation in DEAP. Which fitness values are best is determined by comparing the values and taking the smallest. The std-fitness is calculated by taking the standard deviation of the deviation points of the groups, which often results in a decimal. In the implementation of the algorithm, tiny differences in the std-fitness can still influence which solution is considered to be better, when in actuality these small differences in std-fitness might not matter for our perception of the quality of the solution. In other words, the algorithm might spend too much time trying to perfect the std-fitness when the solution would benefit more from optimizing the sum-fitness.

This behaviour can be prevented by rounding the value of the std-fitness. This makes it that if the similarity of quality of two solutions are similar enough, the focus will lie on improving the sum-fitness.

Rounding the std-fitness introduces a new variable, since it has to be determined what the std-fitness is rounded on. If the std-fitness is rounded on a value that is too small, for example on multiples of 0.1, the initial problem that we tried to solve occurs. If the std-fitness is rounded on a value that is too big, for example on multiples of 0.5, the std-fitness will not be improved enough. We observed that there is a balance between these two behaviours when the value of the std-fitness is rounded on multiples of 0.25.

4.4. Benchmarking

In a genetic algorithm, multiple parameters are used to determine how the algorithm is executed. The values of these parameters have significant consequences on the outcome of the algorithm, therefore it is important to determine values that perform well. This is why we have taken the time to run benchmarks to determine the desired values for these parameters.

To run benchmarks on our algorithm, we have had to determine two things: what input data to use for benchmarking and what values to consider for these parameters when benchmarking.

4.4.1. Data sets

The data used as input for the algorithm when benchmarking is important to consider. Configurations that perform well on one data set might perform poorly on others. Since we want the algorithm to run well when it is deployed on real data, we have to choose our test data input such that it is representative of real scenarios.

Since the amount of members is expected to be between 30 and 100 students, we have chosen data sets with 30, 70, and 100.

We have decided to keep the group size the same for all data sets, since we do not believe this to have a major impact on how the algorithm runs. We have chosen to take the group size of 5 for all data sets.

To be able to determine whether the algorithm works well on a data set, it is necessary to assess the results. However, this turned out to be a difficult task. To be able to asses the quality of a solution produced by the algorithm, a reference solution is needed to compare it to. Since the algorithm approximates the optimal solution, the reference solution should be the optimal solution. However, acquiring such an optimal solution is the exact problem that is attempted to be solved when writing the algorithm. This means that for generic data sets, we are not able to compare the results of the algorithm with an optimal solution, since it is uncertain what the optimal solution is. Moreover, it is worth mentioning that it is not feasible to create such data sets by hand.

However, we have come up with a way to generate data sets in such a way that we know what the optimal solution is. This can be done in two steps:

1. Generate the constraints randomly. This means randomly picking soft-skills that will be in the constraints and randomly picking minima and maxima to go with these soft-skills.

2. Generate the members. This has to be done in groups that are the size of the preferred group size. For these generated member groups, the amount of soft-skills that will be present in the group is determined. These amounts are randomly generated, but within the constraints. Then these soft-skills are randomly assigned to the members in the group to possess. This means that members that were generated in the same group can together make a group that fits all the constraints.

To summarize, this method creates members in groups that we know are within the constraints. So if these members are put into the same group formations in which they were created, these groups would be within all constraints. This means that for these data sets it is possible to create a solution with a std-fitness and sum-fitness of zero.

Furthermore, all solutions in which all groups are within the constraints are considered to be optimal, since such a solution is impossible to improve. This means that if the algorithm has found any solution with an std-fitness and sum-fitness of zero, it has found the best possible result. So by testing on data sets that are generated in this way, it is possible to determine if the algorithm has found the optimal solution and if not, how close it got.

This method, however, does have a limitation: it is only able to create data sets for which it is possible to form groups that are within all the constraints. Generating data in a way that groups are generated to have some random deviation from the constraints does not guarantee that there exists no other group division with less deviations from the constraints.

4.4.2. Algorithm parameters

In genetic algorithms there are a few parameters that determine how the algorithm runs. We will discuss five important ones that we have run our benchmarks with to optimize.

Population size

The population size is the amount of chromosomes that are considered in a generation. For bigger population sizes, more calculations have to be done every generation. This means that generating generations will take longer. However, a bigger population size also means that more of the solution search space is explored, which could mean that less generations are needed to produce an accurate result.

We have considered five values for the population size when benchmarking: 30, 50, 75, 100, and 125. These values were chosen because they seemed promising based on some trial and error.

Crossover chance

The crossover chance is the chance that two chromosomes perform a crossover. However, because the parent chromosomes get removed from the population after the crossover has taken place, favourable properties of chromosomes can be removed from the population if too many crossovers are applied.

For the crossover chance it was difficult to get an initial idea of which values performed better than others. To cover our bases, we considered all chances between 0.1 and 0.9 with increments of 0.1.

Mutation chance

The mutation chance is the chance that a chromosome mutates. If it does, there is a chance that two of its genes are swapped with each other, meaning that the two corresponding members switch groups. Mutating is important to create diversification in the solutions. Not enough mutations would cause the algorithm to only produce solutions that are similar to the original randomly generated solutions. Too many mutations can cause the algorithm to replace its best solutions it has with lesser mutated versions of them.

For the mutation chance it was difficult to get an impression of which values were going to work the best. For this reason it was decided to consider all chances between 0.1 and 0.9 with increments of 0.1.

Gene swapping chance

The gene swapping chance is a chance that is used in our chosen method of mutation. When a mutation takes place, the gene swapping chance is applied to each gene to determine if the gene is going to be swapped with another random gene. If we were to assign the same gene swapping chance to chromosomes of different sizes, each gene would have the same chance of being swapped and, since bigger chromosomes have more genes, on average more genes would be swapped in a bigger chromosome than in a smaller chromosome. This is why we decided to use a gene swapping chance that depends on the size of the input. We defined the parameter for this chance as the integer *x*, which should be the average amount of genes that is expected to be swapped. With this parameter the gene swapping chance $\frac{x}{chromosome\_size}$ is made.

For the gene swapping chance, five different values have been considered to be able to experiment with all different settings of the mutation chance and the crossover chance. The values we considered were 1, 2, 3, 4 and 5.

Generations

The amount of generations that the algorithm runs for determines how close the algorithm can get to an optimal solution. If the algorithm does not run for enough generations, it can happen that there have not yet been enough crossovers or mutations for the solutions to have improved. However, if the amount of generations is too big, a local optimum can make it that there are no longer improvements being made in the solutions. Therefore, the amount of generations should be large enough for the algorithm to be able to make considerable improvements in the solutions, but also small enough for the algorithm to stop when no improvements can be made anymore.

However, the problem of a predetermined amount of generations is that the values for which the algorithm performs best vary per data set. Generally, for bigger data sets more generations are needed to produce an accurate result than for smaller data sets. This is why during the initial tests of the algorithm we have tried to work with an amount of generations that is dynamic. During some test runs of the algorithm, we observed that it was often the case that after the algorithm had run for an amount of generations, the change in fitness values had stagnated and the fitness remained the same until the maximum amount of generations was reached. Furthermore, the fitness values that were reached at that point were often the same for multiple runs. This led us to believe that solutions with these fitness values were either the optimal solutions or the best solutions that the algorithm could find. From this moment, we often ran the algorithm with an added parameter: the amount of generations for which the best fitness values have to remain the same for the algorithm to stop. This added parameter has made it that the algorithm can stop running if there are no longer improvements being made. Since this parameter seemed promising when used with some of the initial tests for the algorithm, we decided to add it to the benchmarking parameters. We have used the following values for this parameter: 25, 50, 75, 100 and 200. A maximum amount of generations for which the algorithm can run was added as well, such that the algorithm is not able to run indefinitely. We decided to make the maximum amount of generations that the algorithm can run 500, since an execution of the algorithm would take too long if it would run for more.

Parameter	Value
Crossover chance	0.7
Mutation chance	0.4
Gene swapping chance	1
Population size	100
Number of same generations	50

Table 4.1: Resulting configurations

4.4.3. Results

To determine for which values of the parameters the algorithm runs best on the three data sets, the algorithm was run on all data sets with all possible parameters. However, since genetic algorithms have random components, running the algorithm twice with the same data set and the same configurations can yield different results. Therefore, a single execution of the algorithm with one set of configurations is not enough to conclude anything about the accuracy or speed. Since more executions with the same configurations generally mean more accurate results, we would have preferred to run each configuration ten or twenty times, however due to time limitations we were only able to run each configuration five times. From these five runs the average fitness and average run-time were taken as measurements of performance and speed.

The data set with 30 members was the quickest to run. From the 10125 different configurations that the data set was run on, 10060 of them produced the optimal result. The data set with 70 members resulted in 8737 of the 10125 configurations being able to find the optimal result. The data set with 100 members had 7405 of the 10125 configurations produce the optimal result.

From these benchmark runs, we were aiming to find some values for the configuration variables for which the algorithm performed well on all data sets. However, we ended up with data that indicated an overwhelmingly big part of all configurations were able to produce optimal results. We suspect that this happened because of the decision to generate one data set for each data input of 30, 70 or 100 members. Therefore we decided to set the population size and the generation amount to two values for which we knew the performed well; 100 and 50 respectively. We generated all possible configurations with these two values set and ran these configurations on 20 different data sets of 100 members. We left out the data sets of 30 and 70 members because often, when a configuration performed well on the 100 members data set, it also performed well on the data sets of 30 and 70 members. The results of these benchmarks were much more distinguishing. From the 405 settings, 60 configurations were able to find the optimal solution for all 20 data sets. From these 60 configurations we looked for the combination of crossover chance and mutation chance that occurred most often and took the mutation amount with which the smallest average time was achieved. The resulting variables can be seen in Table 4.1.

For these configurations, the average run-time of running on the 20 different data sets was 3.2 seconds. With these configurations, the algorithm was always able to find the optimal solution. It took on average 73 generations to find these solutions.

4.5. Performance

After running benchmarks, we were able to conclude that the algorithm should perform well with the configurations in Table 4.1. To give an illustration of the process of the algorithm, we have created an image of the run-time of the algorithm. This can be seen in Figure 4.4. The data in this plot are the smallest std-fitness and the smallest sum-fitness of all chromosomes in every generation. In this plot it can be seen that the std-fitness is rounded to the nearest multiple of 0.25 and that the algorithm stops after obtaining the same optimal results for 50 generations. This execution of the algorithm was done on data that was generated in the same manner as the data used for the benchmarking. The optimal results were found after 2.78 seconds.

In Figure 4.5 an execution of the algorithm can be seen on randomly generated data. For this data set the algorithm has not been able to find group formations which all are within the constraints. However, an effort has been made to optimize both the sum-fitness and the std-fitness. How accurate the resulting solution is can only be evaluated by manually looking at it.

