Tournament Planner

Blal Khwaja Khoja Christian Dannesboe
Frederik Børsting Lund Karrar Sattar Al-Sami
Mark Kloch Haurum Sirojan Sivakumar Søren Lyng

21/12/2015

AALBORG UNIVERSITET
STUDENTERRAPPORT

Institut for Datalogi
CASSIOPEIA
Selma Lagerlöfsvej 300
DK-9220 Aalborg Ø.

TITEL:
Tournament Planner

PROJEKTPERIODE: **09/15-12/15**

PROJEKTGRUPPE: DS313E15

GRUPPEMEDLEMMER:

Blal Khwaja Khoja

Christian Dannesboe

Frederik Børsting Lund

Karrar Sattar Al-Sami

Mark Kloch Haurum

Sirojan Sivakumar

Søren Lyng

SEMESTER: 3

VEJLEDER:
Srinivasa Raghavendra Bhuvan Gummidi

ANTAL KOPIER: 2

ANTAL SIDER: 137

SYNOPSIS

This report describes the third semester project about the development process of an application that can plan tournaments for football clubs. By interviewing tournament planners, it was determined that they spent a lot of time planning the tournaments, and an automated system could save them a lot of time.
The algorithm calculates the match schedule for the whole tournament after pools, teams, divisions and fields have been set or imported to the system.
The system is web-based using Bootstrap, HTML, CSS and AngularJS for the front end, and the back end is created with the C# programming language.

Preface

This report is written for the P3 project on 3. semester for Computer Science (Datalogi) and Software Engineering (Software) students who are attending Aalborg University. The project started on the 4th of September 2015 and have its due date on the 21st of December 2015.

The purpose of the project is to obtain knowledge about developing applications solutions for realistic problems and thereby gaining experience with developing bigger systems, division of work and quality control by testing the developed system. The report are directed to people with interests in these subjects and a basic understanding of the programming languages C#.

The report are written by project project group ds313e15, but the project group couldn't have done it by themselves. Therefore project group ds313e15 would like to thank our supervisor Srinivasa Raghavendra Bhuvan Gummidi for his much appreciated help during the project. Also we would like to thank our informants Magnus Nielsen, Jørgen Trankjær and the persons who participated in our user acceptance tests. At last we would like to thank Piratliga.dk who have helped with the output format for the Excel Sheets.

Reading guide

The report has followed the design and analysis patterns from the textbook OOA&D "Objektorienteret Analyse og Design". [1]

For sources in the report, the Vancouver citation method have been used. Number references are used for the Vancouver method, so the first reference will be (1) and will increment for each new reference in the report. The references will be shown in the form: [Reference number] Author name, Article name, Url for websites, when the website was accessed.

Method

For developing and design this project, two different methods can be chosen, the traditional waterfall method and the agile method. In this project the traditional waterfall approach[2] have been used, the waterfall method are shown in figure 1.

In the first phase of the waterfall method, the requirements for the project will be gathered. This happens through communication with the

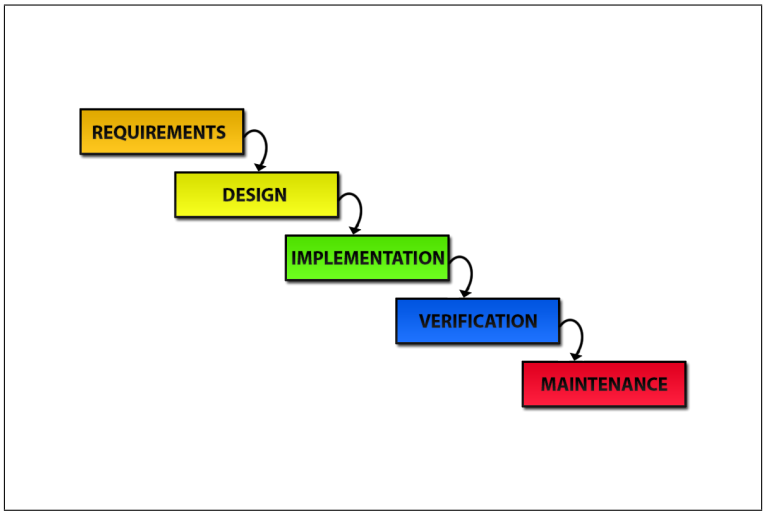


Figure 1: Waterfall Model

user, which could be with interviews or questionnaires.

The second phase is design. In this phase the systems architecture is designed from the before gathered requirements.

The next phase is the implementation. This is where the actual code is written, and is written from the requirements and system architecture.

This phase is the verification phase. This phase is to make sure that the the system lives up to the users expectations. In this project this is done with a user acceptance test.

The last phase is maintenance. The user have started using the application. The errors found in this phase should be fixed, and improvements can be made during this phase. This phase of the waterfall model will not be account for in this project, because of lack of time.

Contents

List of Figures

List of Tables

1 Introduction

In Denmark, the most popular sport is football [3], where thousands upon thousands compete in some of the various tournaments. The tournaments vary in sizes, from a few teams signed up, to hundreds of teams. Many of these tournaments are targeted for the youth football, which usually is for children under 7 years old to teenagers under 17 years old. One of the tournaments targeting youth football is Vildbjerg Cup, with more than 700 teams, and 10,000 players is one of the larger tournaments in Denmark [4].

Whether it's a small tournament or a big tournament, there is a lot of planning and organisation required for the tournament to happen. Usually the tournaments are planned by a club, which means that it's often run by many volunteers. Besides planning the accommodations for the teams and the many meals, the task of planning the tournament schedule is both important but also a difficult task to handle, especially for a tournament like Vildbjerg Cup, as mentioned before, with more than 700 teams participating in the tournament.

Planning the tournament schedule includes choosing the tournament structure. Most commonly, it includes group stages and finals, but the lower placed teams, in terms of group-stage qualification, often qualify for a losers bracket of some sort. The football teams vary in amount of players, in age and in skill. A common way to distinguish age groups and skills, is to divide the tournament into divisions, based on those two. As a division might still have a lot of teams, it's common to divide these divisions into pools. This is just some of the things the tournament planner have to take into account, when they plan a tournament schedule. Other important things is that teams don't have two matches planned for the same time, or right after each other. The amount of the work required to plan these tournaments increases with every team that has signed up.

Making the finals fit the amount of teams, and the amount of hours or days they have for the tournament, can be a hard task. The right teams have to play against the right opponents, at the right time, in order to make the finals work out as planned. The tournament planner also tries to give the teams as many matches as possible, since all the teams came to the tournament to play football and be with their teammates.

Based on the introduction above, this report will draw its baseline from the following problem statement:
How can a software application plan a tournament schedule, based on a number of teams sorted in divisions and pools?

2 | Definitions

A football tournament is in this report defined consisting of the following elements:

Teams, which is the groupings of players from a club, who's playing together against another team in a match.

Matches, which is the organized game between two teams.

Fields, which is the grounds upon which the matches are being played. Only one field is required and can be used for one match at a given time.

Time intervals, which can be used to define the time a tournament is hosted within, and define the time a team can be at the tournament grounds. The time intervals will be defined by the time and date a tournament is hosted.

Pools, which is the groupings of teams, who are to play against each other to qualify for more advanced pools.

Divisions, which is the grouping of pools, who's field size and match duration is the same. A division can also be defined by an age group of the players in the teams, in the divisions pools.

Group stages, which is the qualification pools, a group stage consist of the matches for a pool.

Final stages, which is the advanced pools, a final stage consists of the matches for an advanced pool.

Finalslinks, which is the link from a group stage pool to the final stages. These are defined by the pool placement of a team, linking each pool placement to a final stage.

A tournament, which is the collection of divisions, containing the pools and teams, as well as the fields for all the matches. A tournament also consists of the time intervals for each day a tournament is hosted within.

A tournament schedule, which is the collection of matches for a tournament. The schedule is generated through the collection of matches, at which the match is given a start-time on a given date, and a field to be played on.

3 | Documentation

In this section the problem statement will be examined further through interviews with the potential users of the system. Two interviews have been conducted with two tournament planners from different Danish football tournaments, to understand the process of planning a tournament schedule, and to determine the problems they are facing.

3.1 Interview methods

One of the best ways to find out what people want or need is to interview them. Interviewing people is a qualitative approach and allows the interviewer to dig deeper into the subject, depending on the interview technique used. The method, however, is time consuming and the number of people asked will be restricted. Questionnaires, on the other hand, are much cheaper to conduct and you can get input from a lot people. This method, however, is quantitative and therefore doesn't allow the interviewer to acquire a deeper understanding of the problem statements nature. Given the circumstances that the product will be used under is a very limited user group, the tournament planners, the qualitative interview approach is preferred.

There are mainly three different interview techniques, all with their strengths and weaknesses. These three are divided as structured, semi-structured and unstructured interview methods. The structured interview consists of questions written down before the interview, which the interviewee are then asked during the interview and the interviewer must not deviate from these. This approach is good if you need to interview a larger amount of people in a short time, but it does not let the interviewer get in depth with the problem area. This approach also allows you to do a quantitative analysis, because each interview is easily comparable. The semi-structured interview is a little different, as it consists of pre-made questions and allows the interviewer to ask follow-up questions, or ask to elaborate on certain aspects, that the interviewee has mentioned. This interview method encourages the interviewer to explore even further than what was originally planned, and improvise questions based on the answers the interviewee gives. The last interview technique is the unstructured interview. In this method, there are no questions written down before the interview takes place, which is why this method is suitable if the interviewer has little

knowledge about the subject, and is therefore not able to ask the proper questions. The interviewer just asks whatever comes in his or her's mind during the interview, which can lead to sidetrack conversations.

For the conducted interviews done for this report, the semi-structured approach was chosen. This was because a deeper knowledge and understanding of the planning process was necessary. The unstructured interview method was not used at all, as all members of the group had participated in tournaments before, and therefore had general knowledge of tournament structures.

3.2 Interview data

The interviews are conducted by one group member over the phone and they were, with permission, audio-recorded too. The two interviewees are, as previously mentioned, tournament planners from two different Danish football tournaments. They will also be the target group for the product.

3.2.1 Interview with Magnus

For 24 years, Magnus Nielsen has been planning the tournament schedule for Hinnerup Cup. Magnus can't set an exact time spend on the planning of the tournament schedule, but he says he spends a lot of hours on planning the schedule for just one tournament.

Hinnerup Cup is running for three days, as it starts on a Thursday and ends on a Saturday. Magnus is trying to give the teams as many matches as possible, and all teams are playing group-stages, but the finals are done in many different ways at Hinnerup Cup, where all depends on how many teams there is in each age-group. For example if six teams had singed up for one age-group, he would put them all in one pool. The teams in the pool will play each other one time, and then the first and second place in these matches will play a final, third and fourth will play a placement match for third place and fifth and sixth will play a match for fifth place. If he had 12 teams, he would fit them into four pools, and the teams of a pool would play against each other once. The first place in the pools would qualify for A-playoffs, second place would qualify for B-playoffs and third place would qualify for C-playoffs. Then they have formed new pools, where the winner of the new pools, would then be the winner of either the A, B or C play-off. When Magnus is forming the pools, he is trying to seed the teams so that there wont be a pool full of elite teams, which means one of them have to go to the C-playoffs, and possibly play against teams that might normally play in the B or C league. Even though he is trying to give the teams as many matches as possible, he is also trying to make it as fair for everyone as possible.

Magnus uses a website called Piratliga.dk to help make the tournament schedule. He forms the pools as the first thing, he then sends the pools to Piratliga.dk which sets up the matches for each pool and sends that back to Magnus again. Magnus then adds the time for the matches, the time be-

tween each match, which fields the matches should be played on and sets up the finals/playoffs for the tournament. Then Magnus sends the Excel sheet back to Piratliga.dk. During the tournament they can update the positions of the teams in the group stages and the results of the matches via a mobile phone application. If they find an error in the tournament schedule, Piratliga.dk will then have to reopen the tournament schedule, so that Magnus then can make the necessary changes to the tournament schedule.

3.2.2 Interview with Jørgen

Each year Jørgen Trankjær and two volunteers plan Vildbjerg Cup's tournament schedule, which includes group-stages and finals. They spend about three to five days on planning the tournament schedule, dependent on the amount of teams signed up, and how the pools are going to look like. Jørgen is the one sorting out the pools for the teams, while the other two are specifically planning tournament schedule.

Vildbjerg Cup is running for four days, Thursday to Sunday. They are having the group-stages played at Thursday and Friday, while the finals are played on Saturday and Sunday. At Vildbjerg Cup they are trying to get four teams in each group, but this can change depending on how many teams there is in each division. There are many criteria that Jørgen and the volunteers have to take into account, for example that a team can't play two matches at the same time, and shouldn't play a match right after finishing another one. They are also trying to limit the amount of fields used for one pools matches, so that they can watch the other teams from their pool play, and so that they know where they should play their matches later on.

At the moment at Vildbjerg Cup, the tournament planners are making their tournament schedule in Microsoft Excel sheets. They then import their Excel sheet to a program called CuMap, which they use to run the tournament. It can keep track of the positions of the teams in the group stages, and will send the results online. Jørgen says that other people have recommended software solutions to plan the tournament schedule for them before, but none of the software solutions have been what they were looking for yet.

3.2.3 Follow up questions for Magnus and Jørgen

After the interview with Magnus and Jørgen, the group processed the information by listening to the recorded interviews again. New questions emerged, and more explanation was needed, so an e-mail was sent to them both with some follow up questions. Both e-mails were pretty similar, but yet they were specified for the respondents tournament.

The questions asked and the responses were in Danish, but has been translated to English. Here are some of the questions asked to Magnus and Jørgen:

Is there a break in between each match at the fields? E.g. when team one and team two have just played, will team three and team

four's game start right after, or is there a little break in between the matches? If this is the case, how long is the break?

- **Magnus:** A U13 boy team (boys under the age of 13) plays a match for 2x20 minutes, with a few minutes halftime and a few minutes for the teams to get off the field before the next match begins, so a match duration will be about 50 minutes.
- **Jørgen:** There is a 5 min. break in between each match.

When a team has just played a match, is it a requirement that a specific amount of time should pass before the same team should play the next match? If this is the case, for how long should they wait to play the next match? And is it sometimes necessary to have teams play two matches in a row, to get the tournament schedule to fit?

- **Magnus:** It depends on which kind of teams we are talking about. If it for example is an U8 boy team, who during the tournament plays 7-8 small matches with a duration of 2x6/8 minutes or 1x15 minutes, then i could, if needed, let the teams play two matches in a row. Of course they shouldn't play three or four matches in a row. They will need a break and something to drink and so on.
- **Jørgen:** The only time where a team can get two matches in a row is if there is a team who withdraws in a pool consisting of four teams, before the tournament starts. In this case the tournament schedule changes for the specific pool, so that the remaining three teams will play each other twice. Since the matches in the pools are often played right after each other on the same field, this means that all three teams will have to play two matches in a row, one time each. Alternatively, a lot of changes would be needed for the tournament schedule, since all the fields are normally occupied Thursday and Friday, and we don't want to do that.

Besides the above questions, some questions for understanding purposes were sent to Magnus and Jørgen. The current understanding of how to plan a schedule step by step was presented to both tournament planners, and neither of them had any corrections. Also a list of requirements for a software solution was sent to both Magnus and Jørgen, it can be seen down below in the Summary section. Both Magnus and Jørgen agreed on the requirements, and didn't have any additions to the requirements list.

3.3 Summary

Based on the interview and follow-up questions for Magnus and Jørgen, a list of requirements has been found and defined, as shown below.

- The period of the tournament should be able to be set both date and time.

- It should be possible to set a time for each day, e.g. that a team can play matches Friday from 8.00-20.00.
- Teams shouldn't play two matches at the same time or two matches in a row.
- A field can't be used for two matches at the same time.
- Matches for a specific team or pool should be played at the same fields as earlier matches for that specific team or pool if possible.
- A team can have special needs, e.g. a team might be prevented to arrive for the start of the tournament, so that the time intervals for each specific team should be able to be set for available, when they are able to play a match.

4 | System Choices

Based on the problems and the information from the previous chapter a solution can be developed. This will happen through an analysis process which includes role models, which will be looking at existing software solutions used by the informants. A PACT analysis, to try and understand the people and their activities and in which context the activities take place, to make a interactive system design. A Rich picture will be drawn to try and understand the problem. Last a system definition will be written, which can be used for the problem domain in the next chapter.

4.1 PACT

A PACT analysis is used for scoping the variety of different people, activities, contexts and technologies that revolve around designing a human interactive system, so that concrete scenarios can be developed. A concrete scenario will include the similarities of the four above mentioned topics, that will explain how the user will interact with the system. This will lead to requirements on the interaction between the user and the system, plus requirements on the interface of a system.

4.1.1 People

People differ in many ways, as of this report, people will be divided into four topics of difference: Physical differences, psychological differences, mental models and social differences. From this part of the PACT analysis, the requirements for people's differences will be taken into account.

Physical differences

As of physical differences in our user group, which is the tournament planners, there's a world of differences. It can be anyone, who has the ability and knowledge to make a tournament. Some differences could be weight, height or the user's ability to walk, but a lot of these things aren't relevant, as the system doesn't depend on the user's regular health. Eyesight is a health issue that has to be considered, as colour blindness or reduced eyesight might lead to requirements on the user interface. An example could be: A red button for deleting an object, would need text similar to "Delete"

or “Remove”, if the user is colour blind (as for the colour red). One in twelve men, and one in two-hundred women are colour blind, which makes this disability an issue that has to be considered, when making the user interface for an application [5].

Psychological differences

When considering psychological differences in the user group, some might have close to none technological understanding, which creates the requirements of simplicity and familiarity. As a user with low technological understanding, buttons or sheets that the user hasn’t seen before would be hard to understand, learn and remember. Simplicity and familiarity will let the user understand and remember the functionalities from earlier experience, rather than investigation.

Mental models

A user of a new system always has expectations on how to operate the system, as mentioned before, familiarities makes it easier for the user to operate a new system, if it “works the same way as earlier experiences”. Since our user group has told us, they are familiar with excel and has experience in excel, this would lead to certain familiarities that could be used. The user knows how columns and rows work, they could also know how to drag and drop variables from block to block, which would be familiarities that can be used. The mental model and thereby the understanding of a system, would be determined, not solely, by excel experience.

Social differences

The user group of the software system would be homogeneous persons, commonly volunteers at the club that arrange the tournament, who wants the same thing from the software: A tournament schedule. There’s no differences in what they want from the software, and there’s no requirements set for different social groups. The social differences of the user has no impact on how the system should work in this case.

4.1.2 Activities

In this section of the analysis, the characteristics of activities will be considered. Different activities to be considered have following topics: Temporal aspects, cooperation, complexity, safety critical and nature of the content. The requirements from this section will be taken into account at the summary of the PACT analysis.

Temporal aspects

A tournament regularly don’t appear more than once a year, so the system to plan the tournament schedule for the tournament, isn’t in frequent use of the tournament planner. This means, that it should be easy to learn and

remember, since the user group is infrequent users. But once this system is in use, it has a deadline as for when the tournament is held. This means, that it has a time pressure, so that it cannot spend too long to produce the schedule. Producing this tournament schedule should be one continuous interaction, since it's a work tool. The user gives the complete input, and the system produces the output. As of algorithms, we need to consider the use of time. Since any algorithm takes a certain amount of time, some will use more time than the user would think. This leads to the use of program-indication of the time spent, and the estimated time left of the process, to let the user know that the program has not crashed.

Cooperation

As both informants for this project has pointed out, the work is done either alone, or in cooperation with others, but in individual stages. No more than one person should work with the schedule at once, since this could lead to plan-conflicts.

Complexity

The system in consideration should be a step-by-step process, as this leads to a straightforward way for the user. The system should still be flexible, in the sense that the user can change parts of the process. Users of the system has a well defined task, but the end product could have alternatives, which the user should be able to change without redoing the whole process.

Safety critical

Safety, in form of health and injury, is not in consideration for this program, since it does not require the user to perform any real-life movement beside the mouse and keyboard. The safety in consideration for this system, is the data storage. The user should be able to undo the last, or maybe even last few, actions they made. The system should be able to either save by automatic saves, or by the user clicking a save button. This would enable the temporal aspect of users being able to leave the system at the point it's, and returning later to the same point.

Nature of the content

The system should support keyboard and mouse features, for when the user wants to write something, drag and drop, or click something. The reasons for this are, that there will be a lot of data input, a lot of clicking, and maybe some corrections from the user, that has to be utilized through a keyboard and a mouse.

4.1.3 Context

The activities and people all exist in a context, in this section we will consider three different topics of contexts: Physical environment, social con-

text and organizational context. This will express requirements, that will be taken into account in the design of the system.

Physical environment

The use of the system would regularly be an office, either at home or at work, to enforce the low level of noise. The room would usually be quiet and work friendly, and this means that there should not be any sounds from the system. If this wasn't the case, the system could have a "process complete" sound, but the system should have a visual process-timer.

Social context

As this system would only be in use for one user at a time, and the system isn't entirely a work-office tool, there's little to no social context to consider. As mentioned above, sound should be avoided, tutorials and templates should be included in the solution, as this would help the user to understand differences or similarities between the new system and earlier experiences.

Organizational context

The system for this project should not deskill the user, as they have the power of choosing tournament structures, and reworking the schedule afterwards. Since the user has to choose the tournament structure, the user still has to have the knowledge of tournaments and the structures that suits the amount of teams, to be able to use the system. Therefore the user will not forget, and therefore not be deskilled.

4.1.4 Technology

The current systems include technologies, which has to be considered. The current technologies will help developing an overview of what can be improved, and what works as the user wants.

Output

This system should be able to view the information from adding the teams (or pools) to the system, to and including the finished tournament schedule. It should also include the option to print the schedule, so that it's also possible to have it in paper-format. The output will therefore be either a computer screen, or a paper-print of the schedule.

Input

The system should support input from both a computer mouse, and a computer keyboard. The means of using the keyboard is to give the user the ability to write something in the system, and this is needed to add the teams to the system, and rearrange matches. The mouse (or cursor) is another type

of input, this makes the user able to drag-and-drop, and click on an object in the system.

4.1.5 Summary

The users of this system can be anyone with knowledge of how to make a tournament, who might have eyesight disabilities. That person might require simplicity and familiarity with the systems user interface and functionalities, which the user might know from earlier experience, such as Excel. This user group is homogeneous, commonly volunteers, where social differences has no impact on how the system should work.

When the system is in use, it has a deadline as to when the schedule should be done. This leads to a time pressure, and it should be done in one continuous interaction. An algorithm for this system should be evaluated on the amount of time it takes, and the quality of the end product. The system should only be in use by one user at a time, but should be a step-by-step process. The user needs the ability to configure the process, and the end product. The system should implement data-storage safety, to undo actions, and save the current version of the schedule.

The system should not make use of any sounds, but use visual representations. The system should include tutorials and templates to show the user how to work with the tool. Any user should have the opportunity to choose the tournament structure, to prevent deskilling the user.

The system should view the information for the user and print the schedule. It should support input from a computer mouse and a computer keyboard.

4.2 Role models

In this section the group will explain which role models we've found in association with Jørgen and Magnus, and how they can be used as a form of inspiration to develop a solution.

4.2.1 CuMap

CuMap is a tournament- and match planning system, which is currently being used by many Danish and Swedish sports clubs to plan and organize football and handball tournaments.

This IT-system consists of both a PC application and an online solution. This separation of functionality across different platforms has some downsides, which is why this report focuses on creating a product for just one platform that is able to plan and organize tournaments in these sports. A good thing about CuMap is that you are able to organize the matches yourself and the system will then warn you if any of the matches clash. That seems like a good idea in case the tournament planners would like more control over the matches, but a fully manual match planning functionality is too much of a hassle. So taking the next step and combining the two aspects, you could have an auto-generated match schedule for every team with the opportunity for the tournament planners to make some necessary changes, if any. Another great idea in this existing solution is the opportunity to select different criteria after the completion of the initial group stage. That gives more customization so the tournament planners can use the application with their own tournament structure.

4.2.2 Excel

Excel is a sheet-software created by Microsoft. In Excel there's a lot of empty columns/rows, where you can basically do whatever you want with them. There are many functions in Excel, like for example you can mark a number of columns/rows and move them to a new column/row. Other than that, there's also implemented functions like the SUM-function. This function adds all numbers in a column or row, depending on what needs to be added. Another very popular function is the IF-function. This works just like the if-statement in the programming language, where it needs to check for something, like if A is bigger or equal to B, then it should proceed with something, whereas if A isn't bigger or equal to B, it should just skip that, as it doesn't fit the expression.

This software solution contains many ideas, because you have the option to do whatever you want with it. If you just simply need it to show some information in a better view than Word for example, you can do that. If you want to make a budget form that's possible too. It's also possible to make a tool comparing the wages of every employee in a firm, comparing them with their employment status e.g.. manager, assistant etc. Though the most interesting part of this software for this project, is that people also use this software to create tournament schedules.

