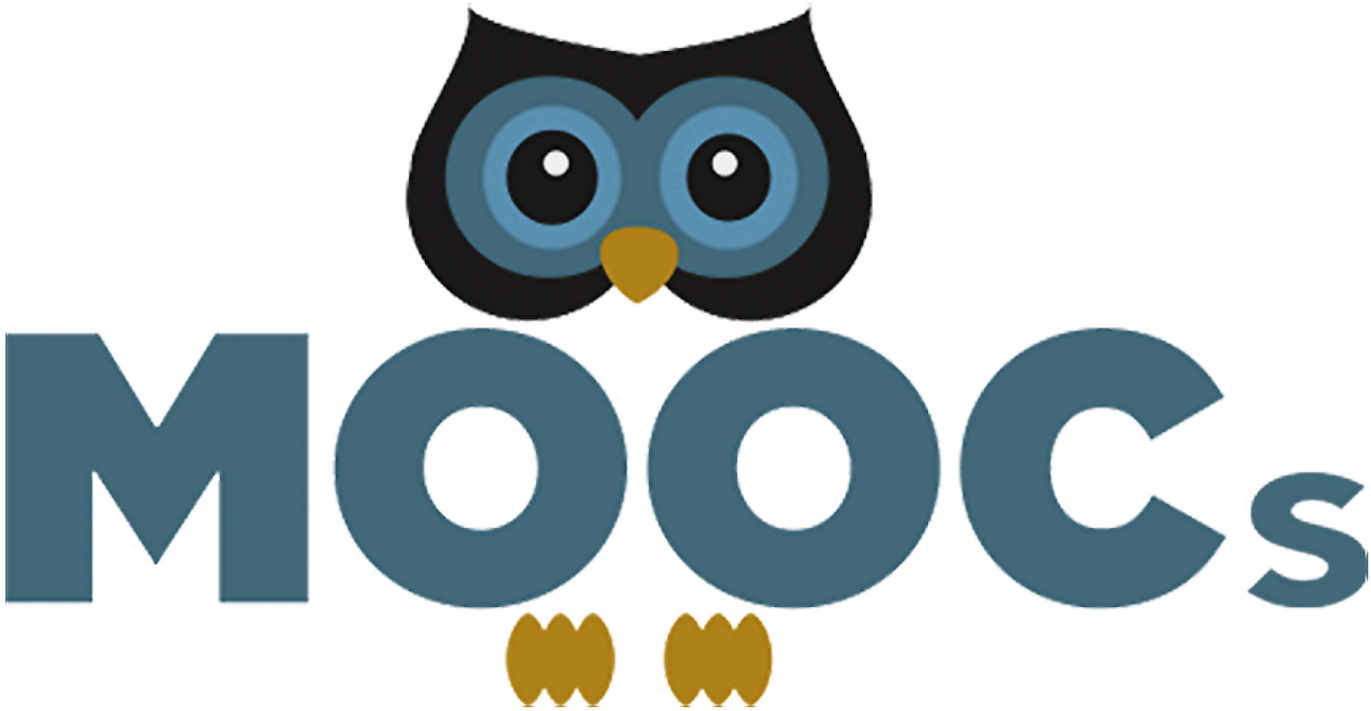
Peer Matching Tool
TI3800 Bachelorproject

Floris Verburg
Freek van Tienen
Marijn Goedegebure

Delft University of Technology



TU Delft Delft University of Technology

Challenge the future

Peer Matching Tool

TI3800 Bachelorproject

by

Floris Verburg
Freek van Tienen
Marijn Goedegebure

in partial fulfilment of the requirements for the degree of

Bachelor of Science
in Computer Science

at the Delft University of Technology,
to be defended on Tuesday June 24, 2014 at 10:00 AM.

Supervisor:	Prof. drs. dr. L.J.M. Rothkrantz
Thesis committee:	Prof. drs. dr. L.J.M. Rothkrantz
	Dr. ir. D. Datcu
	Dr. M.A. Larson

An electronic version of this thesis is available at http://repository.tudelft.nl/.

TU Delft Delft University of Technology

Summary

Massive open online courses (MOOCs) all around the world are growing in size. Therefore, the practical assignments needs to be easy to manage. The growth also means that it is more difficult for participants to find a possible practical partner due to the large amount of people. The difficulty is also affected by the fact that students do not see each other during courses, and most of the time haven't met any of the participants from the course.

For this Bachelor Project we have improved upon this process by creating an application that can be used to create practical groups without the participants being required to search intensively through all the participants. Participants are able to easily access the application, fill in some relevant information about themselves and then get recommendations of possible practical partners or practical groups.

We have divided the project in three phases:

- Research and set-up (phase 1)
- Design and implementation (phase 2)
- Testing, documentation and presentation (phase 3)

Each of these phases describes the things that needed to be done during that phase.

During the implementation phase we used a known software development framework. This development framework is known as scrum. Scrum is based on the agile software development framework which can be used to manage software projects.

We use the Play framework for development of the application. The Play framework gives us the possibility to easily create a web application using Java. This frameworks follows the Model View Controller (MVC) architectural pattern applied to the Web architecture.

For our recommendation algorithm we used our custom designed algorithm. This algorithm uses the skills defined by the user and the teacher to find the best recommendation with the preferences of the teacher.

Creating code with good quality, maintainability and extendability of the system was important during the project. We achieved this using several tools, like Findbugs and Checkstyle.

Another important aspect of the project was testing the code. We have performed different kind of tests to ensure good functioning code, to ensure everything is functioning appropriately. For the maintenance of our code we uploaded our code to the Software Improvement Group (SIG) and received an above average rating.

During the implementation phase, we already started with testing. Because the system was not completed during this phase, only parts of the system could be tested. We performed different kind of unit tests to provide a basic code coverage. We also used continuous integration, which executed the tests ever time we pushed our code to git. Besides the unit tests we have performed integration tests to ensure the connection between the different components.

The testing of the entire system is done after the implementation phase was completed. This was partly done by checking if the final product matches the requirements that were set. It is also done by letting the target audience use the system and document the findings.

A working beta implementation of the complete system was delivered as a proof of concept and most of the initially set requirements were fulfilled.

Preface

This report is the final report which concludes the development of a Peer Matching Application for Practicals for Massive Open Online Courses (MOOCs). This project was executed for the TI3800 - Bachelor Project course at the Delft University of Technology in the Netherlands. The assignment was issued by Prof. Drs. Dr. L.J.M. Rothkrantz, also from the Delft University of Technology.

The project began at April 21st 2014 and finished at June 24th 2014. All research and development took place at the Delft University of Technology, department of Electrical Engineering, Mathematics and Computer Science. This report will provide the assignment, requirements, project methodology, system design, implementation, testing and results of the project.

We would like to thank the following persons in particular:

- Leon Rothkrantz, for supervising the project and providing support and valuable guidance throughout the process.
- Dragos Datcu, from InteliWatch (http://www.inteliwatch.eu), for supervising the project and providing support and valuable guidance throughout the process.
- Felienne Hermans, for coordinating the Bachelor Project course and attending our final presentation.
- Martha Larson, for coordinating the Bachelor Project course.

Floris Verburg
Freek van Tienen
Marijn Goedegebure
Delft, June 2014

Contents

1 Introduction **1**
- 1.1 Problem Description . . . 1
- 1.2 Target . . . 1
- 1.3 Outline . . . 1

2 Assignment and Requirements **3**
- 2.1 Assignment Summary . . . 3
- 2.2 Client and academic supervision . . . 3
- 2.3 Actors of the System . . . 4
- 2.4 Global Goals and Requirements . . . 4
- 2.5 Detailed Requirements . . . 5
  - 2.5.1 User Information . . . 5
  - 2.5.2 Usage of the System . . . 5
  - 2.5.3 Kind of System . . . 7
  - 2.5.4 Peer Matching . . . 7
- 2.6 Changed requirements . . . 7

3 Project Methodology **9**
- 3.1 Global planning . . . 9
- 3.2 Overall process strategy . . . 10
  - 3.2.1 Meetings . . . 10
  - 3.2.2 Software development method . . . 10
- 3.3 Project roles . . . 10
- 3.4 Phase 1 . . . 11
  - 3.4.1 Goal . . . 11
  - 3.4.2 Planning . . . 11
  - 3.4.3 Phase reflection . . . 11
- 3.5 Phase 2 . . . 12
  - 3.5.1 Goal . . . 12
  - 3.5.2 Planning . . . 12
  - 3.5.3 Phase reflection . . . 12
- 3.6 Phase 3 . . . 12
  - 3.6.1 Goal . . . 12
  - 3.6.2 Planning . . . 13
  - 3.6.3 Phase reflection . . . 13
- 3.7 Tools . . . 13

4 System Design and Implementation **15**
- 4.1 Global System Design . . . 15
  - 4.1.1 Database . . . 15
  - 4.1.2 Class Diagram . . . 16
  - 4.1.3 State Diagrams . . . 17
  - 4.1.4 Sequence Diagrams . . . 18
- 4.2 Graphical User Interface Mockups . . . 19
  - 4.2.1 Profile . . . 19
  - 4.2.2 Practical . . . 20
- 4.3 Model View Controller . . . 21
  - 4.3.1 Controller . . . 21
  - 4.3.2 Model . . . 21
  - 4.3.3 View . . . 22

4.4 Implementation . . . 22
4.4.1 Global . . . 23
4.4.2 Invites . . . 23
4.4.3 Recommendation algorithm . . . 24

5 Testing **27**
5.1 Testing methods during the implementation phase . . . 27
5.1.1 Unit testing . . . 27
5.1.2 Integration testing . . . 29
5.2 Testing methods after the implementation phase . . . 30
5.2.1 User test . . . 31
5.2.2 Requirements test . . . 37
5.3 Software Improvement Group Feedback . . . 39
5.3.1 Intermediate feedback . . . 39
5.3.2 Improvements . . . 39
5.3.3 Final feedback . . . 40

6 Conclusion **41**
6.0.4 Recommendations for future work . . . 43

A Appendix A - Plan of Approach **45**

B Appendix B - Assignment **53**

C Appendix C - Research Report **55**

D Appendix D - User test Google form **67**

E Appendix E - User test data **73**

F Appendix F - SIG feedback (Dutch) **79**

Bibliography **81**

1

Introduction

1.1. Problem Description

It is well known that many students come to the University to visit lectures because it provides the opportunity to meet peers and to have discussions also about less academic topics. The meeting place at the University is partly replaced by a virtual meeting place. Students report about their daily activities, opinions, problems using social media.

At this moment social media plays a limited role in the academic process of teaching and learning. But social media offers great opportunities, especially around the development of MOOCs (Massive Open Online Courses).

The basic idea of MOOCs is that students, remote in place and time, follow the lectures. They can use social media as academic communication media, where students discuss about their study activities, put forward their questions and get help and support from fellow students.

The question is how to find the best matching fellow students, especially students outside TUDelft. Inspecting the profiles from all these students via Facebook is too time consuming, therefore are supporting matching tools needed.

At this moment there are many sites trying to find the best matching partner, our interest is finding the best matching study partners. A possible option is to extract features from student profiles via Facebook or LinkedIn, train classifiers or other matching technologies and take care of preferences and expert knowledge.

1.2. Target

In this project we focus mainly on students, but teachers are also important for accomplishing a good result. It is possible for anyone to join these courses, not only for people that are registered to a university. We need to match the students based on their skills, but the teachers are the ones that determine the demands for the assignments and what the optimal level of average skills in a group is.

In the e-learning community there is much interest in cooperative learning. Many researchers report about educational projects how students are able to communicate via social media.[1]

We focus on creating optimal practical groups for students. So it is also very important that students test the application.

1.3. Outline

We will identify the assignment based on the requirements we determined in Chapter 2. Then, in Chapter 3 the strategy and methodology of the process will be explained by going into more detail

on our approach, the project planning and tools used throughout the project. Next, in Chapter 4 the implementation details and design decisions will be explained. We will discuss why we chose for certain implementations of for example the algorithm and the system design. In this chapter, we also give an impression on how the application will look like by means of graphical user interface (GUI) mockups. In Chapter 5 we will discuss our way of testing. We differentiate the tests during and before the implementation. Finally, we conclude the report in Chapter 6 by evaluating the results and providing suggestions for future work.

After the conclusion a Bibliography and several Appendices can be found. These appendices contain intermediate products and more detailed descriptions that are referred to in the text in order to improve the readability of the main report.

2

Assignment and Requirements

2.1. Assignment Summary

The whole original assignment can be found in Appendix B. In this section we just provide a short summary of the assignment.

Massive open online courses (MOOCs) all around the world are growing in size. Therefore, the practical assignments need to be easy to manage. The growth also means that it is more difficult for each participant to find a possible practical partner due to the large amount of people. The difficulty to find the correct partner or group members when the students do not see each other during courses and have not even met most of the people from the course is also affected.

Most of the time participants of MOOCs are searching group members on social media like Facebook or Twitter. The participants try to look through some of the names in the course, search for them on-line through social media and check if they are a capable partner or group member. This approach takes a lot of time and most of the time there are too much MOOC participants to go trough all of them.

Next to the fact that this process is very time consuming, this process also often leads to a sub optimal solution. This is due to the problem that people often look for partners of group members which share the same interests. While this often makes it easy to work together, this will not guarantee that a good functioning group with enough knowledge is created.

We would like to improve upon this process by creating an application that can be used to create practical groups without the participants being required to search intensively through all the participants. Participants should be able to easily access the application, fill in some relevant information about themselves and then get recommendations of possible practical partners or group members.

The interaction with the system has some difficulties. If we want to get some information from third parties, we are required to use the information in the way third parties deliver the data There will be difficulties processing this data and filtering out the relevant data.

The recommendation of possible group members can be quite difficult too. One of the problems we are facing is the question: What user information is useful to determine whether two people can be good practical partners?

2.2. Client and academic supervision

During the project, we had both a TU Delft coach and a company contact from InteliWatch. During weekly meetings, we discussed our progress and the preferences from the client. In Table 2.1 the roles of our supervisors can be seen.

Table 2.1: *Our supervisors*

Leon Rothkrantz	Group Mentor and Supervisor	Supervises the project from the academic point of view and aides the group with difficulties.
Dragos Datcu	Client, Group Mentor and Supervisor	Supervises the project from a client point of view and aides the group with difficulties using his experiences of working for InteliWatch.

2.3. Actors of the System

In the problem description we defined the three actors of the system. Each of these actors can perform different actions and therefore have different requirements. By defining the requirements, as we do in the next section, we take these different actors in account.

Actor 1: MOOC participant

Due to the nature of the MOOCs, there are no requirements set for the participants. Every person with an interest in the given course can enlist for the MOOC and should thus be able to register with or understand our system.

Actor 2: MOOC teacher/MOOC administrator

The MOOC is organised by a professor of a university or a company. This professor will need to register with the system and create a MOOC.

Actor 3: Groups of participants

This actor consist of two or more MOOC participants who accepted each other to their group. These groups can be of variable size, with a maximum, set by the MOOC organiser.

2.4. Global Goals and Requirements

To determine the direction the project should follow, we define goals for the project. These goals will later be used to determine whether the final product completed these goals.

- The final product, that will be presented, will be a working proof of concept which will fit closely with the specified target audience and demands of the assignment
- We want to create a platform for searching other students and letting the students invite other students for a practical group
- We want the final product to recommend good fitting possible group members to the students. By ”good fitting” we mean that there is some heuristic that we use to calculate whether two students would make a good practical group. We want the final product to achieve more optimal solutions than the current situation
- We want the maintainability of the final product to very high and want the code to be easily extendable

The goals do not include a goal for the teachers part, because we mainly want to focus on the student. This is because of the limited time of the project and the current goals having enough depth for an extensive system. Next to this, the assignment specifically focusses on the need of the students to have a platform to search, interact with other students to form practical groups and have help with searching for their best practical group. Although it would be beneficial for the teacher to have an easy to use platform for managing his practical assignments, it is out of the scope of this project.

To provide an overview of what actions the software must be capable of, we define three global requirements for the system. By defining these three global requirements we provide a foundation for the detailed requirements.

- MOOC teachers must be able to create, manage and edit courses.
- MOOC students must be able to register and deregister for a course.
- The system must be capable to use personal data provided by the user to suggest an appropriate practical partner or group member. It must be possible to adjust the definition (parameters) of the appropriate practical partner or group member, and search trough several suggestions.

2.5. Detailed Requirements

This section will mention the different requirements that involve the basic system which the actors interact with. The section starts with mentioning the possible types of user information the application can use and what requirements are associated with that. Followed up by what actions are possible within the system. After that, the kind of system is limited by the requirements listed. Lastly we will list the requirements set for the algorithm.

We use the MoSCoW model to divide the requirements in their appropriate categories. The M stands for must have, the S for should have, the C for could have and the W for would have.[2]

2.5.1. User Information

#	Requirement	MoSCoW
1	Users must have a viewable profile that can be filled with personal information.	M
2	Users must be able to enter the following registration information during registration: Full name, Email address, Password.	M
3	Users must fill in their academic performances. Which knowledge, skills and completed courses does the user have?	M
4	Users should be able to enter basic user information, for example: country, age, short description, skills, education and a picture.	S
5	Users could be able to use a personality test to determine their personality traits.	C
6	Users could be able to fill in their personality traits.	C
7	Users could be able to enter the following extended user information: preferred group role, role of the other group members, etcetera.	C
8	Users could be able to enter a predefined questionnaire.	C

2.5.2. Usage of the System

#	Requirement	MoSCoW
Actor 1: MOOC participant		
1	The potential participant (anyone who visits the site) must be able to view the different MOOCs without registering or logging in.	M
2	The participant must be able to inform him/herself about the usage of the application without registering or logging in.	M
3	The participant must be able to register to the application.	M
4	The participant must be able to log in to the application.	M
5	The participant must be able to register to the MOOCs which they were enlisted for (known to the administrator).	M

6	The participant must be able to view a possible practical partner.	M
7	The participant must be able to communicate with the possible practical partner to verify the matching.	M
8	The participant must be able to accept or decline the practical partner into his practical group.	M
9	The participant must be able to communicate within the group.	M
10	The participant should be able to ask a currently existing group if he can join them.	S
11	The participant should be able to use their LinkedIn profile to log in to the application.	S
12	In case the participant already has an account, the participant should be able to add the course to his profile using the teachers url.	S
13	The participant should be able to receive an invitation from a group.	S
14	The participant should be able to accept the invitation from a group.	S
15	The participant could be able to set his preferences for his possible partner.	C
16	The participant could be able to search for other participants using information that is available in the profile.	C

Actor 2: MOOC teacher/MOOC administrator		
17	The administrator must be able to create a course and add this to his account.	M
18	The administrator must be able to create an URL that he can use to invite the participants.	M
19	The administrator should be able to manage the list of participants and groups of participants.	S
20	The administrator should be able to open and close the participant registration for a course.	S
21	The administrator should be able to set a deadline for a practical.	S
22	The administrator should be able to communicate with every participant.	S
23	The administrator should be able to communicate with the groups.	S
24	The administrator could be able to grade the group for its practical.	C
25	The administrator could be able to provide for his own criteria for each of his course.	C
26	The administrator could be able to add different assignments to the application.	C
27	The administrator could be able to view what groups have finished what assignments.	C
28	The administrator could be able to grade the different assignments.	C
29	The administrator could be able to provide feedback for the different assignments.	C

Actor 3: Groups of participants		
30	Course students group members must be able to log in to the system	M
31	A group should be able to view a possible group member.	S
32	A group should be able to invite a possible group member to join their group.	S
33	A group should be able to accept a request from a participant to join their group.	S
34	The groups must be able to communicate with each other using the application.	C
35	The group could be able to select their preferences for the new group member.	C
36	The participants could be able to share files to other group members using this application.	C
37	The participants would be able upload their source code to the application for verification.	W

2.5.3. Kind of System

#	Requirement	MoSCoW
1	The application must be easily accessible to both participants and course organisers.	M
2	The system must provide account management functionality.	M
3	The system must use a database to store the different data put in by the user.	M
4	The system could be able to be combined with practical systems already used by MOOC organisers.	C

2.5.4. Peer Matching

#	Requirement	MoSCoW
1	The algorithm must be able to use a list of users as input.	M
2	The algorithm must be able to deal with the continuous variation in amount of users.	M
3	The algorithm must be able to return one or more suggestions of practical partners to the user given a list of rejected users.	M
4	The algorithm must be able to return one or more suggestions of possible groups the participant could join.	M
5	The algorithm uses basic user provided data (e.g. education, preferences) to build up a profile for a course participant.	S
6	The algorithm can calculate whether two people will be good practical partners using the basic user provided data.	S
7	The algorithm can return a possible practical partner using the basic user provided data.	S
8	The algorithm uses advanced user provided data (e.g. asked questions, programming tests) to build up an extended profile for a course participant.	C
9	The algorithm can calculate whether two people will be good practical partners using the advanced user provided data.	C
10	The algorithm can return a possible practical partner using the basic user provided data.	C
11	The algorithm could be able to respond to the preferences of groups.	C
12	The algorithm could be able to respond to the preferences of a participant.	C
13	The algorithm would have to be able to use information gathered from rejections to improve the matching.	W

2.6. Changed requirements

During the project we noticed that some requirements were not classified properly. These requirements do not have the priority that we initially thought they would have. Following the goals that we have set for the final product, we reasoned that these requirements were not must haves for the final product. The teacher part of the system is something we would want to look into, the moment that the final product has a focus towards actual use and not at the proof of concept phase. Below we listed the requirements that were reclassified. We have listed both the old and new classification.

#	Requirement	Old MoSCoW classification	New MoSCoW classification
19	The administrator should be able to manage the list of participants and groups of participants.	M	S
20	The administrator should be able to open and close the participant registration for a course.	M	S
33	The groups must be able to communicate with each other using the application.	M	C

3

Project Methodology

In this chapter we will describe the project methodology that we have applied to this project. We will first describe the global planning for the project. Here we will divide the project into three phases. After this we will describe an overall process strategy and what strategy we will apply to which phase and why. Thirdly, we will describe which team member has which project roles and what these project roles imply. After this we will comment on each of the three phases in more detail about the individual planning and deliverables. Here we will also reflect on the phase and talk about what went wrong and what went right. Lastly, we define the tools that we have used during the project.