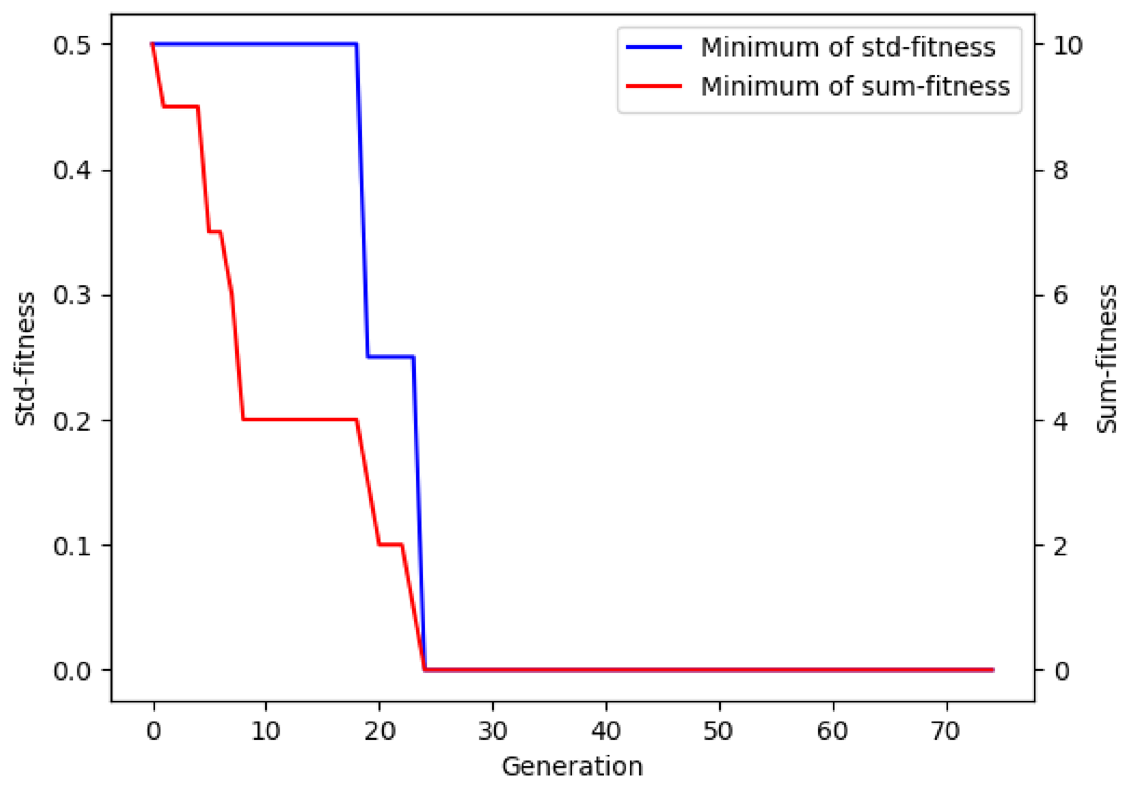


Figure 4.4: Run-time data of the algorithm

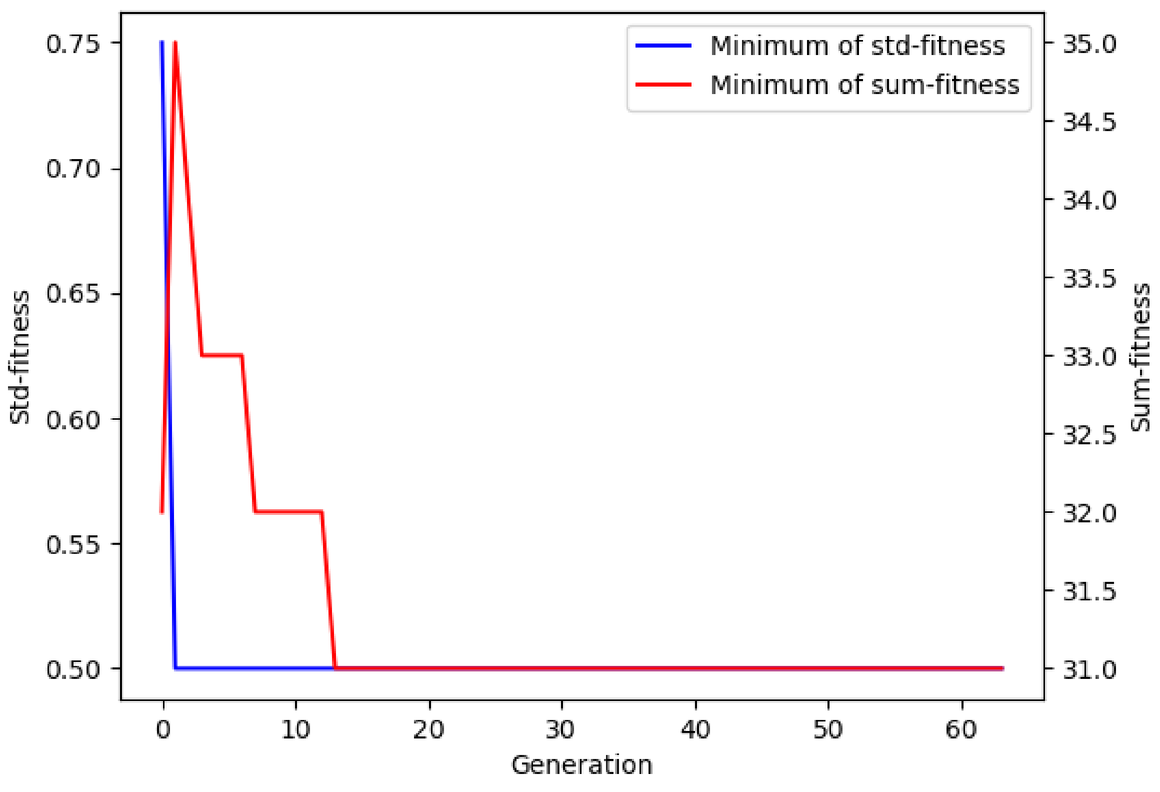


Figure 4.5: Run-time data of the algorithm

5 Soft-skill Profiling System

The client wants the system to be an adaptable web application. It uses frameworks and libraries and it must be tested. These components of the final product will be discussed in this chapter.

5.1. Technologies

Since the system is a web application, HTML must be used to display the application. However, React[1] has been used to write the components for the application, as React makes it possible to reuse the existing code to port the web application to a different type of application, as the client has wished for. React is an open-source JavaScript library which is used to develop the front-end of the web application. It helps creating maintainable and scalable user interfaces and makes the code maintainable. Because of this JavaScript library, almost no HTML code was written.

Web applications use CSS to design the front-end. Material-UI[2], a framework for React, was used in combination with CSS to design the front-end of the application. Material-UI provides its own components with the possibility to make it responsive. Using this framework has helped building a consistently themed application that can be used on any device.

For the back-end, Node.js[3] and Express.js[4] have been used. Node.js is a cross-platform JavaScript run-time environment, which handles various core functionalities of the web server. Express.js provides a web application framework for Node.js to help organize and build the web application. The application used a relational database to store and retrieve data. Choosing a rational model made it easier to analyze and manipulate the data that are related to one another. A conceptual design of the database can be found in Figure 5.1 and the corresponding logical design of the database can be found in Figure 5.2.

5.2. System specification

React has a composition model, which can be utilized to reuse code between components. Figure 5.4 presents an example of this. During the research phase we have created a conceptual page structure, which is adhered to. This structure can be found in Figure 5.3, where blue is only accessible for members, red is only accessible for hosts and green is accessible for both users.

The usage of Material-UI makes all front-end components responsive, which consequently changes the application's appearance depending on the device accessing the system. Examples of this can be viewed in Figure 5.5 and Figure 5.6. This responsiveness provides the users, who we expect to be quite diverse, with the option to use their preferred device.

[1]https://reactjs.org/
[2]https://material-ui.com/
[3]https://nodejs.org/en/
[4]https://expressjs.com/

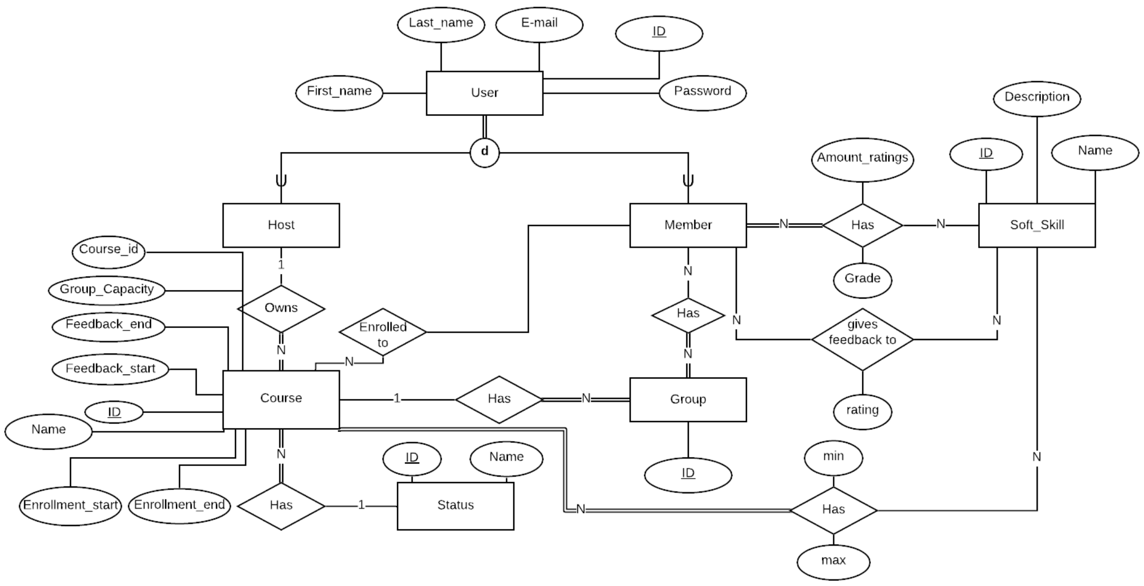


Figure 5.1: Conceptual Database Design

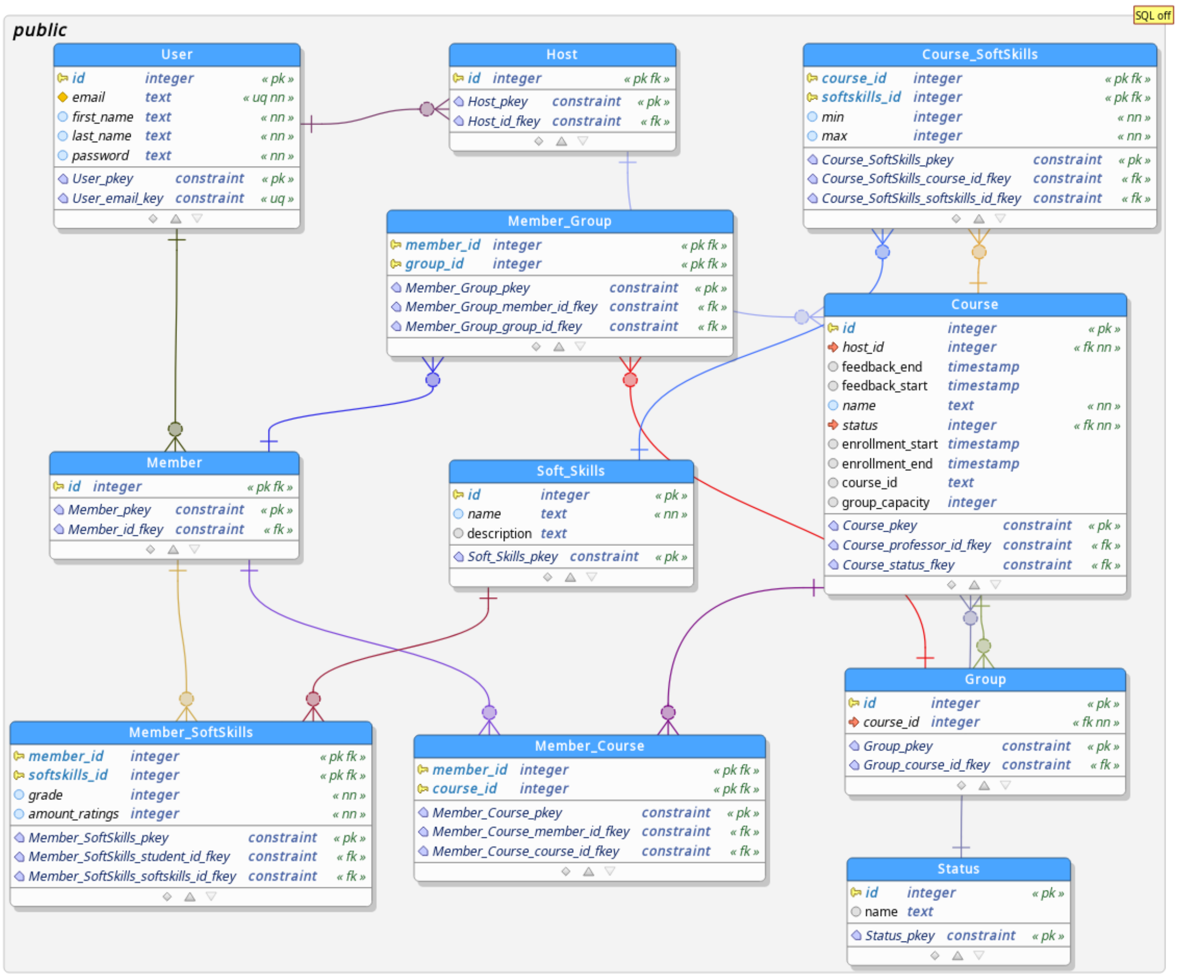

SQL off
public
User
id integer « pk »
email text « uq nn »
first_name text « nn »
last_name text « nn »
password text « nn »
User_pkey constraint « pk »
User_email_key constraint « uq »
Host
id integer « pk fk »
Host_pkey constraint « pk »
Host_id_fkey constraint « fk »
Course_SoftSkills
course_id integer « pk fk »
softskills_id integer « pk fk »
min integer « nn »
max integer « nn »
Course_SoftSkills_pkey constraint « pk »
Course_SoftSkills_course_id_fkey constraint « fk »
Course_SoftSkills_softskills_id_fkey constraint « fk »
Member_Group
member_id integer « pk fk »
group_id integer « pk fk »
Member_Group_pkey constraint « pk »
Member_Group_member_id_fkey constraint « fk »
Member_Group_group_id_fkey constraint « fk »
Course
id integer « pk »
host_id integer « fk nn »
feedback_end timestamp
feedback_start timestamp
name text « nn »
status integer « fk nn »
enrollment_start timestamp
enrollment_end timestamp
course_id text
group_capacity integer
Course_pkey constraint « pk »
Course_professor_id_fkey constraint « fk »
Course_status_fkey constraint « fk »
Member
id integer « pk fk »
Member_pkey constraint « pk »
Member_id_fkey constraint « fk »
Soft_Skills
id integer « pk »
name text « nn »
description text
Soft_Skills_pkey constraint « pk »
Group
id integer « pk »
course_id integer « fk nn »
Group_pkey constraint « pk »
Group_course_id_fkey constraint « fk »
Member_SoftSkills
member_id integer « pk fk »
softskills_id integer « pk fk »
grade integer « nn »
amount_ratings integer « nn »
Member_SoftSkills_pkey constraint « pk »
Member_SoftSkills_student_id_fkey constraint « fk »
Member_SoftSkills_softskills_id_fkey constraint « fk »
Member_Course
member_id integer « pk fk »
course_id integer « pk fk »
Member_Course_pkey constraint « pk »
Member_Course_member_id_fkey constraint « fk »
Member_Course_course_id_fkey constraint « fk »
Status
id integer « pk »
name text
Status_pkey constraint « pk »