The way some of the tournament planners make their tournament schedule is by creating the pools in different columns, like pool 1 in the A-column, pool 2 in the B-column etc. that way they get a clearer view of which teams that's in which pools. From there they plan their match schedule, and here they have two options. They can either use the functions in Excel and use columns instead of writing the names themselves or they can do it manually. If they choose to make it manually it might be faster than doing it with the functions, but it's only going to be the first time that it's faster. If they use the functions in Excel to create the match schedule, they can save the sheet document and every year when they have to make a new schedule, they can use simply use the old one, and just manually changing the names of the teams in the pools. If there are more or less teams in the pools from year to year, they can save the match schedule sheet as a template, where they can name them different things like "pool - 4 teams", "pool - 3 teams" etc., that way they'll have it all saved and the work they need to put in it will be reduced from doing it manually every year.

4.2.3 Piratliga.dk

Piratliga.dk is a website containing different tournaments in Denmark, including the match schedule and results. The way this works, is that there's an Excel-sheet that you'll have to download and fill in the pools and how the matches are going to be played and email it to them on the website, where they'll put it online. A few days before the tournament starts they'll send an email containing a username and password so that the tournament planners can add the results for each match to the website, so the clubs can know which team they'll have to play against after the group stages for example and on which field. So basically what this website does, is showing the matches and results. It doesn't help planning the tournament as you have to make the tournament schedule on your own and email it to them.

This website helps the tournament planner with publishing the tournament schedule online, so that the teams can see which matches they have to play and where. Other from that it also handle the results, giving teams points for each victory/draw and gives a group standing table. It also shows which of the teams that have qualified for the A-finals, and which teams that have qualified for the B-finals etc.

The whole idea of the website seem to be good, as it gives the teams the option to see their schedule online, instead of having them to use the basic tournament schedule "book" where every single match is listed in, even the ones outside of their division. In a team of 14 players or 11 players plus a coach, the possibility of having a smartphone is very high if not certainly. So the idea of piratliga.dk is good, but it doesn't have many functions, other than checking matches and results.

4.2.4 Summary

By analysing existing solutions that already take care of some of the tasks related to tournament planning, a set of ideas and criteria for the system

were introduced. The three existing solution analysed in this section were CuMap, Microsoft Excel and the website piratliga.dk. CuMap is capable of taking care of the tournament- and match planning while being able to keep track of the whole tournament process with match results for all matches. Excel on the other hand is not able to automatically generate a match schedule for the tournament planner right now and it's currently being used to manually organize all the matches, which takes a lot of time. Piratliga.dk is a website that takes care of the match schedule and results after they have been generated. The tournament planners simply fills out an Excel sheet with the required information and send it to piratliga.dk and they'll take care of the rest. A few days before the start of the tournament, piratliga.dk will send the login credentials to the tournament planners so they can add the results of the matches.

4.3 Rich pictures

This section about rich pictures is written using the book "Objekt Orienteret Analyse & Design"[1]. A rich picture is an informal drawing which should express the drawers understanding of a problem. The drawer use the input he or she got from the user or informants to draw the picture. Rich pictures doesn't have to be beautifully drawn and all members of the group doesn't have to agree on all the pictures, it's meant to open up a discussion and give the drawer and in this case the system developers a good overview of the problem.

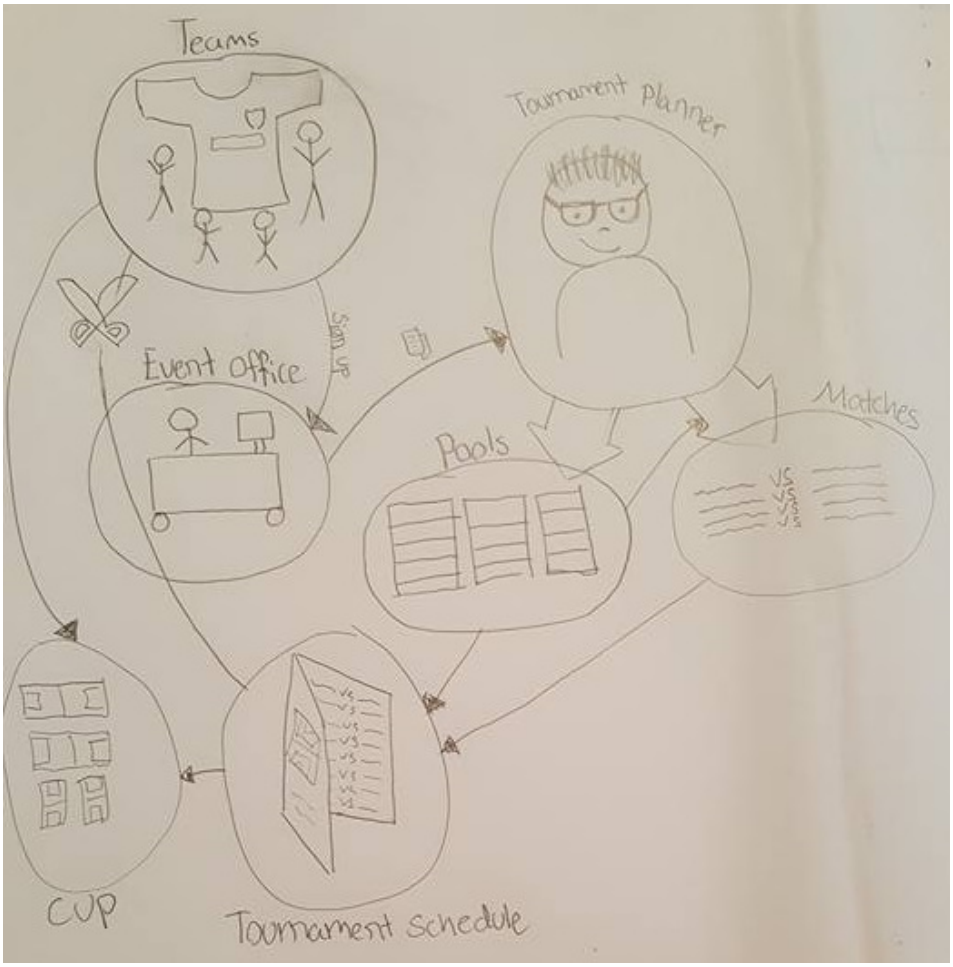


Figure 4.1: This is the Rich picture which is explained in section 3.6

The following paragraphs includes a small explanation of the different drawings in the rich picture in 4.1.

The teams are one of the main objects in a tournament, since they are the ones to play all the matches and compete with one another. A team contain players, coaches and the team leaders.

The event office is the administration of a tournament. The event office is who you contact when you want to sign your team up for the tournament or if you have any problems during the tournament.

The tournament planner is the one making the whole tournament schedule for the tournament. This includes sorting the pools and setting up the matches.

Pools are a grouping of teams, who will play each other during the group stages. The pools sizes will vary from tournament to tournament and will be set by the tournament planner.

Matches includes all the matches both from the group stages and the finals.

The tournament schedule is the final result when the pools and the matches are put together.

The cup/tournament is where the teams will meet and play all the scheduled matches, trying to win the trophy of their division.

First the teams will sign up for the tournament, which is done trough the event office. The event office then gives a final list of teams to the tournament planner. The first thing the tournament planner will do, is to sort the teams into pools with a fitting amount of teams in each pool, which can vary from tournament to tournament and depending on how many teams that have signed up in each division. After the pools have been sorted the tournament planner then plans all the matches. First he plans the matches for the group stages, and then the final stages, which again vary from tournament to tournament and depends on how many teams have signed up. The pools and the matches will then become the tournament schedule. When the tournament schedule is done, it's run through the event office and sent out to the teams. At this point there might be conflicts, maybe the teams found a mistake in the tournament schedule e.g. if the team have two scheduled matches at the same time or if they can get to the tournament by the time they have their first match. When no more objections have been made about the tournament schedule, it's done and ready for the tournament to start.

4.4 System Definition

In this section the system definition is written using the book "Objekt Orienteret Analyse & Design". A System definition will describe the system as a whole and not the individual parts of the system. The system definition will be a short and precise text about the systems general features. The system definition will be checked against the FACTOR criteria. This is a iterative progress, so it will be rephrased until it's seen fit. [1]

The formulated system definition:

An IT-system that will help the tournament planners make a tournament schedule, will be developed. The system will be an automatic planning-tool to make a complete tournament schedule of all the matches for the tournament planner, but will have to be flexible for the tournament planner to be able to reorganize the tournament schedule themselves. The system should be able to be run on any windows PC or Macintosh that have an internet browser with an internet connection. The system should be able to be used by people with different experience with tournament planning, since many tournaments are run by volunteers. The system should give an output as an Excel sheet, that are readable by other programs for managing the tournament.

4.4.1 Factor criteria

- **Functionality** - Support for the tournament planners for planning the tournament schedule, to make a complete tournament schedule for the tournament, with functionalities for reorganizing for the tournament planners.
- **Application domain** - The system is meant for the people who are planning the tournament schedule, which will be the tournament planners.
- **Conditions** - Volunteers with a little experience in excel or other tournament planning software solutions, so that they have an idea of how to plan a tournament schedule.
- **Technology** - A computer with an internet browser and internet connection.
- **Objects** - Teams, pools, fields and matches, but also the tournament and tournament schedule.
- **Responsibility** - An automatic planning-tool to help plan tournament schedules.

5 | Problem Domain

In this chapter the problem domain of the project will be described. Tools like class diagrams, state chart diagrams and event tables will be used to analyze the problem domain. First a class diagram will be presented, which will describe the relations between the systems classes and objects. Based on the class diagram, the state chart diagram will be described with its behavioural patterns, attributes and the event traces. Last in this chapter the event table is shown and the events are described. All these tools are made based on the content from the book OOA&D(Objekt Orienteret Analyse & Design)[1].

5.1 Structure

This section describes the objects and types of objects that will be the focus of this project. It also describes the relationship and interactions of the classes, as it's shown in the class diagram 5.1.

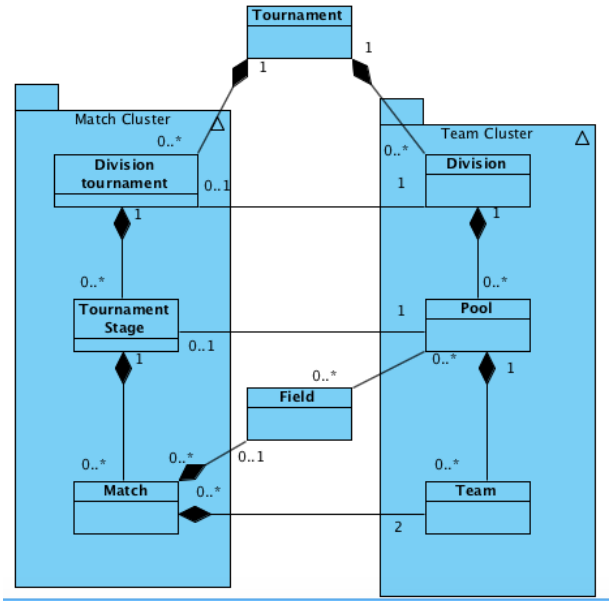


Figure 5.1: Class Diagram

5.1.1 Tournament

The Tournament class contains a time span for each day of the created tournament. The class also contains a list of all the divisions, which includes pools and teams.

5.1.2 Field

The Field class represents the fields which are available for the tournament. The class contains a field size and a name of the field.

5.1.3 Match cluster

The Match cluster contains the classes for scheduling the matches and the matches themselves.

Match

A tournament schedule is, in essence, a grouping of matches. Since there can be no tournament schedule without matches, a match is a class in this system. A match has two teams: a field to play on and a starting time, which are also the attributes for the class. A match can be created and later on be scheduled. These are the events of the class.

Tournament stage

There are different tournament stages, with different structures according to the amount of teams in the tournament pools. A tournament stage is the structure of a piece of the tournament schedule.

Division tournament

Division tournament is the collection of matches for a division. It contains all the tournament stages for a division.

5.1.4 Team cluster

This cluster contains the classes which all consist of different groupings of the teams and the teams themselves.

Team

When a team registers for the tournament, a new object of the Team class will be created. The team will be registered with their team name, added to a pool and have a list of their matches for the tournament.

Pool

A pool is a grouping of teams, as a part of a division. All teams in a pool belongs to the same division. The pools are used to set up a tournament stage between the teams in the pool. There's one pool for each tournament stage, but there can be more pools for a division tournament.

Division

A division is the grouping of teams, so that the level and age of the teams in a division are about the same. A tournament can have more than one division, but a division belongs to only one tournament.

5.2 Behavioural patterns

This section describes the behavioural patterns of the classes. The behavioural patterns of a class is the possible course of events for the class.

5.2.1 Explanation of state diagrams

In each of the behavioural pattern state diagrams, symbols are used. These symbols have different meanings, depending on the different functions.

The iteration function is shown as two arrows pointing to each side (a).

The state function, describing what state the class is in, is shown as a light blue background (b).

The grouping function, which means that a class can have more states but can still call the same event. It's shown as a striped border (c).

The end state function tells when the state is at the end and it's shown as a dot inside a circle (d).

The start state function tells when the behavioural pattern starts and it's shown as a dot (e).

The state change function tells when a state changes and it's shown as an arrow pointing down (f).

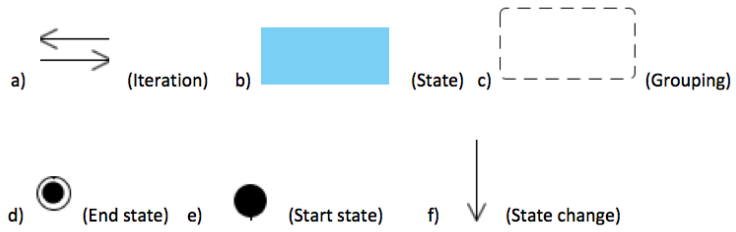


Figure 5.2: Symbols-state diagram

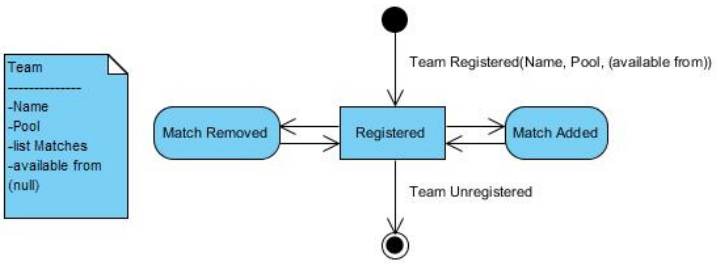


Figure 5.3: Team Behaviour

Team

When a team is registered we have the opportunity to add or remove matches. The user can add or remove matches several times, because it's an iteration. The behavioural patterns ends when a team is unregistered.

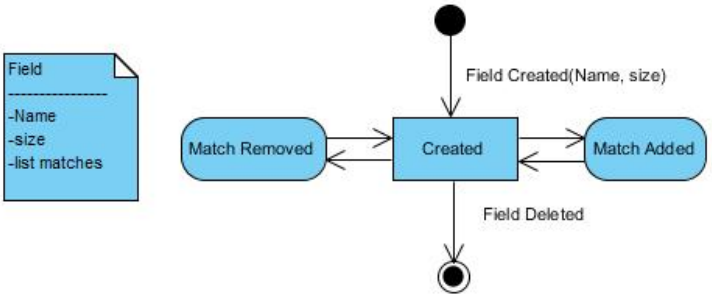


Figure 5.4: Field Behaviour

Field

When looking at the Field class, the behavioural pattern starts when a Field is registered. When a Field is created, the user can add or remove matches several times. The behavioural pattern ends, when a Field is unregistered.

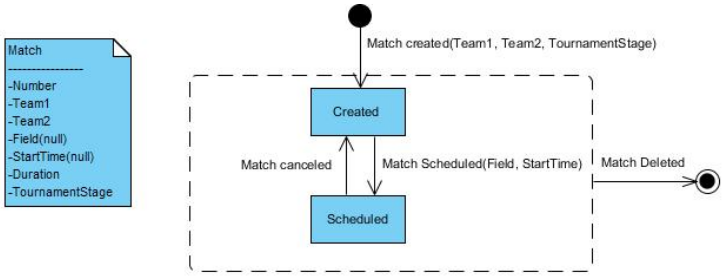


Figure 5.5: Match Behaviour

Match

The Match class' behavioural pattern starts when a Match is created. When a match is created, there's an option to schedule a match and to cancel a match. The match can be scheduled/cancelled several times as it's an iteration. When a match is deleted, the behavioural pattern ends.

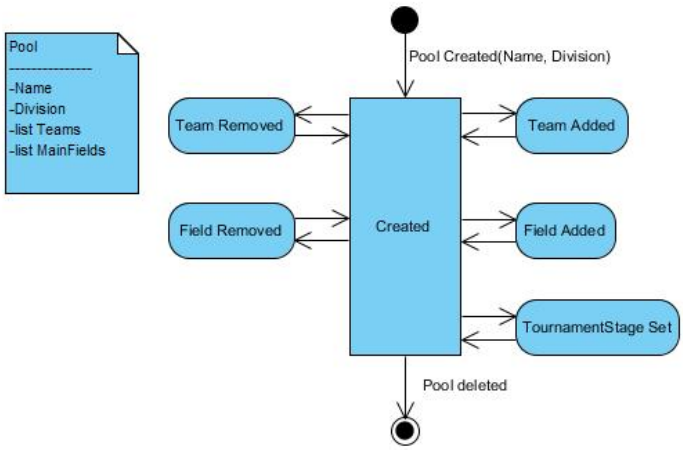


Figure 5.6: Pool Behaviour

Pool

When a pool is created, the behavioural pattern starts for the Pool class. The Pool class has the option to add and remove one or more teams. It also has the option to add or remove a favourite field for the pool. There is also the option to add the pool into the TournamentStage class, through scheduling the matches for the pool. The behavioural pattern ends when a Pool is deleted.

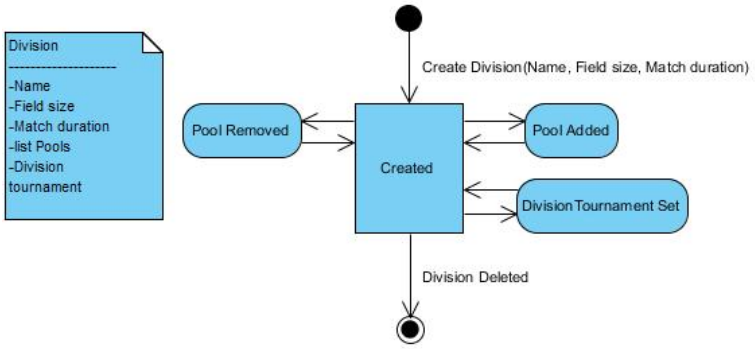


Figure 5.7: Division Behaviour

Division

The behavioural pattern for the Division class starts when a division is created. The division will then have the option to add or remove one or more pools. It also has the option to set the division into the division tournament, through scheduling all the matches for the division. When a division is deleted, the behavioural pattern for the Division class ends.

TournamentStage

The behavioural pattern for the TournamentStage class starts when a Tour-

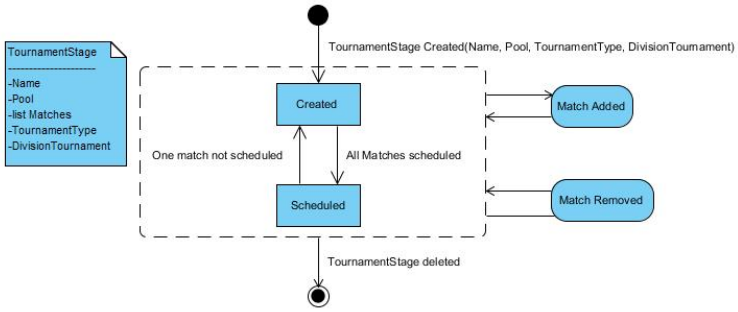


Figure 5.8: TournamentStage Behaviour

namentStage is created. The TournamentStage has the option to add or remove matches. When the matches have been added, the TournamentStage will schedule the matches. When a match is removed, the TournamentStage has the option to schedule all the matches again. The behavioural pattern ends when a TournamentStage is deleted.

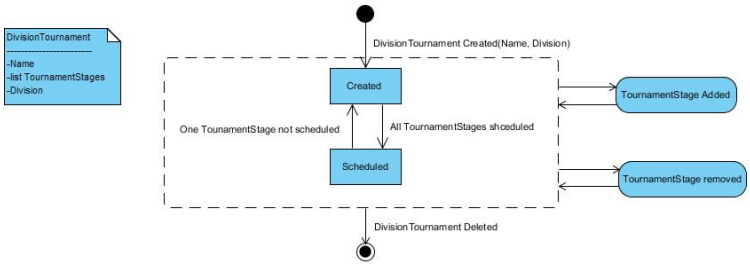


Figure 5.9: Division Tournament Behaviour

Division Tournament

The behavioural pattern for the DivisionTournament class starts when a DivisionTournament is created. The DivisionTournament has the option to add and remove a TournamentStage, where the DivisionTournament will be set as scheduled when all the TournamentStages for a Division is scheduled. The behavioural pattern ends when a DivisionTournament is deleted.

5.3 Events

This section will describe the events in the system and reasons to why these events are important. It's a sequential description of each event.

Match CRUD

When a match is created in the system, it defines the teams for the match. This is needed to have a match that can be scheduled when all matches for a tournament have been created. This is important, as the schedule cannot be created without the matches, but a match cannot be scheduled before

all matches have been created. After being created, a match can be read, updated and deleted.

Match scheduled
When all matches have been created for a certain division and pool, it can be scheduled to a tournament schedule. This is done by scheduling every match, giving it both a field and a starting time. Without having every match scheduled, we cannot assume that every match is in a tournament schedule.

Team CRUD
By creating a team, it's assigned to a pool in a division. It's a requirement that teams are created before matches can be created. Later on, these teams will be assigned the list of matches that the team participate in. After being created, a team can be read, updated and deleted.

Pool CRUD
When a pool is created, the pool will be assigned to the division that it belongs to. This pool will contain the teams that have to play against each other and have a tournament stage appropriate to the amount of teams in the pool. After being created, a pool can be read, updated and deleted.

Division CRUD
The create event creates a division for the tournament. The division will have information about the specific divisions' field size and match duration, and will also have a list of pools for the division. After being created, a division can be read, updated and deleted.

Tournament stage CRUD
The create event of the Tournament stage will schedule the matches for the tournament. It will also create the specific tournament stage and select the tournament type which can be round robin or knockout. After being created, a tournament stage can be read, updated and deleted.

Division tournament CRUD
The create event occurs after all the matches are scheduled and will then have all matches for the tournament as smaller tournaments, so called division tournaments, which all together you could call a tournament schedule. After being created, a division tournament can be read, updated and deleted.

5.3.1 Event table

From the above sections about structure and events, an event table(5.1) has been made. This will describe the classes and their mutual connections.

In the event table, the occurrence of the event is marked with "+" and "*". The "+" means that the event can occur zero or one time, while the "*" means that the event can occur zero or more times.

Table 5.1: Event table for the project

	Team	Field	Match	Pool	Division	Tournament stage	Division tournament
Match CRUD	*		+			*	
Match scheduled		*	*			*	*
Team CRUD	+			*			
Pool CRUD				+	*	*	
Main field added				*			
Division CRUD					+		
Tournament stage CRUD				+		+	*
Division tournament CRUD					+		+

5.4 Summary

As the problem domain is finished, the described classes, behaviours and events will be used to analyse the application domain, and it will be used to describe the model for the system.

6 | Application Domain

This chapter is about who is administrating, monitoring or running the system's problem domain. First the systems actors will be identified and their use cases described, which will end up with an actor table. Next the systems functions will be listed and explained, which will make the actor's use of the system possible. Last in the chapter a prototype of the user interface will be shown and explained. This chapter is based on "Objekt Orienteret Analyse & Design" [1].

6.1 Use

In this section, the interaction between the actors and system will be described. The group will explain which actors that are involved with the project and some relevant use cases, that portrays how the actor is going to interact with the different stages of the system.

6.1.1 Actor

An actor is an abstraction for an user and other systems that interact with the system. The actor of this project has been identified as a tournament planner. The actor will be described with its purpose, its characteristics and some examples. The actor will be described in the following section.

Tournament planner

Purpose: The tournament planner is the person or persons who are making the tournament schedule for a tournament. The tournament planner must comply with many criteria when making the schedule, such as the fields, the time, pause for a team right after each match and so on.
Characteristics: Different users with different needs and a general knowledge in Microsoft Excel and tournament planning.
Examples: Tournament planner A have promised the teams who signed up for the tournament a minimum of four matches over the duration of the tournament, which is from Thursday to Sunday in a given week. Tournament planner A have to fit 4 teams in each pool if possible, where the 1st and 2nd place will qualify for the A-playoffs and the 3rd and 4th will move

on to the B-playoffs. The playoffs are the standard with round-of-16, quarterfinals, semifinals and finals. Tournament planner A likes to have the same pools play at the same fields, so that they can see each other play. Tournament planner A is making his whole schedule in a Microsoft Excel document, where the result management is done by another person's system.