3.1. Global planning

The global planning we have setup can be seen in Figure 3.1 . It mentions all the big deliverables and gives the team a good overview of how far the progress of the project should be. The team can compare the current state of the project with this planning and can conclude if there is sufficient progress. We can then also conclude if we have to take measures to prevent mistakes in the next phase.

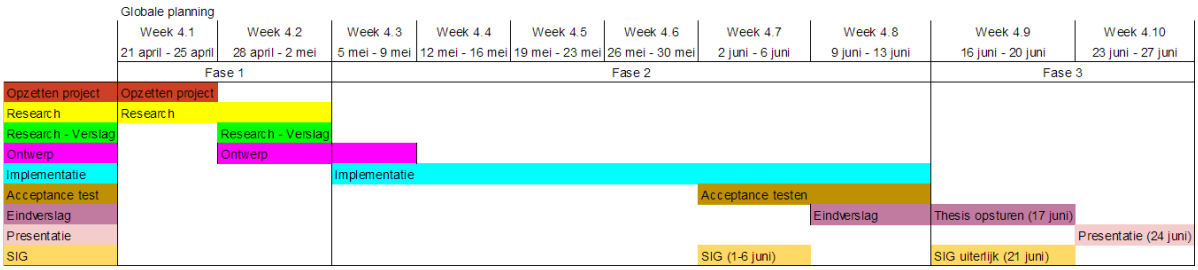


Figure 3.1: *Global planning of the project*

As showed in the image, we divided the project into three phases:

- Research and set-up (phase 1)
- Design and implementation (phase 2)
- Testing, documentation and presentation (phase 3)

Each of these phases describes the things that need to be done during that phase. Not all tasks are completely in one phase, some of these tasks overlap between two phases. To know how far we have progressed, we specified a week where the task should start and when it should end. In the sections about the phases we will go into further detail about the deliverables and planning of each phase.

3.2. Overall process strategy

We started the project off without a strict task distribution. We choose to do so because many of the tasks at hand needed group thinking and group opinions. For example, the interpretation of the problem description is something that can be thought out by a single person, but needs to be communicated and discussed with the other team members. If not communicated enough, the resulting problem description may not be the same as how the team interpreted the client's description.
The tasks that needed to be done were documented in a spreadsheet. In this way, every team member had insight into the things that still needed to be done. Each team member could claim a task for himself when he finished his last task. That way a quick look into the spreadsheet would give a good overview of which tasks were important within a short time notice.

3.2.1. Meetings

Although the team spends most of the week together working on the project, it is still necessary to plan weekly meetings. During these meetings we discussed the current progress of the project and the possible problems that we could encounter. The meetings are planned at the beginning of each week, so that we could discuss what has been done last week and what needs to be done the upcoming week. It also helped identifying the things that need to be done. These things were added to the spreadsheet after the meeting.

3.2.2. Software development method

During the implementation phase we used a known software development framework. This development framework is known as scrum. Scrum is based on the agile software development framework which can be used to manage software projects. Because of the limited implementation time, we couldn't research and design all aspects of the system upfront. We need a software development framework that gave us the possibility to react to changes in requirements, but also reacts to new insights given by our increasing understanding of the system as time progresses. Scrum has a flexible approach to software development and satisfies the requirements that we have mentioned above. In scrum, only a basic design of the system is needed before starting implementing.

During implementation daily feedback is provided on the progress and the team can change the way things are programmed or have been programmed to steer the product in the proper direction. During the research into scrum we ran into the problem that scrum required us to divide roles between team members (e.g. the product owner, development team, scrum master). We choose to also divide the bigger remaining responsibilities among the team members. In this way we could ensure that each team member had his own responsibilities and that no aspect of the project would fall behind. In the next section we define the different roles and divide them among the team members.

3.3. Project roles

As said above, in the beginning there was no strict task distribution. Following the advise of our group mentor, every week a different team member was the designated chairman and another team member was the designated secretary. The chairman of the week had the responsibility to preside the weekly meetings and to maintain an overview of the tasks that were at hand. The secretary had the responsibility of making minutes of both the weekly meetings and the meetings with the group mentors and client. These two roles stayed the same throughout the project.

As said above, we realised that we needed more defined roles. We defined the following roles:

- Lead Development, is responsible for the implementation phase
- Scrum master, is responsible for the scrum process
- Quality Assurance, is responsible for the quality of code and test coverage
- Project Manager, is responsible for maintaining an overview of the project and making sure the product is finished on time

- Lead Documentation, is responsible for the documentation of the various aspects of the project

The following table lists the different group members and their roles.

Name	Roles
Freek van Tienen	Lead Development
Floris Verburg	Scrum master and Quality Assurance
Marijn Goedegebure	Project Manager and Reports

3.4. Phase 1

3.4.1. Goal

The first phase aims to expand the knowledge of the group about the problem and it's requirements. Secondly, it also gives time for the team members to analyse the problem for difficult subjects and to spend time finding a fitting solution. Thirdly it gives the team members the time to set up the project infrastructure and to define the project methodology. Lastly, the remaining time should be spend making a first start with the system design.

3.4.2. Planning

We planned phase 1 to be two weeks long. With the first week having a focus on setting up the project and doing research to the problem and requirements. The second week was more about defining the project methodology, analysing and researching the possible problems that we could encounter and making a first system design. The second week was also used to finish up the research report.

Planned deliverables

At the end of phase 1 we wanted to have the following deliverables done:

- Research report
- Plan of approach
- First system design

3.4.3. Phase reflection

During the first phase we had some trouble with determining the direction that the project would take. Although the Computer Science bachelor has a lot of projects, this is the first project where we are completely free and where we need to make our own decisions. This gave us some problems at the start of the project. After reading up on some projects that were done last few years, we had a better insight into what was expected from us. During the first week we made good progress with defining the requirements, but later realised that we probably should have started with the plan of approach. We made sure to catch up in this regard.

We also spend some time during the first week in setting up the programming environment. Because of this we suffered some delay with the research report. This wasn't a big problem, because otherwise we would have done it right after the research report and both the research report and the plan of approach didn't have a completion date. This might even have worked in our favour by giving us time to have another meeting with our supervisors and discussing the direction and progress of the project. We used the third week of the project to catch up on the research report.

The other delivery was the first system design, this is not a properly defined deliverable but was more an indication that we should have started on this. We did so by choosing to develop a web application.

Despite the rough start, in retrospect we are happy with the result of the first two weeks. It gave us a good start for the implementation phase and we didn't have many set backs.

3.5. Phase 2

3.5.1. Goal

The purpose of the second phase was to provide a time frame in which both a basic system design can be made and the system design will be implemented. During the second phase the roles were more strict since we wanted to follow a structured software development method. This software development method is the framework known as scrum.

3.5.2. Planning

We planned the second phase to be six weeks long. The first week was used to finish up the system design. The other 5 weeks were used to implement this system design. During the final two weeks we looked into the user tests and the acceptance tests. During this time there we also sent our source code to the Software Improvement Group for them to check for the code quality. The feedback that we got back will be discussed later in this paper.

Planned deliverables

At the end of phase two we wanted to have the following deliverables finished:

- Basic system design
- User tests
- Send the source code to SIG for the first review
- A working application

3.5.3. Phase reflection

During this phase the main task of the team was implementing the system design. The first task of the team was to complete the system design. This needed a couple team meetings discussing the database design and the interface interaction. After completing this, the implementation could start. Not many problems were encountered during the implementation. During the implementation we didn't always create a pull request per feature. Some features were just to big to do this. This was caused by not having a lot of experience with big projects and the possibility that features can be so big. We resolved this by splitting up the pull request in multiple parts.

We also thought the reviewing of the pull requests to be very useful. This helps to have all the team members be up to date with the newest features added to the system. It also helped checking the code for human errors.

During this phase we used our continuous integration solution to analyse our code and to run automated tests on the system. We thought the static analysis to be very useful, it increases the readability and maintainability of the code a lot. The automated tests also helped, by decreasing the checking needed by the team members.

We are very happy how the implementation phase went. The amount of features that we implemented did not stall anywhere during the process. We wonder if we made the right choice by spending a moderate amount of time improving our interface. It belongs to the making of a complete product, but it also means that we did not have the chance to implement more features.

3.6. Phase 3

3.6.1. Goal

The goal of the third and last phase is to provide for time to finish up the project. During this phase time has been reserved for finishing up the implementation, the final report and for the user tests.

3.6.2. Planning

During this phase there is time for completion of the implementation, finalisation of the final report and creation of the presentation. The completion of the implementation and the finalisation of the final report will been done simultaneously. The creation of the presentation will be done after the final report has been submitted. After the final report is submitted we will also be looking into the feedback that we received from the SIG. Three days before the presentation, the source code should be send to SIG for a final review.

Planned deliverables

At the end of phase three we wanted to have the following deliverables done:

- Final report
- Presentation
- Improved implementation following the comments provided by SIG
- Final review of SIG

3.6.3. Phase reflection

During the final phase the goal was to finish up the project. The main tasks were the user tests, integration tests and the final report. The planning for the last phase was strict, we did not have much manoeuvrability or time to delay. First up was the final report, this report. We needed to work through the weekend to finish it up in time, but it was mostly to make sure that we weren't stressing too much. That also gave us the possibility to send our report to our supervisors for a review before the final draft.

At the end of the second phase we sent our source code to SIG for checking. Later in this paper we will discuss the comments that SIG made. During the final presentation we will comment on the final review that SIG has made.

While the final report will soon be done, the final presentation and final review will not be complete until after the report has been completed. We have a week from the deadline of the final report until our presentation. In this time the following things need to be done:

- Create the presentation
- Improve implementation using the feedback of SIG
- Prepare the live demo for the presentation

We think that the remaining week until the presentation will be enough time for these last deliverables.

3.7. Tools

We have used several tools to support our project. The tools helped in administrative work, but also in improving code quality and maintainability. In the following list we mention each tool that we used and the benefit that it has.

- Play framework, The play framework gives us the possibility to easily create a web application using Java.
- IntelliJ IDEA, is our Integrated Development Environment (IDE) which will facilitate the development and testing of our code. IntelliJ will be integrated with the Play framework which will be helpful during development.
- GitHub, is our on-line code and documentation repository. It provides for a version control system, an issue tracker and code review possibilities. We will be using it to store all our code and documentation's code (LaTeX).

- Cloudbees will be used to run our test environment, our continuous integration and the release environment. It uses Jenkins for the continuous integration.
- Findbugs, is a plug in for Cloudbees and IntelliJ that gives us the possibility to let our java code be checked for small bugs using static analysis.
- JaCoCo, is a plug in for Cloudbees and IntelliJ that provides us with data analysis about our code coverage.
- Checkstyle, is a plug in for Cloudbees and IntelliJ that checks the code for coding standards. This makes it ideal to enforce the coding standard for our project.

4

System Design and Implementation

4.1. Global System Design

In this section we will explain how we designed the application and why we made several design and implementation decisions. We do this using different diagrams, for example a class diagram, a database diagram, sequence diagrams and state diagrams.

4.1.1. Database

In Figure 4.1, our database diagram can be seen. We will explain the structure of the database shortly.

In the center of the diagram, the user is shown. The user has a many to many relation with a group. There are four different groups. There is a group for students, for teachers, for administrators and one for guests. An administrator can create teachers and edit settings for the application. A teacher can create and manage courses. A student is someone who participates the course and want to find a practical group. A guest is someone who is not logged in to the application. It is obvious that a group can contain multiple users, but a user can also have multiple groups. Users can be a teacher, as well as an administrator. A group is not a practical group. The practical groups are explained below here.

Every group has different rights. The same right can also be assigned to multiple groups. That is why we also have a many to many relation between a group and a right.

A user must also be able to log in. To accomplish this, the user must have an identity. An identity can only belong to one user. If this should not be the case, the system would not be safe because other users have access to protected parts of the system.

A user can be part of multiple practical groups and a practical group consist of multiple users. Therefore user and practical group also have a many to many relation. This is the same for user and practical.

A practical group belongs to only one practical. A practical can have multiple practical groups. Therefore is this a many to one relation.

An invite has two many to one relations with a user. An invite has a sender as well as an receiver. There can only be one sender and one receiver for an invite. But a user can send and receive multiple invites, therefore this is also a many to one relation.

Every invite belongs to only one practical, these therefore also have a many to one relation.

The last relation is between an invite and a message. These messages are for communication between the sender and the receiver of the invite. An invite can have multiple messages and every message belongs to only one invite.

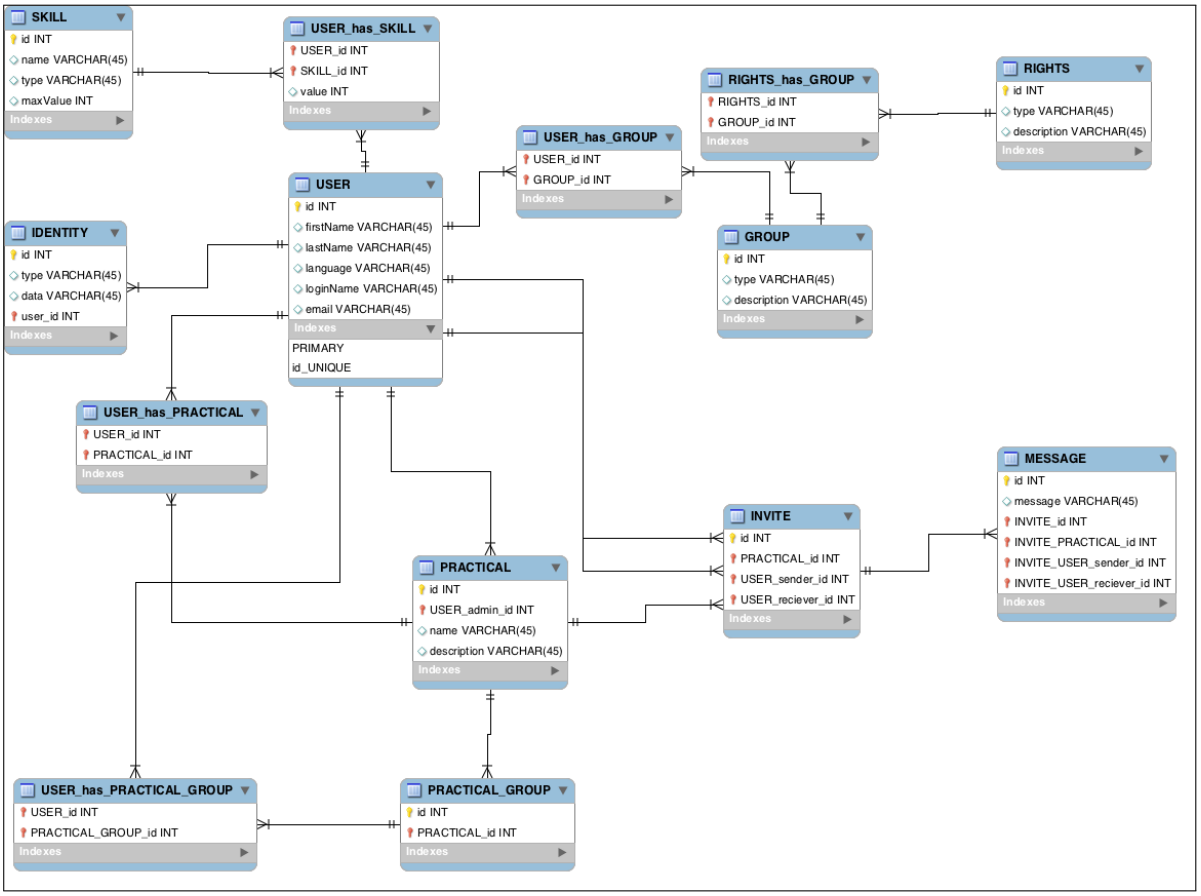


Figure 4.1: *The database diagram*

4.1.2. Class Diagram

In Figure 4.2 our application's class diagram can be seen. This diagram is not complete with all the methods and attributes, but the most important methods and attributes are included. For example, all the get and set methods are not included.

The structure of this diagram looks a lot like the database diagram. In this diagram, we make a difference between a student and a user, because they can perform different actions.

For example, only a teacher can create courses. A student cannot do this, a student can only sign up for this courses.

A student can also find possible group members. These possible group members are determined by the recommender class. With the getNextGroupMember method the next best practical partner will be calculated. More about the matching algorithm can be read in Section 4.4.3

The other methods are needless to explain. The name of the methods tell the function.

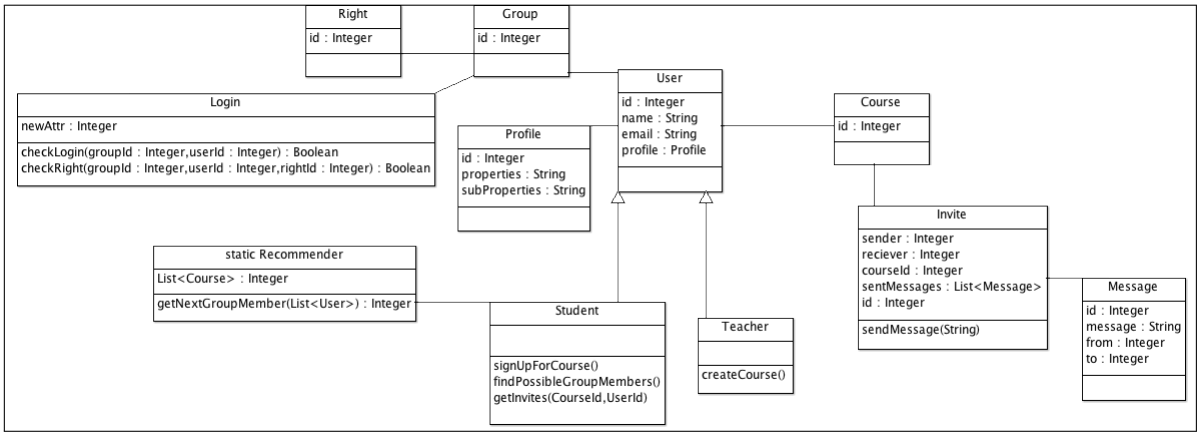


Figure 4.2: *The class diagram*

4.1.3. State Diagrams

In this section we describe the behaviour of our system with the help of state diagrams. We describe this behaviour with two different state diagrams. One diagram for the administrator and another one for a student.

First of all, the administrator must be registered and logged in before he can perform any action. When the administrator is logged in, he can create practicals. When the practical is created, the administrator can edit and manage this practical.

One of the parts of managing the practical is managing the users that are enrolled for this practical. The admin can remove users or even entire practical groups.

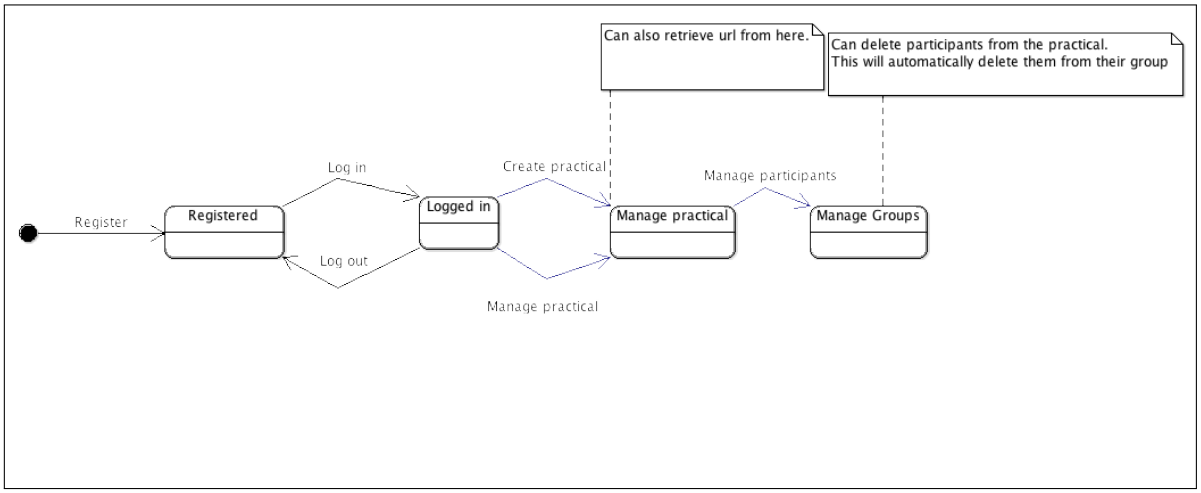


Figure 4.3: *The administrator's state diagram*

The student's state diagram is more complex. The first part is the same as with the administrator. The student also must be registered and logged in to perform any action.

When the student is logged in, he has two options. He can either go to his profile, or manage his practical. When he chooses to go to his profile, he can view or edit it.

The courses have some more options. First of all, the student can see which fellow students are recommended to join his group. The student can invite these students to join his practical group.

The student can also view the overview of his invites. This overview includes both received and sent invites. When the student accepts an invite, both the practical groups are joined.