Figure 5.2: Logical Database Design

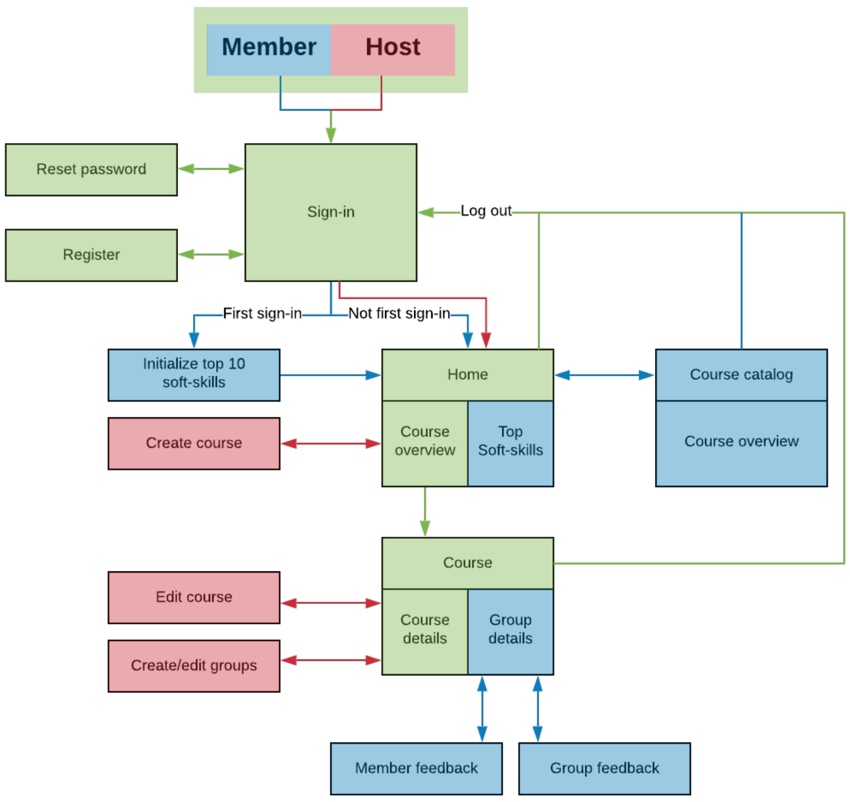


Figure 5.3: Conceptual page structure

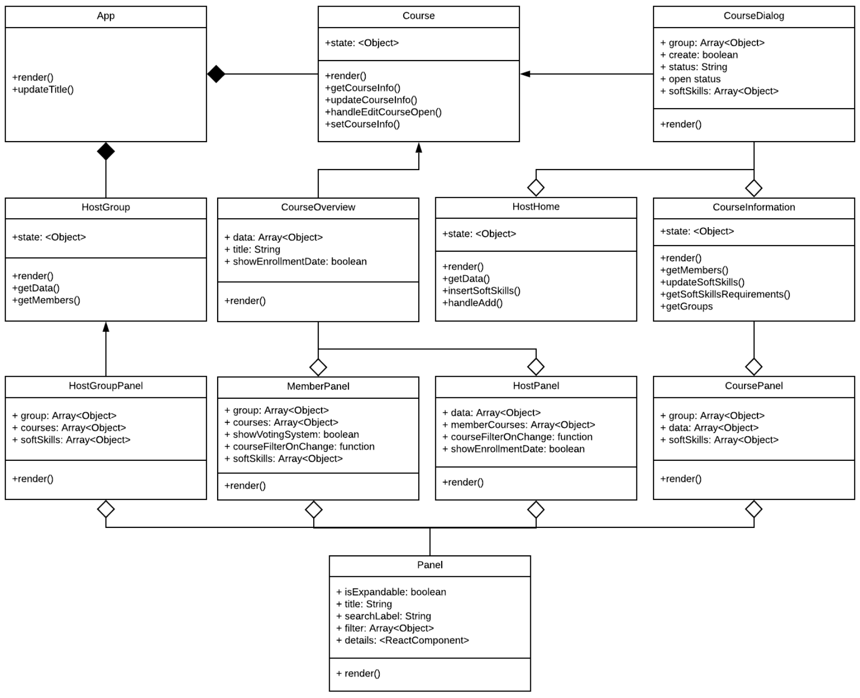
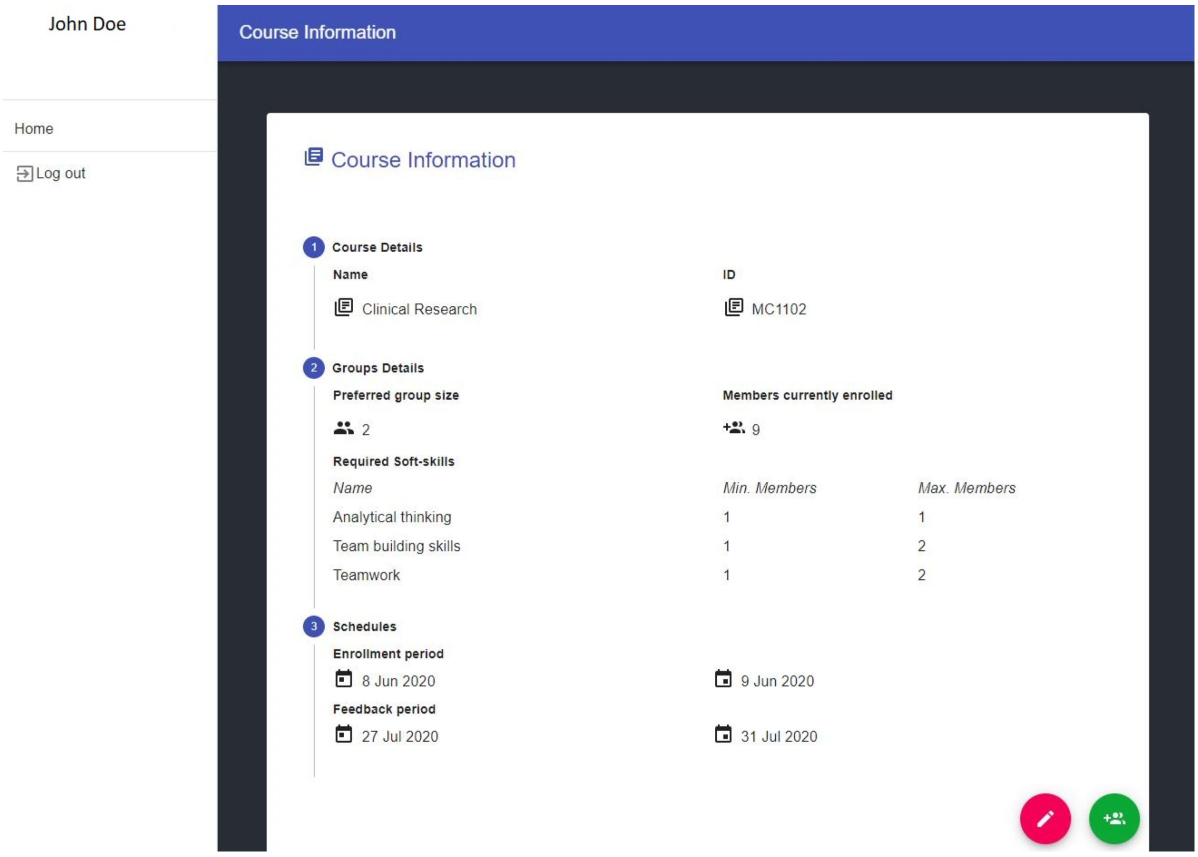


Figure 5.4: An example showing Panel, CourseOverview and CourseDialog components being reused



Course Information

1 Course Details

Name	ID
Clinical Research	MC1102

2 Groups Details

Preferred group size	Members currently enrolled
2	9

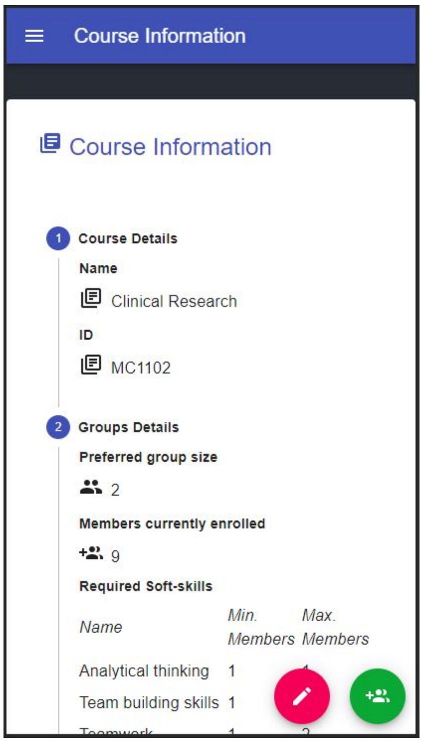
Required Soft-skills

Name	*Min. Members*	*Max. Members*
Analytical thinking	1	1
Team building skills	1	2
Teamwork	1	2

3 Schedules

Enrollment period	
8 Jun 2020	9 Jun 2020
Feedback period	
27 Jul 2020	31 Jul 2020

Figure 5.5: The information of a course, viewed by a host (desktop).



Course Information

1 Course Details

Name

Clinical Research

ID

MC1102

2 Groups Details

Preferred group size

2

Members currently enrolled

9

Required Soft-skills

Name	*Min. Members*	*Max. Members*
Analytical thinking	1	
Team building skills	1	

Figure 5.6: The information of a course, viewed by a host (mobile device).

5.3. Security

A great amount of users will store their personal data to use this Soft-skills Profiling System. Therefore, various technologies have been used to ensure the safety of the data of these users.

Cross-Origin Resource Sharing gave the ability to securely decouple the front-end server and the back-and server. It allows the browser to use the resources of the server. The user interface of the application has one origin and this user interface wants to access the server containing all the data, which is another resource with another origin. This allows better resource management. For example we can dedicate more servers at different locations to our algorithm, which is a CPU intensive task, or put our database on a different server. With the Cross-Origin Resource Sharing, this can be done without fears of facing security issues, for example man-in-the-middle attacks.

Furthermore, to avoid more security risks, the passwords of the users need to be hashed. If it would not be hashed and people could access the passwords, the passwords would be seen in plain text. Hashing the passwords prevents others from being able to use them. For this hashing, BCrypt[5] was used.

For unauthorization issues, JSON Web Tokens[6] were used. These are industry standards for securely representing claims between two parties. This technology was used to identify the user requests. If an unauthorized user would try to get access to the data of others, they can be detected and blocked.

Not just outside sources are not to be trusted; user input should never be trusted as well. To check whether the user input is in the correct format and free of noise, validators and sanitizers were used.

Lastly, prepared statements were used to make the system more resilient against malicious attacks that try to insert data into the database (i.e. SQL injections).

[5]https://www.npmjs.com/package/bcrypt
[6]https://jwt.io/

6 Software Quality

To ensure the software quality, the software should be tested and improved. This chapter will elaborate on what test methods were used and how the code was improved.

6.1. Testing

Since the project is divided into two main parts, testing is also divided. This section will explain how we tested our components and the reasoning behind it.

6.1.1. The multi-objective group formation optimization algorithm

To make sure the algorithm is working as expected, unit-tests and system-tests are used. The combination of these two tests cover all relevant aspects of the algorithm.

Unit-tests

Unit-tests are used to make sure the individual components of the algorithm function as intended. To enforce this, a goal is set to have at least one unit-test for every public method found the application and to have a code coverage of at least 80% for every individual class. With the overall code coverage of the algorithm being 89%, these goals are met for all classes found in the algorithm, except for the $\mathrm{GeneticAlgorithm}$ class and the Main class. For the $\mathrm{GeneticAlgorithm}$ class, this can be explained by its use of the DEAP-framework, making it difficult to test. For the Main class this can be explained by the way uses the console for receiving its input and provides its output, which makes testing with unit-tests difficult.

