IT2901 - Informatics Project II

IDI Open
Programming Contest System

Haakon Konrad William Aasebø
Håkon Gimnes Kaurel
Tino Hakim Lazreg
Filip Fjuk Egge
Anders Sildnes
Eirik Fosse

May 2014

Norwegian University of Science and Technology

Supervisor: Hong Guo

Foreword

Originally inspired by the Nordic Collegiate Programming Contest (NCPC), it has been held at NTNU every spring since 2007. The format is a five-hour contest with competing teams consisting of one, two or three contestants. A team of volunteer judges write the problems and answer clarification requests during the contest, while another team hands out balloons for each solved problem. Usually a rather hectic affair, it is extremely important that everything is well prepared. The number of teams is often more than 100, with the record being 162 teams in 2011

The contest system that verifies solutions is at the heart of the contest when it is in progress, and needs to be working perfectly at all times. The system must handle several submissions per second, while verifying that each one is correct and runs within the set resource limits. Submissions must show up on the high score list, and when problems are solved the team handing out balloons must be notified. In addition to this there were a lot of other functional requirements having to do with the bureaucracy of organizing the contest

A requirement was that new features could be easily added in the future, and the code was written with this in mind. The project will now become open source, and all programming contest enthusiasts will soon be able to request and implement their desired features

All aspects of this project have been pleasing and delightful for us. The team has exceeded all our expectations and their system will be used for years to come.

– *Christian Chavez, IDI Open Manager*

Preface

Before there were computers, there were algorithms. But now that there are computers, there are even more algorithms, and algorithms lie at the heart of computing. Designing a system for eager students to hone their skill in the heart of computing has been a true joy.

Our group never wanted to settle for adequacy and mere requisiteness. For the past few months, weve taught ourselves a new programming language and framework and used advanced development frameworks - while tackling many social and technical conflicts.

We have proven how "Ambition is a dream with a V8 engine", as Elvis Presley once said.

The group would like to thank our eager customers, Finn Inderhaug Holme, Christian Chavez and Christian Neverdal Jonassen for their time to meet us and provide constructive feedback. We also owe a big thanks to our supervisor, Hong Guo, for constructive criticism and reflections; without which, we would not ascertain the peak of our own potential.

Contents

Chapter 1

UI Design

This chapter contains the choices made regarding the process of designing the front-end of the application, for a more technical approach see *System Architecture chapter 6*.

1.1 Design Process

The user interface provided by the previous IDI Open system consisted of a simple web interface for reading news items, registering teams for contests, and delivering submissions. GentleIDI is intended to provide more functionality through its web interface, including but not limited to judge supervision(requirement FJ-11) and user management (requirements FC-01, FC-03 and FC-04). As a consequence we had two options available: reusing and extending the existing interface design, or creating our own design from scratch.

We chose to create our own design from scratch, while still trying to keep a similar placement of elements from the previous design. The customer expressed concern regarding how contestants would react to the transition from the old interface to the new one. With this in mind we started to create mockups modelling core elements of the website. Our initial drafts consisted of simple rearrangements of elements found in the old web interface.