Tournament planner B have promised the signed up teams a minimum of 4 matches, but will try to let them play as many matches as possible over the duration of the tournament, which is from Thursday to Saturday in a given week. Tournament planner B tries to fit 3 teams in each pool where they will play a round robin tournament. Afterwards, they will move on to an A, B or C group according to their placement in the pool. After that, they get into a new round robin group. How the finals are conducted varies, depending on how many teams there are in each division. Tournament planner B are sending some general information of the tournament to another person, who he pays to make a string for all the matches that shall be played. Tournament planner B then finishes the schedule and sends it back to the other person, who will set the schedule up for management during the tournament.

6.1.2 Use case

In this section, our use cases are described. A use case is a pattern describing the interaction between the system and the actor. It establishes a well defined use of the system. A use case contains a description about the use case, but it also contains actor(s), object(s) and function(s). Some use cases may only contain one actor, one object and one function, while others may contain three actors, six objects and eight functions. It all depends on the complexity of the use case.

Create tournament

Actor: Tournament planner
User case: The Tournament planner can create a tournament in two ways. First the tournament planner can use the Microsoft Excel template to set up a tournament with division, pools and teams and then upload it to the system. The tournament can also be created in the system itself, which it does by making an empty tournament, which then can be populated.
Objects: Tournament, Team, Pool, Division
Functions: Create tournament, Create tournament from Microsoft Excel, CRUD team, CRUD match, CRUD pool, CRUD division

Edit tournament

Actor: Tournament planner
User case: After the tournament planner has either created the tournament by uploading it from Excel or made it in the system itself, it's then possible for the tournament planner to edit the work that has been done.
Objects: Tournament, Team, Pool, Division, Field

Functions: Update Tournament, CRUD team, CRUD match, CRUD pool, CRUD division, CRUD field

Validate tournament

Actor: Tournament planner
User case: The tournament planner has to validate the data that are loaded into the program and make sure that the overview is representing the same information as in the file that the tournament planner uploaded. If something isn't in check, the tournament planner has to check the file that was uploaded, and try to locate the error and fix it.
Objects: Tournament, Team, Division, Pool
Functions: CRUD team, CRUD division, CRUD pool, Create tournament from Microsoft Excel

Create tournament schedule

Actor: Tournament planner
User case: When the tournament planner has set up all the information for the tournament, it's then time to schedule the matches, which happens when the tournament schedule is created.
Objects: Tournament, Team, Pool, Division, Field, Match, Tournament Stage, Division tournament
Functions: CRUD team, CRUD match, CRUD field, CRUD tournament stage, CRUD division tournament, Schedule matches

Edit tournament schedule

Actor: Tournament planner
User case: When the tournament's matches have been scheduled, the tournament schedule will be displayed so that the tournament planner can see if the tournament schedule is correct. The tournament planner can also edit the tournament schedule.
Objects: Tournament, Team, Pool, Division, Field, Match, Tournament Stage, Division tournament
Functions: CRUD team, CRUD match, CRUD field, CRUD tournament stage, CRUD division tournament, Display tournament schedule

Validate tournament schedule

Actor: Tournament planner
User case: The tournament planner has to validate the tournament schedule that has been generated. The planner has to make sure, that there haven't been a miscalculation e.g. a team playing two matches in a row, two matches at the same time, or is set to play against the wrong team.
Objects: Match, DivisionTournament, TournamentStage
Functions: CRUD match, CRUD division tournament, CRUD tournament stage

Export tournament schedule

Actor: Tournament planner
User case: The tournament planner has to export the tournament after validating the tournament. The tournament planner has to insert the schedule into a third party solution to deal with the real-time results.
Objects: DivisionTournament
Functions: Display tournament schedule, Export tournament schedule to Microsoft Excel

6.1.3 Actor table

Based on the actors and the use cases, an actor table has been made, which will give an overview of the system's use. The actors are represented horizontally and the use cases are represented vertically. There is a check-mark if the specific actor can use the specific use case.

Table 6.1: Actor table for the project

	Tournament planner
Create tournament	✓
Edit tournament	✓
Validate tournament	✓
Create tournament schedule	✓
Edit tournament schedule	✓
Validate tournament schedule	✓
Export tournament schedule	✓

6.2 Functions

A function makes a model accessible for the user. A function has to be activated or executed and can result in a change of the model or areas in the problem domain or application domain.

6.2.1 Function types and complexity

Functions can be categorized in different types and complexities. The complexity is a way of describing how difficult it is to implement the function in the system. The complexity have three scales which goes from simple, medium and complex, where complex is the most difficult function to implement. When working with objects of a class in this project, there are generally four different types of standard operations. These are creating, reading, updating and deleting, also called CRUD-operations. Creating and deleting are the types of operations used to create and delete e.g. a customer or a product from the program. Reading is concerned with accessing an object, either to display the information to a user or to another function that uses the information. The last one is updating, which is used to update the different attributes on the object. This could be the email or phone number of a client.

6.2.2 System functions

The system has many functions and each of them has to complete their own task in order for the system to work as intended. The following functions have been identified and will now be analysed.

Table 6.2: System functions with type and complexity

Function name	Function type	Function complexity
Create tournament	Updating	Simple
Create tournament from excel	Updating	Medium
CRUD team	Updating/Reading	Simple
CRUD match	Updating/Reading	Simple
CRUD pool	Updating/Reading	Simple
CRUD field	Updating/Reading	Simple
Schedule matches	Calculate/Update	Complex
CRUD division tournament	Updating/Reading	Simple
CRUD tournament stage	Updating/Reading	Simple
Display tournament schedule	Reading	Simple
Export tournament schedule to excel	Updating	Medium

Create tournament

This function is called to set up a new tournament object, so that the user is able to add teams, pools and divisions. The function is only called if the user

has no Microsoft Excel sheet with all the required information to create the teams, pools and divisions.

Create tournament from excel

If the user already has an Excel sheet ready to be imported to the system, then this function is called. It sets up the tournament object with all the teams, pools and divisions, which is extracted from the Excel sheet.

CRUD functions

These functions are called whenever the system needs to perform basic CRUD operations on an object. The "Create" function will create the object, "Read" will return the requested object, "Update" will update the object with any desired changes, and finally, the "Delete" function will remove the object from the system.

Schedule matches

The schedule matches function is called once all the required teams, pools and divisions are ready in the system, and the matches and tournament schedule can be generated. The schedule matches function is by far the most complex function in the system and is the whole essence of it. Firstly, all the matches between the teams in the tournament have to be generated and then they need to have a time and field assigned to them, which is the scheduling part.

Display tournament schedule

The user will be able to verify the tournament schedule after it has been generated. This function is called after the tournament schedule has been generated and it will display the whole tournament schedule for the user and is categorized in divisions and pools.

6.2.3 Export tournament schedule to Excel

When all the matches have been scheduled by the system, this function may be called to export them to an Excel sheet, that CuMap or Piratliga.dk supports, depending on what the user wishes to use.

6.2.4 Details of complex functions

The complex function "Schedule matches" is a combination of smaller functions that combined should compute all matches, in terms of who plays who, on which field, and when. First the function generates every match that has to be played to resolve the tournament. It does not include where and when the match is played at this stage, only which two teams are facing each other. This is done according to the tournament type from each division. After this, the function goes through every match, sets a field and a time and

thereby scheduling the matches. When doing this, the function has to make sure no team is playing two places at once or two matches are played at one field at the same time. When every match generated from the previous function are scheduled, the "Schedule matches" function is done.

Table 6.3: Schedule matches function breakdown

Function name	Function type	Function complexity
Schedule matches Generate matches needed Schedule generated matches	**Calculate/Update**	**Complex** Medium Medium

6.3 User interface

The following section will describe the user interface of the system. A user interface is used by actors to interact with the system. A good user interface is adjusted to fit the user's tasks and how they understand the system. The following analysis will not be the final requirements for the user interface but rather a set of items that the system's user interface should implement. As per the system requirements, the application will be developed as a web application.

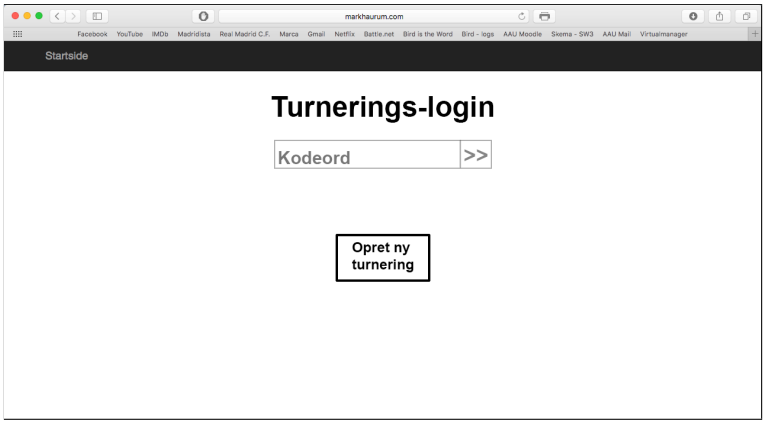


Figure 6.1: Start page

On figure 6.1, the start page can be seen. Here the user has to enter the tournament password that the user of the system has set, when the tournament was created. When the password is entered, the user can view information about the tournament or edit the tournament. The user can also simply create a new tournament.

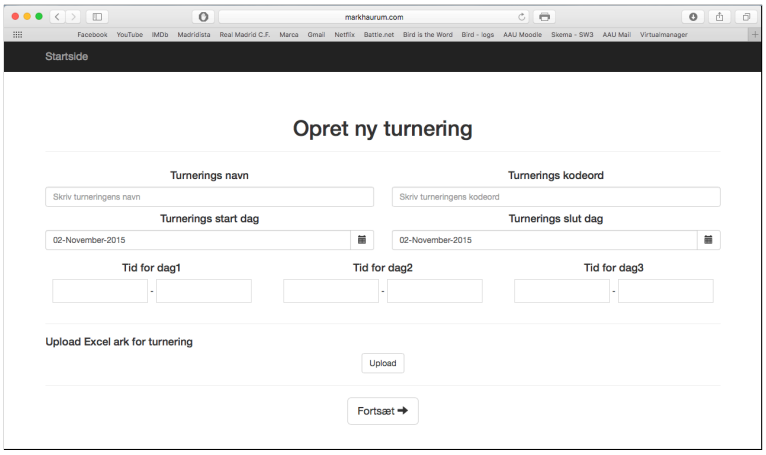


Figure 6.2: Create tournament

Figure 6.2 shows the page where the user can create a new tournament.

There is a form to fill out, with a little information about the tournament which is about to be created. The user can then either upload an Excel sheet containing all the required information about divisions, pools and teams or add them manually in the web application.

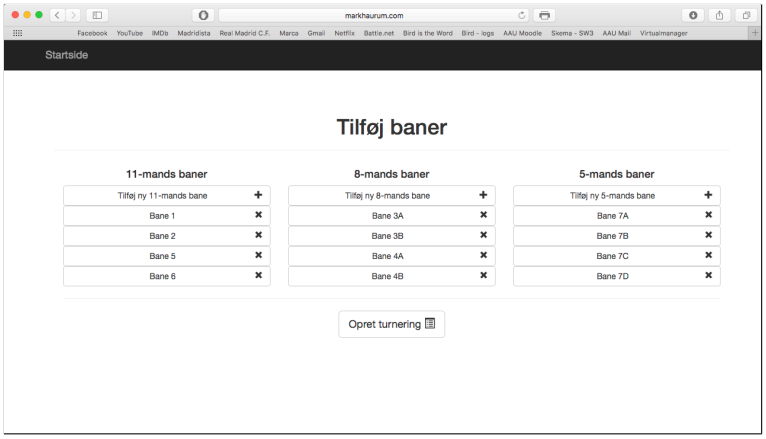


Figure 6.3: Create field

Figure 6.3 shows the screen where the user can add and remove fields. The fields are divided into three different standard sizes for tournaments in Denmark (it may vary in different countries). The user has the option to name the fields whatever they like, since different tournaments may have different naming for their fields.

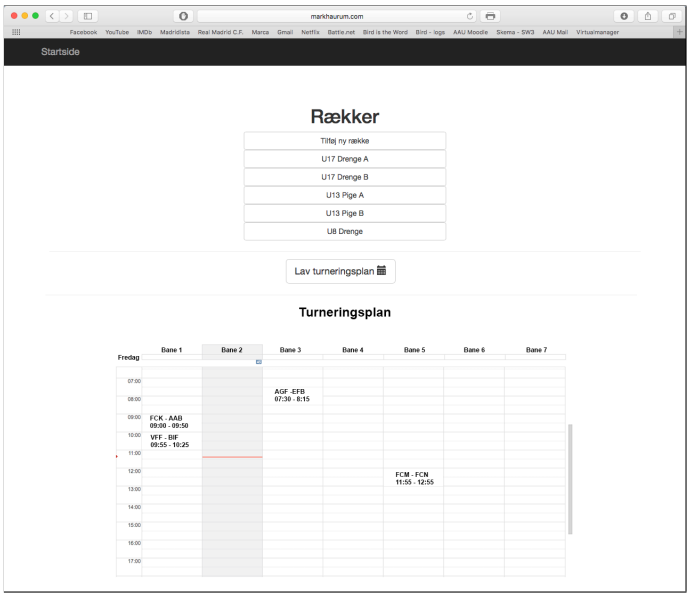


Figure 6.4: Create division

Figure 6.4 is the page where all the divisions can be managed along with

a tournament plan showing all the matches with the day/time following the x-axis and fields following the y-axis. There will not be shown any matches in the grid at the bottom of the page before the button "Planlæg kampprogram" (Make tournament schedule) has been clicked. To be able to click the "Planlæg kampprogram" the user must have added some pools in the divisions, containing different teams.

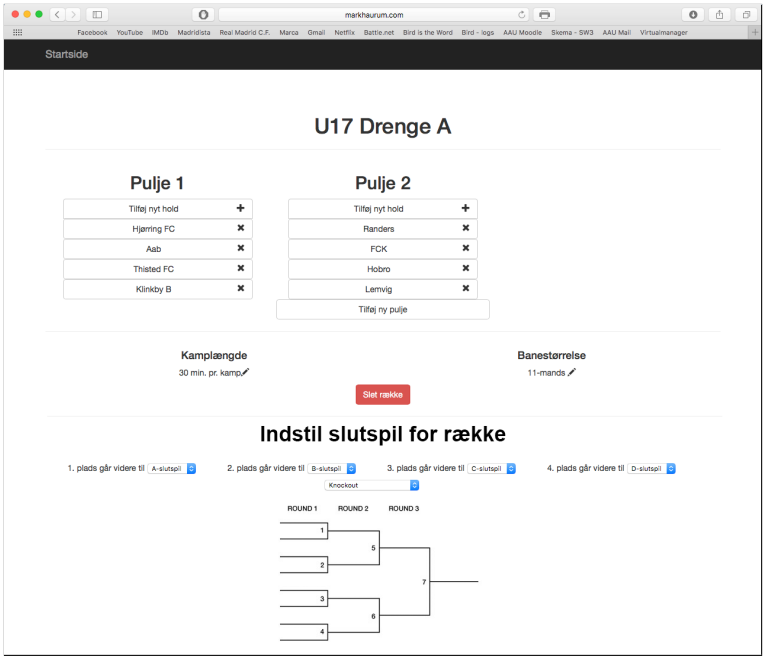


Figure 6.5: Pools

Figure 6.5 is where the specific divisions are managed. Pools can be added to the division and then teams added to the pools. The match duration and the field-size is also set on this page. At the bottom of the page, the structure of the final stages can be set for the specific division.

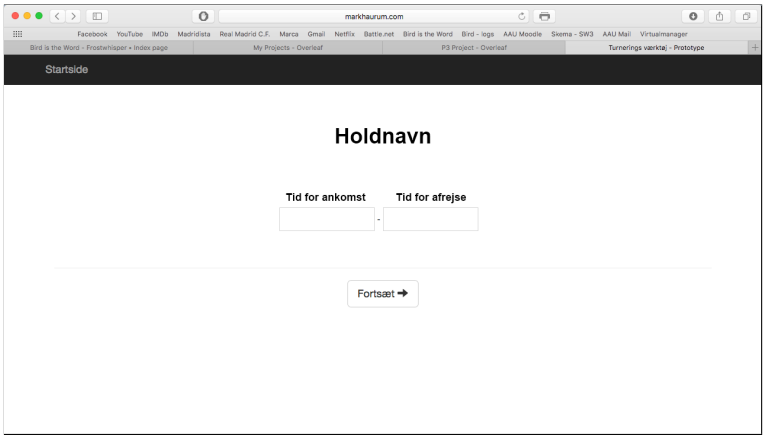


Figure 6.6: Teams

Figure 6.6 shows the page where a team can have their time of arrival and time of departure set, if necessary. If this isn't necessary the time will, by default, be set as the tournaments start and end time.

6.4 Summary

In this chapter the actor and it's use cases have been described. The group have only found one actor, called the tournament planner. Through the use cases there have been found functions which have been described and analysed and given a complexity. At the end of this chapter, the user interface has been described, including some mockups to clarify the idea. This leads to the next chapter which describes the system requirements and the criteria for the system.

7 | Design Specification

In the following section, the system requirements will be specified. It is important to have a set of requirements before implementing the system and each requirement will be examined after the product has been created to see if has been fulfilled.

7.1 System Requirements

1. The system must be designed as a web application.
2. The system must be able to save and edit tournament data provided by the user.
   (a) The user must be able to input the teams, fields, divisions and pools to the system via uploading an Excel sheet and/or manually entering them.
   (b) The user must be able to input the tournament structure for each division to the system.
   (c) The user must be able to edit any data provided by the user.
   (d) The user must be able to input the time period for the tournament.
   (e) The user must be able to select an individual arrival/departure time for each team.
3. The system must be able to generate a tournament schedule.

   Conditions for scheduling a tournament:

   (a) A team must not play two matches at the same time.
   (b) A team should not play during the break (a team has a break lasting one match length after each match).
   (c) Two matches must not be played at the same field at the same time.
   (d) The group stage matches for each division must be finished before any final stage matches in the same division can begin.
   (e) All group stage matches for teams must not all be played on the same day.

(f) The user should be able to choose between knockout and round-robin as final stages.

(g) Matches for a specific team or pool should be played at the same fields as earlier matches for that specific team or pool if possible.

4. The user must be able to edit the tournament schedule, by either moving a match to an empty slot, or swapping two matches.

5. The scheduling algorithm must not be interrupted by closing the browser.

6. The system must be able to output the tournament schedule in a format that can be viewed in a Microsoft Excel sheet.

7.2 Criteria

The following section will present the design criteria for the system. The criteria will be presented in table 7.1 and prioritized accordingly. The criteria table will prioritize the design goals and specify the focus when creating the product. Each criterion will fall under the following five categories: very important, important, less important, irrelevant and trivially fulfilled.

A criterion marked as very important means that the given criterion is essential for the system and must be fulfilled. A criterion marked as important means that it should be fulfilled, but is not essential for the system. If a criterion is marked as less important, it is because that given criterion can be a good thing to implement, but is not needed and should be fulfilled if all the more important criteria are fulfilled already and there is sufficient time left. A criterion marked as irrelevant means that it is not within the scope of the project and will not be taken into consideration at all when implementing. If a criterion is marked as trivially fulfilled, it means that the criterion in question is trivial and very easy to fulfil.

Table 7.1: Criteria table

Criteria	Very important	Important	Less important	Irrelevant	Easily fulfilled
Usable	✓				
Safe			✓		
Efficient			✓		
Correct	✓				
Reliable		✓			
Maintaniable				✓	
Testable		✓			
Flexible			✓		
Comprensehive		✓			
Reuseable				✓	
Portable				✓	
Interoperable				✓	

Usable Denotes a mixture between the user's needs and if they fit the technical platform. This criteria is very important because the user needs to be able to use the product for an important task and therefore it has to be very usable and the user needs to have a good experience using it.

Secure This criteria is about system security. This is a less important criteria for the program as the interest group is small, and the data is not sensitive. Only access to the tournament planning section has to be secured.

Efficient This criteria is about the system running time and it is a less important criteria as the planning of the tournament is done a long time beforehand.

Correct This criteria is very important because it needs to create a correct tournament schedule.

Reliable The system needs to be reliable and therefore this criteria is important. The user expects that the system works, which is what we have to ensure that it does.

Maintainable This criteria is irrelevant because the system is not going to be released and is more or less a finished product at release.

Testable It's important that the system is testable because the user's demands needs to be tested for flaws and errors to ensure the software is correct.

Flexible Denotes how easy system changes will be after it has been used. It's a less important criteria because system requirements may change after the users have used the system, but that is not something that we intend to implement in this project.

Comprehensive It's important that the system is comprehensive because all members have to understand the code and be capable to make any necessary changes.

Reusable This criteria is about reusing the system or parts of the system with other systems. This is an irrelevant criteria because the system is not going to be reused.

Portable This criteria is about portability to other technical platforms. It's irrelevant for us because the system is only going to be used on one platform.

Interoperable This criteria is about the costs to connecting the system to other systems. It's also about the system interface and how the developed system can integrate with other systems. But this criteria is not relevant for our system, because our system will not be integrated with other systems.

7.3 Summary

The system is going to be designed as a web application and it should be able to generate a correct tournament based on a set of conditions and criteria. An important system requirement will also be that the user will be able to put the teams, divisions and pools into the system. Furthermore, a criteria table has been specified and it was determined that it's very important that the product is usable and correct. A description of each criterion and a reason for the categorization have also been specified.

8 | Design

With the system requirements defined and the criteria prioritized, it's now time to design the system. Firstly, the component architecture will be described followed by a description of the technical platform. Finally, the user interface will be introduced and described.

8.1 Design process

The design process for this project was chosen to be the waterfall model. This is because the project group has more experience with this approach than the iterative model, and therefore this will be the easier approach. The iterative model is also more time consuming than the waterfall model, because the program has to be revised multiple times.

8.2 Component architecture

The component architecture is simply the architecture of each of the system components and their relationship to each other. The model and function components will be described in this section. By doing this, it will be easier to understand how the system works as each component and their relationship will be described.

As seen in figure 8.1 the system is divided into client, server and technical platform. Client contains the graphical user interface, the server contains the functions and the model and the technical platform contains the database. The arrow between components represents that the first component knows about and is reliant on the second component, therefore the first component can't function without the second component. Every component will be described in the following sections.

8.3 Component design

In this section the model and function components will be described. The class diagram from the problem domain has been revised and attributes have been added to some classes based on the events in the problem domain.

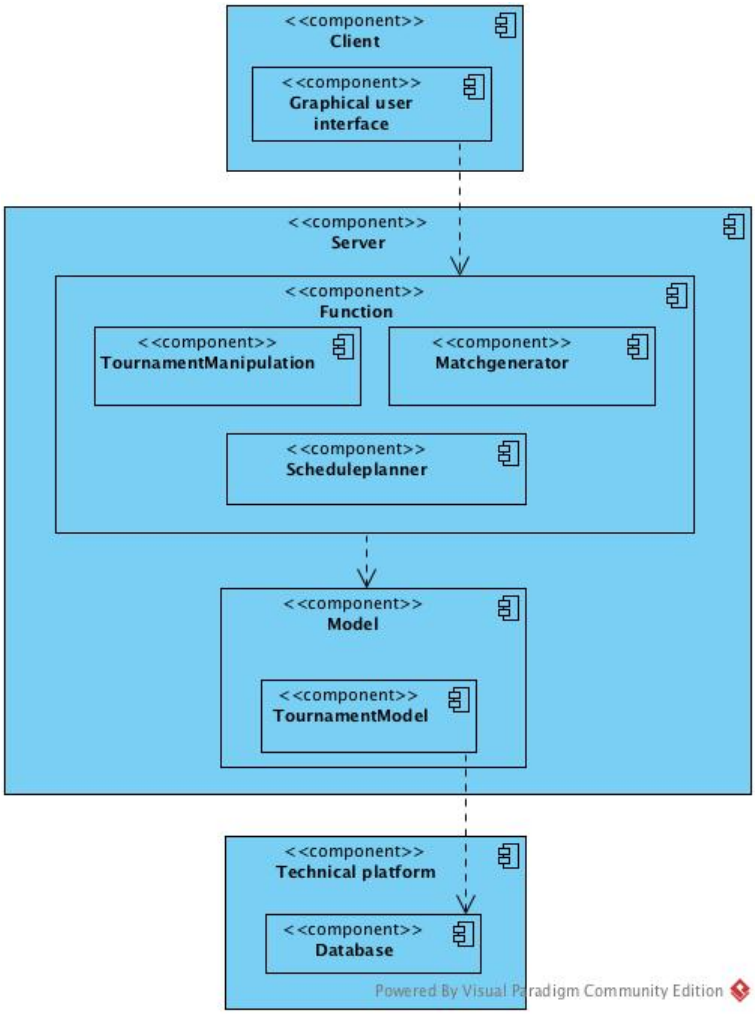


Figure 8.1: Component architecture

8.3.1 Model- & function component

In the model component there are different classes with attributes. They are connected with each other so that they all combined make up the parts of the whole system.