System-tests

To make sure the algorithm is working as expected and to make sure the functionality of the $\mathrm{GeneticAlgorithm}$ class and the Main class is correct, system-tests are defined. These tests simulate commands that can be sent from the web application resulting in specific behaviour. In order to perform the system tests, multiple console commands can be run in console, printing run-time progress of the algorithm. By checking whether the behaviour is as expected, mistakes in the system can be spotted.

6.1.2. Soft-skill profile system

To test the web application, we have written two types of tests: functionality tests and usability tests. The decision was made to test the system manually, since for the number of components, manual testing would be more efficient.

Functionality testing

To test whether everything works as expected, we have created functionality test protocols, which can be read in Appendix F. These tests contain detailed instructions, such as button clicks, which must be followed. These instructions were periodically executed by members of the development team. The expected behaviour is described, after which the developer can state whether the result corresponded to the expected behaviour. These tests resulted into a list of failing behaviours which needed to be fixed.

Usability testing
The web application must be intuitive, which means it should be straightforward for users what they should do on the application in order to achieve their goal. To test this, we have created different scenarios for usability tests, which can be read in Appendix G. These tests contained general tasks which the user should complete. During this task, we asked the user to think out loud. This gave us insight into the user's experience.

The tests highlighted the intuitive parts of the application as well as the things that could be made more user friendly. For example, most users who had the task to enroll for a course could not find the course catalog at first, but rather tried to search for a course in their own course overview panel. This made us change the title to "My courses" and place a button inside the panel, if empty, that redirects the user to the course catalog. Moreover, a user with the task as a host to create a course stated that it is a tedious task to enter the date periods whenever they are a few months away. We took this into consideration and altered the end of the period whenever the start of the period was entered.

The test brought up a few bugs that were not covered during the functionality tests. The reason for this was that the users doing these tasks were motivated to find these bugs, causing them, for example, to click on buttons as fast as they could to see whether it would break. This was the case with the enroll button.

After improving some features and fixing the bugs that were found, the users believed the application to be intuitive.

6.2. Code style

Improving code style is beneficial for the overall program. It makes the code clear, consistent and simple to be read by others. Furthermore, consistent code makes it less complex, which consequently reduces (the introduction of) errors. The proper use of code-style can thus increase both the intuitiveness and adaptability of the application.

6.2.1. The multi-objective group formation optimization algorithm

To keep the code clear and intuitive, the pep8[1] style guide is used. By enforcing this style on all code written for the algorithm, the code remained clear, consistent and simple. Apart from this style guide, a few other standards are set to keep the code intuitive and adaptable:

- The name of any variable and function explains its purpose as clearly as possible. This means that no further explanation is needed for understanding the code. While this results in larger variable and function names and longer lines of code, it also increases the intuitiveness of the code.
- While python allows for dynamic typing, the parameter types and return types of methods are always defined. This helps to understand the purpose of a class and increases the intuitiveness of the code.
- Every method found in the application uses 'restructured text' documentation[2]. This enforces explaining the functionality, parameters and return values of the methods, making the application easy to understand.

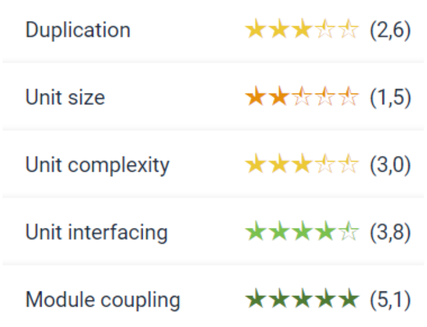
6.2.2. Soft-skill profile system

To style code written in JavaScript and CSS, we used a tool called Prettier[3], which styled the code automatically. For writing SQL queries, we followed the SQL style guide by Simon Holywell[4].

6.3. Software health

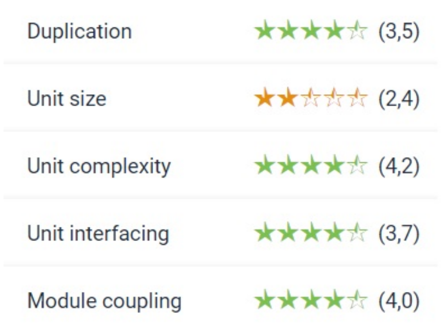
During the development, we received feedback from SIG[5] on our software health. After the first feedback, of which an overview can be found in Figure 6.1, we have improved our code, which we will discuss in this section. Furthermore, we will look back at the feedback we received after our code improvement, which can be found in Figure 6.2. The code for the algorithm, written in Python, is discussed separately from the code of the web application, which was mainly written in JavaScript.

[1] https://www.python.org/dev/peps/pep-0008/
[2] https://docutils.sourceforge.io/rst.html
[3] https://prettier.io/
[4] https://www.sqlstyle.guide/
[5] https://sigrid-says.com//



Duplication	(2,6)
Unit size	(1,5)
Unit complexity	(3,0)
Unit interfacing	(3,8)
Module coupling	(5,1)

Figure 6.1: The score after the first code review from SIG.



Duplication	(3,5)
Unit size	(2,4)
Unit complexity	(4,2)
Unit interfacing	(3,7)
Module coupling	(4,0)

Figure 6.2: The score after the second code review from SIG.

6.3.1. Duplication

The feedback about code duplication contained some inefficient proposed changes. Where we thought this was the case, we did not alter our code (e.g. the case where reusing a function multiple times was seen as code duplication). We created general methods to improve the code where there did seem to be multiple lines of code occurring more than once. Config.js had the sole purpose to contain these general methods. There were no comments on the code for the algorithm. As seen in Figure 6.2, the improvements to our code resulted into an improved grade.

6.3.2. Unit size

For the system, we decided to split the methods such that each executes one task. Furthermore, decreasing the duplication consequently decreased the unit size in some cases. The unit size of render functions were not decreased, since without changes, it would be more clear to see what the component contained.
The algorithm had less comments, which made it possible to review every mentioned function:

- GeneticAlgorithm.run_algorithm(print_progress) - 56 lines of code:
  To improve upon this, we split the method into multiple methods and moved some of them to the GeneticAlgorithmHelper class. Even with these new methods, however, the run_algorithm method still has many lines of code. We decided to keep it this way, since we believe that splitting the method up even more would make the application more difficult to understand. In our opinion it is better to keep the usages of the framework DEAP mostly in one method. We understand that this decision impacts the readability of the code, which is why we have added more comments to this method.

- BenchmarkRun.py:run() - 22 lines of code:
  To improve upon this, the benchmarker is split up into a Benchmarker, a BenchmarkDataIterator and a BenchmarkDataSummarizer. This has made the Benchmarker class more maintainable and easier to customize for specific benchmark runs. However, there is still a main function in the Benchmarker that is being called for each benchmark, which has 28 lines of code. We believe that it is not needed to reduce the length of this method though, since the functionality of this method does not need to be tested. The Benchmarker only sets some parameters and runs the algorithm, which are things that have already been tested individually.

- CrossoverFunction.py:PMXCrossover(c0, c1) - 19 lines of code:
  Since this function was not used, together with all custom genetic-operators, this method is removed.

These alterations resulted into a slightly improved grade.

6.3.3. Unit complexity

In the web application, the main problem for this topic was conditional rendering. Since members and hosts use the same components with different features, almost every component used conditional rendering, increasing the unit complexity. We were able to decrease the complexity in some components by combining decision points for example.
The algorithm had less comments, which made it possible to review every mentioned function:

- GeneticAlgorithm.run_algorithm(print_progress) - McCabe complexity of 12:
  To improve upon this, some of the functionality of the method is moved to a new 'GeneticAlgorithmHelper' class. While the main reason for the high complexity rate appears to be the use of external libraries, moving some of the functionality to another class decreases the complexity and increase the maintainability of the code.

- ___main___.py - McCabe complexity of 9:
  We decided not to improve upon this. This part of the application parses the input given to the application via the command-line and uses it to run the genetic algorithm. Because of this, the file contains calls to nearly all functionality of the project, increasing the McCabe complexity. A way to decrease the complexity is to move the different functionalities to different files. However, this will only decrease the readability of the code, making it more difficult to maintain.

- CrossoverFunction.py:PMXCrossover(c0, c1) - McCabe complexity of 7:
  Because this function was not used, together with all custom genetic-operators, this method is removed.

Although, we did not expect the unit complexity to decrease significantly, its grade had increased the most. Since minor details were changed in the web application, we believe the improvements in the algorithm resulted into the improvement in grade.

6.3.4. Unit interfacing

For the web application, it was hard to decrease the number of parameters. Every function that was mentioned in the feedback required the parameters in order to function. We have thus decided not to improve the code from the web application on this part.
The algorithm had less comments, which made it possible to review every mentioned function:

- DataGenerator.py contains multiple functions having 4 to 8 parameters.
  We decided not to improve upon this. The DataGenerator is used for creating dummy-data. This dummy-data is used for testing the algorithm and is customizable. The customization of the data is done with the use of parameters, resulting in a high number of parameters. Since decreasing the amount of parameters will limit the functionality of the DataGenerator or will increase the complexity of the code, we decided not to change the DataGenerator.

- Configurations.py contains 6 parameters at its constructor.
  We decided not to improve upon this. The configuration class holds all possible configurations of the genetic algorithm, and acts as data class. Because of this, we believe the use of 6 parameters as constructor should be allowed.

- Main.py contains functions needing 5 parameters.
  To improve upon this, we decided to place the functions inside a class, allowing the amount of parameters to be decreased. Main.py contains three functions for running the algorithm: one by loading the data from a file, one by loading the data from an input string and one by generating the data. Because of this, all data has to be passed through the parameters, resulting in some of the functions requiring 5 parameters. Because these parameters are necessary for running the algorithm, the use of at least 3 parameters is required.

- DataManager.py, GeneticAlgorithm.py and SkillReq.py contain methods requiring 3 parameters.
  We decided not to improve upon this. Looking at the complexity of the problem the algorithm tries to solve, we believe that the use of three parameters should not be an issue. For most of these methods the parameters are used to pass data, needed for doing specific calculations. Because of this, we decided not to change anything and to ignore the warnings of the use of 3 parameters.

The grade we received after the second feedback was slightly lower than the first grade. Although we tried to improve on this part, it is possible that the alterations made for other components would somehow have cancelled this out.

6.3.5. Module coupling

One comment was made, concerning $\mathrm{Config.js}$. This file contained the general methods, decreasing the code duplication. We decided not to change anything on this topic.

Although no changes were made, the grade had been decreased. One explanation would be that the improvements made for the other components will have had an impact on this component.

7 Software Development Methodology

Scrum was chosen to be our Software Development Methodology to work with, because it is an agile methodology which divides our work into smaller goals and time-boxed iterations, or so-called sprints. It reduces the risk when building the product and reflects on the sprints to make improvements. Our sprints consisted of one week and started every Monday 9:00h and ended every Friday 17:00h. Scrum is in general designed for and widely used by small teams in the software development. Due to these reasons, we thought Scrum would be the most efficient way for us to build the product as a team.

We divided the labour of the project into two parts: the group formation algorithm and the soft-skills profiling system, which uses the group formation algorithm. Although the algorithm was the most important part of the project, two developers were put on the algorithm and three developers on the system. The reasoning behind this, is that the system contains more issues than the algorithm, thus the system would need more work. With this division, the labour would be well divided.

7.1. Sprint planning

While the sprint planning is usually done at the beginning of a sprint, we have divided the issues over eight sprints at the end of the research phase. These issues were created on GitLab using the built-in scrum board. Issues that were more important were divided into earlier sprints. With this division, we would have all the must-haves and a few should-haves from the requirements list finished by sprint 5. After that sprint, We would then start on should-haves, could-haves and testing.

7.2. Daily scrum

Our sub-teams met every morning via a voice chat. If we had to discuss plans or issues, we would schedule a voice chat with the whole team. During these meetings, we asked each other what we had completed, what we planned on completing and if we had any obstacles that would prevent us from doing our work. Our team had a meeting with our coach every Tuesday. Every Friday, we had a meeting with our client.

7.3. Sprint review

At the end of the sprint, we reviewed our completed and non-completed work. We demonstrated our working product to the client and discussed thoughts about it. We asked him what he wanted to see for the next sprint or what he wanted us to work on and we discussed what we were planning on doing in the next sprint. Based on these conversations, we came to an agreement on what to work on next. The original planning changed very little.

7.4. Sprint retrospective

After the Sprint review, we reflected on the past sprint. We discussed what went well, what did not go well and what could be improved. Our collaboration improved during the first few sprints due to this retrospective. The retrospective is not documented on paper, but only orally discussed and improved.

8 Discussion

This chapter will discuss ethical implications of the soft-skill profile based group formation system, any recommendations for future development and our insights.