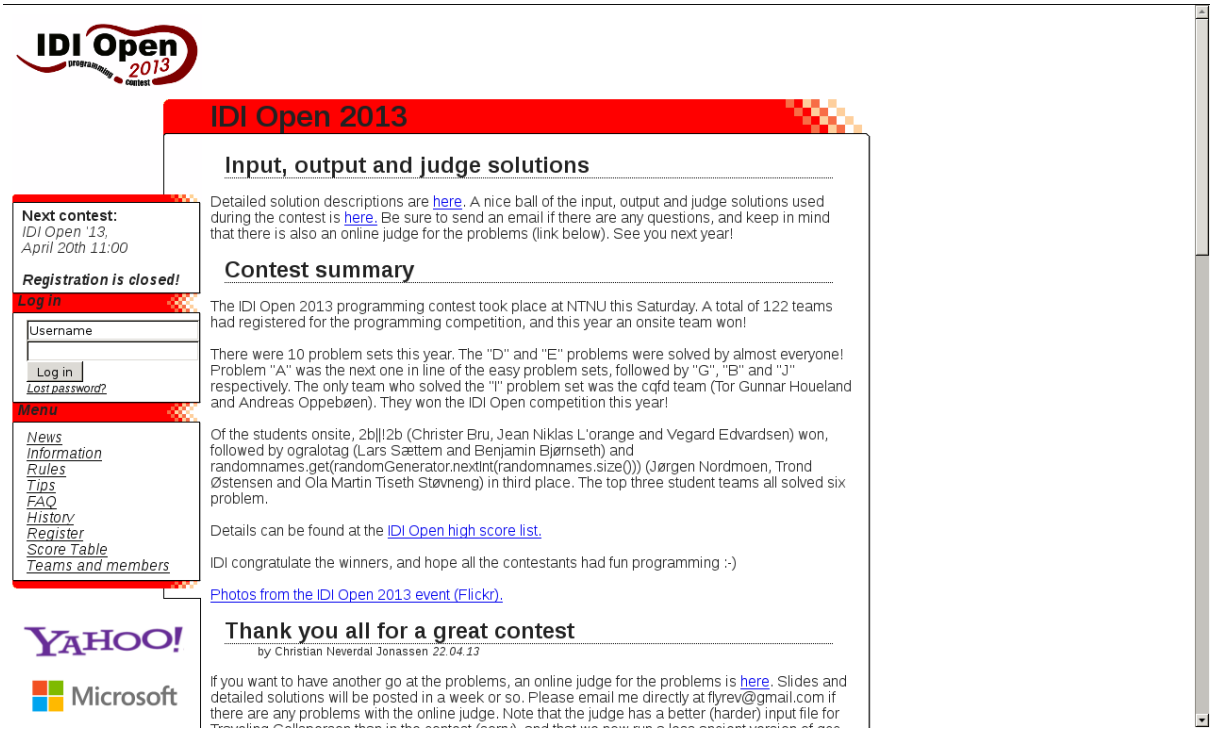


Figure 1.1: User Interface of the old system

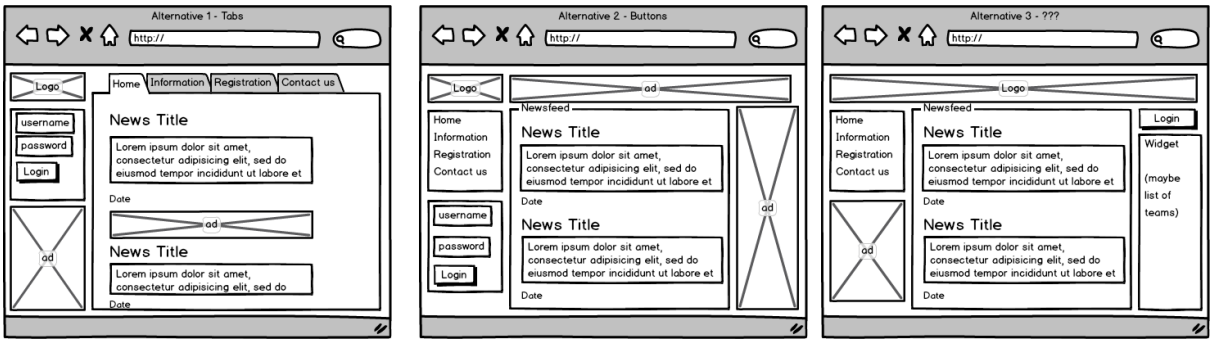


Figure 1.2: Initial mockups

Beyond our three initial mockups we tried a couple of "out of the box" approaches to our designs, but none of them met our standard and was rejected for either being too time-consuming to implement or too far from what our customer wanted. We had a meeting with our customer, where we showed our mockups, and what our thoughts on design had been so far. We wanted to make sure that

the customer was on the same page as us, and that we were not moving beyond the scope of the project. Our customer was not very focused on the design aspect, but one demand they had was that they wanted the new site to have the same structure as the old one. One example of what this means is that the customer wanted us to keep the menu on the left side as you can see that the old system has in Fig 1.1. We agreed, because getting used to a new website can take time, so keeping the structure similar would ease the transition for our users. With this in mind we decided to go for one of our initial mockups, the rightmost one in Fig 1.2, because it had the same structure as the old page, and we personally favoured that design. As a result, most of the elements found in the old interface can be found in the new one, and the transition between using the two is reduced to a minimum.

The task had to be completed in time for milestone M-03, so our main concern was designing for the functionality needed for that particular milestone. However, we also had mockups for functionality outside of this milestone. After milestone M-03 was met, we introduced new designs for new functionality through continuous work on top of a template.

The majority of the front end is stylized using bootstrap[Link til kilde] as a framework, enabling us to create a site which is both highly maintainable and aesthetically pleasing at the same time. The admin interface was created using django-admin-interface. Grappelli was used as a skin to give it a modern look. The look of the final page can be viewed in Fig 1.3.

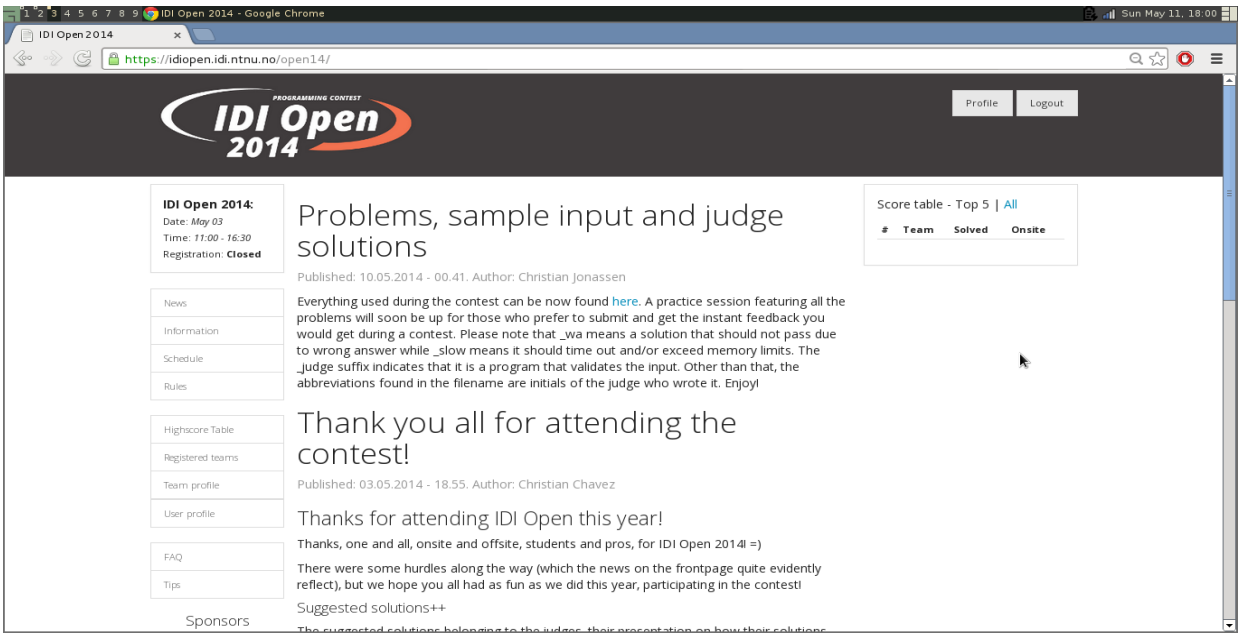


Figure 1.3: Final page

Figure 1.4: Various buttons used on our website. From left to right: the button to go to the contest page, the button to see a user profile, the button to leave a team

The grey header was in our initial design coloured blue, but was changed one week before M-07. This illustrates the strongest functionality of the design, namely customization. It is possible, by only uploading a new CSS file, to change the whole feel of the website and give every contest its own theme. The change from blue to grey was made as a consequence of IDI Open changing to a new logo. By comparing Fig 1.1 and Fig 1.3, you can see that we kept the same structure, but still made some significant changes to the design.