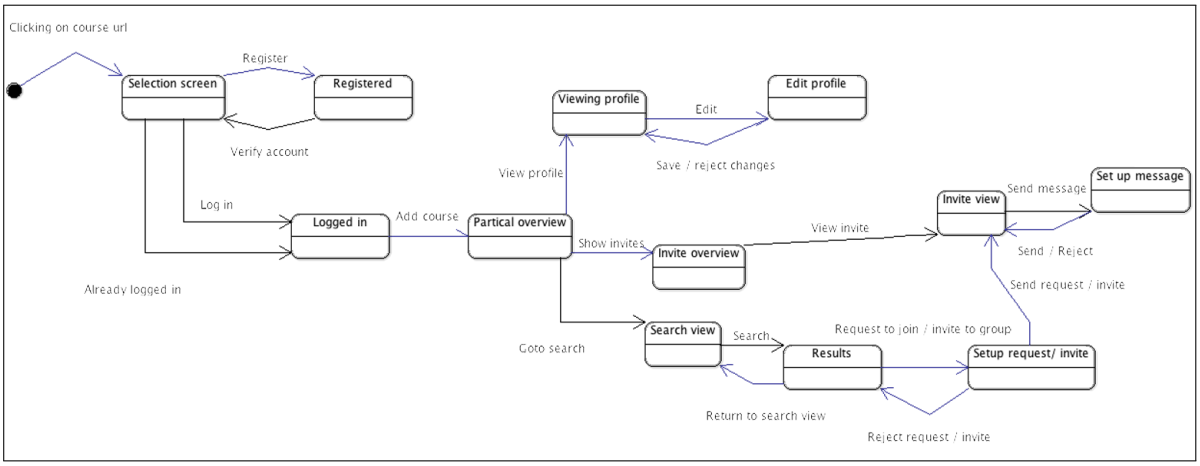


Figure 4.4: *The participant's state diagram*

4.1.4. Sequence Diagrams

In this section we explain the sequence diagrams. We have only made a sequence diagram for logging in, but because of the Model View Controller structure (Section 4.3) most of the actions have a similar sequence diagram.

The user only has direct contact with the views. The views send the information that the user provided to the controllers. After that, the controllers collect the required information from the database via the models.

When the controllers have all the required information, they execute their calculations. After the calculations are fully executed, the results will be sent back to the views. The views display these results and via these views the results will reach the users.

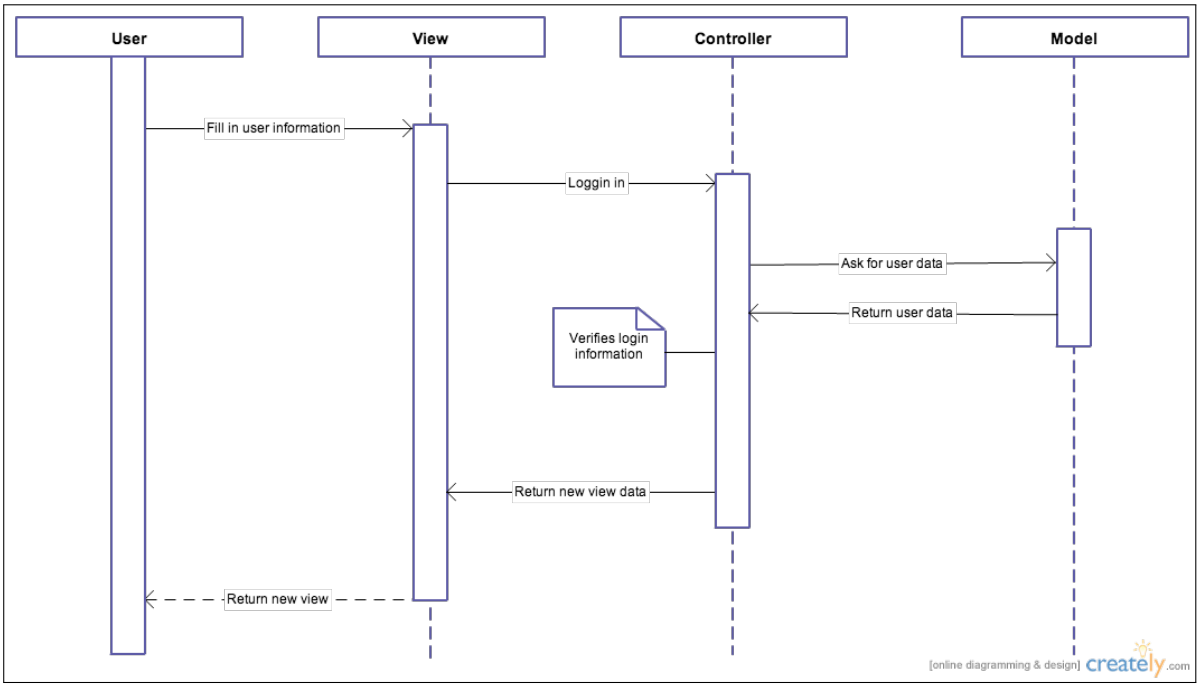


Figure 4.5: *The sequence diagram for logging in*

4.2. Graphical User Interface Mockups

We have multiple important features in our application. Each of these features has an own graphical user interface (GUI). Below, we will provide mockups from the graphical user interfaces of the most important features.

We have made mockups for the profile page and for the practical page. This are the most important features, so it is important for these features to have a clear interface.

4.2.1. Profile

As can be seen in Figure 4.6, a user can view someone's (or it's own) profile. The user can see the basic profile information which the user has filled in himself.

If a user is looking for a practical partner, he can get a better impression of that person when he gets an overview of the profile information of that person. This is just a simple page, because the user should quickly get an impression of the student.

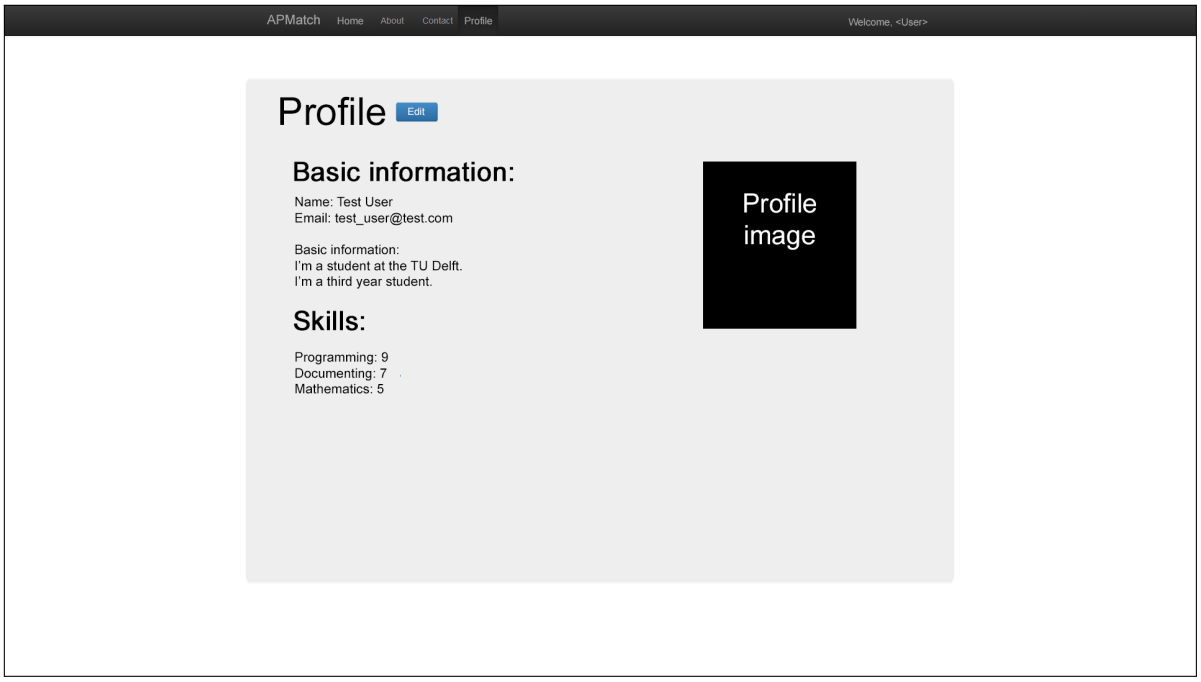


Figure 4.6: *Interface for the profile page*

In Figure 4.7, the page for editing the profile can be seen. On this page, the user can edit his basic profile information as well as the level of his skills. He can also change his password and his profile image. This information will be taken into account while determining the recommended practical partners. So it is important for the users to be able to fill in this information easily.

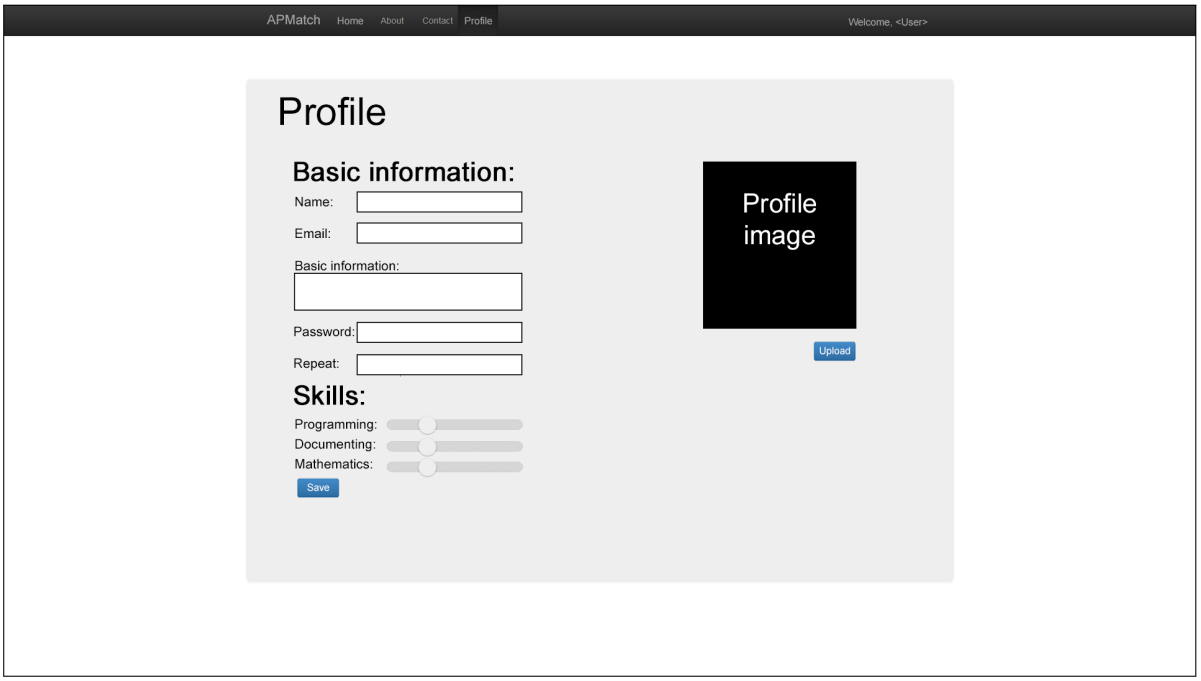


Figure 4.7: *Interface for editing the profile*

4.2.2. Practical

The main feature in our application is recommending practical partners. Therefore the users need to be able to register for a practical assignment. The user receives a link from the teacher and when they have clicked on it, the user is registered to the practical. The practical is then added to the user's practical overview. Such an overview can be seen in Figure 4.8

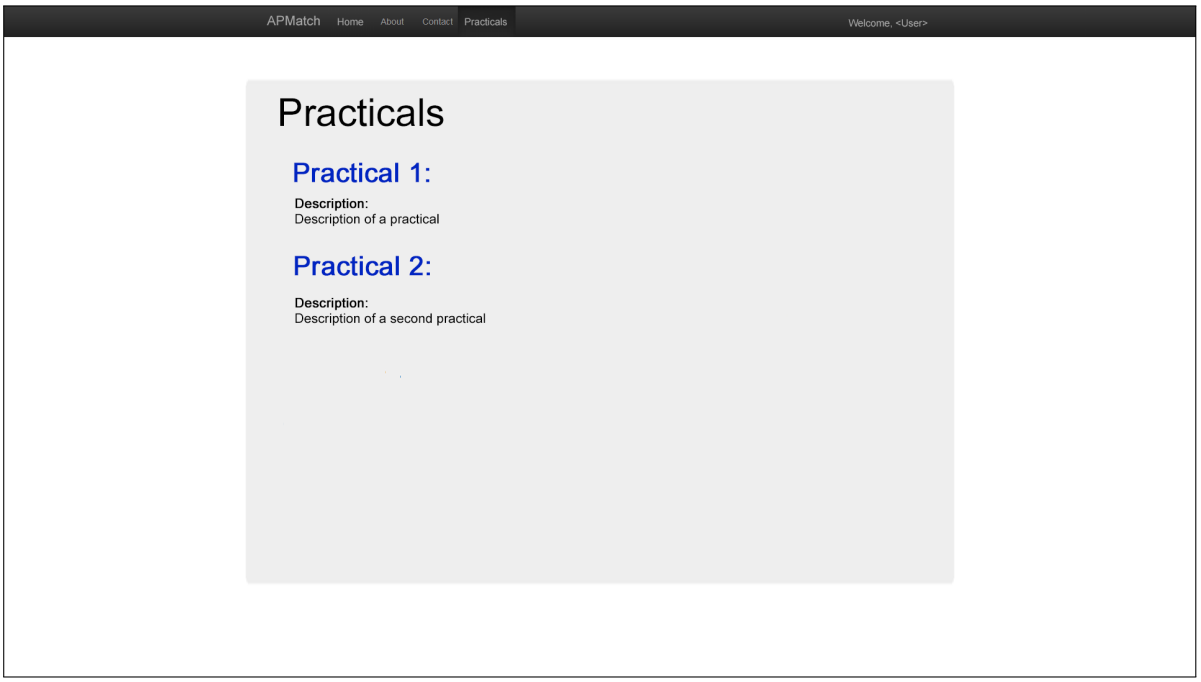


Figure 4.8: *Interface for the practical overview page*

In Figure 4.9, the user can see a page for a certain practical. First of all, there is some information

about the practical itself. We show a short description of the practical.

After that, we show the most important part, the recommendation. The application determines which other practical groups matches the user's group the best. On this page the user can click on the practical group for more information. The user can also invite a practical group to join his practical group.

After the recommendation part, we also show the user's current practical group.

4.3. Model View Controller

As mentioned earlier, we are using the Play Framework. This frameworks follows the Model View Controller (MVC) architectural pattern applied to the Web architecture[3]. This pattern splits the application into separate layers: the Presentation layer and the Model layer. The Presentation layer is further split into a View and a Controller layer.

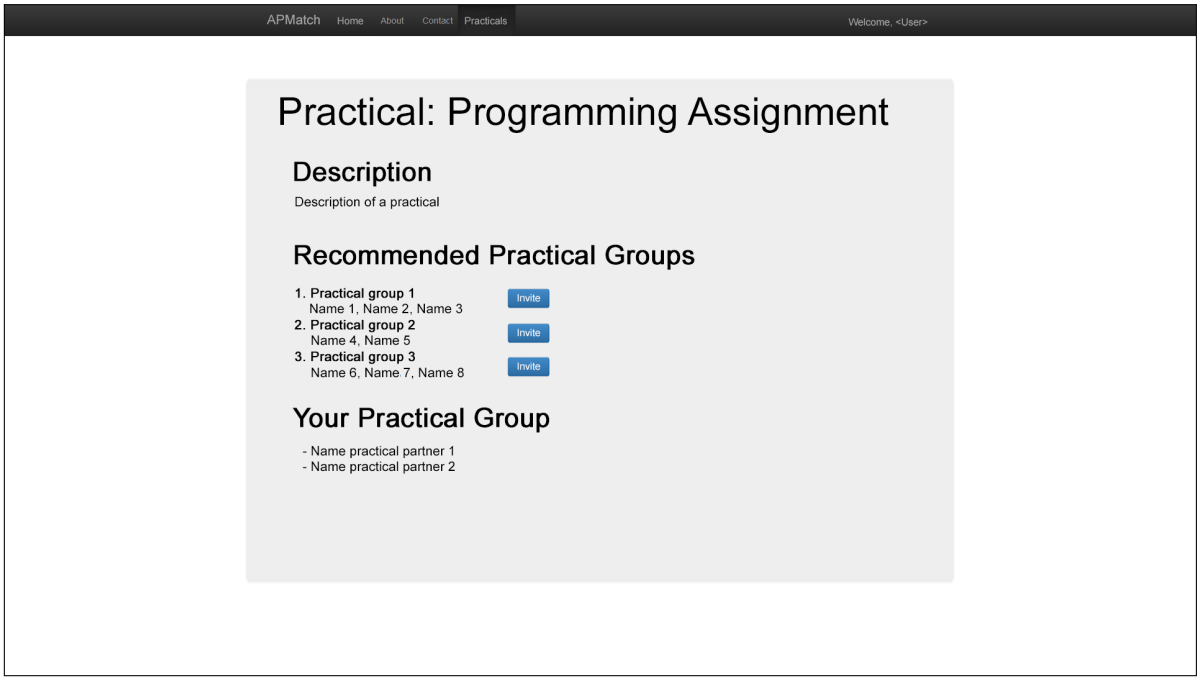


Figure 4.9: *Interface for the practical page*

4.3.1. Controller

The **Controller** responds to events (typically user actions) and processes them, and may also invoke changes on the model. In a Web application, events are typically HTTP requests: a Controller listens for HTTP requests, extracts relevant data from the 'event', such as query string parameters, request headers... And applies changes on the underlying model objects.

In Play, a Controller is a Java class where each public, static, method is an **action**. An action is a Java entry point invoked when a HTTP Request is received. The Java code from the Controller class is not really object oriented: it is mainly procedural code. The action method extracts relevant data from the HTTP request, reads or updates the model objects, and sends back a result which is wrapped into an HTTP Response.

4.3.2. Model

The **Model** is the domain-specific representation of the information on which the application operates. Domain logic adds 'meaning' to raw data (e.g., calculating if today is the user's birthday, or the totals,

taxes, and shipping charges for a shopping cart). Most applications use a persistent storage mechanism such as a database to store data. MVC does not specifically mention the data access layer because it is understood to be underneath, or encapsulated by, the Model.

In Play, the domain model object layer is a set of Java classes using all the object oriented features available from the Java language. It contains data structures and operations on which the application operates. Whenever model objects need to be saved into a persistent storage, they may contain some glue artefacts like JPA annotations or SQL statements.

4.3.3. View

The **View** renders the model into a form suitable for interactions, typically a user interface. Multiple views can exist for a single model, for different purposes. In a Web application the view is usually rendered in a 'Web format' like HTML, XML or JSON. However there are some cases where the view can be expressed in a binary form, e.g. dynamically rendered chart diagrams.

In Play, most of the application views are generated using an efficient template system provided by play. The Controller gets some interesting data from the model layer, and then applies a template to decorate these objects. This packages contains HTML, XML, JSON and other template files with special directives used to dynamically generate the model representation.

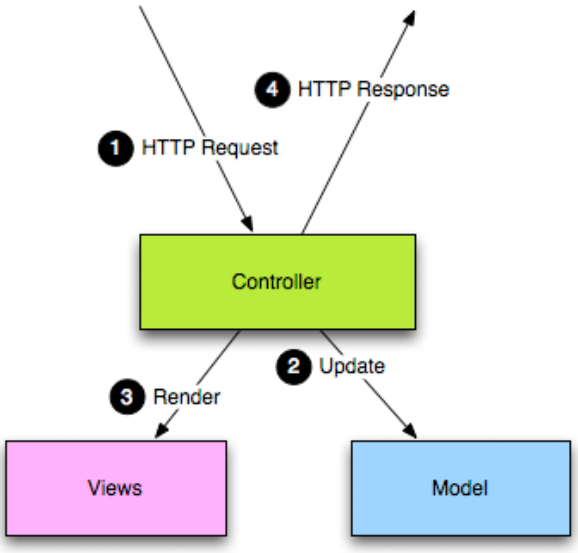


Figure 4.10: *Play Framework MVC structure*

4.4. Implementation

In this section we will describe the software implementation of APMatch. We will start of with a global implementation description of our APMatch system and then we will continue with some more advanced topics.

We chose to explain some more about both the invites and the matching algorithm. For the invites we explain what our difficulties were and which situations were hard to implement, while for the matching algorithm we will explain some more on how it works and why we chose it.

4.4.1. Global

During the implementation we chose for the git feature workflow as described in section 3.2.2. Most of the time this leads us to small and nice feature's which added both the model as well as the visuals to the APMatch system. There were some exceptions like the Invites and Matching algorithm which were a lot harder to implement than expected. These big feature's sometimes broke our workflow, because there were too many dependencies between the different features. Luckily most of the time there were enough other feature's that could be implemented to solve the issue of waiting on the dependencies to be implemented.

Our global implementation went as planned and made use of the MVC design method of the Play Framework. This lead to a nice structured way of models that related closely to the database model, containing most of the functionality of the APMatch system. We also extended the MVC model a bit with a helper package which contains functionalities like hashing, Linkedin and the matching algorithm. The helper classes don't relate to the database model, but will supply the needed functionalities for the controllers.

The controllers of the of the MVC model were defined in six different controllers: Application, Authentication, Invite, Practical, PracticalGroup and Profile. The Application controller only provides the static pages like the home page, contact page and the information page. The Authentication controller provides the functionalities for both the Linkedin login, and the normal authentication and registration. All the other controllers provide the functionality for all the pages which relate to the subjects of their name.

4.4.2. Invites

One of the most underestimated parts of the implementation of APMatch was the invites system. This part took by far the most time during the implementation phase, but was at first estimated as a part which could be quickly implemented. This underestimation was mostly caused by the complexity of the invites system, which we will explain in the following paragraphs. After that we will explain how we solved this difficult invite system, and conclude with what could be improved to make it even better.

So why is this invite system so complex? This all is mainly caused by the problem that occurs when creating practical groups larger than two. Creating a group larger than two requires either a group manager to create and maintain invites, or a different way of inviting people when the user is joined in a practical group. This is needed because of the fact that having the possibility for all the group members to invite other people could lead to several conflicting situations. So the easiest way to solve this is by creating a group manager, which would be the only user that has the rights to send and invite other users.

But then how do we choose a group manager? We could of course easily solve this by doing it random, but that could lead to strange situations and then group manager functionalities also need to be added, to for example change the group manager. To avoid these situations we then chose to instead of doing it random, make the inviter that forms the group the group manager. This will make it sure that when the user accepts a certain invite he also knows who will be the group manager and also can depend his choice on that.

Despite the fact that we now solved both the complex system of having more then two group members, there were still other problems that also caused our underestimation. This was due to the fact that the user could have multiple open invites available at a certain time, both send invites and received invites. Here we could have chosen to deny the user to have multiple outstanding invites to users, but because we expect that there will be a lot of inactive users this seems very inconvenient. Users will then have to wait a certain amount of time for an inactive user, while other possible groups members might join other groups. This is why we chose to implement it that when an invite is accepted and the user is not becoming a group manager, all the other invites will be rejected. Hereby the problem of having multiple open invites at a certain time is solved.