8.1. Ethical implications

The main topic on our ethical insight will be the fact that members are profiled based on soft-skills that they should possess, with corresponding grade. First, soft-skills are initialized by members themselves, creating a profile that they think is accurate. However, after this initialization, their soft-skills and corresponding grades are criticised by their fellow members. A member can never alter their own soft-skills again. This consequently means that their fellow members determine the final soft-skill list, which will be used to place the student at a hospital. The lower the grades on the final soft-skill list, the lower the chances of being placed at a hospital. This is a fact that every member will realize, which could cause members to compare themselves with others and criticise each other accordingly (e.g. a member could give every other member a lower grade to have a chance to make themselves stand out). This would make the system a competition rather than a fair way to keep track of the soft-skills of members.

Another possibility is that the members will not try to negatively criticize their fellow members because of a competition, but based on discrimination. A member could discriminate others based on, for example sex, religion, et cetera, thus lowering the grades for their soft-skills, which would cause this specific group to have lower chances to be placed at a hospital. In a case where the outside world would look at the variety of people working in a hospital, they could think the hospital is the cause for this discrimination.

As a third, in a case where members will criticise other members in a supposedly fair manner, the question arises what these soft-skills will actually claim about a person. A member possessing a certain soft-skill has no direct relation to their personality. This means that placing the best fitting member (as resulted from the algorithm) in a hospital team based on their hard- and soft-skills will not mean that this person's personality will fit with those of the hospital team. Although knowing a member's hard- and soft-skills will make the process of matching the member to a hospital easier, a probation would still be necessary to see whether the member actually fits in the team.

Lastly, soft-skills are skills that cannot be measured, while the system is doing exactly that. It is questionable whether it is ethical to measure or even assign grades to soft-skills. Moreover, the value of a grade is relative, meaning a member could rate a grade to be lower or higher depending on what that member associates with a certain grade.

To conclude, the ethical aspect of the system can be questionable on various topics. Further research is needed to study the impact the system could have on the members, universities and hospitals.

8.2. Recommendations for further development

It is difficult to determine whether the use of the described solution will result in better team compositions. There are two presuppositions made at the beginning at the project: 'the soft-skill profiles of the members created by the application are representative for the soft-skills of the students' and 'there exists a relationship between combining these profiles in a team and the eventual efficiency of a team'. Assuming that the second presupposition is true and the hosts has the required knowledge to set the soft-skill requirements in such a way that it would create an effective team, the question remains whether the soft-skill profiles are actually representative for the members.

Because this is beyond the scope of the project, this question is not addressed in this work. Still, because it is relevant for the success of the solution, this is something for further research to look at.

In the Problem encapsulation the components of a genetic algorithm are discussed along with an explanation of which operators we use for the selection, crossover and mutation. The motivation behind the usage of most of these operators is that they have had positive results in addressing similar problems (Moreno et al., 2012; Zheng et al., 2018). This does not mean no better alternatives exist for our problem. The time that it would take to investigate this has prevented us from exploring this further.

Apart from the operators that are used in the genetic algorithm, there are more variables that influence the execution of the algorithm. By benchmarking the application, we have been able to determine values for the mutation chance, crossover chance and gene swapping chance for which the algorithm performs well. We have determined that the algorithm performs well on these values by testing the results that are obtained with them. Testing these results, however, is only possible if we ran the algorithm on data sets of which the desired results is known. To get these data sets, they are generated to contain the possibility of creating a group formation that would fall within the defined constraints. A disadvantage of using this data is that the algorithm is now calibrated to work well on data that has such a solution. Because of this we do not know whether the chosen configurations are also optimal for the real data that will be used when the application gets deployed.

8.3. Insights

When looking back at the project, important insights were gained. These insights concern the research phase, the algorithm and the process of building the system.

During the research phase, the product idea changed several times. Consequently, different types of algorithms had been researched that were not applicable to the current matching problem, thus making this research nullified. This time could have been spent on research on the more applicable types of algorithms and on creating a plan on how to tackle this. In retrospect, a detailed project description should have been defined before we researched the types of algorithms suitable for us to use in our project.

For the algorithm, it was decided to use the genetic algorithm, however, the initial idea was to create it ourselves. It was clear that *the wheel should not be reinvented*. Even though the process of the creation of the algorithm from scratch was running, a framework was used. In retrospect, if the choice for using a framework was made before the creation from scratch, it could have saved time, which could have been used for implementing the algorithm. For the remaining part of the product, the usage of libraries was correct.

When creating the web application, people were assigned tasks that were slightly overlapping, which lead to duplicate code. Components that were similar, were implemented by different people. In retrospect, when looking at the structure of the application which was made during the research phase, general components could have been made which could have been reused. It would have saved time on refactoring the code.

9 Conclusion

In this project, we were asked by OKademy to create a system in which students can be matched into groups based on soft-skills. The first two weeks consisted of a research phase in which we examined the project idea and made a plan for the process. The main requirements for the system, which followed from the research, were that it should be adaptable to the wishes of the client and that it should be intuitive. Furthermore, it needs to keep track of students' soft-skills, update these soft-skills based on feedback the students give each other and it needed to have an optimization group formation algorithm forming groups based on constraints stated by an instructor.

The requirements for this product were divided using the MoSCoW method. All requirements to create a functional product (Must Haves) have been met. More than half of requirements that the client wished for, but were not critical for its performance, were implemented (Should Haves). A few requirements that were not necessarily needed, but still listed by the client (Could Haves), have been implemented. The product built is adaptable and intuitive, thus making it easy to use for people with little technical knowledge. The system that was created for this product will be built upon in the future by other engineers. To ensure this is possible, the system architecture was built to be extensible.

Since the main part of this project was to create an group formation optimization algorithm, the system would not function without it. The algorithm was based on a genetic algorithm, since this would be able to solve the problem of the client effectively. The biggest advantages are that it is intuitive and hardly makes assumptions about the problem it is trying to solve, allowing it to be highly adaptable. The outcomes of the algorithm created are optimized well, thus enabling the system to produce usable results.

The system is tested by functionality tests, usability tests and unit tests. These tests covered everything to ensure a functional system.

All the wishes of the client and important requirements are met, resulting into a applicable system. Therefore, it can be concluded that this project has been successfully completed.

Bibliography

Marcel R Ackermann, Marcus Märtens, Christoph Raupach, Kamil Swierkot, Christiane Lammersen, and Christian Sohler. Streamkm++ a clustering algorithm for data streams. *Journal of Experimental Algorithmics (JEA)*, 17: 2–1, 2012.

Narendra M Agrawal and Mohan Thite. Nature and importance of soft skills in software project leaders. *IIM Bangalore Research Paper*, (214), 2003.

Qinghai Bai. Analysis of particle swarm optimization algorithm. *Computer and information science*, 3(1):180, 2010.

Flavia Ernesto de Oliveira da Silva, Claudia L. R. Motta, Flávia Maria Santoro, and Carlo Emmanoel Tolla de Oliveira. A Social Matching Approach to Support Team Configuration. In *Groupware: Design, Implementation, and Use*, Lecture Notes in Computer Science, pages 49–64, Berlin, Heidelberg, 2009. Springer. ISBN 978-3-642-04216-4. doi: 10.1007/978-3-642-04216-4_5.

C. García-Martínez, O. Cordón, and F. Herrera. A taxonomy and an empirical analysis of multiple objective ant colony optimization algorithms for the bi-criteria TSP. *European Journal of Operational Research*, 180(1): 116–148, July 2007. ISSN 03772217. doi: 10.1016/j.ejor.2006.03.041.

Sabine Graf and Rahel Bekele. Forming heterogeneous groups for intelligent collaborative learning systems with ant colony optimization. In *International conference on intelligent tutoring systems*, pages 217–226. Springer, 2006.

M. Mahdi Barati Jozan and Fattaneh Taghiyareh. An evolutionary algorithm for homogeneous grouping to enhance web-based collaborative learning. 2013.

Jun Ouyang and Gui-Rong Yan. A multi-group ant colony system algorithm for TSP. In *Proceedings of 2004 International Conference on Machine Learning and Cybernetics (IEEE Cat. No.04EX826)*, volume 1, pages 117–121, Shanghai, China, 2004. IEEE. ISBN 978-0-7803-8403-3. doi: 10.1109/ICMLC.2004.1380626.

Yen-Ting Lin, Yueh-Min Huang, and Shu-Chen Cheng. An automatic group composition system for composing collaborative learning groups using enhanced particle swarm optimization. *Computers & Education*, 55(4): 1483–1493, 2010.

Naseebah Maqtary, Abdulqader Mohsen, and Kamal Bechkoum. Group Formation Techniques in Computer-Supported Collaborative Learning: A Systematic Literature Review. *Technology, Knowledge and Learning*, 24 (2):169–190, June 2019. ISSN 2211-1670. doi: 10.1007/s10758-017-9332-1.

Julián Moreno, Demetrio A. Ovalle, and Rosa M. Vicari. A genetic algorithm approach for group formation in collaborative learning considering multiple student characteristics. *Computers & Education*, 58(1):560–569, January 2012. ISSN 0360-1315. doi: 10.1016/j.compedu.2011.09.011.

V. Selvi and Dr.R. Umarani. Comparative Analysis of Ant Colony and Particle Swarm Optimization Techniques. *International Journal of Computer Applications*, 5(4):1–6, August 2010. ISSN 09758887. doi: 10.5120/908-1286.

Ivan Srba and Maria Bielikova. Dynamic group formation as an approach to collaborative learning support. *IEEE transactions on learning technologies*, 8(2):173–186, 2014.

Anon Sukstrienwong. Genetic algorithm for forming student groups based on heterogeneous grouping. In *Recent advances in information science: Proceedings of the 3rd European conference of computer science*, pages 92–97, 2012.

Anja van den Broek, Kyra de Korte, Joris Cuppen, Froukje Wartenbergh, Joyce Bendig-Jacobs, José Mulder, and Anouk Hellegers. Monitor beleidsmaatregelen 2017-2018. *ResearchNed Nijmegen*, 2018.

Darrell Whitley. A genetic algorithm tutorial. *Statistics and computing*, 4(2):65–85, 1994.

Yaqian Zheng, Chunrong Li, Shiyu Liu, and Weigang Lu. An improved genetic approach for composing optimal collaborative learning groups. *Knowledge-Based Systems*, 139:214–225, January 2018. ISSN 0950-7051. doi: 10.1016/j.knosys.2017.10.022.

A
Info Sheet

Title - Soft-skill profile based group formation system
Client organization - OKademy
Date final presentation - Thursday July 2nd 12:00h

Description - Creating efficient collaborating groups is difficult. Most of the time, they are either randomly assigned or chosen by students themselves and perform poorly. OKademy found a solution for this problem: creating efficient collaboration groups based on soft-skills. They asked us to create a tool in which students' soft-skills can be tracked and matched into groups.

We created a system in which students can initialize their best soft-skills, which will be used in an algorithm that makes theoretically well-functioning groups for courses at the university. After the collaboration, the students will rate each other's soft-skills, of which the grades will be updated until graduation.

The core challenge of this project was deciding and creating the algorithm for this project. The algorithm needed to be optimized based on two requirements: the groups needed to be as optimal as possible and all groups need to equally good.

For the web application and database, we wrote functionality tests, which were executed by ourselves, and usability tests, which were performed by non-computer science people. The algorithm was tested by unit tests.

Process - We used Scrum with weekly sprints and had a weekly meeting with our client and coach. The requirements changed little during the process after the research phase. During the research phase, the idea of the project changed significantly. As a consequence, we had to not only research for a proper algorithm for the product, but also make sure the algorithm would still fit and would be adaptable in case the requirements changed significantly. This made us choose for the Genetic Algorithm.

Outlook - This system will be built upon and used after adaptation and modifications, which is why we made our product adaptable. The missing requirements could be implemented in future work.

Project team - All team members contributed to testing, preparing the report and the final project presentation.

Name	Interests	Contribution	Role
Fianne Stoel	Automata, Languages and Computability	Algorithm	Head Algorithm Optimization
Job Kanis	Computational Intelligence Concepts of Programming Languages	Algorithm	Software Quality Manager
Mostafa Khattat	Operating Systems Embedded Software	Environment, web application	Top-Level Architectue
Nikki Bouman	Web Development Multimedia Computing	Web application	Public Relations Manager
Vanisha Jaggi	Automata, Languages and Computability IT & Values	Web application	Software Quality Manager

Contact persons

Naqib Zarin	Co-owner OKademy	*naqibzarin@okademy.nl*
Christoph Lofi	Web information systems group at Department EEMCS	*c.lofi@tudelft.nl*

The final report for this project can be found at: http://repository.tudelft.nl.

B
Original Project Description

What is OKademy?