1.2 User Interface

The user interface is designed by using a base template. The template is the same for every part of the webpage, and contains a content block that changes while you navigate through the different parts. This makes it easier to add new content to the user interface, because you already have the base, and don't need to worry about the header, footer, or the menu. We wanted to make it easy for future developers to take over GentleIDI after us, and therefore we focused on a versatile user interface, in case they want to add new functionality.

The menu is placed to the left, coping with the western norm stating that eye placement is natural to the left[1]. We designed the menu to be versatile, this was highly prioritized by our customers. Admins can choose what they want to show in the menu, except for *Register user* and *Register team* that are "hardcoded" on request from the customer. As mentioned in Design process 1.1, we designed the user interface after a principle of versatility. Admins can also change the logo, the sponsor images and the contact information in the footer.

Buttons, images and icons were surrounded with boxes, to show that they are different elements. There is also one big box surrounding a group of elements, for example the sponsors. This is consistent with the gestalt law of proximity, that constitutes that humans will naturally group objects that are close to each other, and view them as distinct. This helps the user quickly understand the user interface.

[1] `http://research.microsoft.com/en-us/um/people/cutrell/chi09-buschercutrellmorris-eyetrackingforwebsalience.pdf`

"To strive for consistency" is the first of Shneiderman's eight golden rules of interface design[2], and we tried to follow this while making design decisions. As can be seen in Fig 1.4, we decided to use colours that represents the action each button is connected to. The red button marks that pressing this will have permanent consequences. We added a textbox prompt that the user has to answer after pressing a red button, that constitutes to Schneiderman's fifth and sixth rule, for easy reversal of actions and error handling. This wasn't added initially, but we noticed while testing the system that without a prompt, it could be possible to leave your team by mistake.

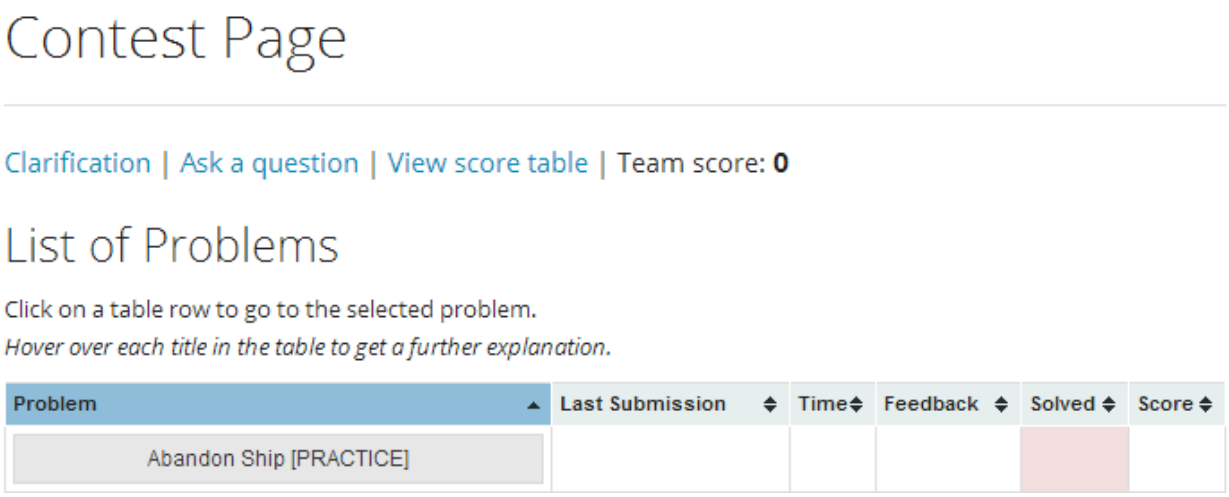


Figure 1.5: Contest page

For the contest page, Fig 1.5, we wanted to give the contestant a good overview of all the problems, their submissions to them, feedback, if they solved the problem and the score. It is important to not bury information to deep in a website. It could be challenging to balance this while trying not to overload the page with too much information. We had this in mind when designing this page. We got valuable feedback from the customer concerning what they wanted to be present on the contest page. They wanted it to be easy for the contestants to access everything they need during the competition, through the contest page. After feedback from the customer, we added links to the clarification page and highscore table on the contest page. This lowers the short-term memory load on the contestants, which is consistent with Shneiderman's eight rule, because they will have everything accessible on the same page.

[2] `https://www.cs.umd.edu/users/ben/goldenrules.html`

1.3 Admin Interface

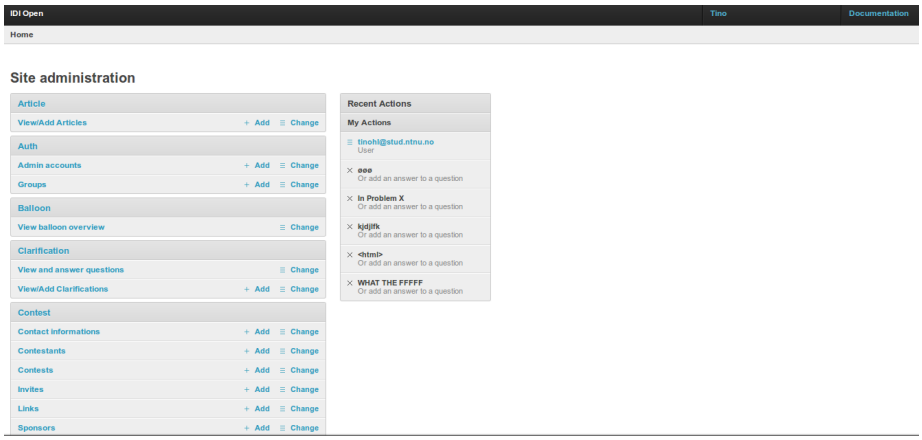


Figure 1.6: Admin Interface

Django comes with an extensive admin interface, that provides functionality for adding, removing and changing parts of the system. The interface consists of everything we as developers want the admins to be able to change. We decided to use Grappelli, an app for the django admin interface that also provided us with more adequate functionality, e.g. auto-completion, rich text editors, drag'n drop and more.