One of the last problems we found during the implementation of the invites system was the fact that the user should be able to invite a group. This caused two different subproblems, one person inviting

a group and a group inviting another group. Both were dealt with in the same way, by defining that a single person is also a group with just one person in it. This only left us with the merging of two groups which was done the same way as joining one by one.

Because of all these all these edge cases we needed to test the invite system very carefully using a lot of unit tests. Finding these edge cases, defining the edge cases and then writing the unit tests for these edge took more time than expected. Initially we didn't taught of all these edge cases and were just thinking of a simple invite system.

For the future we advise to first think out the whole invite system and split it up in smaller features. Then both the progress will be easier to track and writing tests cases will also be easier. We also advise to make it more understandable for the user what exactly is happening during the invites. This means that the users needs to know what happens to his other invites when he for example accepts an invite. Also hiding the fact that the user is in a practical group on its own should make it more understandable for the user what is happening with the invites.

4.4.3. Recommendation algorithm

For our recommendation algorithm we used our custom designed algorithm, which is described in the Research report in Appendix C. This algorithm uses the skills defined by the user and the teacher to find the best matching with the preferences of the teacher. We will first give a description of the algorithm followed by a formal description. After that we will give a description about the complexity and will conclude this section with why we chose this implementation.

Before we start explaining the algorithm we will first describe how to calculate the distance for a certain practical group. Calculating the distance of a practical group is done by going through the skills defined by the practical teacher. We will then calculate the average of the values of the skills of all the users in the practical group and the users' practical group combined. Then all the averages are multiplied with each other and then the root of the amount of skills in the practical is taken. This resulting number will be defined as the distance.

The matching algorithm starts with a practical and user. It will start with going through the different practical groups in the practical. To simplify the algorithm we chose to make sure every user is in a practical group. This means that when a user doesn't have any practical partners he is still in a practical group with only himself in it. While going trough the different practical groups it will calculate the distance as described in the previous paragraph. After calculating the distance with every other practical group and saving this in a map, this result is sorted by descending distance. Then we start recommending practical groups to the user starting by the top of the list. The formal definition of this algorithm is visible in Figure 4.11.

To Analyse the complexity we need the following measurements. We define n as the amount of practical groups in a practical. We define m as the amount of skills defined in a practical. We define j as the amount of users in a certain practical group plus the amount of users in the user's own practical group. Since in the first loop we go through the practical groups to calculate the distance we start with a time complexity of $O(n)$. Inside this loop we go through each of the skills defined in the practical to calculate the average which will give us $O(n * m)$. In order to calculate this average we need to go through all the users in our practical group and the practical group examined. This will lead to a time complexity of $O(n * m * j)$. As one of the last steps we still need to sort the list of the practical groups which will be done in a time complexity of $O(n * log(n))$. All this combined will lead into a total time complexity of $O(nmj + n * log(n))$ which for small m and j will lead to a total time complexity of $O(n * log(n))$.

We chose this implementation, because it is very easy to understand and very adjustable implementation of the algorithm. There are several ways to optimise the algorithm by using precomputing. This can be done for example during the defining of the skills by the user and will save execution time when viewing recommended solutions. Next to that we have shown in the previous paragraph that with both a small amount of skills defined by a practical and with a small amount of users in practical groups, this algorithm will run in $O(n * log(n))$. Since this will be the case, because both sizes are limited this will be fast enough to run in realtime while recommending without precomputing. All these cases prove that the algorithm is good enough for recommendation system.

function Recommend($practical$, $user$)
  $resultList \leftarrow$ Empty map of PracticalGroup and distance
  for all $practicalGroups$ in the $practical$ **do**
    $distance \leftarrow 0$
    for all $skills$ in the $practical$ **do**
      $average \leftarrow 0$
      for all $users$ in the $practical$ **do**
        $average \leftarrow average + SkillValue(user, skill)$
      end for
      for all $users$ in the $user$ practical group **do**
        $average \leftarrow average + SkillValue(user, skill)$
      end for
      $average \leftarrow average/(amount\_of\_users)$ ▯ The amount of users in the practical
      $distance \leftarrow distance * (average - SkillValue(Practical, Skill))$
    end for
    $distance \leftarrow distance^{(1/amount\_of\_skills)}$ ▯ The amount of skills in the practical
    $resultList \leftarrow resultList + (practicalGroup, distance)$
  end for
  Sort the $resultList$ by descending $distance$
  return $resultList$
end function

Figure 4.11: *The recommendation algorithm*

5

Testing

In this chapter we will introduce the different techniques that we have employed to test the different aspects of the system. We will start with describing the different kind of unit tests that we have written. These tests provide a basic code coverage. After this we will describe what kind of integration tests we have performed to ensure the connection between the different components. Then we will describe the user tests we have performed to receive feedback from possible future users of the system. Lastly we tell about the requirement tests that were performed, to test if we achieved our goals.

5.1. Testing methods during the implementation phase

During the implementation phase, we already started with testing. Because the system was not completed during this phase, only parts of the system could be tested. We have tested each feature with unit tests, see Section 5.1.1. We also performed integration test during the implementation phase, see Section 5.1.2

5.1.1. Unit testing

During the implementation phase a feature is only complete when it is fully tested. This is done by test from the programmer while running the system, but also done by running automated tests. These automated tests are required for achieving a good code coverage and for ensuring the code to function appropriately. The tests that are written for other features will be always run before the feature has been merged. This ensures that other features that may change the function of the code does not let the tests fail. So all the features that were already in the system will still function even after the merging of the new feature.

In figure 5.1 the increasing amount of tests that increase with each pull request that has been merged with the development branch can be seen.

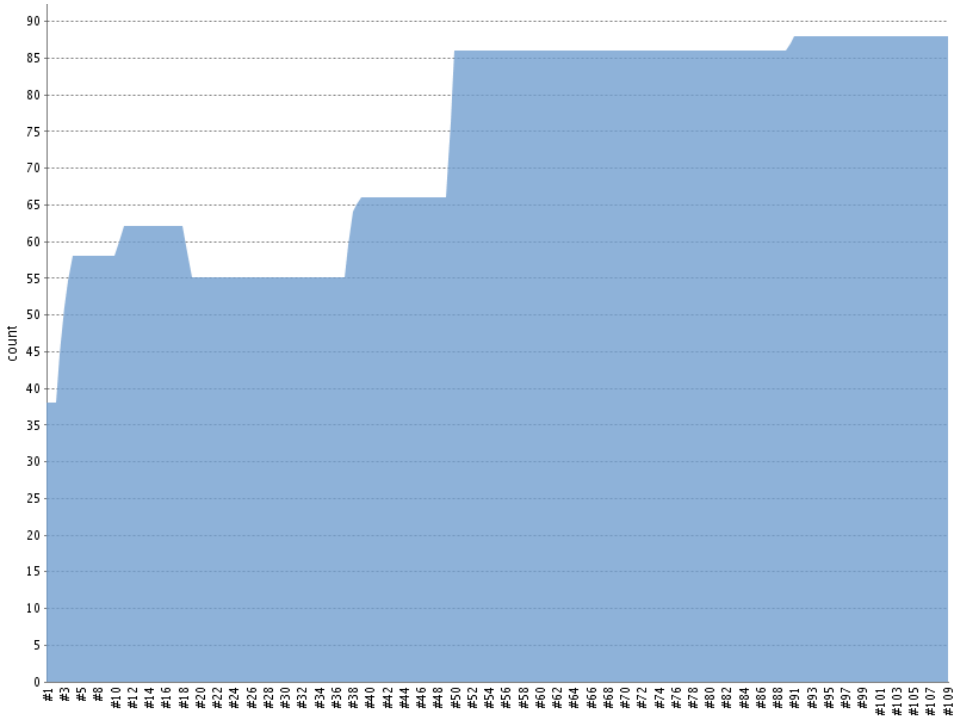


Figure 5.1: *Amount of tests per build*

In figure 5.2 the increasing amount of tests of the different pull requests that have been made can be seen. In this image can clearly be seen that sometimes when a new feature is added, some tests start to fail. Due to our continuous integration we could spot the errors early and make sure that they are fixed before merging with the development branch.

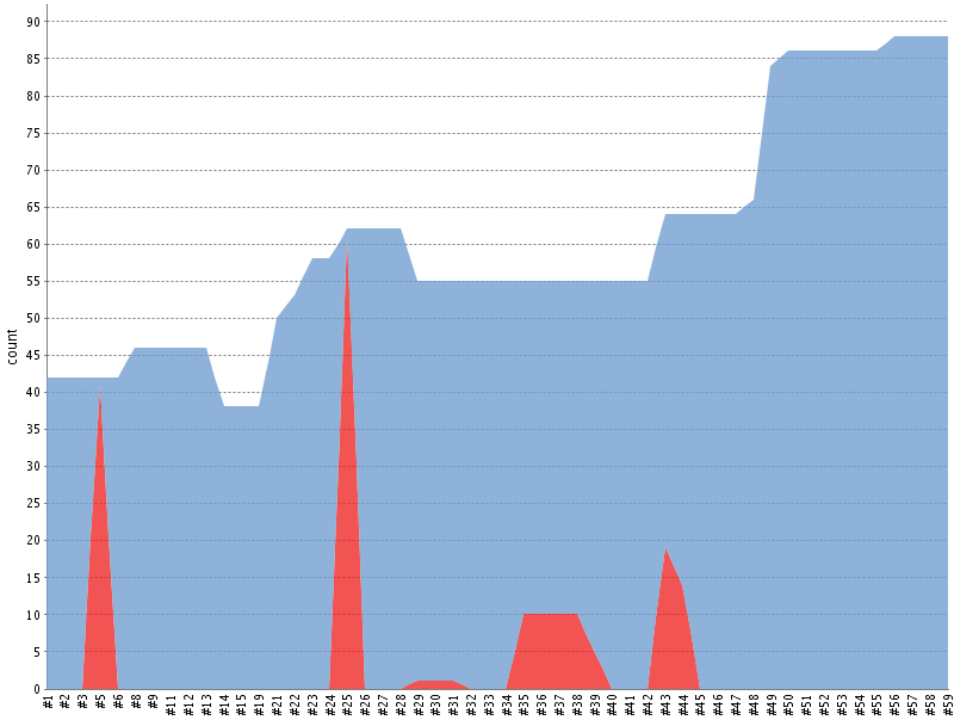


Figure 5.2: *Amount of tests per build of a pull request*

Figure 5.2 shows how not every pull request is without errors. All of these errors first had to be resolved before the pull request could be merged.

Figure 5.3 depicts the code coverage achieved by the tests. The goal is to have the code coverage percentages to be around a certain percentage during the entire implementation. One could argue that a 100% code coverage is desired, but some features are quite trivial and don't need a 100% code coverage. Next to this is the fact that some plugins, in our case the eBean plugin, generates code for classes. This generated code is not something we need to test since it is not written by us.

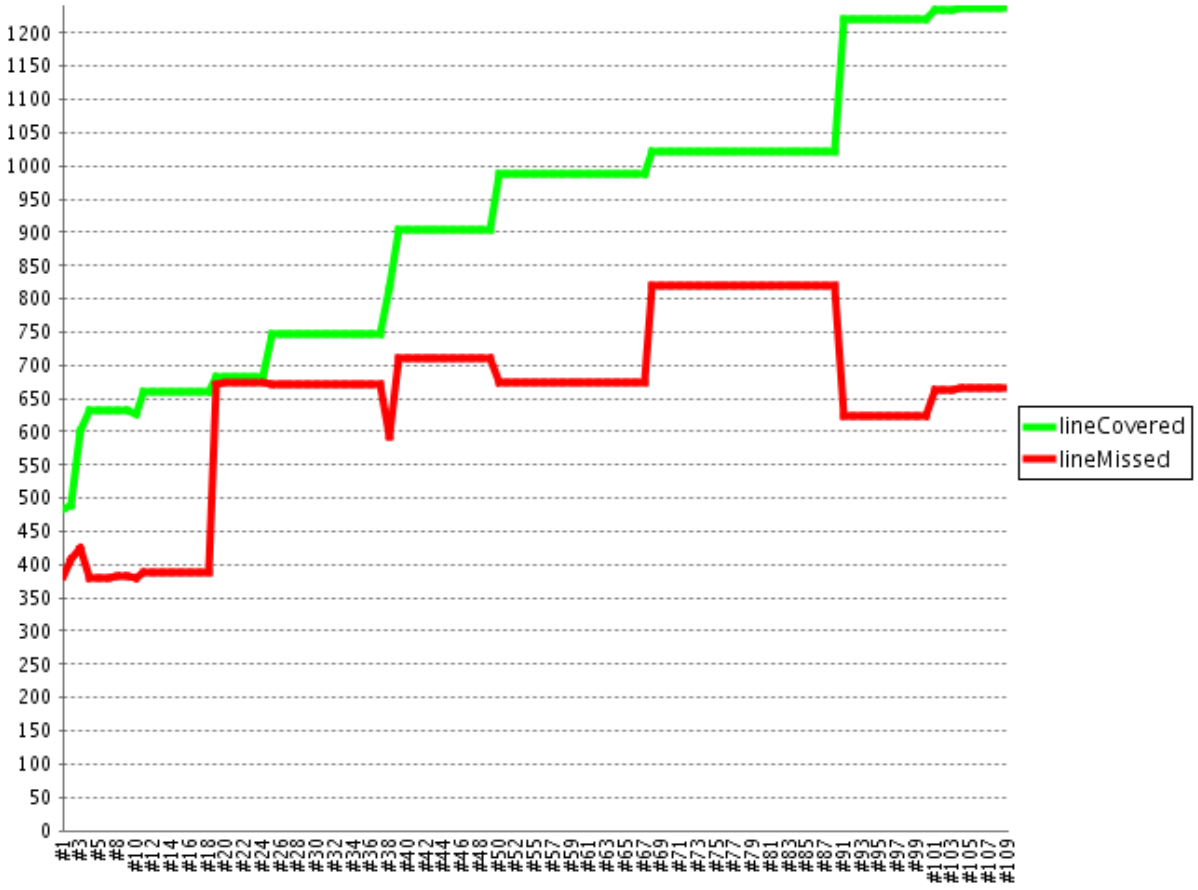


Figure 5.3: *Code coverage trend for the development branch*

We also underestimated the complexity of some features and thus did not wrote the tests for all cases with the first pull request of those features. In that first basic implementation some of the cases were already implemented in part of the system but not actually called upon and made available from the front end. We thus decided to first create a pull request for a basic version and thus split up the amount of changes. Later, when the feature was further implemented, we added tests that would test these extra cases.

5.1.2. Integration testing

Every time a pull request was made, the changes were checked by a member of the team that did not make the pull request. This ensured that the changes made reflected the feature that was selected to be implemented. The member of the team to check the code asked questions about some changes and checked if complex functions could be made more efficient. Although we use Checkstyle and Findbugs to perform code analysis, they did not find everything. So there is much value to achieve with a member of the team checking the code.

In figure 5.4 an example of a pull request and how a team member commented on a part of the code can be seen. The requesting team member responded on the issue and argued why he choose the implementation.

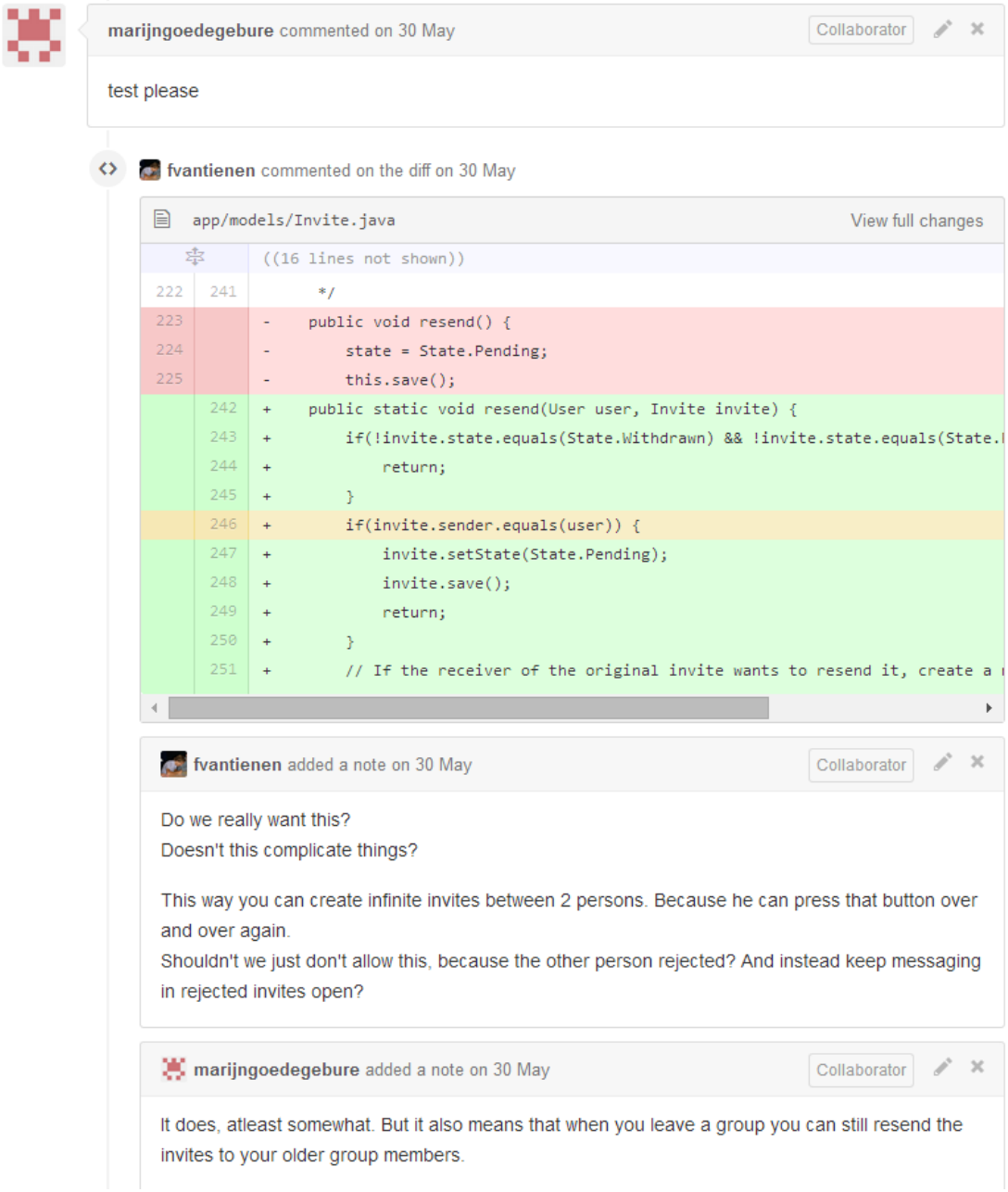


Figure 5.4: *An example of a pull request comment and reply.*

For an overview of the pull request and the comments that were made, more information can be found on the following website: https://github.com/florisverburg/pm-impl/pulls

5.2. Testing methods after the implementation phase

The testing of the entire system is done after the implementation phase has been completed. This is partly done by checking if the final product matches the requirements set. It is also done by letting the target audience use the system and document the findings. We will start with describing the user

tests.

5.2.1. User test

The purpose of the user test for practical participants is to test the system without the limiting factors of a development environment. The system will be tested for his robustness in the different kinds of input a user can generate. The test also let's the user make extensive use of the interface that we provided. Any errors will be revealed by the user's interaction. If the tester finds anything unclear he can comment on that in the appropriate text boxes provided in the survey, see Appendix D.

To test the system we will be defining the people that are valid testing subjects and how we documented the various aspects of the test.

The user

We define a valid subject for our user test as a person that is part of the target audience (which we already defined in an earlier chapter). The user should also be unfamiliar with the system. We will focus our user test on the students, since they will be the biggest group of users of the system.

The test

The user will be asked to test different functionalities which we will define beforehand. The functionalities that are included in the survey for testing are:

- Registration
- Logging in
- Register to practical using url
- Viewing the practical that they registered to
- Invite other student to join your practical group

We hand the user a document which lists the different actions that we want the user to do. Before and after the test we let the user fill in a survey. The survey at the start is about who the user is and the survey at the end is about what they thought of the system. To easily process the responses that we receive, we have used a Google form. In this Google form we have combined the survey and the step by step action plan for the tester. The Google form document can be found in Appendix D.

The user test results

The user test was conducted during two days and provided us with some interesting comments about the system. In this section we will give a brief summary of the results of the survey and what we can do with this feedback.

As you can be seen in the complete survey that is in Appendix D, we started the test with a few basic questions. First three questions about our demographic:

- What is your age?
- What is your occupation?
- Do you know what a MOOC is?

Both questions gave results that we expected. As can be seen in Figure 5.5.