Due to lack of surgery assistants hospitals are forced to cancel planned surgeries. Med-students are eager to help in the operation theater. OKademy trains these students to become surgery assistants and posts them at hospitals after they have successfully finished the program. This way lives can be saved.

What is the problem?

This project focuses on team-forming based on soft skills. Before any candidate is proposed to a certain hospital, the hospital wants to know whether this candidate fits in their teams. In contrast to claims about their hard skills claims about their soft skills is hard to measure and proof.

What will you develop?

You will design/develop a tool that med-students can use at the start of their study. Feedback from other group members at the university will be used to build a reliable profile during the first three years. This profile can be used for matching and team-forming.

Research need to be done on subproblems such as : How to quantify soft skills? How to ensure measurements are reliable? How to match profiles? Etc. etc.

C Requirements

Must have

- Users must be able to sign-in.
- Users must be able to register.
- Members must be able to initialize their top 10 soft-skills.
- Members must be given a grade for their top 10 soft-skills, ranging between a 6 and an 8.
- Members must be able to enroll for a course during the enrollment phase.
- Members must be able to give feedback to five random soft-skills from the top 20 of their team members by either upvoting, staying neutral or downvoting soft-skills.
- Members must be able to recommend at most one new soft-skill to their team members.
- Hosts must be able to create a course.
- Hosts must be able to view their courses and corresponding groups.
- Hosts must be able to create a list of soft-skills for their course.
- Hosts must be able to initialize the allowed amount of people per group.
- Hosts must be able to run an algorithm that produces groups for the course, where each soft-skill is represented by at least one team member.
- Hosts must be able to give a start- and end date to the enrollment phase.
- Hosts must be able to give a start -and end date to the feedback phase.
- When a soft-skill is rated, the amount of ratings must be increased by one.
- When a soft-skill is rated, the grade for that soft-skill must be adjusted accordingly.
- New skills that are added to a member's soft-skill list must be initialized with a 6.
- The soft-skills of a member must be ranked according to their grade.
- The system architecture must be extensible.

Should have

- Users should not be able to own multiple accounts.
- Members should be able to provide a brief anecdote for each soft-skill in their initial soft-skill top 10.
- Members should be able to unenroll for a course during the enrollment phase.

- Members should be able to view their courses and corresponding group.
- Members should be able to view their list of soft-skills, their corresponding grades and amount of ratings.
- Hosts should be able to edit their courses before the feedback phase.
- Hosts should be able to edit the groups before the feedback phase.
- Hosts should be able to run the group formation algorithm multiple times during the matching phase.
- When soft-skills have the same grade, they should be ranked by their amount of ratings.
- Feedback should be processed to affect the grades of a member's soft-skills after all team members have given feedback.
- Skills that are recommended to a member's soft-skill list, but are already in that list, should remain the same grade but the amount of ratings will be increased.

Could have

- Hosts could create a list of soft-skills per member in a group in their course.
- Hosts could only be able to edit the soft-skill list before the work phase.
- Users could verify their e-mail address.
- Users could only be able to sign-in with their school/university e-mail.
- When the host edits the groups, he could be able to see the predicted performance of the current group division.
- Hosts could initialize the amount of groups instead of the amount of members per group.
- A course could be owned by multiple hosts.
- The course details could be imported and exported.
- A profile could have a profile image.
- The system could contain stereotypes based on soft-skills.
- Users could change their profile settings.

Won't have

- Members won't be able to give a preference for team members.
- Members won't be able to give a preference for certain predefined projects.
- The system won't be a different application than a web application.

D Group Formation Algorithms Comparison from Initial Research

Automatically forming well-functioning collaborative learning groups is a complex problem for which no simple algorithm exists. Because of the complexity of the problem and the many, partially unknown, factors collaborative learning depends on, a lot of different approaches to solving this problem exist (Maqtary et al., 2019). In the literature, similar problems often start with defining the criteria what, in their opinion, constitutes a well functioning group, following with an algorithm to form the groups.

As described in a literature review of Maqtary et al. (2019), there currently exist four major ways for approximating the best group formation: using Ant Colony Optimization, using a Clustering Algorithm, using a Genetic Algorithm and using Particle Swarm Optimization. We will now provide a small description about these methods, describe how they could be used for our formation criteria and then provide some advantages and disadvantages of using these methods.

Ant Colony Optimization

Ant Colony Optimization is based on ants' path finding behaviour (García-Martínez et al., 2007). Ants try to find the shortest path from their nests (source) to food sources (destination) by leaving pheromones (chemical trail) on their path. These pheromones guide other ants to follow the same path (Jun Ouyang and Gui-Rong Yan, 2004; Graf and Bekele, 2006). However, ants choose the paths by probabilistic decision: the more ants follow a path, the stronger the pheromones, which consequently results in a higher desirability for ants to follow this path. (García-Martínez et al., 2007; Graf and Bekele, 2006) This leads to ants creating the shortest path from their source to their destination.

The Ant Colony Optimization algorithms are able to solve NP-hard problems, e.g. the Traveling Salesman problem (García-Martínez et al., 2007; Jun Ouyang and Gui-Rong Yan, 2004; Graf and Bekele, 2006). The Ant Colony Optimization itself focuses on distancing problems; it searches the shortest distance, however, it can also be applied to different optimization problems (Graf and Bekele, 2006).

Modifications

The Ant Colony Optimization needs modifications to fit our problem. In the paper Graf and Bekele (2006), the problem of grouping students in groups using Ant Colony Optimization is tackled. This paper aims to measure the difference between groups by measuring the Euclidean Distance and checking the Goodness of Heterogeneity. Students have attributes which have three possible values for which 1 indicates low and 3 indicates high. Each student is represented by a vector. This is made up of the the student's attribute values. Students also have a student-score, which is used to measure heterogeneity and represents the sum of the student's attributes.

In order to solve our matching problem, we need to modify the problem description given by Graf and Bekele (2006). To do so, we need to change the following: the students need to be changed to members, the attributes need to be changed to the relevant soft-skills provided by the host. The members' vectors consist of these relevant soft-skills. These soft-skills have 2 possible values: 1 indicates that the member possesses the soft-skill and 0 indicated the member does not. An addition is needed to transform this problem: in a group, for every soft-skill it needs to be defined what the minimum and maximum amount of members that are allowed to posses the soft-skill is. Due to the addition, we do not need to measure full heterogeneity, which results in the student-score to be removed. As a consequence the Goodness of Heterogeneity is not needed in out problem. Furthermore, we need to transform the Euclidean Distance to give an output which does not show how much members differ, but how well they would do in the same group, meeting the requirement of having a minimum and maximum constraint per soft-skill in one group.

Advantages

- It can be applied to different optimization problems (Selvi and Umarani, 2010).
- It has dynamic applications (Selvi and Umarani, 2010).
- Parallel inheritance (Selvi and Umarani, 2010)

Disadvantages

- It uses a high magnitude of computing (Jun Ouyang and Gui-Rong Yan, 2004).
- It can be inefficient (Jun Ouyang and Gui-Rong Yan, 2004).
- It can arrive at a stagnating state (Selvi and Umarani, 2010).
- Random decisions are not independent (Selvi and Umarani, 2010).
- The research is experimental; not theoretical (Selvi and Umarani, 2010).
- The probability distribution does not stay the same (Selvi and Umarani, 2010).
- The problem needs to be modified in order to take the minimum and maximum amount of members having the soft-skills into account. This would lead to major changes.

Clustering Algorithm

Clustering Algorithms are algorithms that partition a set of objects into clusters, based on the similarity of these objects; similar objects are put into the same cluster (Ackermann et al., 2012). This algorithm can be used to classify very large datasets into similar groups (Ackermann et al., 2012).

In Srba and Bielikova (2014) it is described that, although it can be said that diversity in characteristics, skills and aims are useful when working in a group, sometimes these differences can make it that a collaboration is more difficult. It can be important to make sure that students get along with each other if they have to work together in a group. In Srba and Bielikova (2014) a Clustering Algorithm that uses students' characteristics is used in a proposed group formation method. Students were represented by a vector of characteristics and the algorithm then identifies clusters of compatible members. The proposed method was tested by dividing 110 students into groups and the result was a higher quality of collaboration (Srba and Bielikova, 2014).

Modifications

As shown in Srba and Bielikova (2014), Clustering Algorithms can be used to group students with similar properties together. Groups that are made this way are called homogeneous. For the problem that we need to solve, an algorithm is needed that can create partial heterogeneity in groups, since we need to create groups with members that have different soft-skills. We have not been able to find modifications that make it possible to use a Clustering Algorithm for our problem. Because of this no advantages or disadvantages are listed; these would not be useful for an algorithm that we cannot use.

Genetic Algorithm

Genetic Algorithms can be considered as a family of computational models that are inspired by the evolution principles of Darwin (Whitley, 1994). Genetic Algorithms represent possible solutions as chromosomes composed of genes (Moreno et al., 2012). Starting with an initial population of a number of chromosomes, a fitness function, which is a function determining how good a solution is, is used to determine which chromosomes will produce offspring for the next generation (Moreno et al., 2012). The selected chromosomes will be multiplied and receive random modifications by genetic operators eventually resulting in a new population (Moreno et al., 2012). This process is then repeated until specified stopping criteria are met, eventually resulting in an approximation of the 'best solution' of the problem (Moreno et al., 2012).

Genetic Algorithms are often used for approximating non-linear functions (Whitley, 1994). By traversing the search-space relatively efficient, it is possible to approximate sparse solutions in an exponential search space (Whitley, 1994). Because of this, they are able to approximate several problems in NPO.

Modifications

The Genetic Algorithm can be used in creating a proper group division by either making the genes represent the members and the positions of the genes in the chromosome represent the groups, as done in Moreno et al. (2012); de Oliveira da Silva et al. (2009), or by having the genes represent the groups and the positions of the genes in the chromosome represent the members, as done in Moreno et al. (2012); de Oliveira da Silva et al. (2009). By defining the genetic operators in a way that it cannot create a disallowed group division, the group formations can be adapted. By defining a fitness function to represent the criteria defined by the client, the algorithm will approximate a 'good' group formation.

Advantages

- It is able to approximate a solution in an exponential search space with a sparse amount of 'good solutions' (Whitley, 1994)
- It does not make many assumptions about the problem it is solving (Whitley, 1994). While this will approximate the results less efficiency, it does allow for a very dynamic fitness function.
- Because of the different generations, the algorithm can be stopped before meeting any of the stopping criteria and will still return an approximated result.
- The algorithm allows for a straight-forward way of representing groups and members.

Disadvantages

- The fitness function must be fast (Whitley, 1994).
- The solution should, for the best results, be represented in bit-strings (Whitley, 1994).

Particle Swarm Optimization Algorithm

Particle Swarm Optimization is based on swarms intelligence (Bai, 2010), in particular fish schools and bird flocks (Selvi and Umarani, 2010; Lin et al., 2010). Social insects as well as other animal societies (Selvi and Umarani, 2010) are able to collaboratively identify an optimal solution to complex problems, e.g. birds that can collaboratively find a food source (Lin et al., 2010). The individual behaviour of swarm members is not only influenced by their own experience but also by the behaviour of the swarm (Lin et al., 2010).

In the Particle Swarm Optimization algorithm, all swarm members are represented by n particles. Each particle has a position and a speed. The position is calculated through a fitness function and represents a potential solution for the optimization: optimal position of itself and optimal solution of the swarm (Bai, 2010; Lin et al., 2010). The particles can change their conditions. They can change it to the most optimal position according to themselves, they can change it to the most optimal position according to the swarm and keep its inertia. The optimal solution to this problem based on the iterative process is evolved until the termination criteria or maximum iterations are achieved (Bai, 2010). This process leads the swarm to reach its destination.

For an optimization problem in a search space or model, it is able to find solutions. It can predict the behaviour in the presence of objectives (Selvi and Umarani, 2010).

Modifications

The Particle Swarm Optimization needs modifications to fit our problem. In the paper Lin et al. (2010), the problem of grouping students using the Particle Swarm Optimization principle is tackled. In order to solve our matching problem using Particle Swarm Optimization, we need to modify the problem description given by Lin et al. (2010). In this paper, n students exist, divided into r groups $R_1, R_2, .., R_r$. $T_1, T_2, ..., T_k$ instructors are assigned to k topics to teach. Each student needs to take a pre-test and gets k pre-testing scores. The paper Lin et al. (2010) aims to solve the problem by using a fitness function which is able to find an optimal solution with minimum difference between any two groups. This difference is based on the understanding level and interest in each of the topics of each student. The aim is for each group to have the same level of understanding on each topic.

To form this problem description to fit ours, we need to modify the following: the students need to be changed to members, the instructors $T_1, T_2, ..., T_k$ will be removed, the k topics will be changed to the relevant soft-skills provided by the host and the pre-test scores of each member will be changed to a grade for the corresponding soft-skill; a 1 if the member possesses the soft-skill and a 0 if he does not possess it. The understanding level of the paper is in our case the possession of a certain soft-skill and the topic is the certain soft-skill. Their solution relates to our problem, since we need to find the minimal distance between groups so that the group compositions are equally good. One extra requirement is needed for our problem: a minimum and a maximum amount of members in one group allowed to posses a certain soft-skill.