The structure of the layout is simple. Each category has it's own header and everything in blue is clickable. The "Recent Actions" box is there to help admins remember what they last did, which is important to reduce the users short-term memory load, in accordance with Shneiderman's eight rule.

Originally all the names of the elements were the same as our model names. We decided to change this to more intuitively understandable expressions after a request from the customer. We extended the interface with our own custom views, "Balloon overview" and "Judge views". This allowed us to change what we wanted, while it still kept its consistency with the other parts of the admin site.

The judge views was made primarily for judges, but could also be used by the admins. The motivation behind making this view, is that it gives the judges a better overview of the competition and how the progress is going for the different teams. We were initially told that the judges wanted a way to see if a team was struggling, so they could help that team. We wanted everything to be on one page for the judges, so they wouldn't have to constantly switch between different pages. The judge view can be seen in Fig 1.7.

Fig 1.8 shows the judge views after selecting the team "GentleCoding". It is possible to expand each submission by clicking on it. The third submission has been clicked on, so we can now choose to expand different categories. For example if a judge wants to see the source code for that submission, he/she can click on "Source code" and it will expand. Submissions that haven't been compiled are shown in red, and the other are white.

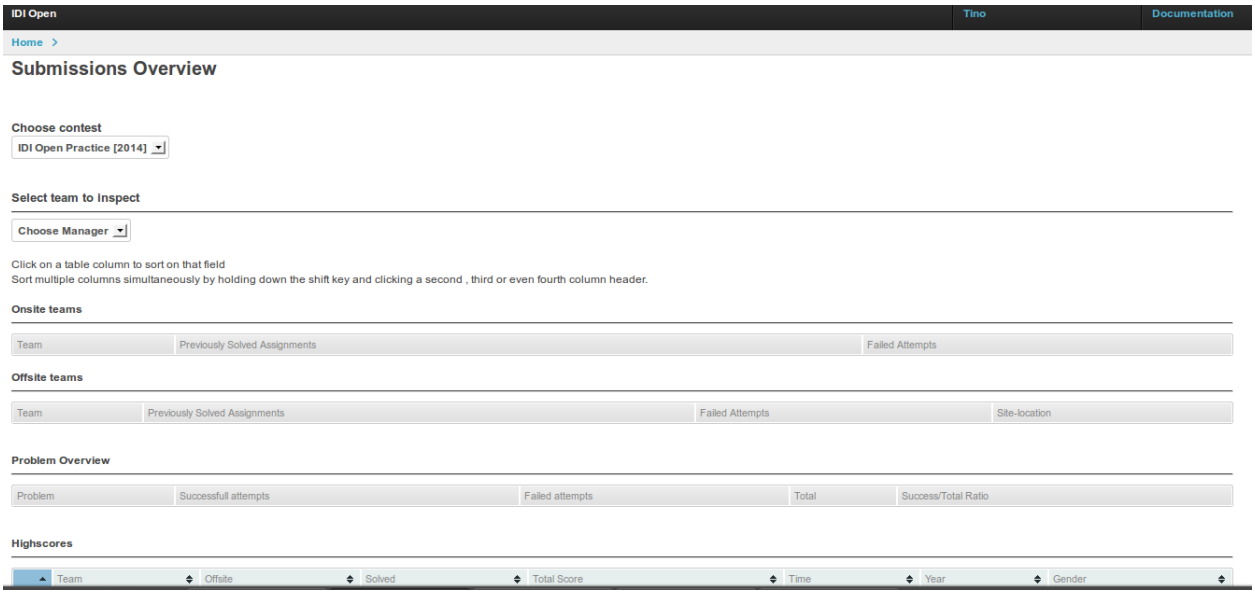


Figure 1.7: Judge views

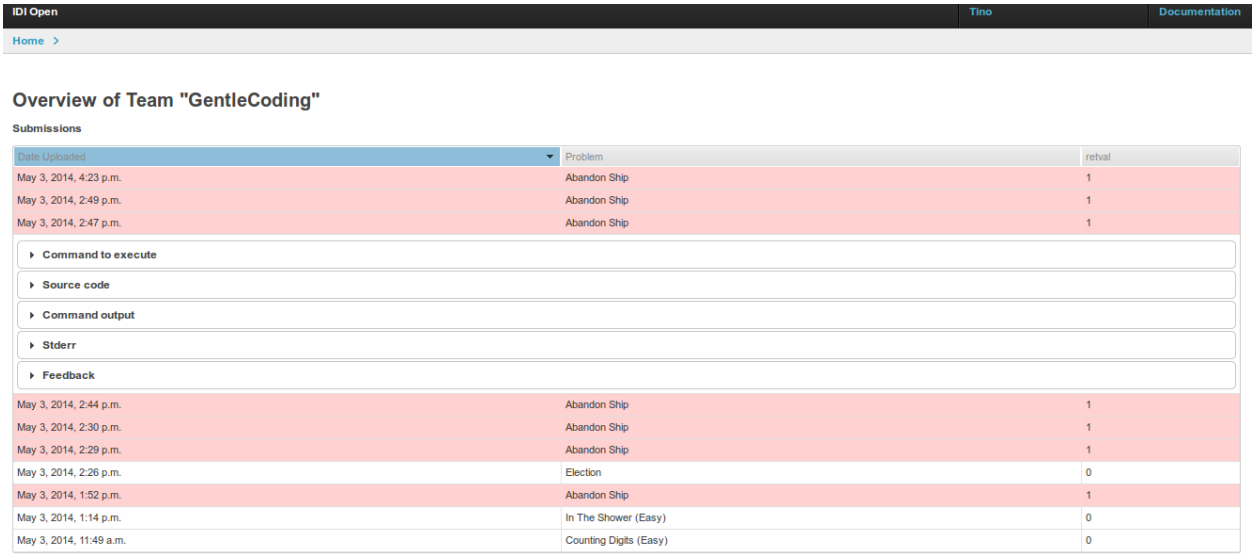


Figure 1.8: Judge views for team

Chapter 2

Implementation

This chapter goes into the details of our implementation. Django follows the MVC pattern in a quite strict manner, and as a consequence so does our project. In addition to MVC our project is divided into several Django apps, which are separate modules containing their own models, views and controllers. Each app serve a specific purpose and provide a certain level of modularity.