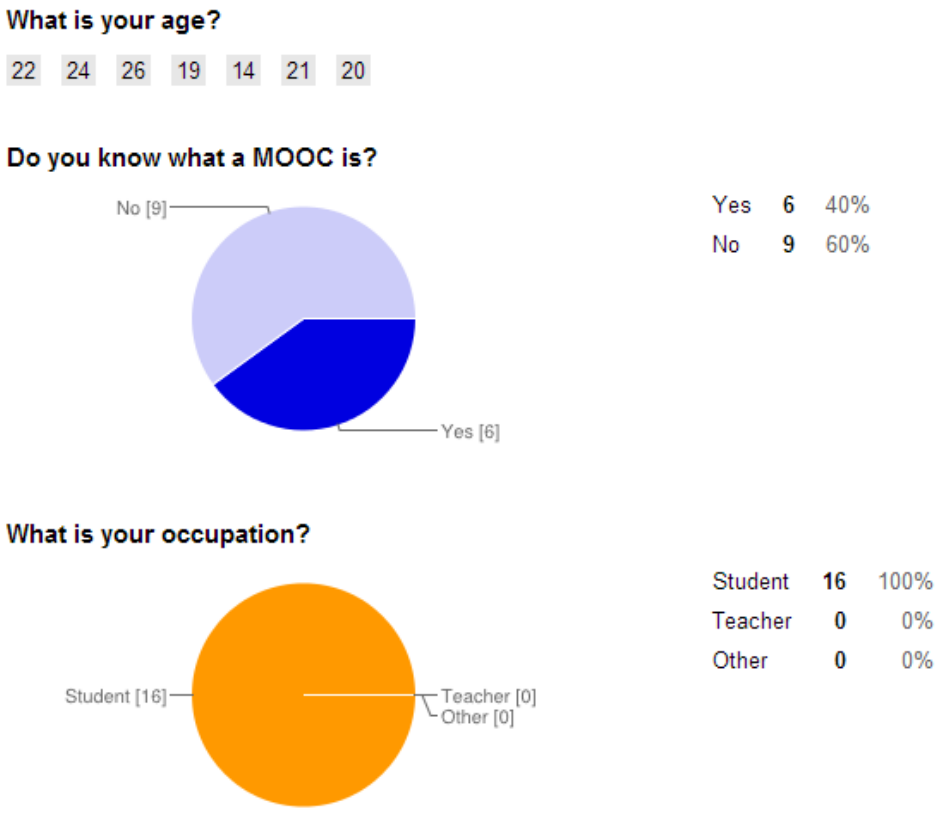


Figure 5.5: *Demographic questions*

The image above shows us that the people that were questioned were all students and between the appropriate age to be in our target audience.

More surprising is the amount of people that knew what a MOOC is. We thought this to be much more than the figures shows.

After the occupation question there was a split up in the survey. Students and teachers would be given different questions from there on. The other that could have been filled in would have submitted the survey directly.

The student part of the survey gives us more information about our demographic, as can be seen in figure 5.6.

Figure 5.6: *Student demograhpic info*

We did not have any respondents that were teachers or ”others” so we do not have any data on the

other part of our target audience. Having more time for an extended survey would definitely give more information about the demographic of our target audience.

After the first survey, the participant had to perform a number of actions on the system. After each action a couple of questions were asked One of these questions was always about possible improvements. This was asked to give the participant the possibility to give suggestions.

The first action to be performed was the registration and logging in to the system. In figure 5.7 the answers to these questions can be seen.

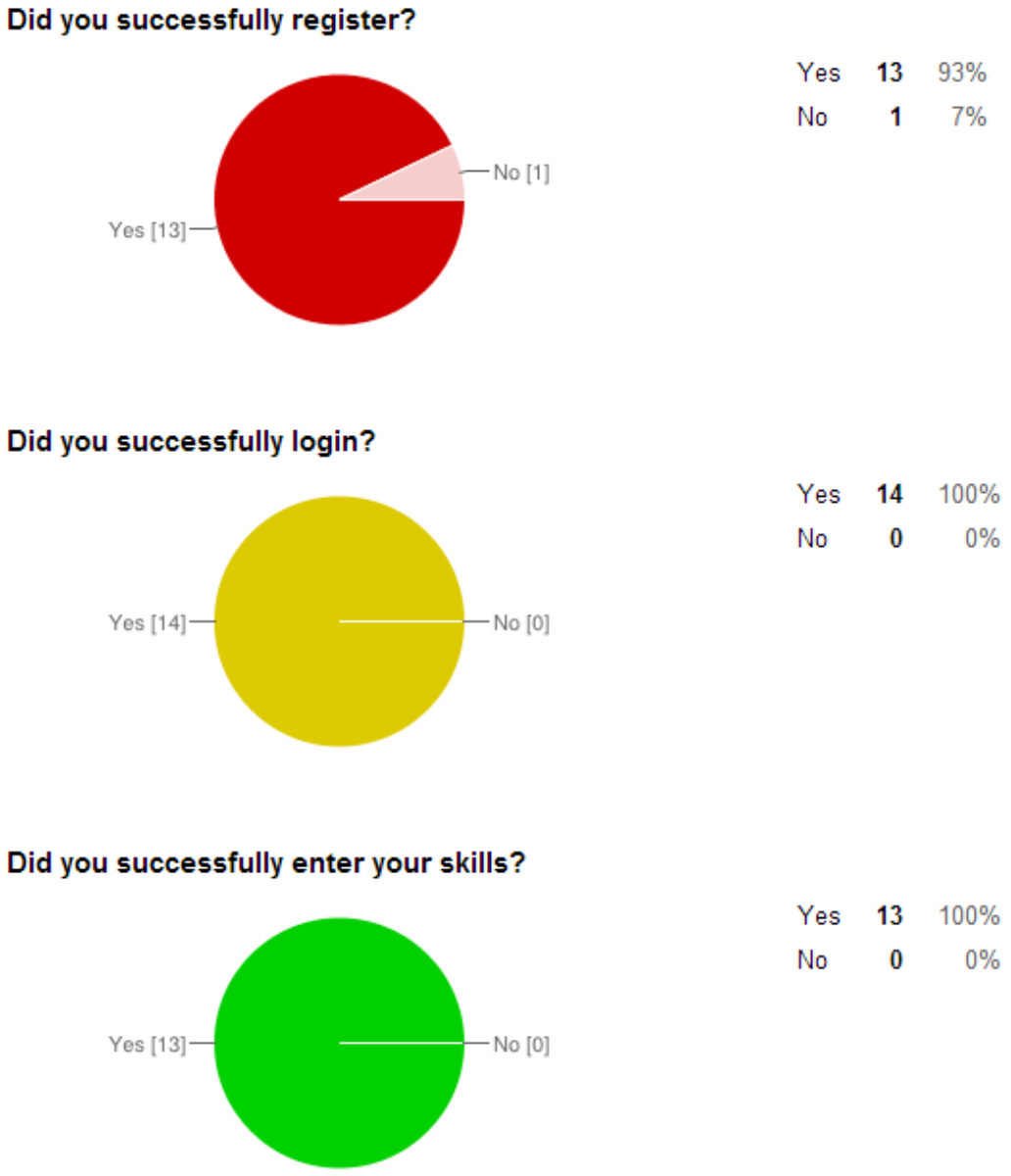


Figure 5.7: *Register and login success rates*

The charts shows a positive reaction to the registration and the log in parts of our system. Almost all of the participants succeeded in registering and logging in.

There were, however, some comments on how some parts had errors in them or how some parts were unclear. We made a list of these suggestions:

- Require fields tooltips sometimes align wrong
- Email activation link was not clickable
- The scale for a skill could be something different then from 1 to 10, since most people do not tend to give themselves a bad ”grade”.
- Send an email with more text, this way people become more involved
- There needs to be more incentive to enter your skills
- The slider value is not correctly updated

Each of these suggestions highlight some aspect of the system that needs to be looked at. All of these things can be used in future improvements of the system. This might also be things to look into during the week between the final report and the presentation.

The next part of the test was the registration to a practical using a URL and viewing the freshly registered practical. The first couple of questions were about whether the functionality worked and whether the participant had any suggestions on the functionality. The second part was focussed on the recommendations.

The charts of the first part can be seen in figure 5.8

The charts indicate that almost all of the participants have successfully registered to the practical and have successfully viewed the appropriate practical.

The suggestions indicate that there are some important improvements that can be done to make the system more accessible. The list of suggestions and remarks is as follows:

- No explanation of the purpose of the practical page
- A notice of a successful registration would be helpful
- Unclear why there are links to other practical groups
- Unclear what the purpose is of the invitations
- Show names instead of practical groups
- Explanation of the recommended skills

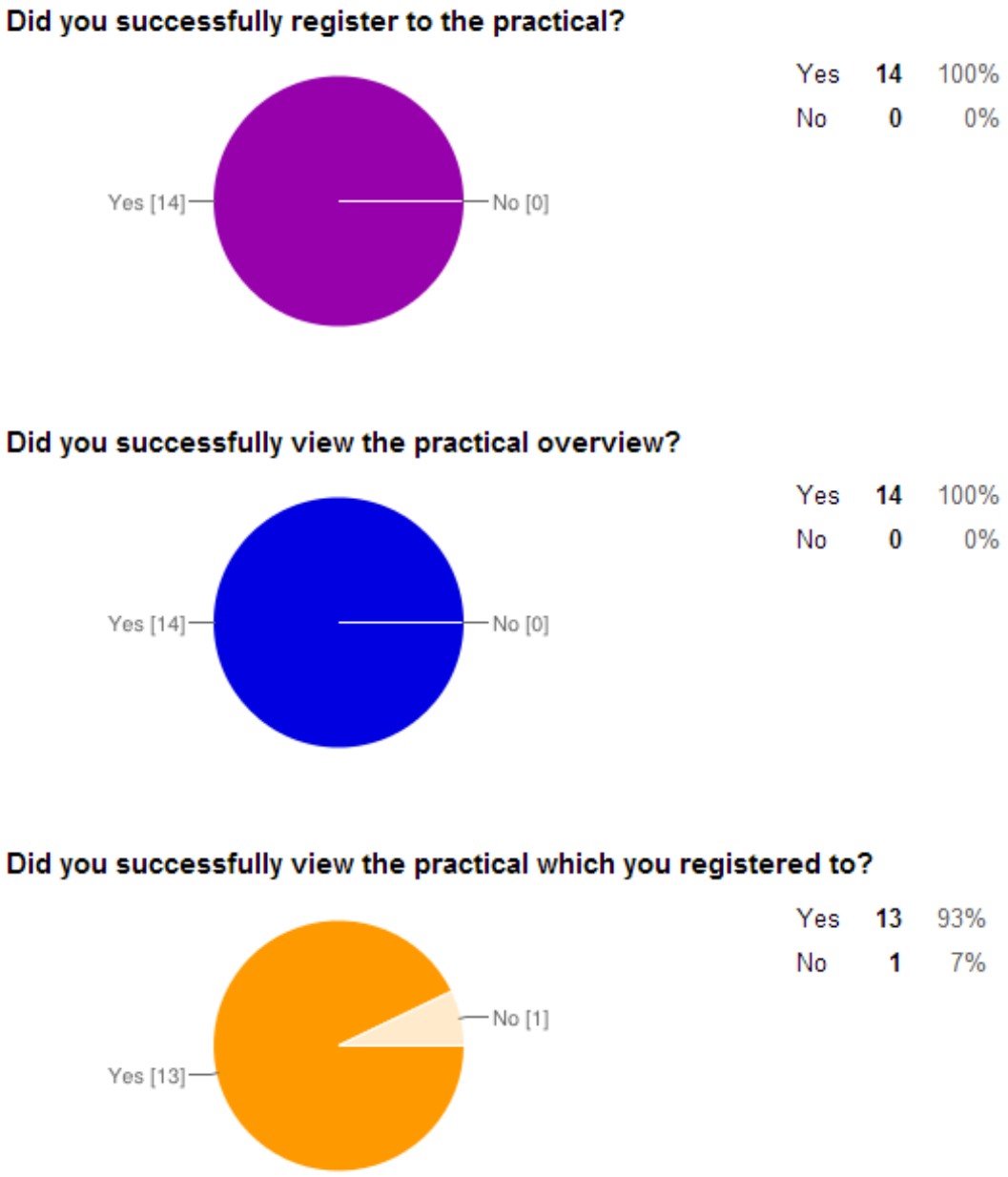


Figure 5.8: *Practical registration and viewing success rate*

We also noticed that the fact that some students were unfamiliar with MOOC's played a role in their ability to deduce the meaning of the numbers on the page and the meaning of the system. Since the system will have to be used by all kinds of users, we need to improve vastly on the interface and the explanation of what the system does. There are many possibilities for improving the system and we will have to look into the different possibilities to select the best one.

The chart about the recommendations can be seen in figure 5.9, followed by the suggestions and remarks.

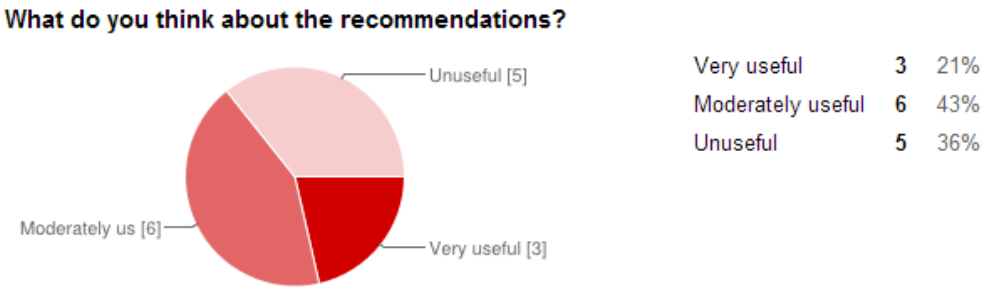


Figure 5.9: *Recommendation usefulness rating*

- Explain what is it recommending me
- For who are the recommendations?
- What are the recommendations based on?
- What is meant by recommendations? Are these my targets?
- What do the values mean and what do I do with it?

This chart is combined with the suggestions and remarks which let's us conclude that there should be a lot more information about the workings of the recommendations. People who will work with the system will require that information to believe that the recommendations are really working in their favour.

The last feature to test was the invitation of another student to join the user's practical group. Figure 5.10 shows the success rate of the participants.

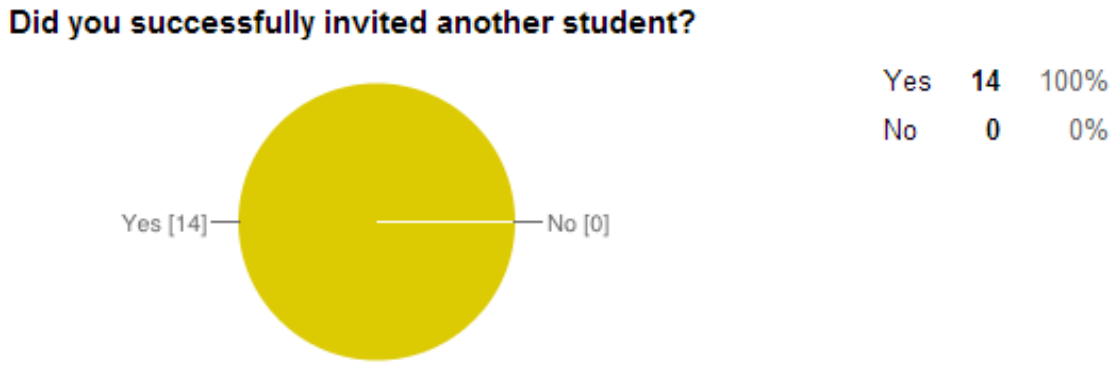


Figure 5.10: *Invite success rate*

The figure shows that the interface for inviting another student is working correctly. The suggestions and remarks are mainly about the participants how the invitation correlate to the practical groups:

- See other person's name and skills without going to a different page
- Invited practical group #14 but in my overview it said that I invited Peter de Jong
- Showing the skills instead of the group names might be more convenient
- Addition of a communication skill to improve the selection criteria
- Should a user start being in a group?

The comments raise an interesting point about whether we should change the user being in a group from the start or that a group should be created when an invite is accepted. It makes sense to do so, but we have multiple ways of changing this. We can either only change the front end representation or change the way accepting invites work. Both quite easily to do and worth changing.

The last part of the final survey asked a very important question concerning the role of the recommendations: What did you think about the usage of recommendations to suggest possible group members? The results are the following:

- Cool that you provide recommendations
- Good idea
- This was useful. With these recommendations you did not have to scan through all the people
- Great addition to courses! Not quite sure whether or not the using of recommendations to create the overall optimal group composition is the best way. A group with members of equal skill might be a better solution

The comments about the recommendations were great. It is nice to know that students appreciate the goal of the system.

User test reflection

The user test gave us the opportunity to test some essential parts of the system by open minded participants who are part of our target audience. They provided us with detailed feedback on the different aspects of the system. The user test did not gave us the possibility to completely test the invite system. To have this part tested by our target audience we need a different kind of test. A test that runs over the course of a week in which multiple users try to use the system at multiple times. Such a test would be a great way and quite necessary to have the system tested before it could be used in a real MOOC.

The feedback that we collected with the user test gives us some great improvements that we can consider for feature work. We will talk more about feature work after the conclusion.

5.2.2. Requirements test

In chapter 2 we define global goals, global requirements and detailed requirements. We will go into the detailed requirements in this chapter, the other two will be discussed in the conclusion. In the detailed requirements section we divided the requirements into four categories following the MoSCoW-model. The categories are:

- Must have, classifies as requirements that have to be in the final product, without these requirements the product would not be usable.
- Should have, these requirements are desired, but without these the product is usable.
- Could have, these requirements will be implemented if there is enough time.
- Would have, these requirements will not be looked into for this project, but can be interesting for a feature project.

Following the MoSCoW-model we can now say that we should at least have all of the must have requirements implemented or a good reason why we have not. We will first list the must haves that we have not implemented and why not. Secondly we look at which should have requirements we did and which should have requirements we did not implement. Using this, we will determine the progress we made during the project and if we correctly used the MoSCoW-model. The could and would have requirements will be discussed in the future work section.

Must haves

In the following table we listed the must have requirements that were not met.

Requirement	MoSCoW
Actor 1: MOOC participant	
The potential participant (anyone who visits the site) must be able to view the different MOOCs without registering or logging in.	M
The participant must be able to inform him/herself about the usage of the application without registering or logging in.	M

During the user test we noticed that the two must haves that we did not implement, were quite essential to the system. Due to the nature of the project, a proof of concept, we focussed more on implementing interesting features than making the system more understandable. So as a proof of concept point of view these requirements were probably more should haves than must haves, but from a user point of view these requirements were far more important than we have treated them.

Should haves

In the following section we will show a list of should have requirements that we have implemented and we discuss why we have chosen for these requirements to be implemented. Following, we will shortly discuss the requirements that we have chosen not to implement and why we have not implemented these requirements.

We will first start with the list of requirements that have been implemented:

Requirement	MoSCoW
Users should be able to enter the basic user information, like: country, age, short description, skills, education, a picture and personality traits.	S
The participant should be able to ask a currently existing group if he can join them.	S
The participant should be able to use their LinkedIn profile to log in to the application.	S
In case the participant already has an account, the participant should be able to add the course to his profile using the teachers url.	S
The participant should be able to receive an invitation from a group.	S
The participant should be able to accept the invitation from a group.	S
A group should be able to view a possible group member.	S
A group should be able to invite a possible group member to join their group.	S
A group should be able to accept an request from a participant to join their group.	S
The algorithm uses basic user provided data (e.g. education, preferences) to build up a profile for a course participant.	S
The algorithm can calculate whether two people will be good practical partners using the basic user provided data.	S
The algorithm can return a possible practical partners using the basic user provided data.	S

Following the goal to create a proof of concept we implemented the should haves that focus on facilitating a more complete proof of concept. We implemented a lot of the recommendation algorithm should haves, since it would make the proof of concept a lot better.

Next to this is the invitation system. While setting up the invite system, we kept in mind that both single students as groups of students will be using this system. This helped with formulating a system where single students can invite other single students and other groups, and where groups can invite other single students and groups. This gives the single students (and groups) complete freedom in creating their practical groups.

The requirements that we have chosen not to implement are mainly requirements about the teacher's part of the system. Due to the goals of the project we thought these requirements to have a lower priority and not be very important to the proof of concept. These requirements will become interesting when the implementation continues and is focussed on usage of the system.

Requirements test conclusion

As we can conclude from the analysis done in the previous sections, we can definitely say that we made a good estimation of the scope of the project. Due to the time and an underestimation of the importance, we did not manage to complete all the must haves. The two must have requirements that still need to be done are quite easy to implement and should be at the top of the list of things to be done. Although we failed to implement all the must haves before the delivery of this report, we did managed to implement a lot of should haves.

5.3. Software Improvement Group Feedback

For analysing how maintainable our created software solution we handed in our code to the Software Improvement Group (SIG). SIG analysed our code for maintainability and gave us some short advice on how to improve on this. The intermediate feedback from SIG can be found in appendix F and is written in Dutch. In this section we will summarise this feedback and give an explanation on how we plan to improve the code maintainability.

5.3.1. Intermediate feedback

On the maintainability scale of SIG we scored almost 4 out 5 stars during the intermediate feedback. This indicates a better than average code maintainability where still some improvements can be made. Three aspects that we could mainly improve on are:

- **Code duplication:** The code that is redundant, because it occurs multiple times and could therefor be removed. Because of maintainability it is suggested to keep the code duplication very low, because adjustments to this code needs to happen in multiple places.
- **Module Coupling:** Here is taken into account the percentage of code that is called the most. Normally this code makes the system less stable, because changes in this code can lead to changes on many different places.
- **Component Independence:** By component independence is looked at the amount of that is used only internally in a component. How higher the percentage of code that is called from other components, how higher the chance is that changes propagate to other components. This has an influence in the future productivity.

SIG also mentions in the report that the availability of the test code is promising and that they hope that it will grow at the moment of adding more features.

5.3.2. Improvements

Based on the intermediate feedback from SIG we tried to analyse most of these problems ourself. We already had some tools for analysing the code maintainability and changed some of the settings and added some new tools.

For the code duplication problems we enabled the 'StrictDuplicateCode' module in CheckStyle to search for duplicate code. This module searches for a defined amount of similar lines of code and ignoring the whitespace in between. We will improve on all these duplicate lines of code and try to remove as much a possible.

Next to adjusting CheckStyle we also added a new tool called Google CodePro for analysing the other aspects from the SIG feedback. Goole CodePro can do a lot of checks both CheckStyle and Findbugs already support, but it can also analyse the dependencies. An example of the dependency graph between the packages is shown in Figure 5.11. In this dependency graph we can see the problems of the module coupling and component independence.