Advantages

- It is applicable into scientific research as well as engineering (Selvi and Umarani, 2010; Bai, 2010).
- No overlapping and mutation calculation, since the search can be executed by the particle-speed. The researching is very fast (Selvi and Umarani, 2010; Bai, 2010).
- It occupies bigger optimization ability in comparison to other developing calculations. It is very simple (Selvi and Umarani, 2010; Bai, 2010).

Disadvantages

- It suffers partial optimism (Selvi and Umarani, 2010; Bai, 2010).
- It cannot work out non-coordinate system problems (Selvi and Umarani, 2010; Bai, 2010).
- The algorithm would need to be modified to be able to take a minimum and maximum amount of soft-skills per group into account.

Selected Matching Algorithm

Based on research done on the Ant Colony Optimization Algorithms, Clustering Algorithms, Genetic Algorithms and Particle Swarm Optimization Algorithms, we compared the algorithms and selected the one which fits best to our problem and has the best optimization solution. Based on our findings, we have selected to use the Genetic Algorithms. We will now explain the reasoning behind this decision by discussing the practicalities of the usage of the algorithms.

Clustering Algorithms can not be modified to our problem. It focuses on homogeneous groups. If we were to change this to have partially homogeneous groups to fit our problem, the point of the Clustering Algorithms is defeated. It will become a different optimization principle; not based on clustering. Based on this, we did not choose this algorithm.

Ant Colony Optimization can be applied to different optimization problems and has many dynamic applications. Still its high magnitude of computing makes the algorithm inefficient. The original optimization problem is mainly focused on distance problems. In this problem, this means it is focused on creating heterogeneous groups. Changes can be made so that the distance function represents members belonging together in a group. In our problem, this implies partial heterogeneity. These changes however, are major and it remains a difficult task since not all relevant soft-skills have the same division and the division changes per course. Based on this, we did not choose this algorithm.

Particle Swarm Optimization occupies bigger optimization ability in comparison to other developing calculations, which makes it easier. However, the optimal solution may not always be as optimal as possible: it suffers from partial optimization. The algorithms focuses mainly on collaboration between members in order to find an optimal location. This can be modified to fit our problem, however, adding a minimum and maximum number of members allowed to have a certain soft-skill within a group remains difficult for this problem to implement using these algorithms. Based on this, we did not choose this algorithm.

Genetic Algorithms are flexible and do not make many assumptions when solving a problem. This ensures less deviation and works towards a better and more accurate solution in comparison the the other algorithms. Furthermore, they allow a dynamic fitness function, which continuously increases the pressure on code correctness. Modification needed to solve our problem can be made. In comparison to the other algorithms, there are no difficulties when doing so. The modification that Ant Colony Optimization and Particle Swarm Optimization struggle with, are the criteria of having a minimum and maximum restriction per soft-skill. Genetic Algorithms can solve this using a fitness function without having the difficulties other algorithms face. Based on this and the previous algorithms, we choose this algorithm.

E

Soft-skill List

1. **Adaptability**
   Adaptability means that you can efficiently continue working under changing circumstances by constantly adapting to the environment

2. **Analytical thinking**
   Analytical thinking is the methodical step-by-step thinking approach that allows you to split complex problems into separate and manageable components

3. **Attitude**
   Attitude is the attitude you have towards a person, place or subject. An attitude is therefore an inner attitude, which is formed by knowledge or experience

4. **Change management**
   Change management is to control or manage change.

5. **Conceptual skills**
   Conceptual skills are skills that allow an individual to understand complex situations to develop creative and successful solutions. In other words, it is a natural talent that tackles difficult scenarios with an innovative approach

6. **Courtesy**
   Courtesy is showing politeness in someone's attitude and behavior towards others

7. **Creativity**
   Creativity is the ability to think in a new or different way about a task or a problem, or the ability to use imagination to create new ideas

8. **Critical thinking**
   Critical thinking is the mental process of carefully gauging details to find the best way to interpret it to make the right judgment

9. **Cross-cultural skills**
   Cross-cultural skill is the ability to function effectively between cultures, to think and act appropriately, and to communicate and work with people from different cultural backgrounds - at home or abroad

10. **Decision making skills**
    Decision making skill is choosing between two or more alternatives to achieve the best result in the shortest time

11. **Empathy**
    Empathy is the ability to accurately place yourself "in someone else's shoes" - to understand the other person's situation, perceptions and feelings from their point of view - and to communicate your understanding to the other person

12. **Flexibility**
    Flexibility is the willingness and ability to take on new responsibilities. It is not just a 'can do' attitude, but a 'want to do' approach. To be truly flexible, priorities must be adaptable to organizational needs

13. **Goal setting**
    Goal setting is a process that starts with a careful consideration of what you want to achieve and ends with a lot of hard work to actually do it

14. **Initiative**
    Initiative is the ability to be resourceful and to work without always being told what to do

15. **Integrity**
    Integrity is following moral or ethical beliefs and doing the right thing under all circumstances, even when no one is looking at you

16. **Listening skills**
    Listening skill is the ability to pay attention to and effectively interpret what other people are saying

17. **Networking**
    Networking is the ability to exchange ideas and information with both groups and individuals with shared interests, so that long-term relationships are developed with mutual benefit

18. **Nonverbal communication**
    Nonverbal communication is the transmission of information through body language, including eye contact, facial expressions, gestures and more

19. **Organizational politics**
    Organizational politics is about selling ideas, influencing others and achieving goals; they are informal and unofficial and can sometimes occur "behind closed doors"

20. **Patience**
    Patience is the ability to accept or tolerate delay, problems or suffering without getting angry

21. **Persuasion skills**
    Persuasion skill refers to the talent to change the attitudes, beliefs or behaviors of a person or group towards a person, group, event or idea

22. **Perceptions**
    Perception is the ability to capture, process and actively understand the information our senses receive

23. **Presentation skills**
    Presentation skills are a set of skills that enable an individual to interact with the audience, convey messages clearly and involve the audience in the presentation

24. **Problem solving skills**
    Problem solving skills refer to our ability to solve problems effectively and in a timely manner without obstacles. It includes being able to identify and define the problem, evaluate and select the best alternative and implement the selected solution

25. **Professionalism**
    Professionalism is the behavior or attitude of an individual in work or business environment

26. **Relationship management**
    Relationship management is knowing how to develop and maintain good relationships, communicate clearly, inspire and influence others, work well in a team and manage conflicts

27. **Responsibility**
    Responsibility is to stand behind what you say. People are committed to fixed values such as honesty and loyalty

28. **Risk-taking**
    Risk taking refers to someone who risks everything in the hopes of achieving something or accepts a greater loss in decisions and tolerates uncertainty

29. **Self-awereness**
    Self-awereness refers to a person's ability to know or recognize their emotions, behaviors, beliefs, motivations, and other characteristics such as strengths and weaknesses, enabling them to identify and understand themselves as a separate entity

30. **Self-management skills**
    Self-management is controlling your impulsive feelings and behavior, managing your emotions in a healthy way, fulfilling obligations and adapting to changing circumstances

31. **Social-awereness**
    Social awereness is the ability to understand other people's emotions, needs and concerns, pick up emotional cues, feel socially comfortable and recognize the power dynamics in a group or organization

32. **Spoken Communication**
    Spoken Communication is the sharing of information between individuals through speech

33. **Strategic Planning**
    Strategic Planning is the process of setting a vision for a company and realizing that vision through small, achievable goals.

34. **Team building skills**
    Team building refers to the activities undertaken by groups of people to increase their motivation and stimulate collaboration

35. **Teamwork**
    Teamwork skills are the qualities and capabilities with which you can work well with others during conversations, projects, meetings or other collaborations

36. **Time management skills**
    Time Management refers to effectively managing time so that the correct time is allocated to the correct activity

37. **Work Ethics**
    Work Ethics is the ability to maintain the right moral values in the workplace

38. **Working under pressure**
    Working under pressure is the ability to work under pressure with restrictions over which you have no control. Constraints can be resources, time, difficulty of the task, or insufficient knowledge required to complete the task

39. **Writing communication**
    Writing communication is necessary to convey your point in writing, which depends on grammar, punctuation and choice of words

F

Functionality Tests

functionality_tests.md

General

Components

- Popups:
  - can be closed by clicking on the X
  - close automatically after some time
  - have the right color of indication (red for error, green for success)
- Dialog
  - if there is an X in the toolbar, this can be clicked on and the dialog closes
- Modal
  - can be closed by clicking on the X
- Button
  - always does something (e.g. lead to other page, show popup, etc)

Register

Sign in page:
This page should create a new account for a user that has the right to have an account (this depends on the e-mail address).

1. Click on "Don't have an account? Sign Up"
   Leads to register page ("create new account")

Register page

1. Don't fill in all required fields and click on "create"
   Error pops up saying you have to fill in all required fields
2. Enter non-student e-mail, click on "send verification code"
   Error pops up saying you need an institution e-mail
3. Enter two different passwords, click on "create"
   Error pops up saying your passwords do not match
4. Enter institutional e-mail on which you already have an account, click "send verification code"
   Error pops up saying you already have an account
5. Enter institutional e-mai and click on "send verification code"
   Success pops up saying the verification code is sent
   E-mail is sent to the address containing a code

- Enter wrong code, click on "create"
  Error pops up saying the code is incorrect
- Enter right code, change e-mail address to another institutional one
  Error pops up saying the code is incorrect
- Enter right code, click on "create"
  Leads to sign in page, success pops up saying the account is made

Sign in

Sign in page:
This page should lead a user to the content of the website if the user enters the correct credentials.

1. Enter wrong email and "correct" password
   Error pops up saying it is incorrect
2. Enter correct email and wrong password
   Error pops up saying it is incorrect
3. Enter wrong email and password
   Error pops up saying it is incorrect
4. Enter correct email and password
   Leads to corresponding homepage (members to memberhome, hosts to hosthome)

Log out

Any page while logged in

1. Click on "log out" in the sidebar
   You are taken back to the sign in page
   You cannot go to any page (e.g. type in the site with "/memberHome", you do not see that page but rather the sign in page, or if the page does not exist you see a "404" error)

Member

First sign-in

Member home
This page should pop up when the user has no soft-skills initialized yet. This page is thus created to initialize the soft-skills. I will call the two divs in which you can place the soft-skill drag/droppables "blocks".

1. Drag a soft-skill to a random place (not in a block)
   It is placed back to where it came from
2. Drag a soft-skill from one list to another
   It is placed in the other list
3. Drag a soft-skill on top of the other (reorder them)
   It is placed where it should be: it is reordered
4. Click on the "i" icon next to a soft-skill name
   A modal pops up telling you the description of that soft-skill
5. Click on save when you have 0 to 9 soft-skills in the block 'My top 10'
   A modal shows up saying you need 10 soft-skills
6. Click on save when you have 10 or more soft-skills
   A modal shows up, showing your chosen soft-skills
   The order (and grades) is correct
   - Click on "no"
     You can drag and drop soft-skills like before and initialize a new top 10
   - Click on "yes"
     You are taken to the homepage. You there see your initialized soft-skills

Enroll

Course catalog
This page is created to let members enroll for (participate in) a course. These members will then be taken into the matching phase, where the groups are created.

1. Click on "enroll" for a course that is in the enrollment period
   The "enroll" button changes color, green to red The course can be seen on your homepage
2. You cannot click on "enroll" for a course that is not in the enrollment period

View course

Member home

1. Click on the "i" next to a soft-skill title in the soft skill panel
   A modal pops up telling you the description of that soft-skill
2. Click on "view course" at a random course in the course panel
   You are taken to the course and group information page\

Course and group information

1. The course is in the enrollment period
   Your group is empty. You can see the course details
2. The course is in the matching period
   Your group is empty. You can see the course details
3. The course is in the ongoing period
   Your group has members. You cannot click on a button to give them feedback. You can see the course details.
4. The course is in the feedback period
   Your group has members. You can click on a button to give them feedback. You can see the course details.
5. The course has ended
   Your group has members. You cannot click on a button to give them feedback. You can see the course details.