The function component determines the functionality of the system. It's split into 3 parts: TournamentManipulation, MatchGeneration and SchedulePlanning. TournamentManipulation has all the controllers for each model component. MatchGeneration will have all the functionality to generate the required matches. SchedulePlanning will handle the scheduling of the matches generated, including validating if a match can be scheduled at a specific time, which will also be used when a user tries to edit the schedule. TournamentManipulation is linked to the whole model as each model component has a controller connected to it. MatchGeneration and Sched-

ulePlanning are connected to the Tournament class, because both functions will operate on a tournament level.

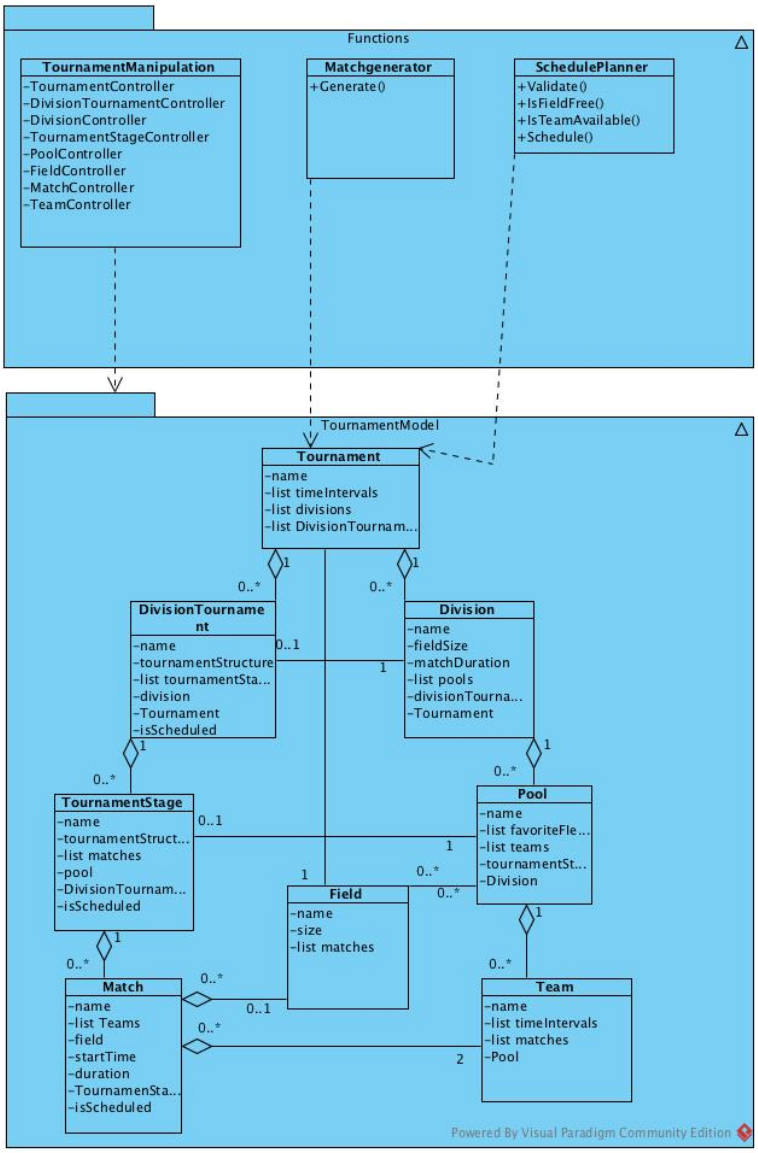


Figure 8.2: Model and function components

8.4 Technical platform

In this section the different tools used to develop the system for the project will be described.

8.4.1 Model-View-Controller

Model-View-Controller(MVC) is the architectural pattern which will be used in this project. The idea of MVC is that each component (model, view and controller) have individual tasks.[6]

Model is handling all the data in the system, and will get the requested information wherever it's stored.

View is where the user is presented the requested data found in the model and the controller have to choose between several different views.

Controller is handling the user requests, calling the right model and presenting the right view.

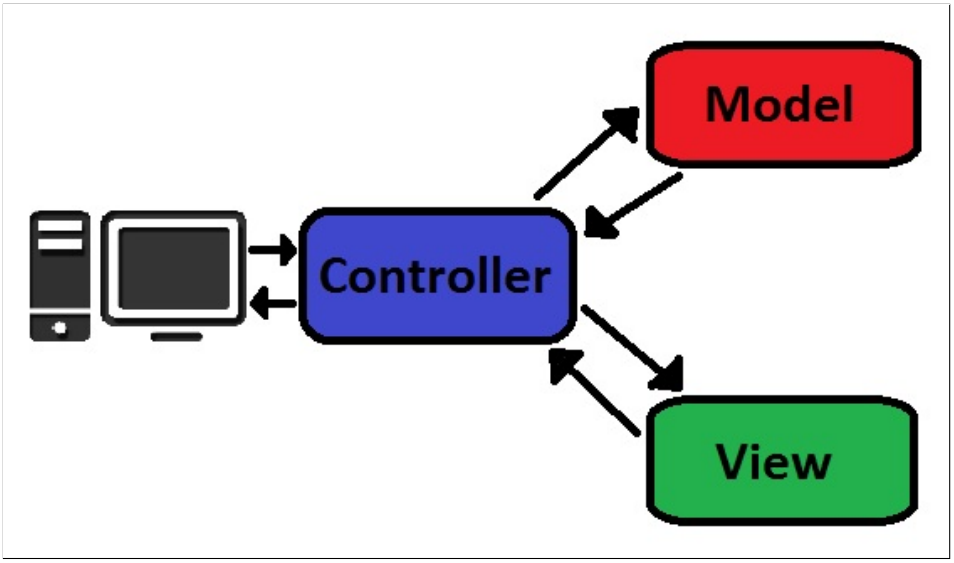


Figure 8.3: Diagram of MVC

As an example, let's say that the user requests to see all matches for a specific division. See figure 8.3. The user sends the request for the specific division to the controller. The controller then asks the model of the specific division, which the model then gets the information from where it's stored and returns it to the controller. The controller then selects the right view for the device which made the request e.g. phone, tablet or computer. The view is returned to the user through the controller again.[6]

8.4.2 AngularJS

Angular is a front end framework, built in JavaScript. In this project, it will work as the view part of the MVC-concept, but has in itself, both a controller and a view. The controller requests data from a back end system, the C# Asp.net MVC back end, and binds the data in a variable called scope, accessible to the views.

The view can then present the data using inline functions, much like Razor in ASP.NET MVC. The difference is that Angular is able to constantly

react to changes in the scope variable, and make changes to the view shown accordingly, while simultaneously communicating with the back end, all without needing to refresh the page.

By utilizing the functionality in Angular, the flow of the program can be optimized, to make it easier and quicker for the user to use the program. [7]

8.4.3 Bootstrap

Bootstrap is a HTML, CSS and JS framework for developing web sites or web applications and is developed by Twitter. Since none of the group members are experts in HTML and CSS, it was decided to use Bootstrap as the basis of the front end part of the project. Bootstrap is essentially a front end framework and makes it a lot easier for the group members to design the web application.[8]

Bootstrap contains HTML and CSS templates for all sorts of interface components, such as typography, forms, buttons, navigation and so on. Bootstrap will also make the website or web application fully responsive, meaning that the interface adapts to the screen size of the device.

8.4.4 Database

To store data permanently, the final system is going to use a relational database. A relational database consist of a number of tables, each representing a type of object, where each row is an object stored. By having extra columns, with the id's (unique key, used to identify an object) of other objects, the database is able to represent relations between objects, for example a team related to a match. [9]

To access, insert and manipulating the tables of the database, Structured Query Language (SQL) is used. SQL is a set of statements, for example SELECT and INSERT, used to access and insert new objects into the database. [9]

In C# and ASP.NET, the ADO.NET framework provides the functionality to connect to a database and perform SQL-queries, but to eliminate the need for the often difficult to write SQL-queries, an Object/Relation Mapping framework (O/RM or ORM) can be used to extent ADO.net. In this project, Entity Framework has been chosen as the ORM. [10]

Instead of SQL, Entity Framework uses Language-Integrated Query (LINQ). LINQ is a part of the C# language as opposed to SQL, and works much like lambda expressions. By doing it this way, Visual Studio provides syntax highlighting and makes it easier to debug. [11]

Entity Framework also makes it easier to work with the database, as it provides functionality to create the database structure either from standard C# classes, or from a designer build into Visual Studio. [12]

8.5 Mapping diagram

In the following paragraph, figure 8.4 will be explained, which will show the mapping of the different views for the system. The names of the views will be marked with bold text, making it easier to understand.

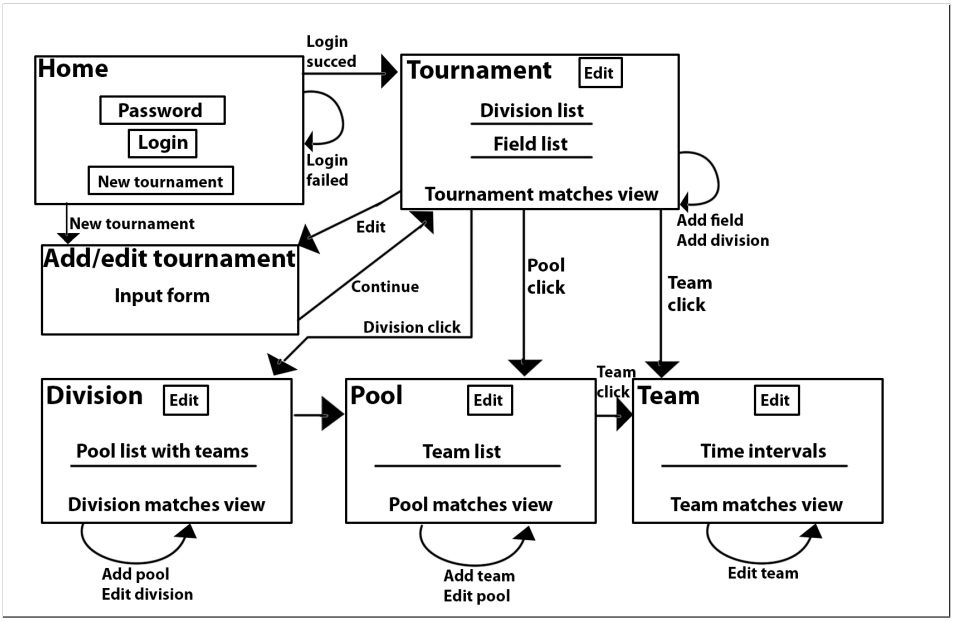


Figure 8.4: Mapping diagram of the user interface

Home is the homepage of the interface. On this page the user have two options: either login to an existing tournament with the correct password or create a new tournament. If the user decides to login in, he/she will be transferred to **Tournament**. If the user enters a wrong or non-existing password an error message will be shown and the user can try again. If the user chooses to make a new tournament he/she will be transferred to **Add/edit tournament**, where the user then have to fill out the forms and then continue to **Tournament**. If the user wants to change the information earlier entered for the tournament, **Tournament** have an edit option which will take the user to **Add/edit tournament**.

Tournament has a list of divisions, a list of fields and a view of all the matches in the tournament, but they will only be displayed when the tournament matches have been scheduled. In **Tournament** the user can add fields and divisions to the tournament. If the user then clicks the added divisions the user will be redirected to **Division** where the user can see a list of pools in a specific division. When the user is done, the user can go back to **Tournament**. If the user had already added some divisions to the tournament, then the user can click a division in **Tournament** and will be redirected to **Division**. In the bottom of **Tournament**, the user could find the view of all matches in the tournament and if the user clicks a specific

pool or a specific team, then the user will be taken to either **Pool** or **Team**.

Division contains a list of the pools including a list of teams for each pool. A view of all the matches for the specific division is shown in the bottom of the page as well. If the user clicks on a specific pool the user will be taken to the **Pool** page for that specific pool. The user also have the option to add a new pool to the division. This is done in the current interface, where the user will have to fill out the forms that are associated with creating a new pool.

Pool has a list of teams which have been added to that specific pool and also contains a pool match view, showing all matches for that specific pool. The user has the option to add a new team to the pool, which is handled in the same interface. If the user had already added some teams to the pool, then the user has the option to click on a specific team and will be taken to **Team** for the specific team.

In **Team** the user can set the time available for the specific team for each day of the whole tournament. If the user wishes to change a team's name, the user can use the edit button and will be able to edit the team's name in the current interface. In the bottom of **Team** the user can see a list of the specific teams and matches for the tournament.

The user will always have the option to go back from the different views, using the browser's back button or the breadcrumbs in the top of each page.

8.6 Summary

In this chapter, the architecture of the system have been described. The model and function component have also been described, including some images to give an overview of it. The different tools used to develop the system have also been addressed, like AngularJS, Bootstrap etc. There is also a section describing how the interface interact with each other from the user's input, in the mapping diagram section.

9 | Implementation

In this chapter the implementation of the system and the different choices made in the process will be described. The back end is made in C#, the front end is made with HTML5/CSS3, the JavaScript library AngularJS and the system uses a MySQL database to store the data for the system. All the comments in the system's code have been removed in the code snippets, and will be explained in the report instead.

9.1 Introduction to the program

In the following section the functionalities of the system/web application will be described. Some of the more interesting functionalities will be described with greater details than other functionalities.

The system made in this project is a tournament planning tool for the tournament planner of a sports tournament, where the informants in this project are planning a yearly football tournament. On the homepage of the web application, the user can log in to an existing tournament with their tournament password, which will connect the user to their tournament. On the homepage the user can also choose to create a new tournament, which will take the user to a new page, where the user will have to fill in a form containing: a tournament name, the aforementioned tournament password, the dates for the tournament and the time intervals for each day picked. The time intervals decides within what hours the matches can be scheduled on each date. If a tournament is set to be played over two days, the time interval for day 1 could be 14:00-20:00 and time interval for day 2 could be 8:00-13:00, and then the matches can only be scheduled between that time interval range. The user also has the option to upload the tournament's divisions, pools and teams from a Microsoft Excel Sheet, which will then be added to the tournament.

When the user has either logged in to an existing tournament or created a new tournament, the user will be taken to the tournament overview. On this page the user can see the tournament's divisions, fields and a schedule button, unless the matches have already been scheduled, then a view of the matches for all fields will be shown instead. New divisions can be added with a name, match duration and a field size. The new division will be added to the list of divisions in the view. Each field will be added to one of the three lists of fields, where each list contains fields of a specific field size. When

the user has added the desired amount of divisions, pools and teams, and has filled in the required information, the user can schedule all the matches for the tournament. When all the matches are scheduled, the view of the all the matches on the specific fields will show on the bottom of the page. If the user wants to change any information filled in when the tournament was created, the user can click the edit button next to the heading at the top of the page, and the user will then be taken to an edit page.

The next step for the user will be to fill out the pools for the divisions. If the user clicks on a division, the user will be taken to the specific division's overview. On the division page, the user can add pools to the division, change the match duration and field size made when the division was created, and an option to delete the division. If the user clicks the edit button next to the heading, the name of the division can also be changed. The user can also set which placement in each pool a team has to be to qualify to a specific user-set final stage. The user can also see a match view for all the matches in the division, which will only be shown when the matches have been scheduled, otherwise there is nothing to show. When the matches have been scheduled, new pools are added to the division, which will be the pools for the final stage.

When the user have created all the pools for the division, the user can click on a pool and will be taken to an overview page for the pool. On this page the user can add teams to the pool, delete the pool, and change the name of the pool, the same way as with the division. At the bottom of the page a view for all the matches in the pool can be found when the matches are scheduled.

When a team has been added to the pool, the user can click on the team and be sent to the team's overview page. On this page the user can change the name of the team in the same way as in pools and divisions, change the team's time intervals for each day, delete the team, and see all the teams matches in the match overview at the bottom of the page. As default, each team has the tournament's time intervals, but if a team is not able to show up until two hours after the tournament starts, then the team's matches will not be scheduled before the team's time intervals.

For navigation, the user can use the breadcrumbs at the top of the page, these links can take the user back to earlier pages the user was on, otherwise the user can use the browser's back button to go back to earlier pages. Also the match views for the divisions, pools, and teams all have a search bar, and if the headings in the table are clicked, it will filter for the clicked heading in alphabetic order, if clicked again it will filter it backwards.

9.2 Front end

The system's front end have been entirely developed with AngularJS and HTML5/CSS3. All the views for the system is written in HTML5 and styled with CSS3 where a simple Bootstrap template have been used for the general design. The logic for making the web application more dynamic is written in JavaScript using AngularJS Framework. [13]

9.2.1 Angular JS

To make the the web application more dynamic, controllers are written in JavaScript to manage the logic in the HTML views. The controllers are containing different functions which e.g. can have the logic of making a post request to the C# back end, calling the Create method in the TeamController, making a new object of a team and saving it to the database. The new team will then be shown in the specific pools list of teams in the front end.

```
1  $scope.addTeamToPool = function(name, index) {
2      $scope.buttonDisabled = true;
3      $http.post($rootScope.apiUrl + "/Team/Create", { name: name, poolId:
           $routeParams.poolId })
4      .success(function(data) {
5        if(data.status === "success"){
6          $scope.getPoolData();
7        } else {
8          $scope.error = "Kunne ikke oprette hold";
9          $scope.buttonDisabled = false;
10       }
11       $scope.buttonDisabled = false;
12     }).error(function(err){
13      $scope.deleteErr = err;
14     })
15     $scope.getPoolData();
16 }
17 $scope.buttonDisabled = false;
```

Listing 9.1: AngularJS addTeamToPool function in the PoolController

The scenario described in the previous paragraph is described in listing 9.1. In the view the user clicks a button which calls the function addTeamsToPool taking the parameters name and index, the name have been filled in by the user in a form in the view as well. The variable buttonDisabled are used so that if the user clicks the button several times, it will not add more than one team with the same name. The function will make a http.post request calling the Team class' Create function, which will make a new object of a team and save it to the database for the specific pool, identified by the pool's ID. Sometimes even though the request have been successful, it might not send the right data with it, which will be checked for in the if/else statement. If the object's status variable = success, the getPoolData function will be called, which will make the team appear in the pools list. If the status variable = error, it will send back an error message and make it possible to click the button again. If the request was an error, the error is sat to a scope variable which can be used as an output.

9.3 Back end

As mentioned before, the back end is developed in C#. First the class diagram was made in Visual Studio and then the database and classes were generated from this model. Not all the code will be explained in the follow-

ing sections, but the code not explained here can be found on the attached CD.

9.3.1 CRUD

Many of the controllers in the program have CRUD methods, which are Create, Read, Update and Delete as mentioned earlier in the report. TournamentController, DivisionController, TeamController, PoolController, FieldController, FinalsLinkController and MatchController are the controllers which implements the CRUD-methods to some extent. The implementation in this project rephrased the CRUD-methods, according to the following words: Create is still named create, read is rephrased to details, update is rephrased to edit, and delete is still named delete. All the CRUD-methods return a JSON object to the front end with a status parameter, indicating whether the CRUD-method was successful or not.

Create

The create method creates an object and the necessary references which gets stored in the database.

Details

The details method reads data from the database and sends the information back to the front end as a JSON object.

Edit

The edit method uses the stored information from the database, and is able to change the information with new user values. The database will then store the new information in the same object.

Delete

The delete method searches the database for the object corresponding to the ID sent with the post-request from the front end. When the object has been found, the method will delete all necessary data, the object and references. This will in most cases include deleting the schedule for a tournament, if one has been made.

9.3.2 FinalsLink

The FinalsLink class is used to link a specific pool-placement in a division to a final stage for the knockouts or round robins, to further qualify for the tournament finals. The FinalsLink has a Division object reference, which references the specific division the FinalsLink is connected to. The final stage variable is an enumeration-type, which indicates whether the FinalsLink is a knockout or a round robin. The ID is the identification for the FinalsLink,

and the PoolPlacement is the pool-placement indicator, indicating what pool-placements this FinalsLink belongs to, and which pool-placements that will be qualifying for the final stage of the FinalsLink.

9.3.3 TimeInterval

TimeIntervals are the schedulable times for a tournament. These are made from the user-inputs in the create/edit tournament. A tournament, and each team, has their own time-intervals, and the teams, as default, has the same time intervals as the tournament they belong to. If a team is late, or has to go early a certain day, the time intervals can be edited accordingly in the team-view. A time interval has a start-time and an end-time, determined by the user's input in the date pickers.

9.3.4 DivisionController.ChangeStructure

The controllers in the system all implements, or partly implements, the CRUD-methods, and some controllers implements other methods too. The division controller implements the ChangeStructure method, which needs the parameters divisionId and type ID. The divisionId is the identification of the instance of the division. The database container then edits the division's TournamentStructure, to match the typeId, which is cast to a TournamentStructure.

9.3.5 FieldController.GetAllTournamentFields

This method takes the parameter called tournamenId and tries to find a tournament corresponding to the tournamentId. When there are fields in the tournament, the tournament's field list will be copied to a new list, which contains the same matches as the tournament's fields. The tournament's time intervals will also be copied, which will be used to the match view in the tournament view. If anything fails it returns an error status in a JSON object, and that object contains a message, which reads "Could not find fields".

9.3.6 Validator.areTeamsFree

This method is a part of the validator class, which is used to validate the user input to make sure there is enough divisions, pools, teams and fields in the tournament to create a schedule. The areTeamsFree method uses two parameters: a Match and a start-time. The time intervals for the tournament (the start-time and date for the tournament) can be found through the match, and the first step of the areTeamsFree method checks if the given start-time parameter starts earlier than the tournament does. If this is true, the method returns a bool, with the value "false", indicating that the requested time is invalid. If it's not true, and the requested start-time therefore is later than the tournament's start-time, it will check whether any team's time intervals starts later than the requested start-time parameter.

If any team's start-time is earlier than the start-time parameter given to the method, it will return false as well. After this, the validator will check for each match for the each team, if the matches are available at the requested time.

9.3.7 Validator.isScheduleReady

This method returns a boolean "true" if the tournament consists of at least one field, one division, and one pool, containing at least two teams, and the division's field size must match at least one field's size. If any of these constraints are not fulfilled, the method will return false at the point of detection.

9.3.8 Schedule generation

When the user clicks the schedule tournament button ("Planlæg turneringsplan" on the web application) a number of algorithms are run one after the other. The following section will explain these algorithms and the order in which they are run.

The front end code is responsible for running the algorithms in the right order with the right parameters. This is done because it makes the program able to display to the user what part of the schedule is being processed at the moment, so that the users doesn't think that the system has stopped working. The first algorithm the front end calls is the DeleteSchedule algorithm, which deletes the current schedule if there is anything to delete.