Figure 2.1 shows the directory structure to one of our apps, all apps follow this structure. An app's root folder contains four files worth taking a closer look at:

- models.py contains the app's models i.e. our database entities. Due to our site being MVC, every aspect of the site is in some way represented by a model defined in a models.py file.
- The file views.py defines the app's functions for handling requests, called views. Though the naming might be confusing, the views defined in this file are not views in the MVC sense of the word. The views are in essence MVC controllers. When an HTTP request is received by Django it is routed to a specific view, and the view then handles the request. Though most views simply serve web pages in response to GET requests, there are no limits as to what a view can be used for.
- The forms.py file contains a set of Django forms, which are simply collections of input fields. The forms can be rendered as HTML, and serve as validators of the input received by POSTs.
- This leaves the admin.py file. Django provides a quite modular and modifiable admin app for managing other apps. The admin page's main functionality is that of viewing, editing, creating and deleting models. However, the admin app does not have access to all of the models in the system by default. The admin.py file is where an app registers which of its models are to be modifiable by the admin page, how the models are to be rendered etc.
- The apps also contain a templates directory. The templates are in essence HTML files extended by Django's template language, making them easily processed/modified by Django. These templates corresponds to the MVC Views. When a Django view sends a response it is usually by inserting dynamic content into a template and then serving the final HTML file as an HTTP response. Though not visible in the figure, most of our templates are extensions of a global base template, this way redundancy is reduced and our user interface stays consistent.

Figure 2.1: App overview

2.1 contest

The contest app contains the most fundamental functionality and models for the system, namely the ones related to creating, hosting, and deleting contests. The contest app defines a couple of models for storing information directly related to a contest, such as sponsor information, support contact information etc. Just about every other model in the project is related to the contest models in some way. A complete overview of the models defined in the contest app can be found in Appenix E.

2.2 article

The article app provides basic functionality for posting news articles. It contains several different views for looking at articles, lists of articles etc. For editing articles the app uses a WYSIWYG editor, available in the admin interface.

2.3 userregistration

As the name suggests this app handles user creation, deletion and modification. The majority of this app is an open source app that we incorporated into our project, however, we made some modifications of our own.

2.4 teamsubmission

When a team has reached something they think might be a valid solution to a problem they submit their source code to the system. The uploaded source becomes part of a submission model which is part of the teamsubmission app. This app also defines some models related to the submissions model.

2.5 execution

The system needs a way of handling the submitted source code. For instance it needs some way of determining which compiler is to be used. When the source has been built the system needs to know what command is to be issued to the system to execute the binary. Both of these things are handled by the execution app. In addition there are restrictions set to limit the resources available to the submissions, for example the number of subprocesses, memory allocated etc.

The models defined in teamsubmission and execution can be found in the Appendix E.

2.6 node_manage

With a well configured system and the previously mentioned apps working properly, a submitted source file will be stored and the outline of how the file should be treated will be set when the file is uploaded. The code for actually performing the actions of building and running is handled by the node_manage app. The node_manage app fetches the appropriate settings for a submission, and submits it to a FIFO queue. Our backend consists of several execution nodes connected in a cluster powered by a framework called Celery. The nodes can be configured to handle any number of concurrent submissions, and when a node has got available capacity it fetches another submission from the queue. Celery relies on the AMQP message passing standard, by means of an open source message broker system called RabbitMQ. All messages passed go through a broker setup on the same host as the web server, the broker then distributes the messages to the appropriate host.

2.7 balloon

When a team has solved a problem, they are to be awarded a helium balloon. This app enables staff users to view problems that have newly been solved by a team, send somebody to deliver a balloon, and then remove them from the list of newly solved. This app simply provides a custom view in the Django admin page.

2.8 changeemail

Since we had to modify the userregistration app that we incorporated, not everything worked as we wanted out of the box. An example was the functionality for changing the email of a contestant, which broke the contestant's pending invites. This app provides a fix for that problem and makes sure that changing email works properly.

2.9 judge_supervise

This app provides judges with an interface in which they can see all submitted solutions and statistics for each team. For each submission, the judges can see compiler errors, execution output and source code.

2.10 clarification

During a contest questions can be asked by contestants to the staff. If a problem is ambiguously formulated, or they are experiencing system errors, these problems can be addressed by requesting a clarification. The questions are posted publicly on the website, as well as their replies.

Chapter 3

Development

This document describes the different phases of development the group went through in order to finish the product. To increase readability the first part of the document describes the process of working towards the milestones, as can be viewed in figure **??**. The second part describes each sprint in more detail including work done/completed.

3.1 Working Towards the Milestones

3.1.1 Milestone M-01 - Preliminary Report

From start to 09.02.2014

Eager to start, we had our first meeting 15.01.2014. During this meeting we discussed which tasks we wanted apply for After receiving the project assignment, we discussed our ambitions for the course and the end product. We agreed that we had a shared goal to receive a top grade in this course, and that we where all prepared to put in the work required to achieve this goal. The group was in doubt if we should try popular, enterprise-level tools and frameworks, or if we should stick to basic, previously used tools. We decided to let each member of the group to explore a tool on his own and present his experience to the others. If the tool seemed usable, we incorporated it into our project.

Our primary concern was that we would spend time on suboptimal tools, methods or frameworks. Thus, the group spent much time discussing and modeling the application to come.

3.1.2 Milstone M-02 - Mid-semester Report

From 09.02.2014 to 09.03.2014

Being aware of the large amount of programming ahead of us, we aimed to have the mid-semester report finished one week before the actual deadline. To shorten meeting time and strengthen our task overview, we had a meeting thoroughly discussing how Scrum worked. We decided to adhere more of the conventional Scrum standard. As a consequence we started to draft release and product backlogs. This resulted in a reduction in the number of hours used to administer and delegate tasks. We also got a better overview of what we wanted the end product to look like. This meant that we could reduce the amount of modeling, and focus more on the code.