Give feedback

Member home

1. Click on a course that is in the feedback period

Course and group information

1. Click on "give feedback" on one of the members
   Feedback page
   You can see on the panelbar who you are giving feedback to
   You can see five soft-skills of that member with corresponding grade and amount of votes
2. Click on the "recommend a new skill" textfield
   You can type a soft-skill and it will show corresponding results by the soft-skill name (not description)
   - Click on a soft-skill that showed up
     That soft-skill is now displayed in the textfield
3. Click on the arrow next to the "recommend a new skill" textfield
   It expands, showing all soft-skills
   - Click on a soft-skill
     That soft-skill is now displayed in the textfield
4. Click on the "i" next to a soft-skill title
   The description of the soft-skill is shown
5. Click on one of the dropwdowns in a soft-skill card that is by default saying "this is sufficient"
   It drops down, giving you three options (lower, sufficient, higher)
   - Click on one of them
     The one you clicked on is now shown in the bar instead of the default "sufficient"

6. Click on "save" at the top right of the screen
   You are taken back to the course and group information page

Course and group information
There is no "give feedback" button on a member that you have given feedback to
The chip says "feedback given" in green

Search bar

Course overview

1. Click on the search bar
   - Type in what course you want to find
     All courses containing this in their course ID or course name appear

Course catalog

2. Click on the search bar
   - Type in what course you want to find
     All courses containing this in their course ID or course name appear

Host

First sign-in

Host home
You see an empty course overview with a "+" in it

Create a course

Host home

1. Click on the "+" in the course overview panel

Create course page

1. Don't fill anything in and click on "create"
   An error pops up saying you should fill in all required fields
2. Click on the name or ID textfield
   You can type now
3. Click on the "preferred group size" textfield
   You can only fill in numbers
4. Click on the "+" inside the "required soft skills" panel
   A row is added
   - Click on the dropdown ("select soft skill(s)")
     You can choose any amount of soft-skills by clicking on them
   - Click on the "min members" or "max members" textfield
     You can only fill in numbers
     - Make "min members" larger than "max members"
       - Click on the checkmark
         An error pops up saying "min members" cannot be larger than "max members"
     - Make "min members" less than or equal to 0
       - Click on the checkmark
         An error pops up saying "min members" cannot be less than one"
   - Click on the checkmark when everything is filled in correctly (one or more softskills are selected, 0 < min < max)
     The requirement is static in the panel
     - Click on the pencil icon next to the static requirement
       You can edit it again with the behaviour like before

        - Click on the delete icon next to the static requirement
          You are asked if you are sure to delete the row
            - Click on the X
              You are taken back to the row like it was before
            - Click on the checkmark
              The row is deleted, not to be seen on the screen anymore

5. Click on any of the date fields in step 3
   A date picker pops up on which you can select a date and click ok
    - Select a date
      That date is displayed in the textfield
        - Make the enrollment or feedback end less than or equal to the enrollment or feedback start and click on "create"
          An error pops up saying the enrollment end must be later than the enrollment start
        - Make the feedback start less than or equal to the enrollment end and click on "create"
          An error pops up saying the feedback period should be after the enrollment period
        - Enter logical dates (start < end, enrollment < feedback) and click on "create"
          You are taken back to the course overview

Course overview
You can now see your added course

View course

Host home

1. Click on "view course" at a random course in the course panel
   You are taken to the course information page

Course information

1. The course is in the waiting for enrollment period

    - One button "edit" is visible on the bottom right
        - Click on the button "edit"

            - Don't fill anything in and click on "save"
              Nothing changes. You go to the course information page

            - Click on the name or ID textfield
              You can type now

            - Click on the "preferred group size" textfield
              You can only fill in numbers

            - Click on the "+" inside the "required soft skills" panel
              A row is added

                - Click on the dropdown ("select soft skill(s)")
                  You can choose any amount of soft-skills by clicking on them
                - Click on the "min members" or "max members" textfield
                  You can only fill in numbers
                    - Make "min members" larger than "max members"
                        - Click on the checkmark
                          An error pops up saying "min members" cannot be larger than "max members"
                    - Make "min members" less than or equal to 0
                        - Click on the checkmark
                          An error pops up saying "min members" cannot be less than one"
                    - Click on the checkmark when everything is filled in correctly (one or more softskills are selected, 0 < min < max)
                      The requirement is static in the panel

        - Click on the pencil icon next to the static requirement
          A row is updated
          You can edit it again with the behaviour like before
        - Click on the delete icon next to the static requirement
          You are asked if you are sure to delete the row
            - Click on the X
              You are taken back to the row like it was before
            - Click on the checkmark
              The row is deleted, not to be seen on the screen anymore A row is deleted
    - Click on any of the date fields in step 3
      A date picker pops up on which you can select a date and click ok
        - Select a date and click on ok
          That date is displayed in the textfield
            - Make the enrollment or feedback end less than or equal to the enrollment or feedback start and click on "save"
              An error pops up saying the enrollment end must be later than the enrollment start
            - Make the feedback start less than or equal to the enrollment end and click on "save"
              An error pops up saying the feedback period should be after the enrollment period
            - Enter logical dates (start < end, enrollment < feedback) and click on "save"
              You are taken back to the course information page

2. The course is in the enrollment period
    - One button "edit" is visible on the bottom right
        - Click on the button "edit"
            - Same behaviour as before
3. The course is in the matching period
    - One button "edit" is visible on the bottom right
        - Click on the button "edit"
            - Same behaviour as before
    - One button "create groups" is visible on the bottom right
        - Click on the button "create groups"
          You can see all softskills and your groups for that course
            - Click on the pencil icon "Assign this member to a different group*
              You can assign the member to a different group
                - Click on "save changes" after assigning the member to a different group*
                  The member should be visible in a different group
            - Click on the minus icon "remove this member from this group
              The member is placed in a "Members Without a Group" block
            - One button "add a group" is visible on the bottom right*
                - Click on the "add group" button
                  A new and empty group appears
                  Click on the "Or remove this group" text
                  The group is removed
            - One button "regenerate groups" is visible on the bottom right*
                - Click on the "regenerate groups" button *A warning appears*
                    - Click on no
                      Nothing changes
                    - Click on yes
                      New groups are made
            - Click on "save" on the top right
              A warning appears
                - Click on "yes"
                  You go back to the course information page
                - Click on "no"
                  The warning disappears

        - Click on the X on the top left
          A warning appears
            - Click on "yes"
              You go back to the course information page
            - Click on "no"
              The warning disappears

4. The course is in the ongoing period

    - One button "modify or create groups" is visible on the bottom right
        - Click on the button "modify or create groups"
          You can see all softskills and your groups for that course
            - Click on the pencil icon "Assign this member to a different group
              You can assign the member to a different group
                - Click on "save changes" after assigning the member to a different group
                  The member should be visible in a different group
            - Click on the minus icon "remove this member from this group
              The member is placed in a "Members Without a Group" block
            - One button "add a group" is visible on the bottom right
                - Click on the "add group" button
                  A new and empty group appears
                    - Click on the "Or remove this group" text*
                      The group is removed
            - Click on "save" on the top right
              A warning appears
                - Click on "yes"
                  You go back to the course information page
                - Click on "no"
                  The warning disappears
            - Click on the X on the top left
              A warning appears
                - Click on "yes"
                  You go back to the course information page
                - Click on "no"
                  The warning disappears

5. The course is in the feedback period

    - One button "view groups" is visible on the bottom right
        - Click on the button "view groups"*
          You can see all softskills and your groups for that course

6. The course has ended

    - One button "view groups" is visible on the bottom right
        - Click on the button "view groups"
          Same behaviour as before

Search bar

Course overview

- Click on the search bar
    - Type in what course you want to find
      All courses containing this in their course ID or course name appear

G
Usability Tests

user_tests.md

Personas

Tiffany Hibbert

Tiffany is a first year medical student and wants to register to the soft-skills profile tool website in order to build up a good profile for her career after graduation. She takes her work and study very seriously and wants to be top of the class. This makes her very responsible, however, she is a lone wolf and like to work on her own to ensure the work is done properly.

Steps

In order to build her perfect profile, she needs to register and sign in. However, she tries to sign in with her private mail account. After finding out she can only sign in with a school account, she registers. After signing-in, she needs to fill in her top-10 soft-skills. She is scrolling through the overview of the courses and looks for the 4 courses she wants to follow: Anatomy, Clinical Research, Patient Care and Radiology. After enrollment, she realizes she took too many courses and needs to unenroll for 1 course.

Requirements

Must Haves

- Users must be able to sign in
- Users must be able to register
- Members must be able to initialize their top 10 soft-skills
- Members must be given a grade for their top 10 soft-skills, ranging between 6 and 8
- Members must be be able to enroll for a course during the enrollment phase
- The soft-skills of a member must be ranked according to their grade

Should Haves

- Members should be able to unenroll for a course during the enrollment phase.
- Members should be able to view their list of soft-skills, their corresponding grades and amount of ratings

Could Haves

- Users could verify their e-mail address
- Users could only be able to sign-in with their school/university e-mail

Note for us: Clinical Research is in the Matching Phase and Patient Care course is Waiting for enrollment.

Bert Justice

He is a second year student Computer Science. and has finished 3 courses: SQT, Web and Database, Reasoning and Logic. For the courses SQT and Web and Database, he still needs to give feedback to his team members. He wants to enroll for OOP and BEP.

Steps

He looks at his account and sees he has finished 2 course and still needs to give feedback for 2 courses. He gives feedback. He forgot what soft-skills he initialized for himself and what they mean. He looks on his profile for clarification. Suddenly, he realizes he is enrolled or the OOP Project course but needs to enroll for the OOP course instead. He switches his enrollement.

Requirements

Must Haves

- Users must be able to sign in
- Members must be able to give feedback to five random soft-skills from the top 20 of their teammembers by either upvoting, staying neutral or downvoting soft-skills
- Members must be able to enroll for a course during the enrollment phase
- Members must be able to recommend at most one new soft-skill to their team members.
- When a soft-skill is rated, the amount of ratings must be increased by one.
- When a soft-skill is rated, the grade for that soft-skill must be adjusted accordingly.
- New skills that are added to a member’s soft-skill list must be initialized with a 6.
- The soft-skills of a member must be ranked according to their grade.

Should Haves

- Members should be able to unenroll for a course during the enrollment phase.
- Members should be able to view their courses and corresponding group.
- Members should be able to view their list of soft-skills, their corresponding grades and amount of ratings
- When soft-skills have the same grade, they should be ranked by their amount of ratings
- Feedback should be processed to affect the grades of a member’s soft-skills after all team members have given feedback
- Skills that are recommended to a member’s soft-skill list, but are already in that list, should remain the same grade but the amount of ratings will be increased.

Could Haves

- Users could only be able to sign-in with their school/university e-mail

Note for us: he is already enrolled in 6 courses of which 2 are ended, 2 in feedback phase, 1 in matching period and 1 in the enrollment period. Soft skills are already initialized.

Warren Vance

He is a Computer Science teacher at a university and needs to register for the soft-skills tool website.

Steps

He is new at the university and needs to sign up for the soft-skills-profile tool website. Since he is a new user, he also wants to try to make 2 accounts. After sign-in he wants to create 2 courses: CPL and Logic Based AI. For CPL, his enrollment period is somewhere in July and the feedback period is somewhere in August. For Logic Based AI, the enrollment period is now and the feedback period somewhere in August. He realizes he made a mistake and switched the information about the courses. Therefore, he wants to edit both courses.

Requirements

Must Haves

- Users must be able to sign in
- Users must be able to register
- Hosts must be able to create a course
- Hosts must be able to create a list of soft-skills for their course
- Hosts must be able to initialize the allowed amount of people per group.
- Hosts must be able to give a start- and end date to the enrollment phase.

- Hosts must be able to give a start- and end date to the feedback phase

Should Haves

- Users should not be able to own multiple account
- Hosts should be able to edit their courses before the feedback phase.

Could Haves

- Hosts could only be able to edit the soft-skill list before the work phase
- Users could verify their e-mail address.
- Users could only be able to sign-in with their school/university e-mail

Meghan Park

She is a school teacher who works in her field for a year.

Steps

She signs in with her account and sees all of her created courses. She is not really sure about one of her courses (Maths) and wants to edit this. Since a new semester starts, she wants to create 2 courses: Physics and Biology of which the enrollment period is in July. She made a mistake in one of the courses and needs to edit this. One of her other courses (English) is in the matching period. She want to create groups. She is not happy with the results and tries again. She is still not satisfied with the result and moves 2 members around. Still not satisfied with the groups, she makes another group and puts members in it.

Requirements

Must Haves

- Users must be able to sign in
- Hosts must be able to create a course
- Hosts must be able to create a list of soft-skills for their course
- Hosts must be able to initialize the allowed amount of people per group.
- Hosts must be able to give a start- and end date to the enrollment phase.
- Hosts must be able to give a start- and end date to the feedback phase
- Hosts must be able to run an algorithm that produces groups for the course, where each soft-skill is represented by at least one team member

Should Haves

- Users should not be able to own multiple accounts
- Hosts should be able to edit their courses before the feedback phase
- Hosts should be able to edit the groups before the feedback phase
- Hosts should be able to run the matching algorithm multiple times during the matching phase.

Could Haves

- Hosts could only be able to edit the soft-skill list before the work phase