If this is successful, either by successfully deleting the schedule or not finding any schedule, the front end will call the IsScheduleReady algorithm which checks the tournament to see if the basic requirements are satisfied.

```
GGS(tournament)
  foreach(division in tournament)
    foreach(pool in division)
      GRRM(pool.teams)
```

Listing 9.2: Pseudo code for GenerateGroupStage

If this is successful the front end calls the GenerateGroupStage algorithm seen in listing 9.2. This algorithm creates matches so that one team from each pool will play each other team in the same pool. It does this by calling GenerateRoundRobinMatches (GRRM) on each pool, as seen in the pseudocode in listing 9.3. GRRM goes through each team and matches it against each of the remaining teams, without matching the same teams against each other twice.

```
GRRM(list of teams)
  for(i = 0; i < number of teams in list of teams; i++)
    for(j = i; j < number of teams in list of teams; j++)
      create match between team i and j
```

Listing 9.3: Pseudo code for GenerateRoundRobinMatches

```
GFT(tournament)
  foreach(division in tournament)
    foreach(finalslink in division)
      if(finalStage in finalslink is different from the previous)
        create a new empty finalstage pool
      foreach(pool in division that is not a finalstage pool)
        create a new team representing the team that ended in the
            (finalslink.poolplacement) place in pool, add it to the
            most recent finalstage pool and set it's timeIntervals =
            sametimeInterval(pool)
```

Listing 9.4: Pseudo code for GenerateFinalsTeams

If this is successful the front end calls the GenerateFinalsTeams(listing 9.4) algorithm. This algorithm creates teams that represent each placement in each pool in the finals stages. In other words, this algorithm creates one team for each team in the tournament, the difference is that they do not represent a specific team but instead whatever team that gets the specific placement in their pool. As an example, take a pool with 5 teams in it, the algorithm will create a team named nr 1 from pool x, nr 2 from pool x . . . and lastly nr 5 from pool x. The algorithm also creates new pools representing each finals stage, according to the users wishes from the FinalsLinks of the division and adds the right teams so these pools. From the example above, say the user wants the top two teams in each pool to go to final stage A and the rest to final stage B. The algorithm will then create two pools, A and B, and add the teams accordingly. The algorithm uses the SameTimeInterval function to set the team's arrival and departure times to the highest arrival and lowest departure time of all teams because the program has no way of knowing which team will end in which placement.

```
GFM(tournament)
  foreach(division in tournament)
    foreach(pool in division that is not a finalstage pool)
      if(finals are round robin)
        GRRM(pool.teams)
      else
        GKOM(pool.teams)
```

Listing 9.5: Pseudo code for GenerateFinalsMatches

```
GKOM(teamList)
  x = 0
  while(2^x <= teamsList.count)
    x++
  if(2^(x-1) != teamsList.count)
    numOfExtraTeams = (teamsList.count - 2^(x-1)) * 2
```

```
    make a new list "extraTeams" with the (numOfExtraTeams) teams
        from teamsList with the lowest poolPlacement
    switch(2^(x-1))
      case 2:
        create a pool "KOPool" representing the semifinals
      case 4:
        create a pool "KOPool" representing the quarterfinals
      default:
        create a pool "KOPool" representing the round of (2^(x-1))
    sort extraTeams by their poolPlacement
    MatchMake(extraTeams, KOPool, teamsList)
  while(teamsList.count > 0)
    make a new list "teamsToAdd" and move all teams in teamsList into
        this list
    make a pool "KOPool" representing a finalstage corresponding to
        the number of teams in teamsToAdd (example: 2 would be
        finals, 8 would be quarterfinals)
    if(teamsToAdd.count = 2)
      add each team to KOPool and set their timeIntervals =
          sameTimeInterval(their previus pool)
      create a new match between the 2 teams
    else
      MatchMake(extraTeams, KOPool, teamsList)
```

Listing 9.6: Pseudo code for GenerateKnockOutMatches

If this is successful the front end calls the GenerateFinalsMatches(listing 9.5) algorithm. This algorithm goes through each division and creates the needed matches for the final stages. If a division has a final stage, set to be round robin, the algorithm calls GRRM, which is described above. If however the division is set to have knockout final stages the algorithm calls GenerateKnockoutMatches (GKM, in listing 9.6). GKM first calculates the lowest integer x such that 2^x is bigger than the number of teams. If 2^{x-1} is equal to the number of teams, the number of teams is ideal for knockout finals, because after each round the number of teams is reduced by half, and with 2^{x-1} teams that means this will end at one team at the end, which is exactly what is needed. If 2^{x-1} is not equal to the number of teams, the algorithm has to reduce the number of teams so that it gets a desirable amount of teams. It does this by calculating the difference between the number of teams and 2^{x-1} and multiplies this number by 2. This number is the amount of teams the algorithm takes from the original list of teams. This is done such that these teams can play each other and reduce the number of teams to a number that is exactly 2^{x-1} which is ideal for knockout finals. After taking these extra teams from the original list the algorithm figures out what pool to create, depending on the number of extra teams. Then the algorithm calls MatchMake, which makes new matches between 2 teams from the list, trying to avoid setting up 2 teams from the same pool and getting the best seeded teams against the worst seeded. it also makes new teams representing the winner of each match and adds them to the original list. After this the original list will contain 2^{x-1} teams, some of them

are from the pools and some of them are from winning the matches created with MatchMake. The rest of the algorithm uses two lists to form the rest of the knockout finals. The first list contains the teams, that are qualified to the next round, the other list contains the teams in the current round. The algorithm continues until there are no more teams for the next round. At first, the teams that qualifies for the next round are the ones in the original list. The algorithm now moves them to the list of current active teams, and does the same as above. It matches the teams against each other, and adds the winner of these matches to the original list, and these qualify to the next round. When two teams remain, the algorithm matches them against each other in the finals and stops by not adding any winner team to the original list.

```
MatchMake(teams, Pool, teamsList)
  for(i = 0; i < teams.count; i++)
    add teams i to Pool and set it's timeIntervals =
        sameTimeInterval(teams i's previus pool)
    if(teams i has no matches)
      for(j = teams.count - 1; j > i; j--)
        if(teams j has no matches and teams i and j are not from
            the same pool)
          create a match between teams i and j
          create a team representing the winner of the above match,
              add it to teamsList and set its previus pool to KOPool
          break
      if(teams i still has no matches)
        for(j = teams.count - 1; j > i; j--)
          if(teams j has no matches)
            create a match between teams i and j
            create a team representing the winner of the above
                match, add it to teamsList and set its previus pool
                to KOPool
            break
```

Listing 9.7: Pseudo code for MatchMake

If this is successful the front end calls the Schedule algorithm. First with 11 man matches, if successful it then tries to schedule the 8 man matches and if that is successful, it then tries with 5 man matches. The schedule algorithm first calculates the minimum amount of fields needed to contain every single match, and calls the ScheduleAll algorithm with this amount of fields. If the algorithm can't find a solution it then tries with one more field. This continues until adding one more field would exceed the number of fields available from the user, in this case the whole algorithm fails. The reason the algorithm is set up like this is to spread the matches over all the available days for the tournament. The users would rather have a field empty, than a day without any matches at all. By restricting the number of fields available to the algorithm, it forces the algorithm to spread the matches on a day-basis instead.

```
ScheduleAll(Tournament, FieldSize, NumberOfFields)
  make a number "selector" equal to 0
  make a number "indicator" equal to 1
  make a number "dayCount" equal to 1
  make a list of tournamentstages "TSToSchedule" and set et equal to
      all tournamentStages that represent groupstages in the
      tournament
  make another list of tournamentStages "unscheduleTS" and set i
      equal to TSToSchedule
  while(Tournament is not scheduled)
    update unscheduledTS so it contains only unscheduled
        tournamentstages
    if(unscheduledTS is empty)
      if(TSToSchedule contains groupstage tournamentStages)
        set TSToSchedule equal to the all tournamentstages that
            represent finalsStages
        continue with the while loop
      else
        set the tournament to scheduled, we are done

    sort unscheduledTS by number of matches in each tournamentStage,
        in decending order
    Select the (selector)'th tournamentStage "TS" from unscheduledTS
    if(any one team in TS has a previus pool which is not scheduled)
      selector++
    else
      make a list "unscheduledMs" equal to all matches from TS that
          are not schedule
      if(unscheduledMs is empty)
        set TS to scheduled and set its endtime to the last match's
            endtime
        continue with the while loop
      else if(indicator is bigger than 0)
        select the first match "matchToSchedule" in unscheduledMs
      else
        select the last match "matchToSchedule" in unscheduledMs
        selctor++

      make a list of fields "fields" equal to the (NumberOfFields)
          first fields with size equal to (FieldSize) from all fields
          in tournament
      for(i = 0; i < dayCount; i++)
        sort fields by number of matches on the i'th day of the
            turnament
        foreach(field in fields)
          if(areTeamsFree(matchToSchedule, field.nextfreetime for the
              i'th day))
            schedule matchToSchedule to be played on field at
                field.nextfreetime
            set field.nextfreetime to the endtime of matchToSchedule
            set indicator equal to 1
```

```
            set selector equal to 0
            break out of the foreach and for loop


    if(selector is bigger than the number of unscheduledTS)
      make a number "newDayCount" equal to 1
      make a number "k" equal to 0
      make a list of fields "fields" equal to the (NumberOfFields)
          first fields with size equal to (FieldSize) from all fields
          in tournament
      while(not done)
        k += 10
        if(all fields' nextfreetime + k are bigger than the
            tournament's endtime for the (newDayCount)'th day)
          if(newDayCount < dayCount)
            newDayCount++
            k = 0
            continue with the while loop
          else if(dayCount < number of days in the tournament)
            dayCount++
            done = true
            continue with the while loop
          else
            return false, algorithm found no solution

        sort fields by number of matches on the (newDayCount)'th day
            of the tournament
        foreach(matchToSchedule that is not scheduled and its
            prevpool is scheduled)
          foreach(field in fields)
            if(field.nextfreetime + k is bigger than the tournament's
                endtime for the (newDayCount)'th day)
              continue with the inner foreach loop
            if(areTeamsFree(matchToSchedule, field.nextfreetime for
                the (newDayCount'th day) + k )
              schedule matchToSchedule to be played on field at
                  field.nextfreetime
              set field.nextfreetime to the endtime of matchToSchedule
              set indicator equal to 1
              set selector equal to 0
              break out of the both foreach loops
    indicator *= -1
  return true
```

Listing 9.8: Pseudo code for ScheduleAll

The ScheduleAll(listing 9.8) algorithm tries to schedule all matches set to a specific field size in a tournament on a specific number of fields. This is done in a while loop that will run until every match is scheduled or till it determines that there is no solution. The first thing it does is to make a list of all tournamentStages that are groupstages, it then copies everything from this list into a list of unscheduled tournamentStages. Then it sorts the

unscheduled tournamentStages by number of matches in descending order, the groupstages with the most matches is scheduled first, this is done to avoid having only matches from one tournamentStage available to schedule at the end. It then selects one of the tournamentStages, preferably the first but if it doesnt work it will work its way through every tournamentStage in unscheduled tournamentStages. Now it checks that none of the teams in the tournamentStage's pool has a previous pool which is not scheduled, if any of them does, it counts up the variable "selector" so that the algorithm will choose the next tournamentStage in the list. If none of them have a previous pool which is not scheduled the algorithm will do the following:
Make a list of all unscheduled matches from the tournamentStages choosen above. If this list is empty, set the tournamentStage to scheduled, otherwise it will choose the first or last match in the list.

This is done so that the algorithm does not clutter up, because of the first few matches will contain the same teams making these matches impossible to schedule at the same time of right after one another. As it can be seen in the pseudocode(INDSÆT REF), when the algorithm tries to use the last match, it increments the selector as well to make the algorithm go to the next tournamentStage, if the last match could not get scheduled. The algorithm now tries to schedule the selected match. To do this, a daycount variable is made to ensure the algorithm fills out the first day as much as possible, before considering the next. The daycount is the number of days the algorithm should consider when scheduling at the particular moment in time. The algorithm always considers one day at a time. For each specific day the algorithm will make a list of all avaivable fields and sort them by number of matches on the specific day. It will then go through each of these fields and check whether the match can be scheduled at their NextFreeTime for the specific day, by calling the areTeamsFree algorithm. This algorithm checks that both teams are not playing or resting at the specific time and that the match will not exceed the time for either team's departure or arrival. If the algorithm returns true, the match will be scheduled at the specific time on the specific field, and that field's NextFreeTime will be moved forward, to make it point at the end-time of the newly scheduled match. All this happens if only the teams do not have a previous pool that is not scheduled.

Now if the match could not be scheduled anywhere and the selector variable has reached the end of the list of unscheduled tournamentStages, the following happens:

A list of all fields that are available to the algorithm is constructed and a variable k is introduced. The k variable will be used to make holes in the schedule, such that a match can be scheduled later, at another time than the current NextFreeTime for a specific field. This is done inside a while loop to force the algorithm to find a solution or stop altogether. In each iteration of the while loop the k variable will be counted up by 10 representing 10 minutes. At first the algorithm will try to schedule on the first day, until adding k minutes to all the NextFreeTimes would make every fields NextFreeTime exceed the end-time for the tournaments timer interval

for that particular day. If this happens the algorithm will try the next day, unless the daycount variable has not been counted up this far. If it hasn't, the algorithm will see if any more days are available to it, if there is one more day/are more days the algorithm will count up daycount and jump out of the while loop to let the regular schedule functions try with a day extra. If there are no more days available the algorithm has failed and will return false to indicate that it needs more fields or time to find a solution. If at least one field's NextFreeTime did not exceed the end-time for the tournament on the particular day, the algorithm then goes through every match that is able to be scheduled and tries it at every field's NextFreeTime added k minutes. If it finds a match it schedules the match and jumps out of the while loop. If no match can be scheduled anywhere the while loop continues and k get incremented by 10 and the process starts over until a match can be scheduled or the algorithm adds another day or it fails altogether. A flow diagram of the algorithms can be found in the Annex under section Flow diagram, which might give a better overview of the schedule generation.

9.3.9 Excel import and export

To implement Excel import and export in the final product, two different libraries where necessary. In C#, Microsoft has built-in functionality to work with Excel files, but it requires Excel to be installed on the machine, as it's only a way for the code to communicate with Excel. It doesn't in itself create, read or make changes to a document. For reading Excel files, ExcelDataReader was used, an open-source library found here: [14] (license can be found on the CD provided with the project). ExcelDataReader handles .xls and the newer .xlsx files, and works by loading the entire document into a DataSet object in C#. The data can then be accessed by regular dot notation, e.g. to access A1 on the first sheet in an Excel document, the statement would look like this:

string content = excelreader.Tables[0].Rows[0][0];

By having a predetermined format, of how the document should be structured with tournament information, pools and teams, the create function is able to create a complete tournament based on the sheet.

EPPlus [15] license can be found on the provided CD) was used for exporting the tournament to Excel, in a structure/format, compatible with piratliga.dk. The product can only export to .xlsx, as EPPlus only supports that format. To provide compatibility with .xls, another library would have to be implemented as well, but seeing as .xlsx has been standard on Excel applications since 2007, and Piratliga supports it, it wasn't prioritized. Writing to a document with EPPlus works by accessing cells with the same notation as used in Excel. To show an example, cell "A1" can be accessed as shown here:

string content = sheet.Cells["A1"].Value;

9.4 Working with the database

When the database structure has been created in Entity Framework designer, and “Generate Database” is pressed, a class deriving from System.Data.Entity.Context.DbContext is created, with a name specified during the setup of Entity Framework. In this case CupDBContainer. CupDBContainer contains information about the classes and how they map to the actual database, and has the functionality to read/write data to and from the database including how to handle the data. It uses the Connectionstring in the Web.Config file, to gain actual access to the database. When database access is required in a controller, an instance of CupDBContainer is created. An example of getting a single tournament based on an Id, getting all pools belonging to a tournament and setting the name of a tournament is provided below.

Initializing the CupDBContainer.
CubDBContainer db = new CubDBContainer();

Initializing tournamentId
Int tournamentId = 1;

Getting a single tournament:
Tournament tournament = db.TournamentSet.Find(tournamentId);

The statement “Find” matches a single tournament based on its Id.
Getting all pools from a tournament:
List<Pool> pools = db.PoolSet.Where(pool => pool.TournamentId == tournamentId);

The LINQ query matches all pools that have the same tournament Id as the variable set earlier.
Setting the name of a tournament:
tournament.name = “new name”;
db.Entry(tournament).State = EntityState.Modified;
db.SaveChanges();

The tournament is marked as changed by setting its state, and then making a call to db.SaveChanges(), to force a database update.

9.5 Screenshots of the application

In this section the application will be explained through screenshots of the system.

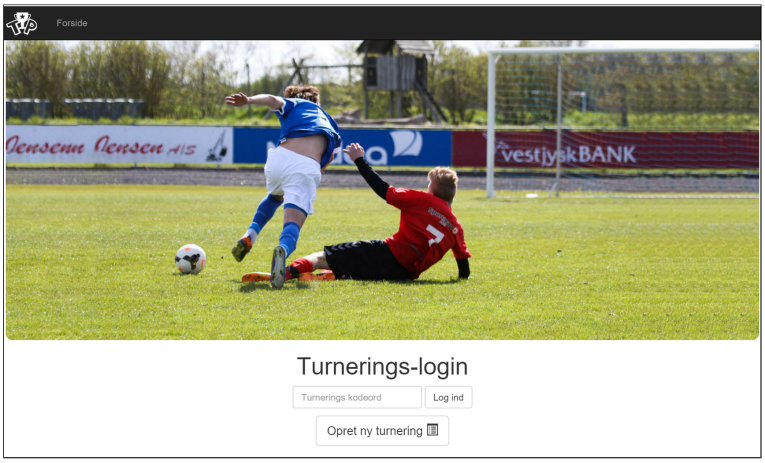


Figure 9.1: Login page

9.5.1 Login page

Figure 9.1 show the homepage where the user are able to login to an existing tournament, with an existing password. When the user clicks on the create tournament button, “Opret ny turnering”, the user will be able to create a new tournament.

9.5.2 Create tournament

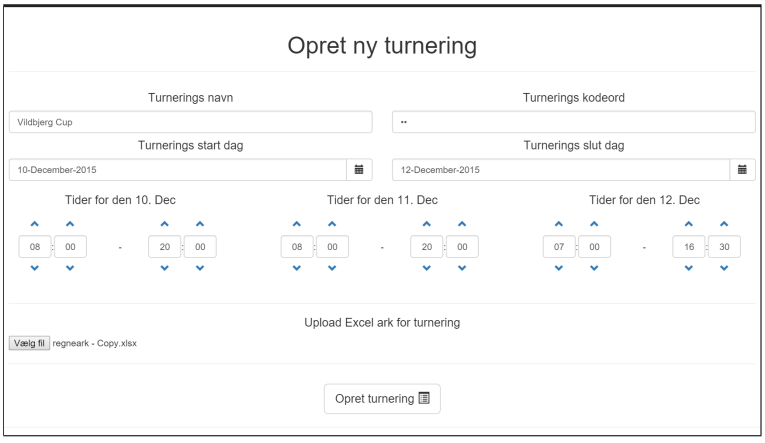


Figure 9.2: Create Tournament

Figure 9.2 shows the create tournament page. Here the user will see a form, where the user can fill in the name of the tournament, the password and the time intervals for the tournament. The user can then select one or more days with different start- and end-times. The user is also able to upload an excel sheet to the tournament. When the form is filled out, the user can click on the "Opret turnering" button and create the tournament.

9.5.3 Tournament view

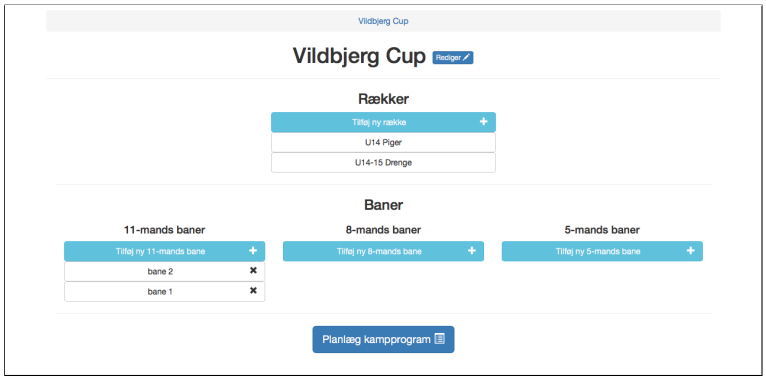


Figure 9.3: Tournament view

In figure 9.3 the user's tournament is shown. The user can create divisions, when the user clicks on "Tilføj ny række". The user can edit the tournament, by clicking on the button "Rediger". On the bottom of the page, the user have the possibility to add fields to the tournament, there are 3 different sizes of fields. The blue button at the bottom is the tournament schedule button, when the user click on that button it will shows the complete tournament schedule.

9.5.4 Add field

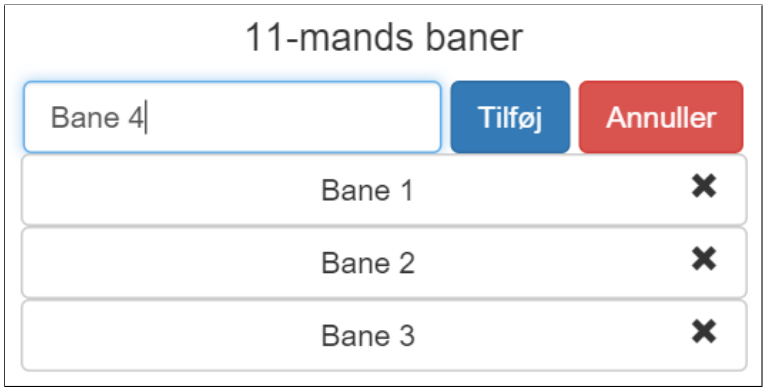


Figure 9.4: Add field

Figure 9.4 shows when a user is adding fields to the tournament, the user can click the add button "Tilføj". The user can see the added fields under the specific field size.

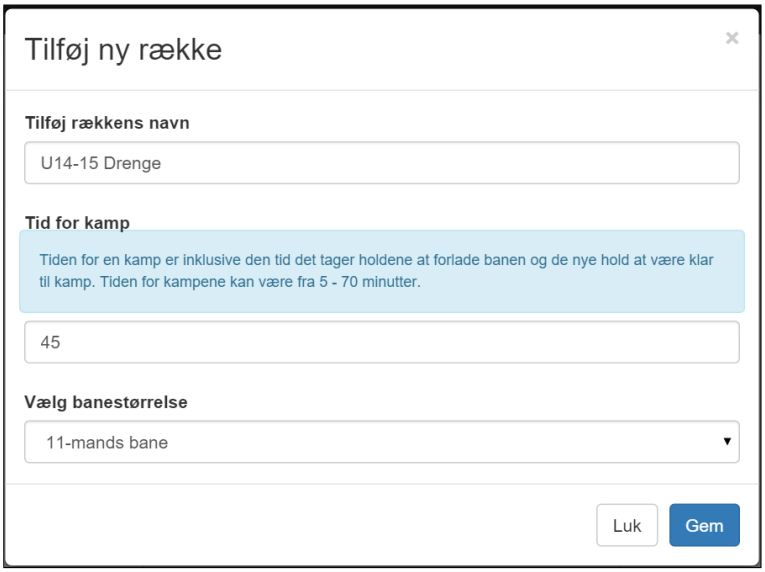


Figure 9.5: Add division

9.5.5 Add division

Figure 9.5 shows up by clicking on the add new division button “Tilføj ny række” from figure 9.3. The user will have the options to add a name, match duration and field size to the division. The user can submit the division by clicking “Gem”, or abandon the creating of the division by clicking the “Luk” button.

9.5.6 Division view

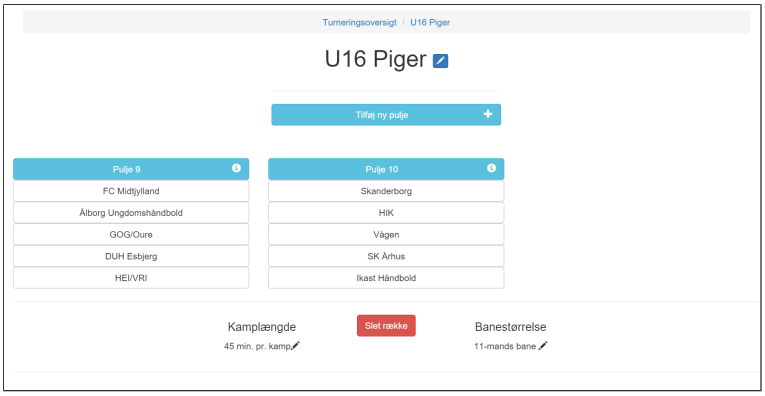


Figure 9.6: Division view

When the user clicks on a division that is created, the user will be transferred to an overview of that division as in figure 9.6. Here the user can add pools to the division, by clicking on the button "Tilføj ny pulje". After

the pools have been created, they will appear on the page. At the bottom of the page, the user has the option to delete the division and also the option to change the assigned field size, and also change the match duration for the division. At the top, there is an edit button, where the user can edit the name of the division.

9.5.7 FinalsLink settings

Figure 9.7: Finallink settings

Figure 9.7 show the FinalsLinks for the division. Before the user is able to schedule the matches, the user should set which pool placements are going to which final stage for each division. The user should also choose whether it's a knockout or round robin final stage.

9.5.8 Pool view

Figure 9.8: Pool view

When the user clicks on a pool, the user is transferred to the pools overview page as in figure 9.8. Here the user can add teams to the specific pool, change the pool name and delete the pool if needed. At the bottom of the page, when the schedule is created, a match view for the pools matches can be found.

9.5.9 Team view

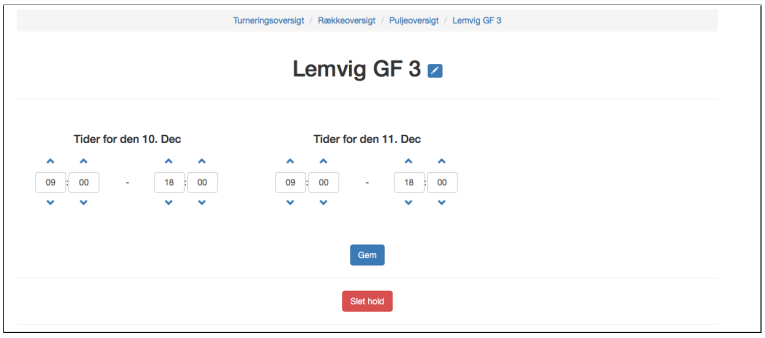


Figure 9.9: Team view

Figure 9.9 shows the team view, where the team can be edited. This page is accessible through clicking a team in the pool view. The user can edit the arrival and departure time for a team and the team name, or delete the team if they desire. When the team's arrival and departure times needs to be changed, the user has to remember to save the changes made by clicking the "Gem" button.