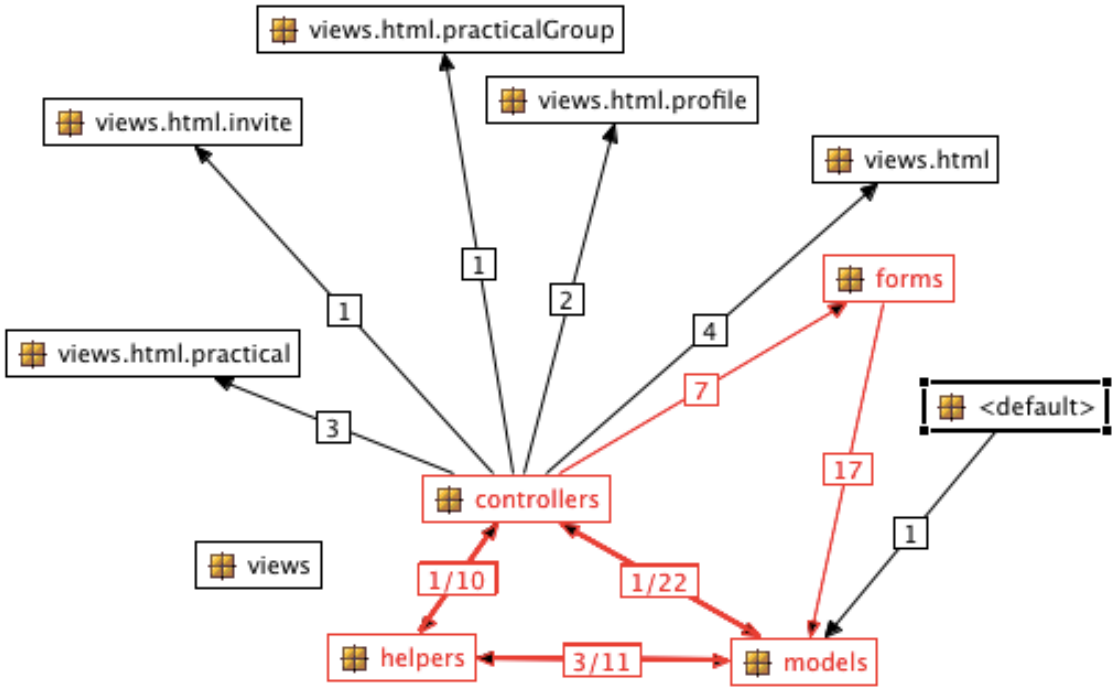


Figure 5.11: *Dependency graph*

In Figure 5.11 we see that we have cyclomatic dependencies between controllers, forms, helpers and models. We can improve on that by eliminating the dependencies between packages that go both ways. Since we now are able to analyse this in the dependency graph this can easily be improved.

5.3.3. Final feedback

After implementing the improvements from the previous section we uploaded our code to SIG again, to analyse if our code was improved. In total they say that both the maintainability and the size of the system has grown.

The code duplication that was described above was mainly improved by using more abstraction and implementation of methods. For the module coupling they analysed that we divided most of the functionality that was available in the User class and moved them to several other classes. As last they also mention that we have removed all the cyclomatic dependencies.

From this observations they conclude that we have used SIG's advice during the rest of our implementation. They are happy to see that next to the improvement of the maintainability also the amount of test code is improved.

6

Conclusion

During the past weeks we have created a platform where teachers can create the practicals associated with their MOOC and where students can enrol for practicals and form practical groups to work together on assignments. This platform serves as a proof of concept for the various problems that were mentioned in the problem description. One of the problems is the scale of MOOCs, which makes searching for a good group member is not a trivial problem. To assist the student in forming a good group, the platform recommends possible group members to the searching student.

We split the project in three phases: the orientation phase, the implementation phase and the final phase. The project started off with the orientation phase in which the different aspects of the problem description were analysed. During this time we looked into what kind of recommendation algorithm we would be using. We also made an extensive list of requirements that helped us set out the different design decisions. Lastly, we made a basic system design. In the next phase, the implementation phase, we incrementally implemented the different features of the system. At the end of the implementation phase we started the user tests. With these user tests we tested the interface and basic usage of the system. During the last phase we finished up the last features and we focused on documenting the project in this report. We also received the feedback of SIG on the maintainability of our source code, one of the things done in the last phase was to improve the source code using this feedback. The last thing to be done in the last phase is looking in what ways we can improve on the final product. This is discussed in the future work section later this chapter.

This report described the entire project. We started with introducing the problem and our target audience. Secondly, the report described our supervisors, the different actors of the system, global goals, global requirements and detailed requirements. Following this, we described the project methodology we have chosen and we discussed how it panned out for us. Per phase we described the goal, planning and deliverables of the phase. We also reflected on each phase separately. In chapter 4 we described the design we made of the system and how we implemented specific parts. Things that we discuss in this chapter: Database diagram, class diagram, state diagrams, sequence diagrams, MVC, implementation of the recommendation algorithm and implementation of the invitation system. Next, we described the different kinds of testing that has been done during the project. We started with describing the different kinds of testing that we have done during the implementation phase, followed by the testing that we have done after the implementation phase. During the implementation phase we made use of unit testing and integration testing. After the implementation phase we made use of a user test and requirements test. Lastly, we described the feedback that we have received from SIG and how we thought on improving this feedback.

The final product, that will be presented, will be a working proof of concept which will fit closely with the specified target audience and demands of the assignment.
To determine whether this goal has been met, we can look at the requirements. The requirements have been set up with the target audience and the problem description in mind. The problem description states the following important problems that need to be solved by the system:

- Difficulty finding a good practical partner.
- Searching for a practical partner is a time consuming process.
- Sub optimal solution, people often look for group members which share some interests. While this often makes it easy to work together, this will not guarantee that you have a good functioning group with enough knowledge.

Looking at the requirements, we can definitely see that these problems are well represented by the requirements. During the project we focussed on implementing these requirements. It is no surprise that the resulting product is a proof of concept for the solutions that we have selected to solve the problems mentioned above.

To determine how well our proof of concept solves the problems we will have to look at to what extent we have implemented the requirements. We will make that comparison in the other goals, since those goals were focussed more on what the system should be able to do.

We want to create a platform for searching other students and letting the students invite other students for a practical group.
We have achieved this goal by creating a website and implementing an invitation system for this website. Using this invitation system, students can search and invite one another. This invitation system gives students the possibilities to invite both single students as groups of students. It also gives groups the possibilities to invite both single students as groups of students. The invitation system gives the user all the possible actions needed for the process of creating a practical group.

We can verify the completion of these features by looking at the requirements test. The requirements clearly states which requirements have been implemented and which requirements have not. The requirements test also clearly states that all the must and should haves concerning the creation of a platform for searching and inviting other students have been implemented.

We want the final product to recommend good fitting possible group members to the students. By ”good fitting” we mean that there is some heuristic that we use to calculate whether two students would make a good practical group. We want the final product to achieve more optimal solutions than the current situation
In our research report we wrote extensively about what aspects are important to our recommendation algorithm. Questions that we asked ourselves at the start of our research were:

- How does the algorithm determine a possible practical partner?
- How does the algorithm make a difference between two possible partners and determine which one is a better fit?
- How does the algorithm make use of the extra information provided if the could haves are implemented?
- How does it make sure the partner that is recommended hasn’t already been suggested?

We tried answering these questions using known techniques like knowledge-base recommendations, content-based recommendation, collaborative recommendation and demographic recommendation. After doing research we concluded that the known techniques did solve all our problems. We thus choose to select aspects of some techniques and combine these to a complete solution. We decided upon the following aspects:

- The algorithm uses the idea of social interaction platform that is based off the platform that the Tinder app uses.
- The algorithm uses a knowledge-based approach to deal with the various kinds of user information.
- We use the geometric mean that is used by OKCupid’s algorithm to calculate the differences and similarities between users.
- We would like to make the algorithm robust for cold-start problems, the usage of a knowledge-based approach should help with this.

We do not have a guarantee of the performances of our algorithm, but we can say that our algorithm is perfectly tailored for the problems that were mentioned in the problem description. We can improve on our understanding of the implemented algorithm and it's results by creating simulations. These simulations would simulate the different actions of the system and will simulate both the use of the system as the use of the algorithm. Due to the time limits of the project we did not get to implementing these simulations.

Using the requirements test of chapter 5 we can say that we have done a good job implementing the must and should haves that are focused on these problems. The only thing we have fallen short in, is the fact that we deal with a wide target audience. MOOC's are open to all people and we have not spend enough time writing an explanation of the system.

We want the maintainability of the final product to very high and want the code to be easily extendable.
Maintainability is very important in creating a software product, because it is important to easily solve defects, replace worn-out components, meet new requirements, etc. While programming, all these aspects are very important. In chapter 5 we have discussed the SIG feedback and have seen that our maintainability scores above average. This will be improved after the analytics we have done on the improvement points on the SIG feedback.

After the analysis we will try to improve our maintainability by making the software more modular. The improvements we have made after finishing this report will further be discussed in our presentation. For now we are happy with the three stars that we have scored.

6.0.4. Recommendations for future work

Although we managed to implement most of the must and should haves that were important for the goals of the project, there is still much room for improvement. There is room for improvement in different aspects of the system and some have a higher priority than others. Below we will list the different aspects that can be improved upon, we introduce a general direction for improvement, discuss shortly why improving this aspect will improve the system in general and it's priority.

- First on the list of improvements is implementing the last two remaining must haves. This improvement has a high priority because it makes the proof of concept much more complete and usable. The proof of concept would be a better representation of the system that is needed. Our goal is to implement these two before the presentation.
- The recommendation above already describes some of the improvements that can be made to the user interface. Other improvements can be found in the interface mock-ups that are not yet implemented. Implementing these interface mock-ups will drastically improve the user interface. These have a moderate priority due to the focus being on having a working system. The priority will increase when the focus will be shifted towards the usage of the system.
- The user test revealed some errors in our system that our tests did not find. Fixing these errors is high on our priority list and definitely something we want to look into during the time until presentation. Due to it being errors, they are high on our priority list. We want to deliver a complete and working proof of concept and this is a part of that.
- Although we deliver a working proof of concept, it is difficult to verify the performance of our recommendation algorithm and our system. This would need either a large user test or a simulation. We think that the implementation of a simulation would help to understand and verify the working of the current system. This simulation will simulate the process of using the system and by doing this try to measure the performance of the recommendation algorithm and the system.
- To improve the recommendations that the algorithm gives more information is needed. This can be done by adding more skills for the user to fill in. Increasing the amount of skills that the user provides will increase the accuracy in which the algorithm can recommend. This could be done by letting the user rate him-/herself for more skills, this is the easiest solution for this improvement. But other ways of rating user skills might be more beneficial to the system. These other ways might be the user answering questions which are related to the subjects. The answers can then be rated to determine the score of the different users.

- We already mentioned some possible improvements that will increase the accuracy of the algorithm. These improvements were based on using more data in the recommendation. Another way of improving the algorithm is by adding another technique to the recommendation algorithm. An example of this is a rating of the recommendation by the user. The algorithm has to be adjusted so it can use these ratings to further improve his recommendation.
- In the requirements test we have mentioned that we have implemented all the must and should have requirements that are connected to the focus and goals of the project. The next step for this product is towards work to version which will be used in a live situation. This will result in more features that will have to be implemented. One of the things that need to be done will be implementing a more advanced practical management system for the teacher. In that way the product will be more attractive to the people that have the choice to start using it. It will not have a big initial priority, but will increase in priority when the product will be focussed more on live usage.

A

Appendix A - Plan of Approach

Preface

This document is the Plan of Approach report that has been written for the Peer Matching assignment for the Bachelor Project course. The assignment was issued by Dr. L.J.M. Rothkrantz of the TU Delft. In this document, we will outline the Plan of Approach for this project. In the first section we will introduce the assignment and explain why it was issued. In the second chapter we will describe the current situation, the goal of the project, the project assignment and the deliverables. In the third chapter we will define our approach and planning for the time that is available to us. In the fourth section we will define the project approach, who will be working on the project, who is in charge of what and administrative procedures. We will also be describing how we finance the resource that we require, how we will report the progress and what technical resources we will be using. In the final section we will describe our approach to assuring quality of the final product.

Introduction

In this section we will provide an introduction to the assignment. We will start by giving a company outline followed by giving a background and motivation of the assignment. This will answer the question: why has this assignment been issued?

Company outline

The assignment is issued by Dr. L.J.M. Rothkrantz who is part of the Interactive Intelligence group in the Intelligent Systems department. This department is located in the EEMCS faculty at the TU Delft. The research mission of the Computer Science department is as follows: "The research mission of Computer Science within the Faculty of EEMCS is to contribute to the advancement of science, engineering, and design in the broad fields of autonomous distributed systems and information analysis and interaction."[4]

Background and motivation of the assignment

Each year many students go to an university to follow the predefined bachelor and/or master programs, but not everyone has the time and intent to follow an entire program. There is a demand from all sorts of people to be able to follow a course while their main occupation isn't that of a student. For example: a working parent could be very interested in such a course. That is why universities introduced MOOCs.

The basic idea of MOOCs is that students remote in place and time follow the lectures, can participate in a practical assignment and can make an exam. If they score sufficient, they are rewarded with a certificate that states their success in the course.

The concept of the MOOCs is a very new concept that still has many areas to improve upon. One of these areas is the area of practical assignments. Currently, for a Computer Science MOOC at the TU Delft, multiple practical assignments are required. The participants are required to form groups to make these assignments. After the announcement the participants either use mail, Facebook or Twitter to contact other people in order to form the groups.

We would like to improve upon this concept and try to provide a more fitting approach.

Assignment description

In this section we will describe the definition of our Peer Matching assignment. We will describe which people are involved in the assignment, what the exact assignment is and what the difficulties can be.

Client

The assignment was written by Dr. L.J.M. Rothkrantz of the TU Delft and is our main client. Dr. L.J.M. Rothkrantz asked Dragos Datcu from the TU Delft to advise us on our design and implementation of the Peer Matching assignment.

The target users of the system are MOOC participants and MOOC teachers, specifically for the Computer Science courses. The system is easily extendable to provide more different MOOCs then Computer Science courses, but will not be supported by default.

Contact information

Team

Marijn Goedegebure *(marijngoedegebure@gmail.com)*

Floris Verburg *(floris.verburg@gmail.com)*

Freek van Tienen *(freek.v.tienen@gmail.com)*

Project supervisors

Dr. L.J.M. Rothkrantz

Dragos Datcu

Problem description

MOOCs around the world are growing in size and thus the practical assignments need to be easy to manage. The growth also means that it is more difficult for each participant to find a possible practical partner due to the large amount of people. It also becomes really difficult to find the correct partner or group members when you don't see each other during courses and haven't even met most of the people from your course.

Most of the time participants of MOOCs are searching group members on social media like Facebook or Twitter. They just try to look through some of the participants names in the course, search for them on-line through social media and check if they are a capable partner or group member. This approach takes a lot of time and most of the time there are too much MOOC participants to go trough all of them.

Next to the fact that this process is very time consuming, this process also often leads to a sub optimal solution. This is due to the problem that people often look for partners of group members which share the same interests. While this often makes it easy to work together, this will not guarantee that you have a good functioning group and enough knowledge inside the group.

We would like to improve upon this process by creating an application that can be used to create practical groups without the participants being required to go search intensively trough all the participants.

Participants should be able to easily access the application, fill in some relevant information about themselves and then be recommended possible practical partners or group members.

The interaction with the system has some difficulties. If we want to get some information from third parties, we are required to use the information in a way this third party delivers the data. There can be difficulties processing this data.

The recommendation of possible group members can be quite difficult too. One of the problems we are facing is the question: What user information is useful to determine whether two people can be good practical partners?

Goal

To provide an overview of what the software must be capable of, we define four global requirements for the system. By defining these four global requirements we provide a foundation of what the system must be able to do.

- MOOC teachers must be able to create, manage and edit courses for which participants of their MOOC are able to register.
- MOOC students must be able to register and deregister for a MOOC practical assignment.
- The system must be capable to use personal data provided by the user to find a suggestion of an appropriate practical partner or group member. It must be possible to adjust the definition (parameters) of the appropriate practical partner or group member, and search trough several suggestions.

Assignment formulation

During the course of this project we will develop a Peer Matching system for finding appropriate practical partners or group members during MOOCs practical assignments. To be able to find an appropriate practical partner or group member, we split up the assignment in two main areas of focus:

The basic system makes it possible to register to MOOC practicals, find and create practical partners or group members, and communicate with other participants. The research in this area focusses on which information a participant would like to know when choosing a practical partner or group member.

The algorithm makes it easier for the users to search through all the available practical partners by giving suggestions of appropriate practical partners or group members. Here the research is focussed on how to find suitable group members or practical partners, based on skills and personality.

Deliverables

At the end of the project we will deliver a working prototype of the Peer Matching system, containing as much requirements as possible. Because of the complexity of the Peer Matching algorithm and the short timeframe of 10 weeks, we choose to develop a prototype instead of a full working product. We also choose to focus mainly on MOOCs from the study Computer Science, because the development team has more experience with these courses which makes it possible to make a more feasible Peer Matching algorithm.

Risks

We have concluded the following main risks:

- Finding a suitable algorithm for matching practical partners and group members could be very difficult to evaluate and requires a lot of time. This is because it is very difficult to define a ”suitable” match between practical partners and group members based on skills and personality. Every student is different and has its own preferences about a ”suitable” practical partner or group member. This makes it almost impossible to evaluate if the algorithm gives correct suggestions to the students.

- Overestimating our knowledge and underestimating the time required for tasks. Since this is a common problem with Software Engineering there is a good change that this will also be happing to us. Estimating the knowledge in de software design team is very difficult because it relies on too many factors. Because of that estimating the time required for several tasks is very hard.

Approach and time schedule

In this section we will be describing the methods, techniques and tools that we will be using to develop our application. We will also be giving a global overview of the planning.

Methods and techniques

We will be developing our application in the programming language Java. We will combine this with the Play framework to provide us with a solid basis with which we can develop our web application. Next to this we will be using scrum as our software development, more on that in the appropriate section.

Applications and technologies

We will being using the following applications and technologies:

- IntelliJ IDEA, is our Integrated Development Environment (IDE) which will facilitate the development and testing of our code. IntelliJ will be integrated with the Play framework which will be helpful during development.
- GitHub, is our on-line code and documentation repository. It provides for a version control system, an issue tracker and code review possibilities. We will be using it to store all our code and documentation's code (LaTeX).
- Cloudbees Cloudbees will be used to run our test environment, our continuous integration and the release environment. It uses Jenkins for the continuous integration.
- Play framework The play framework gives us the possibility to easily create a web application using Java.
- Findbugs is a plug in for Cloudbees and IntelliJ that gives us the possibility to let our java code be checked for small bugs using static analysis.
- JaCoCo is a plug in for Cloudbees and IntelliJ that provides us with data analysis about our code coverage.
- Checkstyle is a plug in for Cloudbees and IntelliJ that checks the code for coding standards. This makes it ideal to enforce the coding standard for our project.

Software development method

We will be using scrum as our software development method. Scrum is based on the agile software development framework which can be used to manage software projects. We choose scrum because of its flexible approach to software development. We do not have the time to research all of the aspects of the system. Scrum gives us the possibility to react to changes of requirements, but also to react to new insights given by our increasing understanding of the system as time progresses. We will define the different roles associated with scrum (e.g. the product owner, development team, scrum master) in the next section.

Planning

We have set up a global planning that splits up the project in phases. Each of these phases has deliverables attached to them. We will be mentioning our goals for each week. Other important dates are also added to the planning.

Phase 1, Set up and research

During the first phase we will be researching the requirements of the system and which requirements are essential for the application and what requirements are not. We will also be looking into the set up of the project for our implementation phase.

Week 1: 21-04-2014 - 25-04-2014
Drafts of Plan of Approach and literature study, setup of the development environment.
Week 2: 28-04-2014 - 02-05-2014
Finished Plan of Approach, literature study, first system design.

Phase 2, Implementation

During the second phase we will be developing the application. During this time we will be using Scrum to support our development. During week 3 to 8 we will be developing the application. During this time we won't have big deliverables, but we will be using the weekly meetings with our supervisors to measure our progress.

Week 3: 05-05-2014 - 09-05-2014

Week 4: 12-05-2014 - 16-05-2014

Week 5: 19-05-2014 - 23-05-2014

Week 6: 26-05-2014 - 30-05-2014
Start of the acceptance testing.

Week 7: 02-06-2014 - 06-06-2014
End of the acceptance testing, first draft of the final report and send code to SIG for review.

Phase 3, Final report and presentation

Week 8: 09-06-2014 - 13-06-2014
Final development week, final draft of the final report.

Week 9: 16-06-2014 - 20-06-2014
Send code for final review to SIG, send final report to supervisors for review, creation of the presentation.

Week 10: 23-06-2014 - 27-06-2014
Presentation

Project approach

In this section we will be explaining our approach to the project. This begins with by defining the stakeholders of the project and what roles they will be having during the course of the project. Following this, we will be explaining the administrative processes that will be used to monitor the project and to make sure that it will achieve the goals set. Next, we will explain the means by which we finance the resources we use. We will also define the ways in which we report our progress to the client. Lastly, we will define the resources that we will be using during the project.

Stakeholders

The following table lists the different group members, their roles and a short description. This description describes the role and responsibilities of the group member.

Name	Roles	Description
Freek van Tienen	Lead Development	Responsible for implementation phase.
Floris Verburg	Scrum master and Quality Assurance	Responsible for the Scrum process, the quality of the code and test coverage.
Marijn Goedegebure	Project Manager and Reports	Manages the project and ensures that the requirements are met.
Leon Rothkrantz	Client, Group Mentor and Supervisor	Supervises the project, aides the group with difficulties and has issued the assignment.
Dragos Datcu	Group Mentor and Supervisor	Supervises the project and aides the group with difficulties.

Administrative processes and reporting

In this subsection we will be describe the administrative processes that we will be using to make sure that the project will achieve the goals set.

Monday meetings

We will have weekly meetings on Monday with just the group members to determine the weeks deliverables and talk through their requirements. For each week we set up an agenda at the start of the meeting with all the things we would like to talk about. During the meeting one of use will make minutes about the meeting which will be available to all the group members on the google drive. During this meeting we will single out tasks that can be done and add those to a google drive file. This google drive file is a log that has all of the tasks that have been done, are in progress and need to be done. Each task has an assigned group member, a phase, a category, a yes/no if it is finished, a deadline and a finishing date. When a task is finished the finishing date is filled in and the task is set to finished. The group member that just finished his task can now use the log to determine what to do next. This log will also be used as the backlog for the scrum period.