The mid-semester report finished as planned one week before our deadline. We more or less completed our testing plans and concluded on management structure. The biggest challenge was how to implement support for user handling.

3.1.3 Milestone M-03 - First Release

From 09.02.2014 to 19.03.2014

Having finished the mid-semester report, the group now had a structured overview of the requirements specification, and approach to development. We had much coding to do in order to reach the third milestone. We tried to agree on an optimal approach, but concluded that we had to "just get started". In our sprint backlogs the amount of coding assignments grew. To induce more coding, we arranged informal coding nights in order to trigger "learning by doing" and improved our progression.

By the time we had finished the necessary prestudies and requirements, we already had some functionality. However, there was still work remaining, as suggested by our work breakdown structure. In addition we had a meeting with the customer where they proposed some new requirements, and reprioritized a few others.

In advance to the first release we had some meetings with the customer We were a little nervous regarding some of the design choices, however, the meeting discussing the design went well. We had formerly agreed on our mock up-design, although there were a few discrepancies between the delivery and what the customer wanted.

The deadline for our first delivery to the customer was 19.03.2014, but the actual release of the website was delayed to after the weekend, for external reasons.

3.1.4 Milestone M-04 - Presentation

From 09.02.2014 to 19.03.2014

Since the presentation was scheduled at the same time as our first release, we did not have time to prepare for this presentation. Nevertheless, we received valuable feedback from other groups.

3.1.5 Milestone M-05 - Beta Release

From 19.03 to 11.04

Working toward the beta release was challenging. Increasingly, we experienced that modeling the application before coding was not an optimal solution. Thus, we began to code without relying on diagrams to aid us. We sustained this approach until the end of the project.

With limited time, it became necessary to prioritize some tasks over others. Our improved product backlog proved to be a big benefit. As mentioned previously, we felt that it was hard to predict the outcome of the development process, so we decided not to update the Gantt diagram. Instead we relied on our own options and customer prioritizations. This was due to our new understanding of what needed to be completed when.

We did make some progress with our development, but still had some aspects of our frameworks that needed to be researched. As the weeks went by, we increased our work estimates and grew more familiar with the framework. Still our models seldom related to the actual end result. It was not something we felt was a big problem, as we where making progress.

3.1.6 Milestone M-06 - IDI Open Test Event

From 11.04.2014 to 26.04.2014

We still had quite a few packages to implement, and we were uncertain how much time we needed to spend on each of them. As a consequence we had to shorten our easter vacation. Spending this much time together, every day for weeks, may cause tension in groups. We felt it was important to create an environment to ease the tensions. Therefore we took breaks from the coding, eating pizza and playing foosball. We started every day discussing what we were suppose to do, similar to a daily scrum. We believed all members had a good tacit understanding of what needed to be done, so we transitioned from sprint backlogs to daily TODO lists. These lists were written informally for the sake of brevity.

The days were long, lasting from 09:00 to 24:00. Packages were implemented at a high pace, and the pieces where finaly starting to fall into place. The biggest challenges were to get the execution node up and running, highscore table, and contest management for the judges. Testing was also completed. We also had sufficient time to implement some of the lower prioritized requirements.

During the test event, we sat at our own table and received feedback from the judges and volunteers that had shown up. The fact that some of the judges were considered really good programmers made us a little nervous. They did give us feedback and a list of new requirements to be implemented. These were minor fixes, mostly related to the user interface. The test event itself was considered a success: all the judges approved our system.

3.1.7 Milestone M-07 - IDI Open

From 26.04.2014 to 03.05.2014

After the test event we got a new list of requirements. There was only one week to the actual event, and we had to carefully pick those we and the customer felt were the most important. We implemented support for several execution nodes, refined the contest management, and fixed small bugs. Some tasks were complex, so it was a challenging to predict if we would be able to finish them on time. The most advanced task we were given after the test event, was that the judges wanted a better overview of the contest. I.e. they wanted access to the whole competition and all the functionality, before the contest started. The customer also wanted to be able to export data to CSV and LaTeX. This task seemed lightweight at first, but turned out to be much more extensive. While finishing on time, this consumed more hours than initially planned.

In total there were 92 teams taking part in IDI Open 14, and a total of 214 registered users in the system. When the contest officially started and the problem set was released, all users simultaneously accessed the same resource. This caused a spike on the system load. We had been told by our customer that the old system had previously buckled under the pressure from this spike. Our system did, however, handle this well. Thus, the start of the contest went well.

At one point the system went down for a few minutes. This was because we ran out of hard disk space on our main server. In other words, the system had nowhere to store its data, and was unable to handle the requests made by users. After a couple of minutes of deleting unnecessary files, we discovered that for every file that we removed, we only bought ourselves a couple of more minutes of uptime. Somewhere in the file system there was a file growing at an alarming pace. Identifying this file was challenge. By monitoring the server's processes we found that the database was logging extensively. This resulted in a 1MB/s disk write rate. The rate was small enough that we could easily monitor and periodically erase the log to clear out disk space. We could have disabled logging, however, that would have required a restart of the database server and thereby downtime.

After this problem was resolved the rest of the contest went without any significant issues. Our system where capable of handeling a total of 12 concurrent submissions, which was more than enough. All parts of the website where responsive and working properly, except the highscore list, which we knew had performance issues. These issues did not have a significant impact on the user experience.

3.1.8 Milestone M-08 - Final report

from 03.05.2014 to 30.05.2014

After the final event we were all exhausted. The following week we only did some administrative tasks. We started working on the report based on the feedback we got from the supervisor and external sources.

Sprint by sprint

We have documented each sprint. These are given in appendix A.

Chapter 4

Testplan

To determine requirement, structural, and architectural coverage of our product, software testing has been performed. The tests are formalized to make it easier to agree on the coverage between the customer, maintainers, and us. The results and process is documented in this chapter.

4.1 Testing Strategy Overview

It is common practise to structure tests in three categories. This way, tests can be communicated to developers, stakeholders, and high-level non-technical users. Following is our interpretation of each category.