9.5.10 Tournament schedule

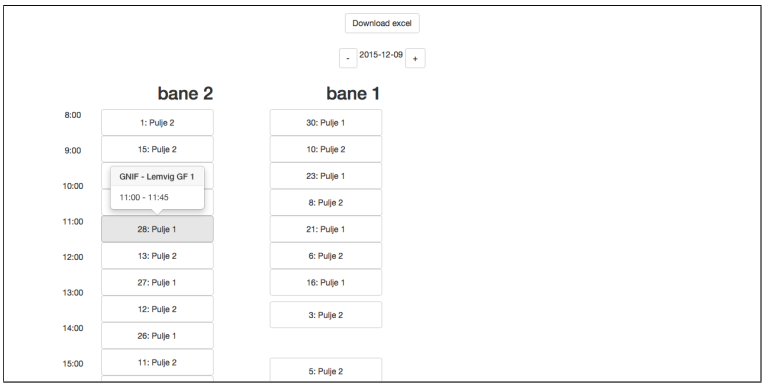


Figure 9.10: Tournament schedule

In figure 9.10 the user is able to see the whole tournament schedule, which will show the match number and time for the match, if the user hovers over a match, more information about the match will be shown. The user also has the option to download an output of the schedule, as an Excel sheet.

9.5.11 Division Schedule

The user is able to see the division schedule shown in figure 9.11, by clicking on a division in the tournament from figure 9.3. In the schedule the user

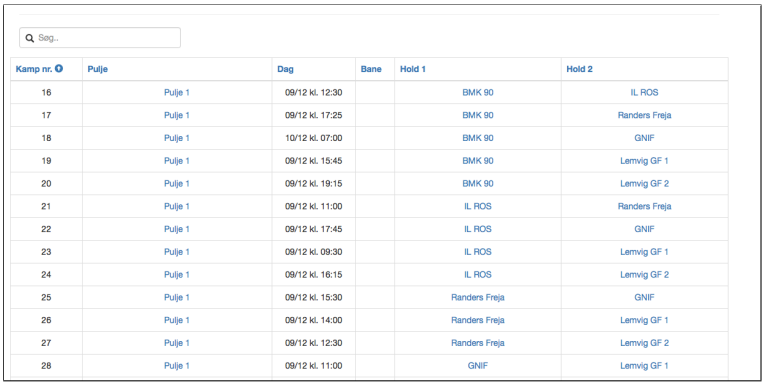


Figure 9.11: Division schedule

is able to see the whole schedule for that specific division. The schedule shows the match number, the pool, start time for the match, what field the match is played on and the two teams. In the pool and team view, similar schedules can be found.

10 | Test

Now that the system has been implemented, it's time to test the program to ensure that everything works as expected. The system will be tested thoroughly so that bugs and errors are found, as it's better to fix these now rather than after release. The system must be of high quality, meaning that the functions are correct and they work as intended. During the testing, the quality of the program will be evaluated and further improved after the issues have been taken care of. It will also be concluded in this section whether or not the system is as described in the system definition from before the implementation phase. Firstly, different kinds of tests will be described and then the ones that are relevant for this project will be chosen and performed, followed by a detailed description of the results.

10.1 Test types

There are many ways to test a program, but only a few will be described in this section. First, unit testing is done, which tests the smallest parts of the program called units, and it is done so using white box testing. Once the unit tests have been set up, the project group tested the program in black box testing. Lastly, before the program gets released, the end user has to test it in a user acceptance test (UAT) so that everything works as intended and final modifications can be made.

10.1.1 White Box Testing

In White Box Testing, the tester is aware of the internal code structure and chooses the inputs to test the software through different paths in the code and determine the outputs. The project group has tested the program using white box testing to ensure that there wasn't anything missing or wrong before the end user would test the program.

Unit testing

Unit tests were made for the project. The tests were made in Visual Studio using the built-in Unit Testing Framework. Most methods made changes to the database, which required the project group to store the ID when the

create functions in the controllers were tested, so these newly created entities could be manipulated within the other tests. In the listing 10.1 a Unit test from DivisionControllerTests.cs can be found, which is the CreateTest. The code is an example of a test from the system. This unit test tests if a new division can be added to the tournament with the proper information needed. In the second call it's testing if a new division can be added using null as the name for the division. The third call tests if the function can add a new division using a tournament ID that does not exist in the system. The first call should be successful while the second and third calls should both return an error, since it can't accept null as the division name for the second call, the tournament does not exist in the third call and therefore the division can't be assigned to it. Assert.AreEqual is used throughout the testing project, to check if the result is the same as the expected result. All of the conducted unit tests passed.

```
[TestMethod()]
public void CreateTest()
{
    //Create a new division
    dynamic jsonResult = ((JsonResult)controller.Create("U20
        Drenge", ID.TournamentId, 60,
        FieldSize.EightMan)).Data;
    ID.DivisionId = jsonResult.id;
    Assert.AreEqual("success", jsonResult.status);

    //Create a new division, using null values
    jsonResult = ((JsonResult)controller.Create(null,
        ID.TournamentId, 60, FieldSize.EightMan)).Data;
    Assert.AreEqual("error", jsonResult.status);

    //Create a new division, but to a non-existing tournament
    jsonResult = ((JsonResult)controller.Create("U20 Drenge",
        999999, 60, FieldSize.EightMan)).Data;
    Assert.AreEqual("error", jsonResult.status);
}
```

Listing 10.1: Example on a Unit test

The CreateTest method uses the controller to create a division and getting the result back as a JSON result, which is a dynamic type that gets created at runtime, because we already know the properties of the result in that particular controller method. We then set the ID of the newly created division to a global variable that can be used by other test functions to manipulate the division, and finally remove it again from the database at the end. The number 999999 is used afterwards to test for an error status, since there is no tournament with an ID of 999999 in the system. All the controller methods, except the "IdFromPass" method from TournamentController.cs, are using try and catch blocks, so they will all either return a status response of "success" or "error".

TournamentController

The tournament controller handles the returning the details of a tournament, creating, editing and deleting a tournament. Below is a list of the unit tests for the create function. A complete list of all the unit test and the results can be found in the Annex. The code for the unit tests can be found on the test project, which was handed in, along with this report. All the tests resulted in what was expected of them.

Test		Condition	Expected result	Result
1	Tournament-Controller			
1.1	Create			
1.1.1	Create new tournament		A new tournament will be added to the list of tournaments	Tournament was added
1.1.2	Create new tournament using null values		A new tournament will not be added to the list of tournaments	Tournament was not added
1.1.3	Create new tournament with existing password		A new tournament will not be added to the list of tournaments	Tournament was not added

Table 10.1: Create new tournament

10.1.2 Black Box Testing

In Black Box Testing, the tester is unaware of the internal code structure of the program and only knows the input and the expected output. This method of testing was used in both unit and user acceptance testing. User Acceptance Testing can be seen as a type of Black Box Testing because during the unit acceptance testing, the user isn't aware of any code base of the program when testing, but only the requirements which the software should meet. When the tester is conducting the user acceptance tests, the tester is only aware of what the program is supposed to do, but the tester is not aware of how it's being done.

User acceptance test

When the project group has tested it thoroughly and fixed errors and bugs that have surfaced in the previous testing phases, it's now time for the end

user to test the product. The process of verifying that the solution works for the user is called User Acceptance Testing. It's important that the system lives up to the user's expectations and the system definition. When the users have tested the program and everything works as expected, the testing process is done and the product can be released. User acceptance tests are useful because of several reasons[16]:

- They provide an overview of how much the system lives up to the user's expectations
- They identify problems that unit testing and integration testing have missed
- They capture user requirements in a directly verifiable way

The user acceptance test we've conducted, includes two parts. First a preparation phase, where we've made some realistic tasks for the users, so the results we get from the tests are actually useful. The second part is the results of the tests. The user acceptance tests were conducted with Mikkel Hessellund from Hinnerup Cup and Jan Elhom from Piratliga.dk.

Preparation phase

Prior to conducting the interviews with the end users, we made a sheet containing different realistic tasks (See Annex 14.3), that we've send to the end users a day before the interview, so that the users had the opportunity to be prepared for the interviews.

The test was split into two parts as well, where the first part is about the end users making their own tournament from scratch, using the tools we've implemented. The user also had to to make some errors, and see if the system could handle the errors. The last part is about the end users using an Excel file (that we've made for them), and schedule the tournament plan, just by using that file.

Before we started conducting the interviews, we asked the end users to read out loud the questions, the way they understood them, and try to tell us what they're thinking while performing the tasks. The reason behind this, is so that we get their perspective of the system, instead of having us simplifying every step for them, as we know how the system is supposed to work.

After each of the end user test, we asked the users on their thoughts about the system, if there were anything they felt was missing and/or if there is anything we could do better. The reason why we asked for these questions, is so that we can have something to work with, other than analysing the recorded conversation.

Results

When the two interviews had been conducted, we had stumbled across some errors/problems. In our Annex section, all the errors and problems can be found, we'll only be discussing the more important errors.

The errors and problems that the users had stumbled on can be divided into three categories: Cosmetic, serious and critical.

- Cosmetic: Low delay (>1 minute), low annoyance.
- Serious: Medium delay (<1 minute), medium annoyance.
- Critical: High delay or not executable, high annoyance.

In the conducted interviews both of the end users had one critical error, one serious error and different cosmetic errors/suggestions. The critical error is the most important one, and since both of them had the same error, that was our main priority of errors to fix.

Down below there is a description of the critical error, the serious error and one cosmetic problem, the rest are described in the Annex 14.2 section.

Problem 1 is categorised as a critical error, because it's the main function of our system. When the users had created a tournament and added a division with a couple of pools that includes teams, and added a decent amount of fields and tried to schedule the tournament, it couldn't. The reason behind this, can be two things. The first possible error, is that if there is more teams in one pool than the others, the users doesn't seem to understand the finalslinks very well, as they let every team in the pool play against the other teams in the other pools with the same position, which means, that if there are 4 teams in pool 1 and 2, and 5 in pool 3, then team 5 in pool 3 will go to a final stage alone, which isn't possible. Therefore there should be something in the code handling this issue. The other issue, can be that there isn't enough time on the selected days, and therefore, the user have to add an extra day to the tournament. The way we show this information at the moment of the tests, we're saying that it could be either that or that there are missing fields, so the users just keep adding fields, which won't necessarily help. Therefore, we should also make it more clear, if the problem is that there isn't enough time or that it's the fields that are missing.
The screenshot 10.1, shows how the first user, Mikkel Hessellund failed to schedule his tournament, while doing the tasks that the group have made, even though the fact that he had added it correctly.

Problem 3 is categorised as a serious issue as it's part of problem 1 and can confuse the user, because if he had 5 teams in one pool, and 4 teams in two other pools, then if the user delete the 5th team, the finalslinks will still be visible for the 5th position, which don't make any sense. The system should handle this, when a team is deleted, which may help solve problem 1.
In the screenshot 10.2, it is shown, that there is a finalslink for the 5th team, even though that the 5th team was deleted by the user, a couple of minutes ago.

Problem 11 is a cosmetic error. When the users tries to edit the match

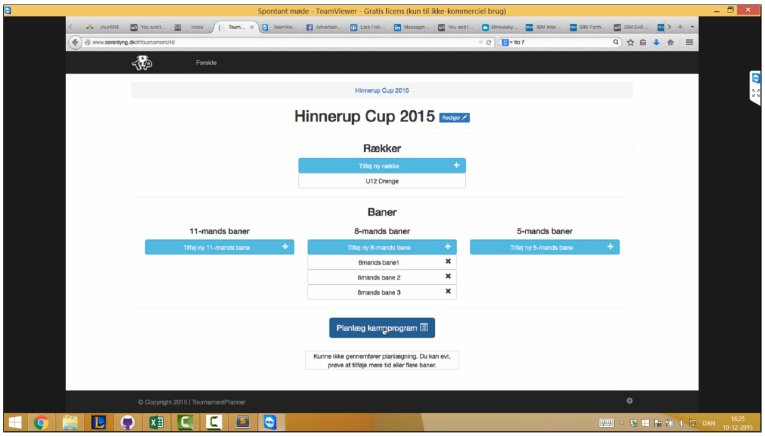


Figure 10.1: Scheduling failed after during the user acceptance test

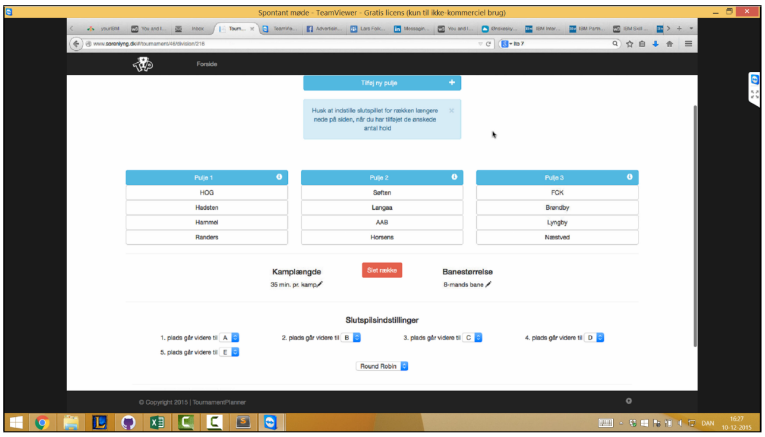


Figure 10.2: The finalslinks for the 5th team is still there

duration of a division by using the arrows, it doesn't work, as it doesn't use the current match duration as text, but just as a placeholder. This should be be handled, so that when the user clicks on the edit button he gets the current match duration as text instead of a empty text-box. Also when creating a new division, if the arrows are used, the value can go down to negative values, which aren't the idea.
The screenshot 10.3, shows that the match duration for the row being added to the tournament is negative, because Jan Elhom used the arrows instead of typing it himself.

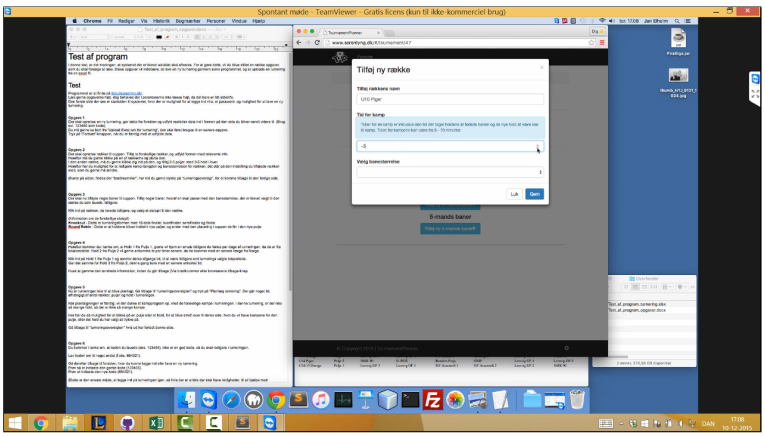


Figure 10.3: Negative match duration value appears when using the down arrow

10.2 Summary

In this chapter, unit testing and user acceptance testing were conducted. The unit tests tested the controllers and the helper classes to ensure that the units were properly handled. The input was fixed and the output was verified with Asserts to ensure that everything works as planned and that the quality is ensured. User acceptance tests were also done so that the end user could test the program before release and allowed us to get somewhat of a guess of how much the system lives up to the user's standards and works as the user expects.

11 | Discussion

In the following section, the requirements which were partially passed or not fulfilled, will be discussed. In the conclusion chapter, the table shows if the requirements were passed, partially or not fulfilled.

All group stage matches on one day

This requirement was not implemented in the system. When the group got to the implementation part of the project, and thought the requirements through, this requirement didn't seem usable. First of all it is possible for the tournament planner, to make a tournament only lasting one day, then the group stages can't be spilt out on more days, and the group stages and final stages will have to be on the same day. If the tournament lasted two days, and would have round robin as it's final stage. If the group stage matches can fill out the first day alone and still have to be spilt out over two days, then there will not be room for the round robin matches. The group then decided not to implement this requirement into the projects system.

Edit schedule functionality

In the beginning of the project, the plan was to implement a function for the user, to be able to edit the schedule the systems algorithm would produce. The intention was for the user not to schedule the whole thing, more than once, and then be able to use a drag and drop function to either switch the matches around, or have a box to drag the matches to temporarily. This function would make the schedule more flexible for the user, getting the schedule exactly as the user wants. The function would have been a big part of the system, but the project group didn't have the time to implement this to the system.

Not interruptible while scheduling

When the user presses the schedule button, the system calls the back end a number of times in a sequence. This makes it easier to provide an overview of how far the system has come with the scheduling algorithm. Unfortunately, it also prevents the algorithm from finishing should the user close the browser during scheduling, as all requests has to be send. This goes against the criteria, that the scheduling algorithm must finish, even if the

browser is closed, but we chose to prioritize a way for the user to see the progress of the algorithm than keeping the algorithm running.

Tournament passwords

As of now, a tournament has a password, but in this case, the password is not for safety, it is for sharing the tournament, and giving the user an easier way to enter the tournament, than with a long list, where the user has to search for the specific tournament. If a password would be implemented for safety, an ideal way would be to use an already existing login system, using sessions. So far, there has been no lectures in login safety, and therefore the project group has little to no experience with these systems. If safety is critical, this would need expertise and experience within the topic. The current state makes the user able to hand the password to volunteers or others, to let them view and edit the tournament. Safety was never a critical topic of this project, and that is why the tournaments can be access without the passwords, so the data is never safe. The password is only a gateway to the tournament, it is not an obstacle to enter the tournament and data.

Favourite field

One of the requirements was: “Matches for a specific team or pool should be played at the same fields as earlier matches for that specific team or pool if possible.”, this was not implemented to the system. At first it this was implemented, but because of the algorithm this was removed. In the beginning the user had the option to select some specific fields for a pool, which the pool’s team’s matches would be prioritized on. Later this was done in the algorithm, but it was completely removed later on. The biggest reasons this functionality was removed, was that the algorithm could not handle scheduling a tournament with one division and one pool, it would prioritize one field and fail because of the time constraint of the tournament. The scheduling could have succeeded if it had spread the matches on more fields, but with the favourite field functionality the algorithm would fail, no matter how many fields there was available. Therefore this was removed to make the algorithm able to schedule the matches for all the fields. The ideal solution to this project would include the favorite field functionality, since it was a requirement from the user.

12 | Conclusion

This conclusion chapter will be the culmination of the project and its chapters, where a conclusion on the projects processes and the systems requirements fulfilment is described. The discussion chapter, have described what could have been done in other manners, regarding the execution of the system requirements.

After a completed introduction to the topic 'tournaments', the project has revolve around the problem statement:

How can a software application plan a tournament schedule, based on a number of teams sorted in divisions and pools?

Through interviews, the topic was researched, and the two interviews done in this project, lead to the system choices chapter. In this chapter, the people, activities, contexts, technologies and role models lead to the system definition. After ended system choice chapter, the problem domain was analyzed, picturing the objects in this domain, and describing the behavioural patterns, and events, of these objects. Along with the problem domain, the application domain was described, mapping the use (actors, use cases) of the system, the functions of the system and the user interface of the system, leading to the design specification. This specification included the system requirements, and the criteria for the system. After the design specification, the design of the system was described, modelling the component architecture, component design and technical platform. The previous chapters, including the design chapter, all contributed to the implementation chapter, showcasing some of the code for the system, and describing essential and interesting parts of the implementation of the system. This included the front end, back end, database and the user interface, including screenshots of the systems interface. Implementation in itself was not sufficient for fulfilling the system requirements, this also needed documentation through testing, which is the chapter after the implementation. This project included two test-types: unit testing and user acceptance test. These tests would be the documentation for the fulfilment of the system requirements, which this chapter will describe and conclude upon.

12.0.1 Requirements

Table of the requirements from the Design specification chapter.

Requirement 2 was partially passed because of the sub item 2.c, since

		Requirement	**Status**
1		The system must be designed as a web application	Passed
2		The system must be able to save and edit tournament data provided by the user	Partially
	a	The user must be able to input the teams, fields, divisions and pools to the system via uploading an Excel sheet and/or manually entering them	Passed
	b	The user must be able to input the tournament structure for each division to the system	Passed
	c	The user must be able to edit any data provided by the user	Partially
	d	The user must be able to input the time period for the tournament	Passed
	e	The user must be able to select an individual arrival/departure time for each team	Passed
3		The system must be able to generate a tournament schedule	Partially
	a	A team must not play two matches at the same time	Passed
	b	A team should not play during the break (a team has a break lasting one match length after each match)	Passed
	c	Two matches must not be played at the same field at the same time	Passed
	d	The group stage matches for each division must be finished before any final stage matches in the same division can begin	Passed
	e	All group stage matches for teams must not all be played in the same day	Not fulfilled
	f	The user should be able to choose between knockout and round-robin as final stages	Passed
	g	Matches for a specific team or pool should be played at the same fields as earlier matches for that specific team or pool if possible.	Not fulfilled
4		The user must be able to edit the tournament schedule, by either moving a match to an empty slot, or swapping two matches	Not fulfilled
5		The scheduling algorithm must not be interrupted by closing the browser	Not fulfilled
6		The system must be able to output the tournament schedule in a format that can be viewed in a Microsoft Excel sheet	Passed

Table 12.1: Conclusion of the requirements from the design specification chapter

it's not possible for the user to edit the fields when they have been added to the tournament. If the user named the field wrong and they want to change the name, they will have to delete the field and then add a new field with the desired name.

Requirement 3 was partially passed because of sub item 3.e. This requirement was intentionally not implemented. A tournament can last only for one day, and therefore will all the matches be played on that day. Also if it's a two day tournament, and the final stages will also be played as round robin as the group stages, it isn't sure if all the matches can be be scheduled during the two days. Also requirement 3.g was not fulfilled, as mentioned in the discussion chapter.

Requirement 4 was not fulfilled. Due to lack of remaining time, this requirement was not implemented.

Requirement 5 was not fulfilled. When the schedule button "Planlæg kampprogram" is pressed, it will make a request to the back end, if this is successful, it will make another request and so on. Therefore if the user closes the browser during scheduling, the system will not make all the necessary request to the back end, to schedule the matches.

A sizeable system have been developed in C# for the back end, using the JavaScript Library AngularJS and HTML5/CSS3 for the front end. Most of the requirements from the design specification chapter have been implemented to the system, a few larger requirement have not been fulfilled though, these will be discussed in the following section.

13 | Future work

After completed implementation and testing, the future work for the project will now be described. This chapter will explain the future work to implement other features, edit some of the existing features and showcase how the user expectations on other aspects could be satisfied. This chapter aims at explaining the essence and theory of the future work, but does not showcase how it should be implemented. The bugs and optimizations of the system is also part of the future work, but this chapter does not include those two things. The choice of not including those two were based on the thought, that any system can be optimized and debugged after any completed version of the system, and therefore a system is always able for improvement.

Edit the schedule

In the reports design specification one of the requirements for the system was “The user must be able to edit the tournament schedule, by either moving a match to an empty slot, or swapping two matches”, as mentioned before this was not implemented. How the system looks like at the moment it doesn’t allow the user to edit in the generated schedule, only by deleting the teams in the pool and adding them in a different order.

In future work the group would like to have implemented an edit view/-function for the schedule. Either the user should be able to drag one match on top of another, release it on the other match and they should then switch place. There should then be a validator which will indicate if this is an legal action. The validator will check if the teams have a match at the same time as the new or if they had a match right before the new match, since they need a match length break between each of their matches.

The other option could be to take it user to an edit view. There should be a box where the user can drag matches to temporarily, for then switching the matches around as the user likes. Again there should be a validator which should indicate if how the schedules matches have been moved around is allowed. As before it should validate if the teams have a match at the same time as the new or if they had a match right before the new match. The validator should also make sure if no matches are left in the box where the matches should only be temporarily.

Result processing

At this point the system are generating the schedule for the tournament and the user can then download the tournament schedule for Microsoft Excel. The group have tried to make the Excel output look like the Excel input Piratliga.dk uses to do the result processing, but in future work the group would like to make it available for user to do the reuslt processing in the system.

An whole new view should be created for each pool, showing the group standings for each team, who will proceed to which final stage and the result processing of the final stages as well. The group standing is a table of all the teams in a pool, showing their goals score, goals against, goal difference, wins, draws, losses and their points.

The results should most likely be possible to sent in by a mobile application, where the result of a match, identified on the match id, would be send in and the standings would be calculated each time new data was received.