Friday meetings

We will have weekly meetings on Friday with our supervisors. During this meeting we will be talking about the progress that has been made past week and what we plan to do next week. We also talk about design decisions during these meetings. There will also be minutes documenting the important information and decisions of this meeting.

Development period

During the development period we will be having daily meetings, as prescribed by Scrum. These short meetings will follow the Scrum paradigm.

Financing

We do not have a budget provided so our project must use open source software and free to use services. All of the tools and techniques that we use follow this rule.

Reporting to the client

We will use the Friday meetings to report our progress to the client (and supervisors). The deliverables that we set will be sent to the supervisors for review.

Resources

We will be using the different meeting rooms and public studying areas as our working environment. Although not perfect, it will be enough for the short duration of the project. The meeting rooms provide for a good place to hold meetings and work together as a group. The public studying areas are a good place to work alone on a task.

Quality assurance

For quality assurance we use several tools and techniques, to make sure our code is maintainable and is well documented. In this section we will first describe some of the techniques we will use and explain why they are useful for quality assurance. Next we will describe how several tools are providing use useful help and knowledge about our quality.

Techniques

As described before we will use scrum as our software development method, which will give us the opportunity to write tests very early in the process. Due to the fast scrum cycles of one week and splitting the software in small features, we can test each feature in the same scrum cycle. This way we will detect bugs and programming faults very early. We will also detect possible problems in the group process early.

Next to scrum we will make use of Git branches to separate our feature implementations in different branches. When a feature is completely tested and reviewed by another member of the programming team, it will be merged into the staging branch. For this, we will use pull requests on Git, so everyone of us must review the changes before they will be added to the staging branch. After that acceptance testing will be done on the staging branch and when all tests are passed it will be merged into the master branch. This will always assure that we have a stable release available in the master branch.

As last technique we will use unit testing together with acceptance testing with our client and future users of the system and together with user testing. Unit testing will be happing during each scrum cycle, and will be required for each feature to be merged into the staging branch. Acceptance testing will happen in the last two weeks of our project implementation and will give us feedback on which requirements we finished or didn't finish. User testing also will happen in the last two weeks of our project implementation. These tests will give us feedback on the accessibility of our system and if the system interacts in a natural way.

Tools

For quality assurance our main tool will be the Jenkins (Cloudbees) continuous integration testing environment. This environment gives us the opportunity to automatically perform several testing techniques and code analysis during each push to the Github repository. In this environment we will run several tools we will explain in the following paragraphs in more detail.

At first all our Unit test will be run on Jenkins, which will give us fast feedback about the functionality of classes and functions. Bugs can easily be found, and when code is changed later on each test will be run again to make sure it keeps on working.

Next we will run Findbugs to statically analyse our code for small bugs, and wrong naming conventions. This will not make sure there are less bugs in our code, but at least will give a helping hand in finding bugs very easily.

After Findbugs we will run JaCoCo, which will give a really good insight in the data analysis of our code coverage. It will help use find parts of code where more testing needs to be done.

As last we will run Codestyle, to make sure that every programming holds itself to the same conventions. Codestyle will rive a report of the errors in the conventions and need to be checked before pushing to staging, This will make sure our code is more maintainable, because it provides more easy to read and understand code.

B

Appendix B - Assignment

This was the first version of the assignment. After the start of the project, the assignment has changed.

Project description

It is well known that many students come to the University to visit lectures because it provides the opportunity to meet peers and to have discussions also about less academic topics. The meeting place at the University is partly replaced by a virtual meeting place. Students report about their daily activities, opinions, problems using social media. At this moment social media play a limited role in the academic process of teaching and learning. But social media offer great opportunities, especially around the development of MOOCS (Massive Open Online Courses).The basic idea of MOOCS is that students remote in place and time follow the lectures, they can use social media as academic communication media, where students discuss about their study activities, put forward their questions and get help and support from peers. The question is how to find the best matching peers especially peers outside TUDelft. Inspecting all the profiles via Facebook is too time consuming. Supporting matching tools are needed. At this moment there many sites finding the best matching partner, our interest is finding the best matching study partners. A possible option is to extract features from student profiles via FaceBook , take care of preferences and expert knowledge and train classifiers or other matching technologies. Students involved in the peer matching project are supposed to communicate via social media. But at the end of the project there will be a written report. The TUDelft supervisor Leon Rothkrantz is involved in supervising students for many years. Currently he participates in an European project FETCH aimed at the development of new didactical models for distant learning. This gives the current Peer matching project an European dimension and offers cooperation with students from other European Universities.

Company description

The TUDelft is well known, so no additional info is provided. At this moment there is a lot of interest in the development of MOOCS. The Board of the University (Anka Mulder) supports the developments for many reasons. In the framework of The European Life Long Learning Program TUDelft is a partner in the FETCH project (Future Education and Training in Computing). More than 55 European Universities are involved in this project. There is a special Work Package on the use of social media in distant learning. Prof. drs. dr. L.J.M. Rothkrantz is the coordinator of this project. There are possibilities for students to take part in this project and present their work on Educational Conferences organised by the project.

Auxiliary information

The company FeedbackFruits will be represented by Ewoud de Kok (ewoud@feedbackfruits.com +31630496917). The company is cofounded by TUDelft graduates (computer science). The have experience in developing software products according to well known standards. last years many students did an internship at the company He will have the role of case holder or client of the project. An interesting aspect is to study how online educational platforms will change the content and didactical aspects. It is now a fact that many students are online almost the whole day. Regular courses at the University are too static, boring and most important not interactive. It proves that even workgroups and interaction during classes are less effective. It is difficult to measure the effectivity/efficiency and to make effective changes in the way of teaching.Online courses have a broad spectrum of didactical forms. It ranges from video lectures, courses with quizzes and tutorials at regular times, courses modelled as games, and a lot of gimmicks which gives learning a lot of funs Online courses have a lot of social activities (not only knowledge transfer), students will be stimulated and enabled to help each other during hangouts and group meetings via Skype-like interfaces. This enables students to give feedback and discussions about the content, with or without the lecturer. The educational challenge is to integrate for example the use of social media in online courses. This will have a huge impact on the way Universities will offer educational and training material.

Company: TUDelft, Fetch
TU Delft coach: Leon Rothkrantz
TU Delft coach email address: L.J.M.Rothkrantz@tudelft.nl
Company contact: Leon Rothkrantz
Company contact email: l.j.m.rothkrantz@tudelft.nl

C

Appendix C - Research Report

Preface

This document is the research report that has been written for the Peer Matching assignment for the Bachelor Project course. This course is the last project based course of the Bachelor of Science at the TU Delft in the Netherlands. The assignment was issued by Prof. drs. dr. L.J.M. Rothkrantz of the TU Delft. The report provides a summary of the requirements that are set for the system and will provide for a literature study about different possible algorithms to solve the peer matching problem presented in the problem definition.

Introduction

Many students go to an university each year to follow the predefined bachelor and/or master programs, but not everyone has the time and intent to follow an entire program. There is a demand from all sorts of people to be able to follow a course while their main occupation isn't that of a student. For example: a working parent could be very interested in such a course. That is why universities introduced MOOCs.

The basic idea of MOOCs is that students remote in place and time follow the lectures, can participate in a practical assignment and can make an exam. If they score sufficient, they are rewarded with a certificate that states their success in the course.

The concept of the MOOCs is a very new concept that still has many areas to improve upon. One of these areas is the area of practical assignments. Currently, for a Computer Science MOOC at the TU Delft, multiple practical assignments are required. The participants are required to form groups to make these assignments. After the announcement the participants either use mail, Facebook or Twitter to contact other people in order to form the groups.

TUDelft has been a forerunner in the field of Open & Online Education since 2007 and is one of the sustaining members of the OpenCourseWare Consortium and since 2013 charter member of the edX Consortium. In March 2014, the board of the TU Delft approved an innovation program to accelerate the development of open & online education. The goal is to create a Delft Extension School that will bundle all open & online courses for a global population of life long learners.

We would like to improve upon this concept and try to provide a more fitting approach.

Problem definition

In this subsection we will continue on the case provided by the introduction. We will pinpoint the points we would like to improve upon.

MOOCs around the world are growing in size and thus the practical assignments need to be easy to

manage. The growth also means that it is more difficult for each participant to find a possible practical partner due to the large amount of people. It also becomes really difficult to find the correct partner or group members when you don't see each other during courses and haven't even met most of the people from your course.

Most of the time participants of MOOCs are searching group members on social media like Facebook or Twitter. They just try to look through some of the participants names in the course, search for them on-line through social media and check if they are a capable partner or group member. This approach takes a lot of time and most of the time there are too much MOOC participants to go trough all of them.

Next to the fact that this process is very time consuming for every participant, it also leads to a sub optimal solution. This sub optimal solution will be focussed on a group or set of partners which share the same interest and this doesn't say anything if they would form a good group.

We would like to improve upon this process by creating an application that can be used to create practical groups without the participants being required to go search intensively. Participants should be able to easily access the application, fill in some relevant information about themselves and then be recommended possible group members. Our peer matching tool is an add on for the MOOCs. It can be parallel used for every MOOC with group assignments. The teacher can create a course in our tool to represent a MOOC.

The interaction with the system is a pretty straight forward application, but the recommendation of possible group members is more difficult. One of the questions we are trying to answer is: What user information is useful to determine whether two people can be good practical partners? We will also need to figure out a good way to recommend a practical partner, but still give the possibility to reject such a request.

We have two problems. First, how can we find appropriate matches for people? And second, how can we create optimal groups?

These questions and difficulties will be answered in this report.

Introduction to requirements

The following sections will describe what requirements are set for the product. This section will define four global requirements who cover the entire project. These requirements are then split up in the next two sections, the first being the requirements for the basic system and the second being the requirements for the peer matching algorithm. We choose to split these requirements up, because of the literature study required to further specify the requirements. In these sections, the global requirements are split up in smaller requirements. It is easier to verify these smaller requirements for their completion and these requirements are more transformable to functions of the system. We also assigned these requirements into their appropriate MoSCoW [5] method category.

Actors of the system

In this subsection we will clarify the different actors used in the system and throughout this report.

MOOC participant

Due to the nature of the MOOCs, there are no requirements set for the participants. Every person with an interest in the given course can enlist for the MOOC and should thus be able to register with our system.

MOOC teacher/MOOC administrator

The MOOC is organised by a professor of an university. This professor will need to register with the system and create a MOOC.

Groups of participants

This actor consist of two or more MOOC participants who accepted each other to their group. These groups can be of variable size, set by the MOOC organiser.

Global Requirements

To provide an overview of what the software must be capable of, we define four global requirements for the system. By defining these four global requirements we provide a foundation of what the system must be able to do.

- MOOC teachers must be able to create, manage and edit courses. Participants of their MOOC are able to register to this course.
- MOOC students must be able to register and deregister for a course.
- The system must be capable to use personal data provided by the user to find a suggestion of an appropriate practical partner or group member. It must be possible to adjust the definition (parameters) of the appropriate practical partner or group member, and search trough several suggestions.

Requirements

This section will mention the different requirements that involve the basic system which the actors interact with. The section starts with mentioning the possible types of user information the application can use and what requirements are associated with that. Followed up by what actions are possible within the system. After that the kind of system is limited by the requirements listed. Lastly we will list the requirements set for the algorithm. Each section will use the MoSCoW model to divide the requirements in their appropriate categories.

User information

Must have:

- Users must have a viewable profile that can be filled with personal information.
- Users must be able to enter the following registration information during registration: Full name, Email address, Password and Language.
- Users must fill in their academic performances. Which knowledge, skills and completed courses does the user have?

Should have:

- Users should be able to enter the basic user information, like: country, age, short description, skills, education, a picture and personality traits.

Could have:

- Users could be able to use a personality test to determine their personality traits.
- Users could be able to fill in their personality traits.
- Users could be able to enter the following extended user information: preferred group role, role of the other group members, etcetera.
- Users could be able to enter a predefined questionnaires.

Would have:

Usage of the system

Actor 1: MOOC participant

Must have:

- The a potential participant (anyone who visits the site) must be able to view the different MOOCs without registering or logging in.
- The participant must be able to inform him/herself about the usage of the application without registering or logging in.
- The participant must be able to register to the application.
- The participant must be able to log in to the application.
- The participant must be able to register to the MOOCs which they were enlisted for (known to the administrator).
- The participant must be able to view a possible practical partner.
- The participant must be able to communicate with the possible practical partner to verify the matching.
- The participant must be able to accept or decline the practical partner into his practical group.
- The participant must be able to communicate within the group.

Should have:

- The participant should be able to ask a currently existing group if he can join them.
- The participant should be able to use their LinkedIn profile to log in to the application.
- In case the participant already has an account, the participant should be able to add the course to his profile using the teachers url.
- The participant should be able to receive an invitation from a group.
- The participant should be able to accept the invitation from a group.

Could have:

- The participant could be able to set his preferences for his possible partner.
- The participant could be able to search for other participants using information that is available in the profile.

Would have:

Actor 2: MOOC teacher/MOOC administrator

Must have:

- The administrator must be able to create a course and add this to his account.
- The administrator must be able to open and close the participant registration for a course.
- The administrator must be able to manage the list of participants and groups of participants.
- The administrator must be able to create an url that he can use to invite the participants.

Should have:

- The administrator should be able to set a deadline for a practical.
- The administrator should be able to communicate with every participant.
- The administrator should be able to communicate with the groups.

Could have:

- The administrator should be able to grade the group for its practical.
- The administrator could be able to provide for his own criteria for each of his course.

- The administrator could be able to add different assignments to the application.
- The administrator could be able to view what groups have finished what assignments.
- The administrator could be able to grade the different assignments.
- The administrator could be able to provide feedback for the different assignments.

Would have:

Actor 3: Groups of participants

This actor consist of two or more course participants who accepted each other to their group.

Must have:

- The groups must be able to communicate with each other using the application.

Should have:

- A group should be able to view a possible group member.
- A group should be able to invite a possible group member to join their group.
- A group should be able to accept an request from a participant to join their group.

Could have:

- The group could be able to select their preferences for the new group member.
- The participants could be able to share files to other group members using this application.

Would have:

- The participants would be able upload their source code to the application for verification.

Kind of system

Must have:

- The application must be easily accessible to both participants and course organizers.
- The system must provide account management functionality.
- The system must use a database to store the different data put in by the user.

Should have:

Could have:

- The system could be able to be combined with practical systems already used by MOOC organizers.

Would have:

Peer matching

Must have:

- The algorithm must be able to use a list of users as input.
- The algorithm must deal with continuous variation in amount of users.
- The algorithm must be able to return one or more suggestions of practical partners to the user given a list of rejected users.
- The algorithm must be able to return one or more suggestions of possible groups the participant could join.

Should have:

- The algorithm uses basic user provided data (e.g. education, preferences) to build up a profile for a course participant.

- The algorithm can calculate whether two people will be good practical partners using the basic user provided data.
- The algorithm can return a possible practical partners using the basic user provided data.

Could have:

- The algorithm uses advanced user provided data (e.g. asked questions, programming tests) to build up an extended profile for a course participant.
- The algorithm can calculate whether two people will be good practical partners using the advanced user provided data.
- The algorithm can return a possible practical partner using the basic user provided data.
- The algorithm could be able to respond to the preferences of groups.
- The algorithm could be able to respond to the preferences of a participant.

Would have:

- The algorithm would have to be able to use information gathered from rejections to improve the matching.

Design choices

In this section we will be describing our choices for the design. For each choice we will be arguing why we have chosen for this. We will argue using scientific literature or the requirements. We will first describe the choices we have made concerning the kind of system. Secondly we will describe the choices we have made concerning the framework that we will be using. Following this, we will be describing the choices we have made concerning the algorithm.

Global system design

In the global system design we will be describing a first concept of the system. In this concept we will describe the different actions, that different actors will need to be able to perform.

The first thing that has to be done is for the teacher to create an assignment in the application. It must be possible for the teacher to open and close this assignment. While creating, the teacher has to define some properties of the assignment. When the assignment is created successfully, a link is generated that can be sent to the students.

When the students open the link, they have a choice between logging in or registering. For logging in and registering, the student could use LinkedIn, so a lot of data is gathered automatically. We can also look for LinkedIn alternatives, like Yammer and Ututi, which are non-commercial. After filling in the basic information, the student must provide information about his personality. This can be done by means of an questionnaire.

After logging in, the assignment can be added to the profile of the student. After adding the assignment to the profile, some recommendations are provided. The student can form groups by means of this recommendations.

Web application

Our main decision about choosing to make a Web application was relatively simple. This is due to the fact that MOOCs itself are also Web applications, which are available for everyone. Because our application is specifically designed for MOOCs it should also be widely available for all the MOOC participants. One of the best ways to do this is by developing a web application.

Framework

Because Web applications are getting more and more popular, a lot of frameworks are appearing which we could use for our Peer Matching system. Choosing the right framework is therefor really difficult, because they are hard to compare and not a good way to say which is better. We therefor listed a couple of frameworks which we had experience with, to make sure we have a lot of knowledge available in our programming team.

We first started looking at the PHP based Kohana Framework[6].

The Java based Spring Framework[7] was also a possible solution. Spring Web Flow builds on Spring MVC and allows implementing the ”flows” of a web application. The programming team doesn't have that much experience with the Spring Framework, but because they are all well known with the Java programming language it shouldn't be a problem. It has a lot of useful features like an ORM (Object-relational mapping), transaction management and a MVC setup. This makes the Spring Framework a good choice.

Next we started looking at the Python based Django Framework[8]. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. But because the programming team didn't have that much experience with both Python and Django, we chose to avoid learning a totally new programming language and framework.

The last framework we looked at was the Play Framework[9], based on the programming language Java and Scala. Play is a high-productivity Java and Scala web application framework that integrates the components and APIs you need for modern web application development. The Play Framework looks a lot like the Spring Framework, it includes the same features and is also MVC based.

To conclude, we chose the Play Framework to work with. All of the team members already had experience with programming in Java. One of the team members also had worked with the Play Framework before, so we had some experience with this framework in our group.

Literature survey recommender systems

Using the requirements stated in the previous section we see that the requirements all state the need for the handling of a specific set of input and outputting the needed information. The requirements do not state any limitations about the inner workings of the algorithm. Some questions remain:

- How does the algorithm determine a possible practical partner?
- How does the algorithm make a difference between two possible partners and determine which one is a better fit?
- How does the algorithm make use the extra information provided if the could haves are achieved?
- How does it make sure the partner that is recommended hasn't already been suggested?

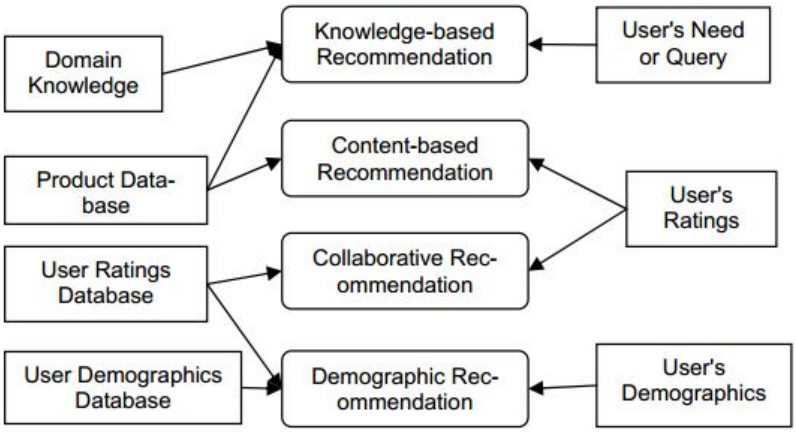
Due to the unique requirements set to how the system works, we need to look for all kinds of different algorithms that could help us solve our problem. Because of the unique requirements, these techniques will probably not solve the entire problem, but we can possibly use parts of these algorithms to formulate our own solution.

We have done a literature survey into recommender systems since these kind of systems are closely related to what we want to achieve.

As can be read in these papers [10],[11], there are roughly five kinds of algorithmic recommendation approaches:

- Demographic filtering [11]
- Collaborative filtering [11][10]
- Content-based filtering [11]
- Knowledge-based filtering [11][12]
- Hybrid filtering [11]

The following image supports the existence of these five classes. The image shows how the different classes use different kinds of resources. Next to these recommendations we added a subsection for the mentioning of other possibilities that we encountered while researching the subject. We will now first discuss each of the techniques and what aspects are usable and what aspects are not.



Demographic filtering

In the paper Privacy-preserving demographic filtering[13] demographic filtering is explained. Recommender systems based on demographic filtering aim at categorising users based on their demographic information and recommend services accordingly. More precisely, demographic information is used to identify the types of users that like similar services.

Demographic filtering can be used by any recommender system that offers services by using data on individual users. The key element of demographic filtering is that it creates categories of users having similar demographic characteristics, and tracks the aggregate behaviour or preferences of users within these categories. Recommendations for a new user are issued by first finding to which category he belongs and then by applying the aggregate preferences of previous users in that category.

In the paper A framework for collaborative, content-based and demographic filtering[14] an example is provide. For example, Table C.1 shows information on the age, gender, education, etc. of people that rated a restaurant together with their rating of the restaurant. One might expect to learn the type of person that likes a certain restaurant.

	gender	age	area code	education	employed	Dolce
Karen	F	15	714	HS	F	+
Lynn	F	17	714	HS	F	-
Chris	M	35	714	C	T	+
Mike	F	40	714	C	T	-
Jill	F	10	714	E	F	?

Table C.1: *Demographic filtering example*

We could use data like these to recommend similar groups for people with the same demographic background.

Collaborative filtering

The book Adaptive Web [11] describes what collaborative filtering techniques are. He says that these techniques generally involve matching the ratings of a current user for objects with those of similar users. In this way it produces recommendations for objects that the users has not yet rated or have not been seen by the active user. Traditionally, the primary technique used to implement a collaborative filtering is the use of k-Nearest-Neighbor (kNN) algorithm. This algorithm compares a target user's

profile with the historical profiles of other users in order to find the top k users who have similar tastes or interests.