4.1.1 Unit Testing

Unit testing is the process of testing program components individually. The tests invoke methods and structures in the code using different input parameters. These are usually written before or immediately after a module is completed. This way, it is easier to assert that the module does what it is intended. Each test case is independent from each other, so several people can write test cases simultaneously without having to worry about dependencies.

4.1.2 Integration Testing

In development, many features are bundled into different components. The components are then joined together to form a system. The interfaces to each of these components, and how they communicate with each other, are tested during Integration testing. The purpose is to ensure that communication between the components is correct, and that the components work as intended. It can be extensive if those responsible for integration have to review the code in each component, so integration testing abstract code away. If there are any errors, then one will either review the unit tests or notify the author.

4.1.3 System Testing

System testing is a high-level test of the system. It is performed after all of the integrated system parts have been tested and joined together. System testing is a black box test, as anyone should be able to perform the test without having any knowledge of the underlying code. The purpose of system testing is to test if our system fulfills the requirements in the requirement specification. This is important to find out if we meet the expectations from the customer.

4.1.4 Acceptance Testing

Acceptance tests are usually executed by the customers. They are written after agreeing on the requirements specification for a delivery. The tests are then verified by the customer. Once both the customer and developers agree on the acceptance test, it will be possible to formally agree on whether or not a delivery meets the given requirements.

4.1.5 Testing Coverage

We wanted to provide complete test coverage, unfortunately, due to time contraints we where unable to achieve this goal. Thus, we needed to prioritize which components of the system were most prone to error, and most important to test. The following were our software assurance objectives:

- Ensure that the system can be used by many users
- Ensure that the contest can be held without any error that would critically impact the contest

Errors that solely impacted user experience were not prioritized to test. The majority of these were intended to be found from debugging the system. Since the developers would work closely with each other, we concluded that we would fix small errors in regression. If our team had more members, or if we had been working in different locations, this would have been a higher priority.

In most projects, testing is used to ensure requirements coverage. In our case, however, with frequent customer-meetings and iterative development, we have not had a strong need for this. The customer has had access to prototypes of our solution and our source code. In order to see that the product does as intended, they could simply try it out for themselves.

As per our software assurance objectives, our largest focus has been simulating the role of a contestant. To meet our objectives, we intended to do a full coverage of all contestant scenarios. The privileged users were believed to be technically experienced and without intention to do harm. We still felt it was important to prevent user errors, but our coverage was not as complete for these usergroups.

Since we were developing a website that would feature many users, developer testing alone could never simulate peak values for system demand. We have relied on load testing, giving our Web server a fixed amount of HTTP requests per second, hereafter RPS. What pages were used in the simulation was determined by us. Thus, our testing also extends to cover simulated peak values for high loads.

4.2 Our Approach to Testing

This section describes our approach to planning the different testing categories.

4.2.1 Unit Testing

We performed unit testing after the completion of a testable module. The unit tests use the PyUnit framework, and is written by another person than the one who produced the code for the module. I.e. if person A makes module M, then person B will write the unit tests for module M. The reason for having another person writing the test for a module is because that will give more people insight in the code, and make it easier to discover problems. The unit tests reside in the test.py file in each Django app.

4.2.2 Integration Testing

Each integration test will test a different interface. The interface is defined as the connection between the different components in our system. The pre- and post-condition sets the boundaries for the test. Input and output is used to determine if the test produces the expected output with a corresponding input. The motivation behind integration testing is that we can determine whether a module has been successfully integrated. By going through the accompanied tests made for the interfaces that interact with the module

4.2.3 System Testing

Each separate test in the system test is linked to one or more of the requirements from the requirements specification. This is to ensure that the system meets all the requirements set by the customer. The template for system testing starts with specifying which function is being tested. After that we say what the action/input should be, and what the expected result is. The expected result needs to be achieved for the test to be considered successful.

4.2.4 Acceptance Testing

The customer performed an acceptance test before each release of the system, so they could confirm that we met the expected requirements. The acceptance test was based on our system test, with the customer executing the tasks in the system test. It was approved when the customer was satisfied with how we implemented the requirements.

4.3 Testing Results

The results from our testing is presented in this section.

4.3.1 Integration Test

Each test has a unique identifier, name, pre/post-conditions and corresponding input and output. An example is given in table 4.1.

Table 4.1: Integration test for adding a sponsor

ID	IT-01
Interface name	Add sponsor
Pre-condtion	Contest is created
Post-condition	Sponsor and image
Input	Image, URL
Output	Sponsor in contest

In section X.X[12. Evaluation of testing methods] we explained why our coverage by integration testing was not extensive. The written integration tests are from our milestone M-03, First release, and only cover the requirements that was necessary for that milestone. As such, we have chosen to move all the integration tests to Appendix D

We formally agreed on what modules our system was made out of and their interfaces. Figure 4.1 shows our view on the system as per milestone M-03. In figure 4.1, we have replaced some default UML symbols and replaced them with the equivalent UML stereotype. The explanations are given in table 4.2. The integration tests we did make are given in appendix D.

Table 4.2: Symbiology for our UML component diagram

UML stereotype	**Function**
<<provides>>	The component delivers the given functionality
<<requires>>	For the component to work it must have the given interface