A view should then be available for the players, coaches, parents and other people who have an interest on the outcome of the cup, could be able to check the results of the matches and see the winner of the different division.

Validate final stages

In our current version of the system, we're only handling part of validation of the final stages. The part that is missing, is the part that checks if there is a team assigned to a finals-link alone. This should return an error message to the user, or make the schedule button not available until it's fixed, as sending one team to a finals-link alone, would mean that the team is out of the tournament after the group stage, which isn't the purpose of it. Therefore, this is also something that should be implemented in the near future.

Field overview

Like the current division match overview, there should be a view for each field in the near future. This idea came from one of the end users, during our user acceptance test. He suggested that this view would help them a lot when they're dealing with the assignments of the referees, as the referees are most likely to stay on the same field for 4-6 matches at the time, and therefore he should have an overview of who is going to to play who, in case one of the teams doesn't show up. The group had already talked about the referees at the very beginning of the project, like if we should also try to make it possible to schedule referees into the tournament, but we quickly decided to leave them out, and focus on the match scheduling part instead. But after we talked with Mikkel from Hinnerup Cup, an overview showing all the fields like the division overview, seemed to be very important for him and therefore it's something that should be considered for a future version of the current system.

Browser compatibility

As of now, the current version of the system may look different depending on which browser it's opened on, and which resolution the screen of the computer has. This is something that should be implemented better in a future version, to ensure that it looks the same and is compatible, no matter which browser or resolution that is being used by the user.

Login system

The current login system in the web-application include very little security. It is required that a password must be entered for being able to enter a tournament, but all that does is actually only redirecting the user to the tournament ID that fits the password. In the URL it's possible to change the ID number to another number, to enter other tournaments without even knowing the password for that tournament. Also when editing a tournament password, the system only asks for a new password, it doesn't ask for the old password, before it's able to change it. Also when creating a tournament with an already used password, the error message is that, that password is already being used, which means, that the user now has access to another tournament. In a future version of the system, there should be a better login system, containing either an email or username. It also shouldn't be possible to just edit the URL to a different ID. That way the security of the individual tournament will be better and safer.

Dynamic handling of user inputs

The handling of inputs, such as the datepicker calendar; when the user sets the start-time and end-time for the first day, the rest should default to the same. If a user has a cup spanning ten days, it would be much easier to put in the first day, and edit the ones that are unlike the first. Or, as one of the user tests stated, a weekend date would have a default time, and a weekday date would have a certain default time as well. If the user picks a date from a month that is different from the current when editing, the next datepickers, for all the other days, should also change the default value for the month to the same as the user-picked month. The current state, as the user has to change the month in every datepicker, could lead to wrong user-input, as they get confused, or do not notice the month difference. Editing the dates would reset all the start- and end-times, which is not needed. The users requested that it would keep the times for the days that are not removed.

The user cannot edit a field name by now, so this would be another part of the future work. After the schedule is done, the user cannot edit the field name, without deleting the whole schedule, due to the need to delete a field, and create a new one, to 'edit' the name. In-line editing was suggested by one of the users under the user tests, which would be included in all name-editing. While editing the match duration of a division, the value of the current match duration should be used, instead of starting from zero. When

the user wants to add something new, clicking the buttons for adding the item, doesn't mark the text-field. Users in the user tests started writing without clicking on the text-field, so marking the text-field on pop-up would be a great solution for the user.

Bibliography

[1] Lars Mathiassen, Andreas Munk-Madsen, Peter Axel Nielsen, and Jan Stage. *Objekt Orienteret Analyse & Design*. Marko ApS, 2000.

[2] Douglas Hughey. The traditional waterfall approach. http://www.umsl.edu/~hugheyd/is6840/waterfall.html. Accessed: 15-12-2015.

[3] Dansk Idræts Forbund. Medlemstal. http://www.dif.dk/da/om_dif/medlemstal. Accessed: 25-9-2015.

[4] Vildbjerg Cup. Om vildbjerg cup. http://www.vildbjerg-cup.dk/om-vildbjerg-cup-9. Accessed: 25-09-2015.

[5] Colour Blind Awareness. Colour blind awareness. www.colourblindawareness.org/. Accessed: 30-09-2015.

[6] Pablo Pastor. Model-view-controller. http://code.tutsplus.com/tutorials/mvc-for-noobs--net-10488. Accessed: 5-11-2015.

[7] Google-Angular. Angular. Accessed: 6-11-2015.

[8] Mark Otto & Jacob Thornton. Bootstrap. http://getbootstrap.com/. Accessed: 5-11-2015.

[9] Oracle. Database. Accessed: 6-11-2015.

[10] Wikipedia. Ado.net. https://en.wikipedia.org/wiki/ADO.NET. Accessed: 6-11-2015.

[11] Microsoft. Linq. https://msdn.microsoft.com/da-dk/library/bb397926.aspx. Accessed: 6-11-2015.

[12] entityframeworktutorial.net. Entity framework. http://www.entityframeworktutorial.net/what-is-entityframework.aspx. Accessed: 6-11-2015.

[13] AngularJS. What is angular? https://docs.angularjs.org/guide/introduction. Accessed: 11-12-2015.

[14] Excel Data Reader. Exceldatareader. https://github.com/ExcelDataReader/ExcelDataReader. Accessed: 01-12-2015.

[15] EPPlus. Epplus. http://epplus.codeplex.com/. Accessed: 01-12-2015.

[16] Thomas Peham. User acceptance testing. http://usersnap.com/blog/types-user-acceptance-tests-frameworks/. Accessed: 30-11-2015.

14 | Annex

14.1 Unit tests

14.1.1 TournamentController

TournamentController is handling the details, creating, editing and deleting a tournament.

Create

Is creating the tournament and adding it to a list of tournaments, if it is not created with an existing password.

Test		Condition	Expected result	Result
1	Tournament - Controller			
1.1	Create			
1.1.1	Create new tournament		A new tournament will be added to the list of tournaments	Tournament was added
1.1.2	Create new tournament with null as tournament name		A new tournament will not be added to the list of tournaments and will send back an error	Tournament was not added
1.1.3	Create new tournament with null as start dates		A new tournament will not be added to the list of tournaments and will send back an error	Tournament was not added
1.1.4	Create new tournament with existing password		A new tournament will not be added to the list of tournaments and will send back an error	Tournament was not added

Create new tournament

IdFromPass

This function is getting a tournament ID from a password, which will allow the user to log in to that specific tournament. If the password is unknown an error message will be shown.

Test		Condition	Expected result	Result
1	Tournament - Controller			
1.2	IdFromPass			
1.2.1	Find created tournament with password		The tournament with the unique password should be found in the list	Tournament ID was returned
1.2.2	Find created tournament with null as password		It will not find a tournament since the password is null	0 was returned
1.2.3	Try to find a tournament which doesn't exist	No tournament has the password "ATournamentThatDoesNotExist"	It will not find a tournament since no password is matching	0 was returned

Get tournament ID from password

Details

The function Details should get all the information about the tournament, which were filled out in the forms when created. Also later it should show the tournament's divisions and its fields.

Test		Condition	Expected result	Result
1	Tournament - Controller			
1.3	Details			
1.3.1	Get the details of the created tournament		The details of the tournament should be found	The created tournament was found, returned and had the expected member values
1.3.2	Try to find a tournament that doesn't exist	Tournament with ID 999999 does not exist	It shouldn't find a tournament and sends back an error	A tournament was not found so an error was returned

Get details of tournament

Edit

Setting the tournaments values to new values, in case the user for example wants to change the name of the tournament or the time intervals of the days.

Test		Condition	Expected result	Result
1	Tournament - Controller			
1.4	Edit			
1.4.1	Edit the information in the tournament		The tournament will be edited with the new information	Tournament is edited
1.4.2	Check if the edited information have been saved		The edited information should be saved as the new information for the tournament	The edits have been saved
1.4.3	Edit a tournament using null as parameters		Null is not accepted, so an error is returned	Tournament was not edited
1.4.4	Try to edit a tournament that doesn't exist	Tournament with ID 999999 does not exist	It can't edit the information of a non-existing tournament and will send back an error	Tournament was not edited

Edit tournament

Delete

Takes care of deleting tournaments and all connected entities

Test		Condition	Expected result	Result
1	Tournament - Controller			
1.5	Delete			
1.5.1	Delete a tournament		The tournament will be deleted	Tournament was deleted
1.5.2	Try to delete a tournament that doesn't exist	Tournament with ID 999999 does not exist	The tournament can not be found so an error is returned	Tournament was not deleted

Delete tournament

14.1.2 DivisionController

The DivisionController is handling the details, creating, editing and deleting of a division.

Create

Is creating a division for a specific tournament and adding it to a list of divisions for the tournament.

Test		Condition	Expected result	Result
2	Division - Controller			
2.1	Create			
2.1.1	Create new division for a tournament		A new division will be added to the list of divisions	Division was added
2.1.2	Create new division using null as division name		Null is not accepted, so an error is returned	Division was not added
2.1.3	Create new division to a non-existing tournament	Tournament with ID 999999 does not exist	A new division will not be added to the list of divisions and sends back an error	Division was not added

Create new division

Details

Is fetching the details for the specific division

Test		Condition	Expected result	Result
2	Division - Controller			
2.2	Create			
2.2.1	Get the details of the specific division		The details of the division should be found	The correct division will be returned with the expected member values
2.2.2	Find a division that doesn't exist	Division with ID 999999 does not exist	Division will not be found and therefore sends back an error	Division was not found and an error was returned

Details on division

Edit

Makes it possible to change the divisions values, which were set on creation.

Test		Condition	Expected result	Result
2	Division - Controller			
2.3	Edit			
2.3.1	Edit the information in the division		The information of the division should be edited	
2.3.2	Check if the edited information have been saved		The edited information should be saved as the new information for the division	
2.3.3	Try to edit a division with null as the new division name		Null is not accepted, so an error is returned	Division was not edited
2.3.4	Try to edit a division that doesn't exist	Division with ID 999999 does not exist	It can't edit the information of a non-existing tournament and will send back an error	Division was not edited

Edit division

Delete

Will make it possible to delete a division in the specific tournament.

Test		Condition	Expected result	Result
2	Division - Controller			
2.4	Delete			
2.4.1	Delete the specific division from the tournament		The division should be deleted from the tournament	Division was deleted
2.4.2	Try delete a division in a tournament that doesn't exit	Division with ID 999999 does not exist	It can't delete a division in a tournament that doesn't exist and will send back a error	Division was not deleted

Delete division

14.1.3 PoolController

The PoolController handles the details, creation, editing and deletion of a pool in a division.

Create

This function will create a new pool to a specific division, connecting it to the divisions id.

Test		Condition	Expected result	Result
3	Pool - Controller			
3.1	Create			
3.1.1	Creates a new pool for a specific division		The created pool will be added to a list of pools	Pool was added
3.1.2	Tries to create a new pool with null as the pool name		Null is not accepted so an error is returned	Pool was not added
3.1.3	Tries to create a new pool to a non-existing division	Division with ID 999999 does not exist	A new pool can't be created to a non-existing division and will send back an error	Pool was not added

Create new Pool

Detail

The function will get all the details on the pool.

Test		Condition	Expected result	Result
3	Pool - Controller			
3.2	Detail			
3.2.1	Get the details for a specific pool		The details of the pool should be found	The correct pool was found and had the expected member values
3.2.2	Tries to get the details on a pool that doesn't exist	Pool with ID 999999 does not exist	The details can't be found on a non-existing pool and will send an error	The pool was not found and an error was returned

Details on Pool

Edit

Makes it possible to change the pools values, which were set on creation.

Test		Condition	Expected result	Result
3	Pool - Controller			
3.3	Edit			
3.3.1	Edit the information for the specific pool		The information of the pool should be edited	The pool was edited
3.3.2	Check if the edited information have been saved		The edited information should be saved as the new information for the pool	Pool had the new changes
3.3.3	Try to edit a pool with null as new pool name		Null is not accepted so an error is returned	Error is returned
3.3.4	Try to edit a pool that doesn't exist	Pool with ID 999999 does not exist	It can't edit the information of a non-existing pool and will send back an error	Error is returned

Edit pool

Delete

Will make it possible to delete a pool in the specific division.

Test		Condition	Expected result	Result
3	Pool - Controller			
3.4	Delete			
3.4.1	Delete the pool from the division		The pool should be deleted from the division	Pool was deleted
3.4.2	Try delete a pool in a division that doesn't exit	Pool with ID 999999 does not exist	It can't delete a pool in a division that doesn't exist and will send back a error	Pool was not deleted

Delete pool

14.1.4 TeamController

The TeamController handles the details, creation, editing and deletion of a team in a pool.

Create

This function will create a new team to a specific pool, connecting it to the pool's ID.

Test		Condition	Expected result	Result
4	Team - Controller			
4.1	Create			
4.1.1	Create a new team for a specific pool		The created team will be added to a list of teams	Team was added
4.1.2	Tries to create a new team with null as team name		Null is not accepted so an error is returned	Team was not added
4.1.3	Tries to create a new team to a non-existing pool	Pool with ID 999999 does not exist	A new team can't be created to a non-existing pool and will send back an error	Team was not added

Create new team

Detail

The function will get all the details on the team.

Test		Condition	Expected result	Result
4	Team - Controller			
4.2	Detail			
4.2.1	Get the details for a specific team		The details of the team should be found	The correct team was found and had the expected member values
4.2.2	Tries to get the details on a team that doesn't exist	Team with ID 999999 does not exist	The details can't be found on a non-existing team and will send an error	Team was not found and an error was returned

Details on team

Edit

Makes it possible to change the teams values, which were set on creation.

Test		Condition	Expected result	Result
4	Team - Controller			
4.3	Edit			
4.3.1	Edit the information for the specific team		The information of the team should be edited	Team was edited
4.3.2	Check if the edited information have been saved		The edited information should be saved as the new information for the team	Team edits were saved
4.3.3	Try to edit a team with null values		Null is not accepted so an error should be returned	Team was not edited and an error was returned
4.3.4	Try to edit a team that doesn't exist	Team with ID 999999 does not exist	It can't edit the information of a non-existing team and will send back an error	Team could not be found and an error was returned

Edit team

Delete

Will make it possible to delete a team

Test		Condition	Expected result	Result
4	Team - Controller			
4.4	Delete			
4.4.1	Delete the team from the pool		The team should be deleted from the pool	Team was deleted
4.4.2	Try delete a team that doesn't exit	Team with ID 999999 does not exist	It can't delete a team that doesn't exist and will send back a error	Team was not deleted

Delete team

14.1.5 FinalsLinkController

The FinalsLink Controller handles the details, creation, editing and deletion of the FinalsLink

Create

The create function will create a new FinalsLink for the specific division

Test		Condition	Expected result	Result
5	FinalsLink - Controller			
5.1	Create			
5.1.1	Create a new FinalsLink for a specific division		A new FinalsLink will be created	Finalslink was created
5.1.2	Create a new FinalsLink to an invalid division	Division with ID 999999 does not exist	The new FinalsLink can't be created to an invalid division and will send back an error	Finalslink was not created

Create new finalslink

Detail

The function will get all the details on the finals link

Test		Condition	Expected result	Result
5	FinalLinks - Controller			
5.2	Detail			
5.2.1	Get the details for a specific finals link		The details of the finalslink should be found	The finalslink was returned
5.2.2	Tries to get the details for a field that doesn't exist	Finals Link with ID 999999 does not exist	The details can't be found on a non-existing field and will send an error	Error was returned

Details on finalslink

Edit

The edit function allows the finalslink to be edited, which were set on creation.

Test		Condition	Expected result	Result
5	FinalsLinks - Controller			
5.3	Edit			
5.3.1	Edit the finals-link for the specific division		The finals-link for the division should be edited	Finals link was edited
5.3.2	Check if the edited information have been saved		The edited finals link should be saved as the new information for the division	Edits were saved
5.3.3	Try to edit a finals-link that doesn't exist	Finals Link with ID 999999 does not exist	It can't edit the information of a non-existing finals-link and will send back an error	Error was returned

Edit finalslink

Delete

The delete function makes it possible to delete a finalslink for a specific division.

Test		Condition	Expected result	Result
5	FinalsLink - Controller			
5.4	Delete			
5.4.1	Delete the finals-link		The finals-link should be deleted	Finals link was deleted
5.4.2	Try delete a finals-link that doesn't exit	Finals Link with ID 999999 does not exist	It can't delete a finals-link that doesn't exist and will send back a error	Finals link was not deleted

Delete finalslink

14.1.6 FieldController

The Field Controller handles the details, creation, editing and deletion of a field.

Create

This function will create a new field to a specific tournament, connecting it to the tournament id.

Test		Condition	Expected result	Result
6	Field - Controller			
6.1	Create			
6.1.1	Create a new field for a specific tournament		The created field will be added to a list of fields	The field was added
6.1.2	Tries to create a new field to a non-existing tournament	Tournament with ID 999999 does not exist	A new field can't be created to a non-existing tournament and will send back an error	Error was returned

Create new field

GetAllTournamentFields

The function will get all the fields for a tournament.

Test		Condition	Expected result	Result
6	Field - Controller			
6.2	GetAllTournamentFields			
6.2.1	Get the fields for a specific tournament		The fields of the tournament should be found	The list of fields were returned
6.2.2	Tries to get the fields for a tournament that doesn't exist	Tournament with ID 999999 does not exist	The fields can't be found on a non-existing tournament and will send an error	An error was returned

Fields for a tournament

Detail

The function will get all the details on the field.

Test		Condition	Expected result	Result
6	Field - Controller			
6.3	Detail			
6.3.1	Get the details for a specific field		The details of the field should be found	The field was found and had the expected member values
6.3.2	Tries to get the details for a field that doesn't exist	Field with ID 999999 does not exist	The details can't be found on a non-existing field and will send an error	An error was returned

Details on field

Edit

Makes it possible to change the fields values, which were set on creation.

Test		Condition	Expected result	Result
6	Field - Controller			
6.4	Edit			
6.4.1	Edit the information for the specific field		The information of the field should be edited	The field was edited
6.4.2	Check if the edited information have been saved		The edited information should be saved as the new information for the field	The edited field had the correct values
6.4.3	Try to edit a field that doesn't exist	Field with ID 999999 does not exist	It can't edit the information of a non-existing field and will send back an error	An error was returned

Edit field

Delete

Will make it possible to delete a specific field.

Test		Condition	Expected result	Result
6	Field - Controller			
6.5	Delete			
6.5.1	Delete the field		The field should be deleted	The field was deleted
6.5.2	Try delete a field that doesn't exit	Field with ID 999999 does not exist	It can't delete a field that doesn't exist and will send back a error	The field was not deleted

Delete field

14.2 Errors and problems

In this section, all of our errors and problems from the user acceptance test will be described and given a category fitting the the problem.

Problem number	Problem state	Problem
1	Critical	Schedule tournament isn't working
2	Cosmetic	Dynamic field sizes
3	Serious	Finalslinks aren't deleted when they're supposed to
4	Cosmetic	End-time calender should automatically go to the same month as start-time
5	Cosmetic	Better definition on end-time (last match ends or start at the end-time)
6	Cosmetic	Time-intervals are resat when editing the tournament
7	Cosmetic	Focus on text-field when clicking on add something
8	Cosmetic	Tournament plan view should also support a field view in same format as pool or division view
9	Cosmetic	The Modal and search function looks weird in some browsers
10	Cosmetic	Default time-intervals for all days, matching the first day's time-intervals.
11	Cosmetic	Arrows not working to increment and decrement match duration
12	Cosmetic	Edit field name
13	Cosmetic	In-line name editing

Problem table

Problem 1 is categorised as a critical error, because it's the main function of our system. When the users had created a tournament and added a division with a couple of pools that includes teams, and added a decent amount of fields and tried to schedule the tournament, it couldn't. The reason behind this, can be two things. The first possible error, is that if there is more teams in one pool than the others, the users doesn't seem to understand the finalslinks very well, as they let every team in the pool play against the other teams in the other pools with the same position, which means, that if there are 4 teams in pool 1 and 2, and 5 in pool 3, then team 5 in pool 3 will go to a final stage alone, which isn't possible. Therefore there should be something in the code handling this issue. The other issue, can be that there isn't enough time on the selected days, and therefore, the user have to add an extra day to the tournament. The way we show this information at the moment of the tests, we're saying that it could be either that or that there are missing fields, so the users just keep adding fields, which won't necessarily help. Therefore, we should also make it more clear, if the problem is that there isn't enough time or that it's the fields that are missing.

Problem 2 is cosmetic suggestion/problem from one of the end users. It came up when he was adding fields, and saw that the three field sizes were hardcoded, then he suggested to try and make it possible to make it dynamic, so you could add fields that fits the need of the user, as some tournaments still use the old 7-man fields instead of the new 8-man fields.

Problem 3 is categorised as a serious issue as it's part of problem 1 and can confuse the user, because if he had 5 teams in one pool, and 4 teams in two other pools, then if the user delete the 5th team, the finalslinks will still be visible for the 5th position, which don't make any sense. The system should handle this, when a team is deleted, which may help solve problem 1.

Problem 4 is another cosmetic problem. When the user select a month in the future for the start day, and then wants to select the end day, the month goes back to the current month, instead of starting from the start day month selected. This is a minor error that should be handled.

Problem 5 is also a minor cosmetic problem. One of the end users asked whether the end-time is the last time a match can be planned for, or if the last match should end before the end time.

Problem 6 is another cosmetic problem. When the users tried to edit the tournament, the time-intervals for the tournament resat. There should be a function that handles this, as it's time wasting for the users, that they have to select new time-intervals, every time they want to add a new day to the tournament, or cut a day off the tournament.

Problem 7 is also a cosmetic problem. When creating teams for a pool, the user has to click add new team, then click the text-box for every single team that the user wants to add. This make the process take much more time than needed, and if there is 100 teams that needs to be added, it could start becoming annoying.

Problem 8 is a cosmetic suggestion that one of the end user pointed out.

When the user had scheduled the tournament, and saw the different views, he felt like there was missing a view like the division/pool/team view for the fields. That could be useful, when assigning referees to the tournament, as he needs to know which team have to play, in case one of the teams doesn't show up.

Problem 9 is a cosmetic problem that appears when opening the web-application in different browsers. It primarily the mac browsers that have the issue, as the two informants both we're using mac, one with Firefox and one with Google Chrome. And the interface didn't quite look as it does with Safari e.g., we haven't noticed any problems with windows computers so far.

Problem 10 is a cosmetic suggestion an end user pointed out. When the tournament length is more than a day, the time-intervals should by default just select the same time-intervals as the first day set by the user, and if the tournament needs to end earlier the last day e.g., the user can just do that.

Problem 11 is a cosmetic error. When the users tries to edit the match duration of a division by using the arrows, it doesn't work, as it doesn't use the current match duration as text, but just as a placeholder. This should be be handled, so that when the user clicks on the edit button he gets the current match duration as text instead of a empty text-box. Also when creating a new division, if the arrows are used, the value can go down to negative values, which aren't the idea.

Problem 12 is a cosmetic suggestion from one of the end users. His thought was that it should be possible to edit a field name without having to delete the field and then reschedule the whole tournament.

Problem 13 is also a cosmetic suggestion from the end user. It should be possible to edit a tournament name by double-clicking the tournament name, instead of having to to click the edit button, change the name and click the save button.

14.3 User acceptance test tasks

Beginning from the next page, the tasks for the user acceptance tests.

Test af program

I denne test, er det meningen, at systemet der er blevet udviklet skal aftestes. For at gøre dette, vil du blive stillet en række opgaver, som du skal forsøge at løse. Disse opgaver vil indebære, at lave en ny turnering gennem selve programmet, og at uploade en turnering fra en excel fil.

Test

Programmet er at finde på http://sorenlyng.dk/.
Læs gerne opgaverne højt, dog behøves det i parenteserne ikke læses højt, da det bare er lidt sideinfo.
Den første side der ses er startsiden til systemet, hvor der er mulighed for at logge ind vha. et password, og mulighed for at lave en ny turnering.

Opgave 1
Der skal oprettes en ny turnering, gør dette fra forsiden og udfyld realistisk data ind i formen på den side du bliver sendt videre til. (Brug evt. 123456 som kode)
Du må gerne se bort fra "Upload Excel ark for turnering", den skal først bruges til en senere opgave.
Tryk på "Fortsæt"-knappen, når du er færdig med at udfylde data.

Opgave 2
Der skal oprettes rækker til cuppen. Tilføj to forskellige rækker, og udfyld formen med relevante info.
Herefter må du gerne klikke på en af rækkerne og slette den.
I den anden række, må du gerne klikke dig ind på den, og tilføj 2-3 puljer, med 3-5 hold i hver.
Herefter har du mulighed for at redigere kamp længden og banestørrelsen for rækken, det står på den indstilling du tilføjede rækken med, som du gerne må ændre.

Øverst på siden, findes der "brødkrummer", her må du gerne trykke på "turneringsoversigt", for at komme tilbage til den forrige side.