Examples of collaborative filtering algorithms are found in the on-line dating branch. An example that provides some insightful information about the subject is OKCupid. The video ,that can be found on the website[15], describes the workings of the OKCupid algorithm. The OKCupid algorithm uses the geometric mean. This is a type of mean or average, which indicates the central tendency or typical value of a set of numbers. OKCupid's algorithm is an algorithm that uses questions and preferences to determine whether two people would make a good pair. Instead of the questions they ask, we could fill in our own questions or values. The geometric mean would still give an overall score for the combining of two people.

Content-based filtering

The book Adaptive Web [11] also describes what content-based filtering techniques are. He describes content-based filtering systems as systems where in a user profile represents the content description of items in which that user has previously expressed interest. The content description of items are represented by a set of features or attributes that characterise that item. The recommendation in these kind of systems usually consist of comparing unseen or unrated items with the user profile. Items that overlap with the interests of the user are recommended.

The paper Using Content-Based Filtering for Recommendation [**?**] describes an example of a recommender system that uses user information to determine what to recommend.
In this paper it becomes clear that a built up user profile can give a much better recommendation than a starting profile. Although this can work for many applications, this means that the user needs to build up a training set. The first couple of recommendations could be possibly bad and this is not in the users interest. The user wants the best recommendation from the start. Although it might look like content-based filtering can not be used for our domain, we would like to keep them in our thought as a possible way to improve upon the algorithm. By using the user's history we can improve the recommendations by listening more to the users preferences.

Knowledge-based filtering

In the book Adaptive Web, they also mentioned knowledge-based filtering as a class of recommendation techniques. This is supported by a paper called Knowledge-based recommender systems [12]. In this paper they describe it as a recommender system that uses knowledge about users and products to generate a recommendation. It will use this knowledge to have a different approach to the generation of a recommendation and the user. Knowledge-based recommendation systems do not have the problem of a so called cold-start problem. The cold-start problem is the problem when a learning-based system does not have enough information to provide for a proper recommendation. Learning-based systems are systems that learn from each user or product added to the system. With each step they learn more about the users that are in the system and their preferences. This can occur when a new user is registered to the system, a new product or when there are only a few current users. This is a big advantage and is something we should keep in mind when designing our algorithm

One of the downsides of knowledge-based filtering is that it can't learn. It will always be limited to the knowledge it has. An expert can improve the knowledge the system has, but it will not do so autonomous like a learning-based system.

Hybrid filtering

Adaptive web sites offer recommendations to the user by using a well studied technique. This technique can either use collaborative, knowledge-based and content-based recommendation, but it can also use a combination of these techniques. These kind of systems are called hybrid filtering techniques. Each of these techniques has its own strength and weaknesses and by combining these researchers try to improve the performance of the techniques. The resulting systems are called hybrid recommender systems and are defined as the combination of two or more of the different recommendation techniques.

A variety of techniques have been proposed as the basis for hybrid recommender systems: collaborative, content-based, knowledge-based and demographic techniques.

The learning-based techniques (collaborative, content-based and demographic) suffer from the cold-start problem. Most hybrid recommender systems are combining techniques to deal with the cold-start problem. This is something we need to keep in mind when designing our algorithm.

The converse of this problem is the stability versus plasticity problem. Once a user's profile has been established in the system, it is difficult to change one's preferences. Many adaptive systems make the older ratings have less influence and thus prevent the unresponsiveness of the system.

We can use a combination of different techniques, and thus have a hybrid filtering, to have a resulting algorithm that perfectly adheres to the requirements.

Other possibilities

Recommender system all try to use data that has been collected to improve their recommendations, but there are other approaches to the matching problem. A great example of an user based approach is the Tinder mobile-app. This application allows users to get in contact with other people. They only provide for a platform where the user can view people who are near you and adhere to some basic requirements (E.g. age limit and amount of kilometres away). The user is shown a small profile of a potential match and let the user decide whether it is a go or not based on that info.

We would like our user to decide upon skills that are not very applicable to the Tinder format. Another downside of the format is that it does not provide for a way to influence the outcomes of the groups. For example, the administrator should be able to set values for the different skills that are advised for each group. Lastly, we would like to narrow down the possible people an user will see to the ones that could be a positive addition to his group. Although we can not use the entire format, we can use the accepting and/or rejecting of a possible group member. It is also a good idea to let the possible group members talk to each other before they have to decide to join one's group.

Conclusion

Each of these techniques are very interesting for having a good recommendation algorithm, but we do not have the time with this project to expand so vastly on the subject. We will thus have to simplify the problem to something where for we can implement a basic recommendation algorithm and that still adheres to the requirements.
We concluded that the techniques weren't appropriate for the requirements that were set. There are three reasons why the techniques mentioned above aren't appropriate for our solution. First is the fact that the requirements state the need for the administrator to be able to influence the group characteristics. Secondly the requirements state the possibility that the participants could be able to influence the recommendation. Lastly not every method is able to easily provide for groups of different sizes. The techniques mentioned above do not provide for the flexibility that we require. We are best suited with an algorithm that can easily be implemented with a basic result and can easily be extended with extra functions.

The Algorithm

Following up on the literature survey we need to decide what aspects of what techniques can be used to construct an algorithm and to determine how it will be used. That is why we took another look at the requirements and how the different actors should interact with each other. During this evaluation we thought up a algorithm that combines multiple known techniques that adheres to the requirements.

Aspects

We decided upon the following aspects:

- The algorithm uses the idea of the social interaction platform that is based off platform that the Tinder app uses. This part of the Tinder app provides for all the interaction possibilities needed for our system. We only need to extend upon the creation of groups.
- The algorithm will use a knowledge-based approach to deal with the various kinds of user information. It is important to use the proper scales for each information type, else the algorithm can not properly determine whether two people would be good group members.
- We use the geometric mean that is used by OKCupid's algorithm to calculate the differences and similarities between users. By using this we can accurately determine how much two people defer from each other. Using this data, we can determine who would and who wouldn't be good group members.
- We would like to make the algorithm robust for the cold-start problem, the usage of a knowledge-based approach should help with this.

What does the algorithm do?

The algorithm will be used to recommend a possible group member to a person. It will do so by analysing information the user has filled in. This information will be compared to other users to determine their similarities and differences. The algorithm will then calculate how good of a group member others users are. While computing this it will track the top partners to eventually return these to the user. It will do so by combining the two group members and compute a resulting value for each information property. Whether two user form a good group can then be determined by comparing the resulting values with predefined values. These predefined numbers can either be set by the administrator or later adjusted by the searching participant.

Formal description of the algorithm There are two cases in which the algorithm should be able to recommend a new group member:
A single participant searches for a first group member Given m properties and n participants currently enlisted to the course the algorithm will search for a possible practical partner for participant X. The algorithm determines for each possible practical partner how the creation of a group of that pair would bring them closer to the predefined numbers. It will do this by taking the average for each property and taking the difference between this average and the predefined number. It will then sum these differences and take the m root of this number. This will result in an number that represents the overall difference between the group of the two participants and the predefined numbers. A group with a low overall difference means that the two participants form a group that as closely meet the predefined numbers as possible. During this calculation the algorithm keeps track of the top possible practical partners which it will return as a result to the participant.

A group searches for a new group member The second case is similar to the first, the calculation is completely the same. The only difference is that you do not start with a participant X, but with a group of participants. The properties of these participants are already averaged per property. This results in a number per property which can be used by the algorithm mentioned above.

Possible improvements of the algorithm

- The user can use extra properties (which the administrator can't define) to further specify their groups.
- The user can adjust the predefined numbers (the predefined numbers of the administrator are not required).
- The algorithm uses the group size to ensure that there are more versatile people in a group.
- The algorithm can incorporate a content and/or collaborative-based recommendation.

D

Appendix D - User test Google form

Survey user test

This survey will question the user about their involvement in MOOC's and the usage of our system. The system is meant to be used

* Required

1. **What is your name? ***

   .....

2. **What is your age? ***

   .....

3. **Do you know what a MOOC is? ***
   Mark only one oval.

   - ◯ Yes
   - ◯ No

4. **What is your occupation? ***
   Mark only one oval.

   - ◯ Student *Skip to question 7.*
   - ◯ Teacher *Skip to question 5.*
   - ◯ Other: ..... *Stop filling out this form.*

Teacher survey

5. **Do you teach a MOOC? ***
   Mark only one oval.

   - ◯ Yes
   - ◯ No

6. **Would a system to arrange your practical assigments interest you? ***
   Mark only one oval.

   - ◯ Yes, alot.
   - ◯ Moderate
   - ◯ No

Stop filling out this form.

Student survey

7. **What do you study?** *

8. **What is your starting year?** *

Skip to question 9.

Register/Login Test

Register & login

Please register to the website and after completing this, log into the system.

After you are logged in, please fill in your skills.

9. **Did you successfully register?** *
Mark only one oval.
- Yes
- No

10. **Did you successfully login?** *
Mark only one oval.
- Yes
- No

11. **Did you successfully enter your skills?** *
Mark only one oval.
- Yes
- No

12. **Any thoughts about possible improvements?**

Skip to question 13.

Student test

Register to practical

Please browse to this url (http://localhost:9000/practical/5/subscribe?secret=abc) in the appropriate browser to register to the practical. After this, view the practical overview and browse to the practical which you registered to.

13. **Did you successfully register to the practical?** *
Mark only one oval.

- ◯ Yes
- ◯ No

14. **Did you successfully view the practical overview?** *
Mark only one oval.

- ◯ Yes
- ◯ No

15. **Did you successfully view the practical which you registered to?** *
Mark only one oval.

- ◯ Yes
- ◯ No

16. **Any thoughts about possible improvements?**

.....

.....

.....

.....

.....

17. **What do you think about the recommendations?** *
Mark only one oval.

- ◯ Very useful
- ◯ Moderately useful
- ◯ Unuseful

18. **Any comments on the recommendations?**

.....

.....

.....

.....

.....

Invite other student to join your practical group

Viewing the practical that you have registered to, you can invite another student to join your group. Please do so and view the invite that you have created. Also look at the practical that you registered to, has anything changed?

19. **Did you successfully invited another student?** *

Mark only one oval.

- Yes
- No

20. **Any thoughts about possible improvements?**

.....

.....

.....

.....

.....

Skip to question 21.

Final survey

21. **What did you think about the system in general?**

.....

.....

.....

.....

.....

22. **What did you think about the usage of recommendations to suggest possible groupmembers?**

.....

.....

.....

.....

.....

E

Appendix E - User test data

Timestamp	What is your age?	What is your occupation?	What do you study?	Did you successfully register?	Did you successfully login?	Any thoughts about possible improvements?	Did you successfully register to the practical?	Did you successfully view the practical overview?	Did you successfully view the practical which you registered to?	Any thoughts about possible improvements?	What do you think about the recommendations?	Did you successfully invited another student?	Any thoughts about possible improvements?
6-12-2014 12:18:51	21	Student	applied mathematics and applied physics	Yes	Yes		Yes	Yes	Yes	Explanation of the recommended skills.	Moderately useful	Yes	
6-12-2014 12:29:32	26	Student	Computer Science	No	Yes		Yes	Yes	Yes		Moderately useful	Yes	
6-12-2014 12:55:45	22	Student	Computer Science	Yes	Yes	"Documenting" Is a weird skill	Yes	Yes	No	No idea what the purpose of this page is supposed to be. No explanation	Unuseful	Yes	See other person´s name and skills without going to a different page
6-12-2014 16:03:03	20	Student	Computer Science	Yes	Yes		Yes	Yes	Yes		Moderately useful	Yes	Link to the user's profile on invite overview
6-12-2014 16:59:18	19	Student	Bachelor Computer Science	Yes	Yes	Require fields tooltips sometimes align wrong. Email activation link was not clickable.	Yes	Yes	Yes	A notice of my succesful registration would be nice.	Very useful	Yes	
6-13-2014 10:30:27	21	Student	Technische Informatica	Yes	Yes	De 10 schaal vind ik niet zo fijn om je skills aan te geven; je geeft jezelf niet zo snel een onvoldoende, dus 1-5 zullen bijna nooit worden ingevuld. 1-4 of 1-5 is beter, een 2/3 zou ik mezelf eerder geven bij een 5 schaal.	Yes	Yes	Yes	Meer uitleg, ik krijg nu een pagina met weinig informatie; Ik weet niet wat ik er mee kan en waarom er linkjes staan naar andere practical groups. Daarnaast weet ik ook niet wat ik met invitations kan.	Unuseful	Yes	Ah, nu snap ik het misschien. Een practical group is ook een student? Het lijkt mij logischer om studenten uit te nodigen ipv hele groepen? Daarnaast laten zien wat de "skills" zijn ipv van de groepsnamen lijkt mij een stuk beter: dan kun je beter je invites plaatsen. Daarnaast lijkt mij een "communication" skill ook heel belangrijk, iemand die twee 10'en heeft voor programming en documenting kan alsnog super kut zijn om mee samen te werken

6-13-2014 10:50:49	21 Student	Technische Informatica	Yes	Yes	niet op dit moment	Yes	Yes	Yes	Het is niet echt duidelijk wat ik hier moet doen. Misschien dat er nog wat meer uitleg kan komen wat de bedoeling is.	Moderately useful	Yes	Ik zit in een groep, maar ik kan ook mensen uit andere groepen uitnodigen. Moet het niet zo zijn dat je eerst nog niet in een groep zit en dat je dan een groep kan maken door iemand te inviten?
6-13-2014 11:37:03	21 Student	Technische Informatica	Yes	Yes		Yes	Yes	Yes		Moderately useful	Yes	
6-13-2014 11:55:50	22 Student	Technische Informatica	Yes	Yes	The slider value is not correctly updated	Yes	Yes	Yes	Show names instead of practical groups	Very useful	Yes	No
6-13-2014 12:08:57	24 Student	CS	Yes	Yes	Sent an email with more text, this way people become more involved. Also give more incentive to enter your skills.	Yes	Yes	Yes	Who is User? I thought I would be subscribed to the practical. I answered yes on the questions because I think I did so. I have no clue on what I am exactly doing in this system.	Unuseful	Yes	I have sent an invite to practical group #14, but in my overview it all of a sudden says that I invited Peter de Jong. This feels strange. It also does not feel logical to invite groups to join my group.
6-13-2014 12:20:52	19 Student	Technische Informatica	Yes	Yes		Yes	Yes	Yes		Very useful	Yes	Explain the goal of the program before giving the survey
6-13-2014 12:33:18	21 Student	Mathemathics	Yes	Yes		Yes	Yes	Yes	Ik snap er niks van.	Unuseful	Yes	

What did you think about the system in general?	What did you think about the usage of recommendations to suggest possible groupmembers?	Would a system to arrange your practical assigments interest you?	Do you know what a MOOC is?	What is your starting year?	Any comments on the recommendations?	Did you successfully enter your skills?
works oke, but not sure what the purpose is.	good idea		No	2010		Yes
			Yes	2006		Yes
I do not know what its purpose is and therefore do not consider it to be useful	That´s fine, I guess....?		Yes	2009	What is it recommending me?	Yes
			No	2011		Yes
Seems easy to use and performs just fine. Layout may be improved for deployment.	Great addition to courses! I'm not quite sure whether or not I like the idea of using recommendations to create the overall optimal group composition. I'd prefer a group with members which have equal skills.		No	2011	Seems to work great!	Yes
Nog erg minimaal, meer uitleg en begeleiding lijkt mij essentieel. Verder handig dat je op deze manier projectgroepjes kunt samenstellen.	Dat lijkt mij erg handig. Kun je in dit systeem ook elkaar beoordelen? Dat lijkt mij ideaal, dan kun je bij latere projecten een betere keuze maken, omdat mensen die zichzelf beoordelen niet altijd helemaal eerlijk hoeven te zijn!		Yes	2011	Wat kan ik ermee?	Yes

Ik denk dat het een handig systeem is voor mensen die elkaar nog niet goed kennen.	Ik denk dat het op de universiteit in de sjaars jaren niet veel gebruikt gaat worden, omdat ze daar toch al vrienden hebben die het graag met elkaar willen doen. Maar misschien voor instanties waar veel mensen komen die elkaar niet kennen kan het goed gebruikt worden	Yes	2011	Ze zeggen wel iets, maar ik weet of de reccommendations er zijn voor mij, of om een persoon te zoeken die die recommondations heeft. De group recommendations zijn wel handig, maar ik zie niet goed waar ze op gebaseerd zijn. waarschijnlijk om tot een zo dicht mogelijk bij de cijfer recommendation te komen die erboven staat.	Yes
Seems handy. I would pay a little bit more attention to the GUI. There is a lot of information on a single page (e.g. the practical page) and it wasn't directly clear where to look for what.					
Finally, I saw some average grade being 4.666666667. Not so fancy.	Cool that you provide recommendations.	Yes	2011		Yes
Nice layout. System seemed to be working pretty well.	This was useful. With these recommendations you did not have to scan through all the people.	No	2011	No	Yes
View my previous comments.	?	No	2009	An explanation on what exactly we are doing misses altogether. What do the values mean and what should I do with it?	Yes
		No	2012		Yes
Het zou allemaal wat duidelijker mogen zijn		No	2011	Wat wordt er met de recommendations bedoeld? Zijn dit mijn targets of zo? Hoe moet ik die bereiken? Snap er vrij weinig van.	Yes

F

Appendix F - SIG feedback (Dutch)

De code van het systeem scoort bijna 4 sterren op ons onderhoudbaarheidsmodel, wat betekent dat de code bovengemiddeld onderhoudbaar is. De hoogste score is niet behaald door een lagere score voor Duplicatie, Module Coupling en Component Independence.

Voor Duplicatie wordt er gekeken naar het percentage van de code welke redundant is, oftewel de code die meerdere keren in het systeem voorkomt en in principe verwijderd zou kunnen worden. Vanuit het oogpunt van onderhoudbaarheid is het wenselijk om een laag percentage redundantie te hebben omdat aanpassingen aan deze stukken code doorgaans op meerdere plaatsen moet gebeuren. In dit systeem is er bijvoorbeeld duplicatie te vinden binnen 'view.scala.html', maar ook tussen de 'ProfileForm'- en 'RegisterForm'-classen. Het is aan te raden om dit soort duplicaten op te sporen en te verwijderen.

Voor Module Coupling wordt er gekeken naar het percentage van de code wat relatief vaak wordt aangeroepen. Normaal gesproken zorgt code die vaak aangeroepen wordt voor een minder stabiel systeem omdat veranderingen binnen dit type code kan leiden tot aanpassingen op veel verschillende plaatsen. Wat hier opvalt is dat de drie grootste classen ('User', 'Invite' en 'Practical') gezamenlijk 30% van de Java code bevatten. Alhoewel het commentaar vermeld dat bijvoorbeeld 'User' een 'user representation of the database' is, bevat deze class naast een representatie van eigenschappen ook configuratie en functionaliteit voor het versturen van emails. Om zowel de grootte als het aantal aanroepen te verminderen, zouden deze functionaliteiten gescheiden kunnen worden. Dit leidt er toe dat de afzonderlijke functionaliteiten makkelijker te begrijpen, te testen en daardoor eenvoudiger te onderhouden worden.

Voor Component Independence wordt er gekeken naar de hoeveelheid code die alleen intern binnen een component wordt gebruikt, oftewel de hoeveelheid code die niet aangeroepen wordt vanuit andere componenten. Hoe hoger het percentage code welke vanuit andere componenten wordt aangeroepen, des te groter de kans dat aanpassingen in een component propageren naar andere componenten, wat invloed kan hebben op toekomstige productiviteit. In dit geval valt met name op dat er meerdere cyclische afhankelijkheden zijn tussen componenten, bijvoorbeeld tussen 'models' en 'helpers', maar ook tussen 'helpers' en 'controllers'. Het lijkt er dan ook op dat het niet duidelijk is welk component verantwoordelijk moet zijn voor welke functionaliteit. Om het systeem ook in de toekomst onderhoudbaar te houden, is het aan te raden om dit duidelijker te maken.

Over het algemeen scoort de code bovengemiddeld, hopelijk lukt het om dit niveau te behouden tijdens de rest van de ontwikkelfase. De aanwezigheid van test-code is in ieder geval veelbelovend, hopelijk zal het volume van de test-code ook groeien op het moment dat er nieuwe functionaliteit toegevoegd wordt.

Bibliography

[1] Rothkrantz, *Survey of the role of social media in e-learning,* (2014).

[2] Coley, *Moscow prioritisation method,* http://www.coleyconsulting.co.uk/moscow.htm .

[3] PlayFramework, *The mvc application model,* http://www.playframework.com/documentation/1.0/main .

[4] C. S. department TU Delft, *Research mission of the computer science department,* http://cs.tudelft.nl/ .

[5] J. Highsmith and A. Cockburn, *Agile software development: The business of innovation,* Computer **34**, 120 (2001).

[6] K. Framework, *Kohana: The swift php framework,* http://kohanaframework.org/ .

[7] *Spring,* http://spring.io/ .

[8] *The web framework for perfectionists with deadlines | django,* https://www.djangoproject.com/ .

[9] *Play framework - build modern & scalable web apps with java and scala,* http://www.playframework.com/ .

[10] J. S. Breese, D. Heckerman, and C. Kadie, *Empirical analysis of predictive algorithms for collaborative filtering, Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, UAI'98, 43 (1998).

[11] P. Brusilovsky and E. Millán, *The Adaptive Web* (Springer, 2007).

[12] R. Burke, *Knowledge-based recommender systems,* Encyclopedia of library and information systems **69**, 175 (2000).

[13] E. Aimeur, G. Brassard, J. M. Fernandez, and F. Onana, *Privacy-preserving demographic filtering,* in *Proceedings of the 2006 ACM symposium on Applied computing* (ACM, 2006) pp. 872–878.

[14] M. J. Pazzani, *A framework for collaborative, content-based and demographic filtering,* Artificial Intelligence Review **13**, 393 (1999).

[15] *Ted talk about the okcupid match making algorithm,* "http://ed.ted.com/lessons/inside-okcupid-the-math-of-online-dating-christian-rudder" .