Opgave 3
Der skal nu tilføjes nogle baner til cuppen. Tilføj nogle baner, hvoraf en skal passe med den banestørrelse, der er blevet valgt til den række du selv lavede, tidligere.

Klik ind på rækken, du lavede tidligere, og vælg et slutspil til den række.

(Information om de forskellige slutspil)
Knockout - Dette er turneringsformen med 16-dels finaler, kvartfinaler, semifinaler og finale.
Round Robin - Dette er at holdene bliver inddelt i nye puljer, og ender med den placering i cuppen de får i den nye pulje.

Opgave 4
Herefter kommer du i tanke om, at Hold 1 fra Pulje 1, gerne vil hjem en smule tidligere de første par dage af turneringen, da de er fra lokalområdet. Hold 2 fra Pulje 2 vil gerne ankomme et par timer senere, da de kommer med en senere færge fra Norge.

Klik ind på Hold 1 fra Pulje 1 og ændrer deres afgangs tid, til at være tidligere end turnerings valgte tidsperiode.
Gør det samme for Hold 2 fra Pulje 2, denne gang bare med en senere ankomst tid.

Husk at gemme den ændrede information, inden du går tilbage (Via brødkrummer eller browserens tilbage-knap.

Opgave 5
Nu er turneringen klar til at blive planlagt. Gå tilbage til "turneringsoversigten" og tryk på "Planlæg turnering". Der går noget tid, afhængigt af antal rækker, puljer og hold i turneringen.

Når planlægningen er færdig, vil der dukke et kampprogram op, med de forskellige kampe i turneringen. I denne turnering, er der ikke så mange hold, så der er ikke så mange kampe.

Her har du så mulighed for at klikke på en pulje eller et hold, for at blive smidt over til deres side, hvor du vil have kampene for den pulje, eller det hold du har valgt at trykke på.

Gå tilbage til "turneringsoversigten" hvis ud har forladt denne side.

Opgave 6
Du kommer i tanke om, at koden du lavede (eks. 123456), ikke er en god kode, så du skal redigere i turneringen.

Lav koden om til noget andet (f.eks. 654321).

Gå derefter tilbage til forsiden, hvor du kunne logge ind eller lave en ny turnering.
Prøv så at indtaste den gamle kode (123456).
Prøv at indtaste den nye kode (654321).

(Dette er den eneste måde, at logge ind på turneringen igen, så hvis der er andre der skal have rettigheder, til at hjælpe med turneringen, så skal koden, gives til dem af dig, personligt.)

Opgave 7
Du vil gerne gemme kampprogrammet som en Excel fil, så du kan videresende dette til piratliga.dk f.eks., klik på "Eksporter"-knappen i bunden af kampprogrammet i "turneringsoversigten".

En fil bliver downloadet til din computer.

Opgave 8
Gå tilbage til forsiden, hvor du har mulighed for at logge ind på turneringen eller lave en ny, igen. Lav igen en ny turnering. Denne gang, kald turneringen, noget andet og vælg en anden kode, da du ikke kan bruge den samme kode.
Udfyld igen formen, med relevante informationer. Denne gang skal du prøve at uploade en Excel fil. Find Excel filen, der skal uploades, og tryk på åbn.

Klik herefter på opret turnering. Du vil nu se, at der allerede er tilføjet rækker og baner, og hvis du klikker dig ind på de forskellige rækker, er der tilføjet puljer og hold. Dette er hvad Excel filen indeholdt.

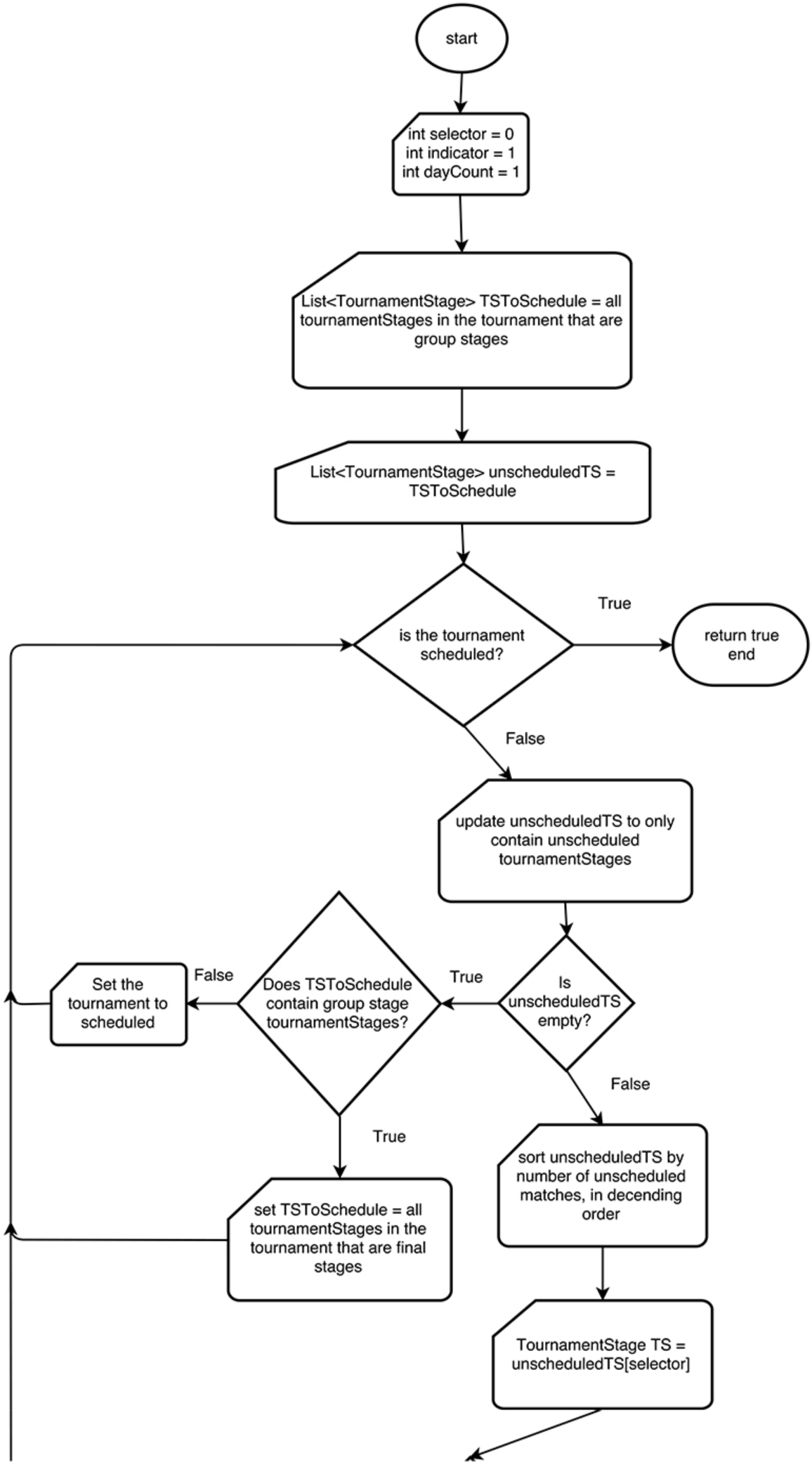
Det eneste der mangler, at du trykker på “Planlæg turnering” og venter på at systemet udregner de forskellige kampe.

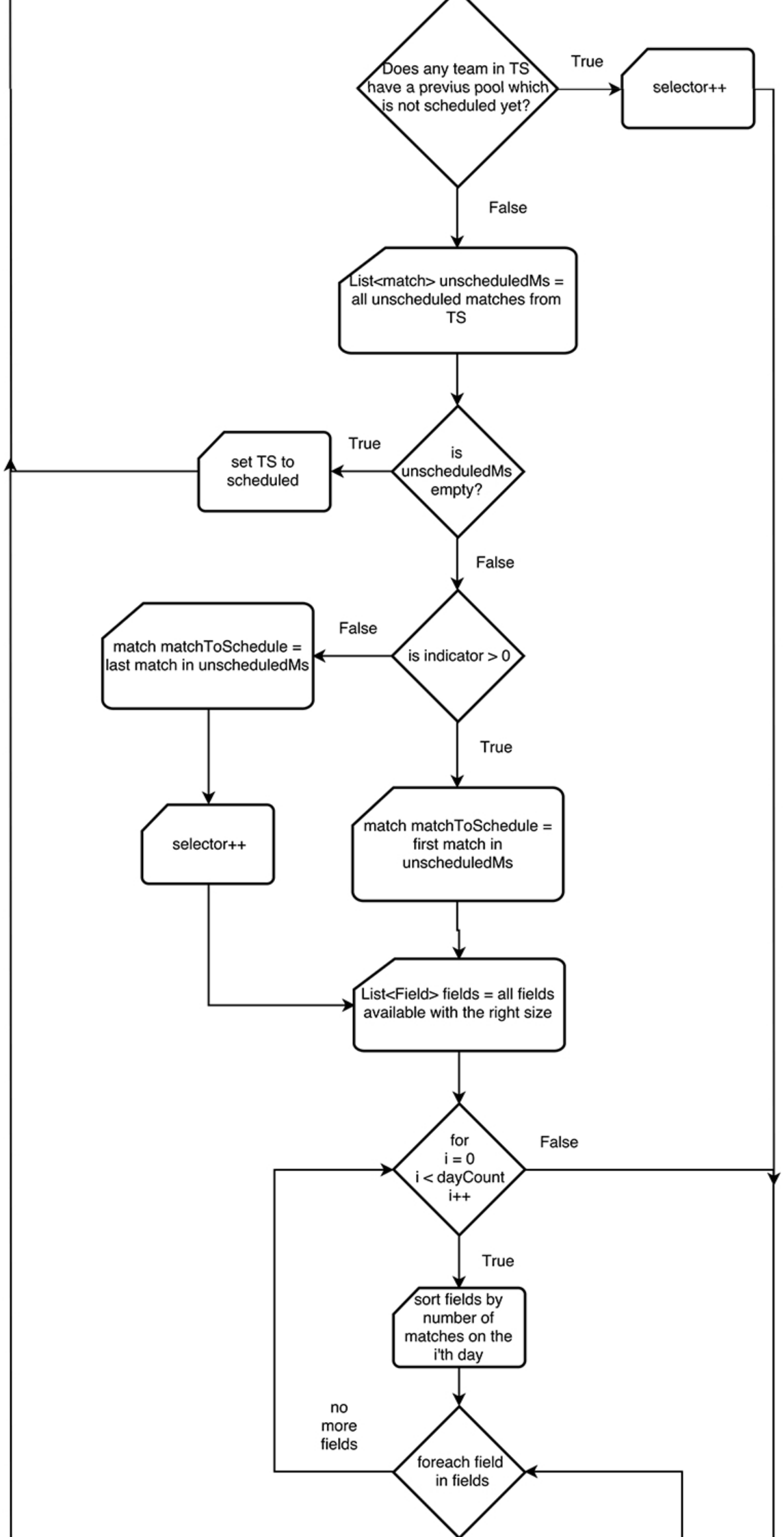
Når dette er færdigt, vil turneringsplanen igen dukke op, denne gang skulle den gerne være længere end den fra den forrige turnering. Hvis du klikker på de forskellige puljer, vil du blive taget over til den valgte pulje, hvor alle kampe for puljen vil blive vist, i stedet for alle kampe i turneringen.

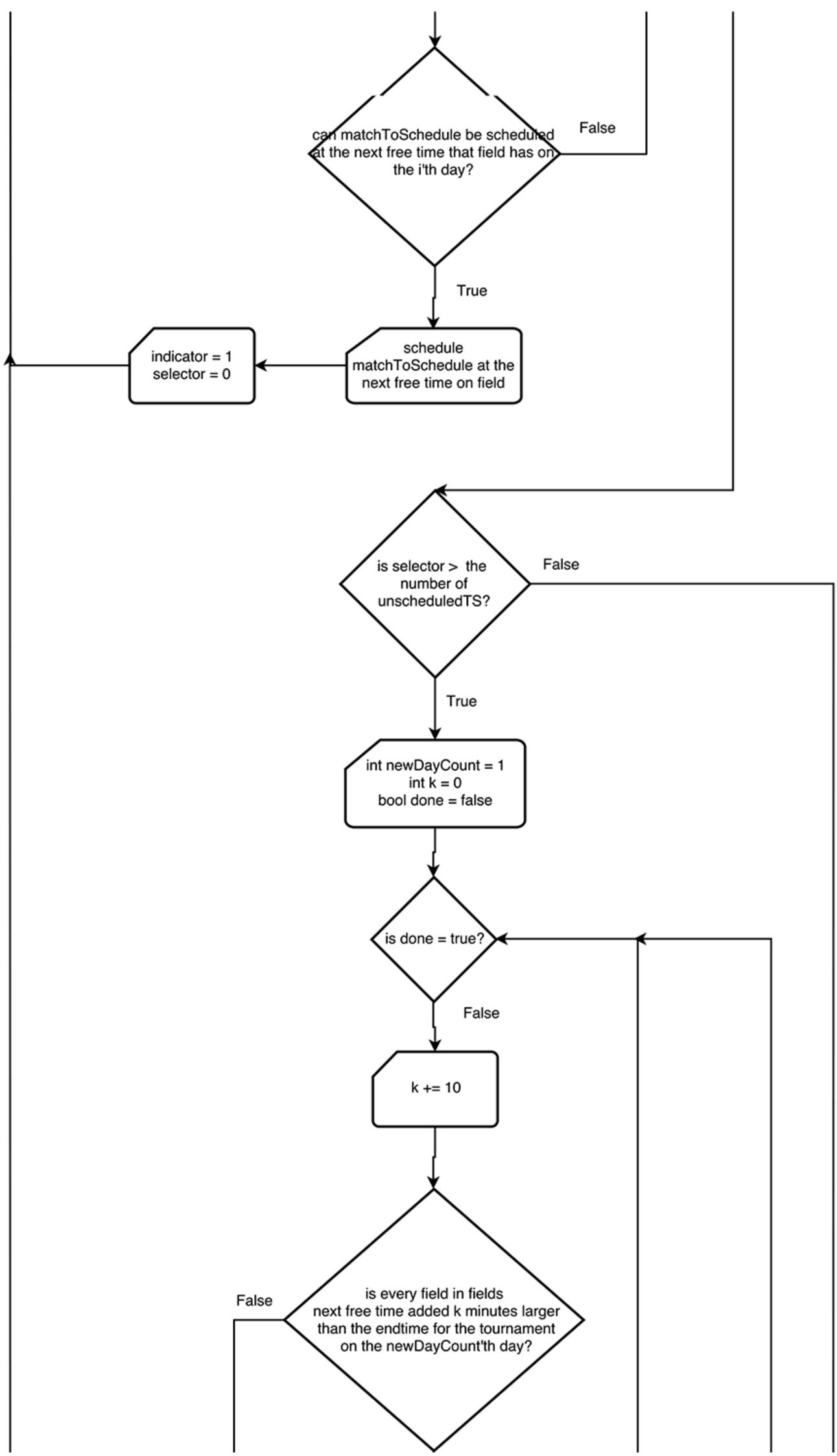
Igen her, har du mulighed for at eksporter turneringsplanen til en Excel fil.

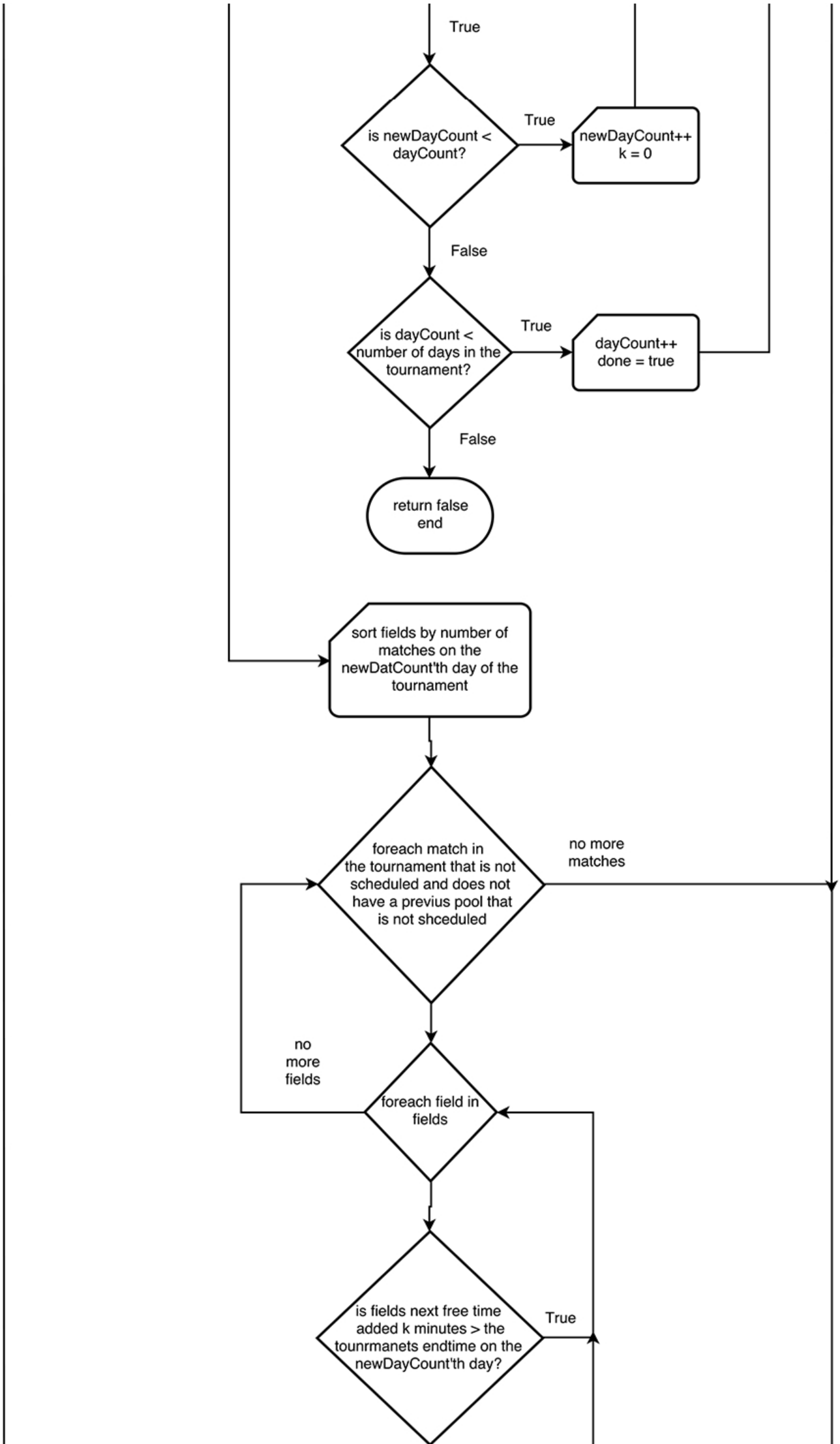
14.4 Flowdiagram

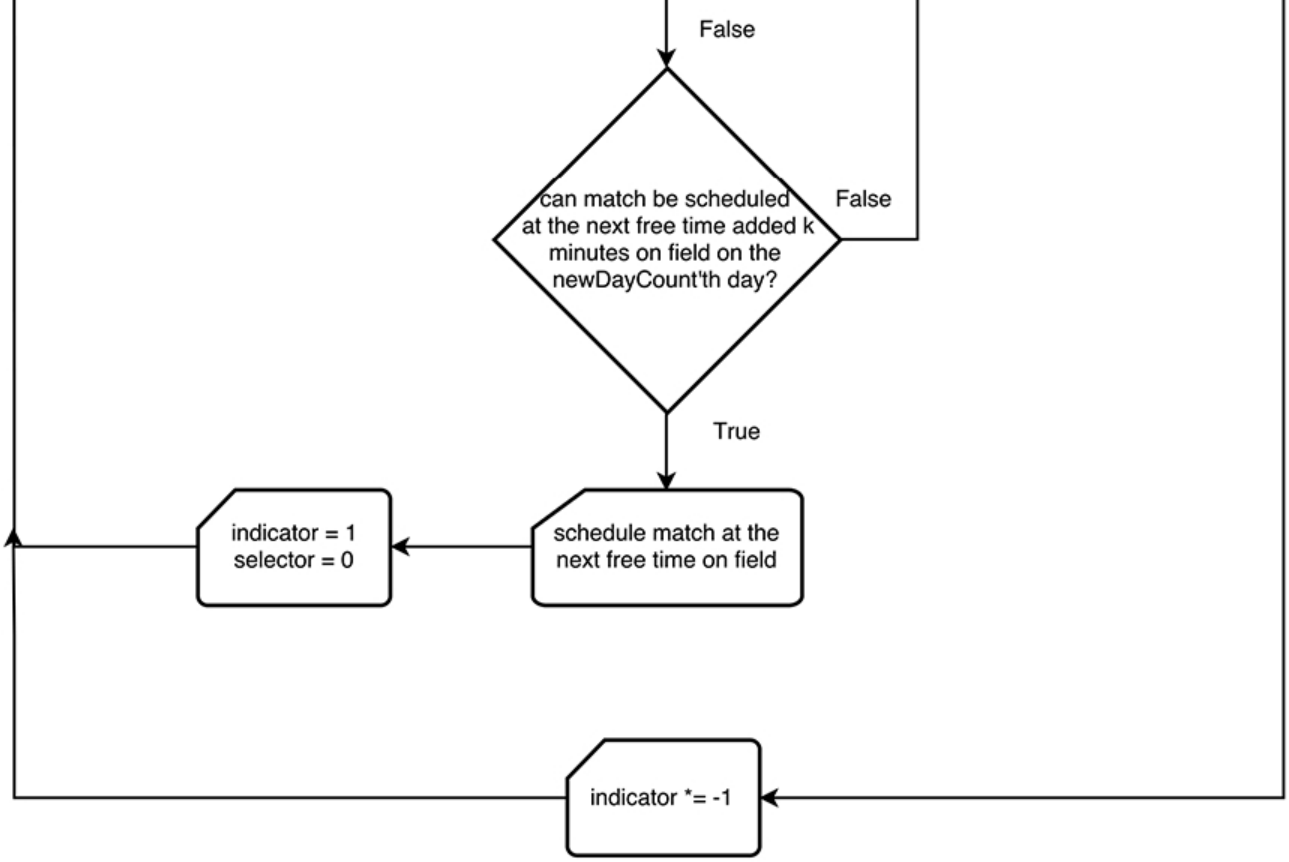
Beginning from the next page, the flow diagram for the algorithms can be found.


start
int selector = 0
int indicator = 1
int dayCount = 1
List<TournamentStage> TSToSchedule = all tournamentStages in the tournament that are group stages
List<TournamentStage> unscheduledTS = TSToSchedule
is the tournament scheduled?
True
return true
end
False
update unscheduledTS to only contain unscheduled tournamentStages
Is unscheduledTS empty?
True
Does TSToSchedule contain group stage tournamentStages?
False
Set the tournament to scheduled
True
set TSToSchedule = all tournamentStages in the tournament that are final stages
False
sort unscheduledTS by number of unscheduled matches, in decending order
TournamentStage TS = unscheduledTS[selector]


Does any team in TS have a previus pool which is not scheduled yet?
True
selector++
False
List<match> unscheduledMs = all unscheduled matches from TS
is unscheduledMs empty?
True
set TS to scheduled
False
is indicator > 0
False
match matchToSchedule = last match in unscheduledMs
selector++
True
match matchToSchedule = first match in unscheduledMs
List<Field> fields = all fields available with the right size
for i = 0 i < dayCount i++
False
True
sort fields by number of matches on the i'th day
no more fields
foreach field in fields


can matchToSchedule be scheduled at the next free time that field has on the i'th day?
False
True
schedule matchToSchedule at the next free time on field
indicator = 1
selector = 0
is selector > the number of unscheduledTS?
False
True
int newDayCount = 1
int k = 0
bool done = false
is done = true?
False
k += 10
is every field in fields next free time added k minutes larger than the endtime for the tournament on the newDayCount'th day?
False


True
is newDayCount < dayCount?
True
newDayCount++
k = 0
False
is dayCount < number of days in the tournament?
True
dayCount++
done = true
False
return false
end
sort fields by number of matches on the newDatCount'th day of the tournament
foreach match in the tournament that is not scheduled and does not have a previus pool that is not shceduled
no more matches
no more fields
foreach field in fields
is fields next free time added k minutes > the tounrmanets endtime on the newDayCount'th day?
True


False
can match be scheduled at the next free time added k minutes on field on the newDayCount'th day?
False
True
indicator = 1
selector = 0
schedule match at the next free time on field
indicator *= -1

14.5 CD

With the report a CD with the following have been attached:

- The program code written in C#, HTML5/CSS3 and AngularJS
- A copy of the report
- A folder with different licences used through the project
- A read-me-file
- Picture of the flow diagram