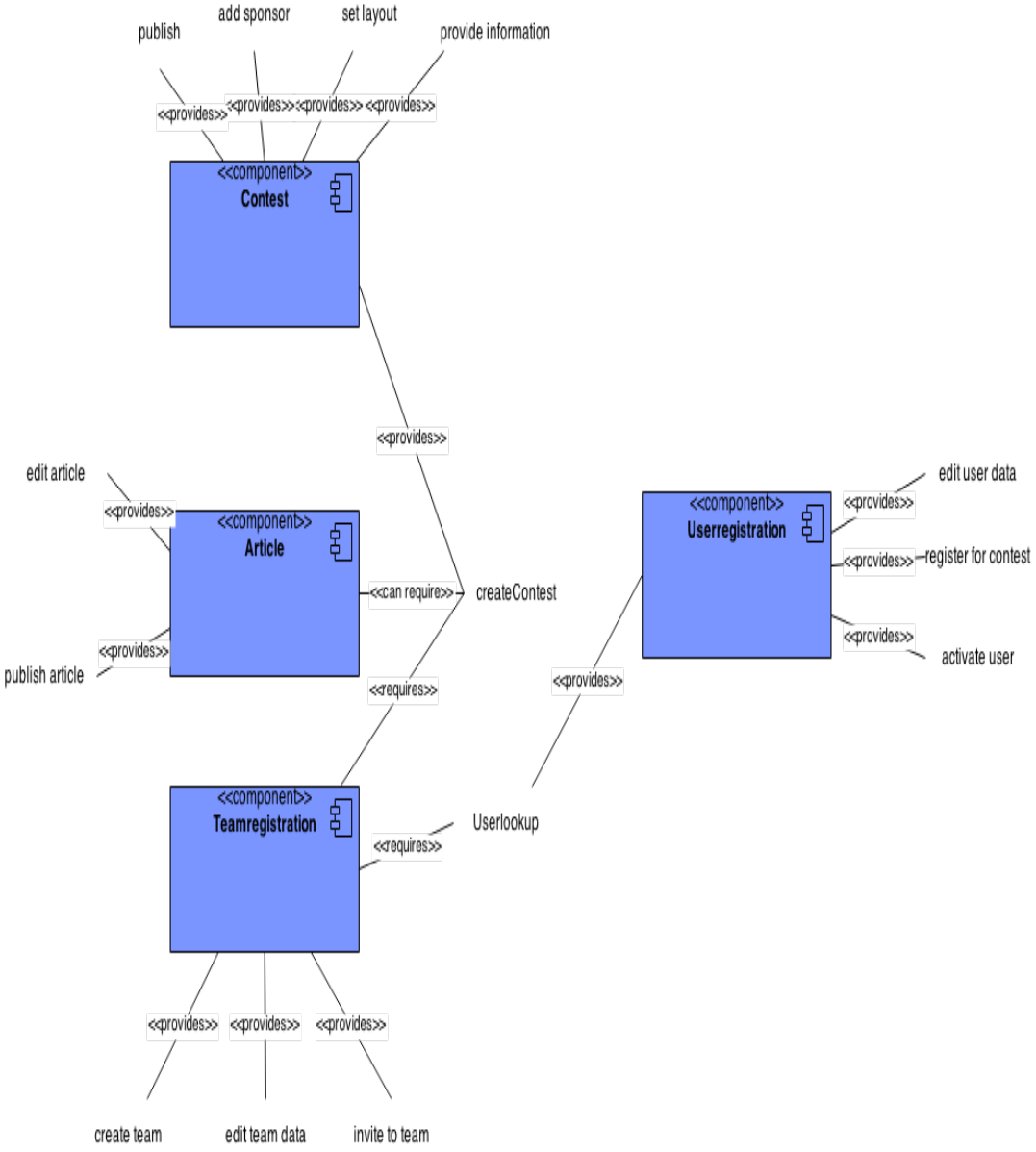


Figure 4.1: Diagram from milestone M-03. Each interface connection, especially “createContest” has been tested

4.4 System Test

Our system tests cover all the functional requirements. All tests are written as successive cases. This means that the tests do not cover scenarios for how the system should respond when a user performs an error or another external fault occurs. The complete listing is in table 4.3.

Table 4.3: System test

ID	Function	Action/Input	Result	Req	Pass/Fail
ST-01	Create a contest, and publish an article to that contest. Edit article.Then, delete the contest.	Contest name, article text	Contest and article is no longer publicly available	FA-16,	PASS
ST-02	As a contestant, create a team and invite contestants. Go to profile page and see which team the contestant is amember of. Then, delete the team	Team, contestants, contest	First contestant in team, then contestant not in team	FE-01 FE-02 FE-04 FE-06 FC-04	PASS
ST-03	Add custom css, specify custom settings,	Existing contest, css, compiler flags, penaltysystem, maximum numbers of contestant, maximum number of contestant per team	Contest with custom css and settings	FA-05	PASS
ST-04	Log in as admin, and enable all judges to createa contest. Then remove and add a judge, by escalating and de-escalating privileges from contestant.	Admin account, contestant account	Zero changes to system.	FA-09	PASS

Table 4.3 – continued from previous page

ID	Function	Action/Input	Result	Req	Pass/Fail
ST-05	Log in as judge, create a problem and upload cases. Upload different solutions; one correct, one erroneous, and one that loops forever. After that, modify the problem before deleting it.	Problem, solutions, erroneous code, judge account	Only the correct solution should give points.	FJ-01 FJ-02 FJ-03 FJ-04 FJ-05 FJ-06 FJ-07	PASS
ST-06	Add two execution nodes with different compiler supports. Change both nodes, such that they take each other's compiler setting. Then remove both nodes.	Compiler profiles, available nodes, production server, administrator account	zero added nodes, no errors in execution	FA-12 FA-13	PASS
ST-07	As a contestant, submit a question to the judge. As a judge, receive a notification, and answer both the contestant and globally.	Contestant, contest, question, answer	All contestants should be able to see message, successful communication between judge and contestant	FJ-08	PASS
ST-08	Create a contestant account. Activate the account via email, and change the email. Ask for lost password on the new email.	Contest-data, emails	Activation data received on the email, and all links word	FC-01 FC-02	PASS

4.5 Non-functional testing

The tests related to performance usually comes in pairs, a value and the double of that value. This applies to the input and expected result. This is to ensure that system performance does not scale down in a non-linear way. E.g. if "X" transactions are processed and the server begins using swap memory instead of RAM, this would mean that a high load would cause an exponentially slower load rate for a high number of transactions.

Often, as mentioned in section X.X, we did inspection tests. Thus, table 4.4 does not contain all tests that are executed, and the table only covers the first 12 non-functional requirements. The documented tests do, however, ensure some requirements coverage.

Table 4.4: System tests

Case	Input	ID	Expected Result	Pass/Fail
Adding 500 contestants	500 users	NF-04	Ability to add yet another	PASS
Adding 200 teams	200 teams	NF-05	Ability to add yet another	PASS
Adding 20 judges	20 judges	NF-06	Ability to add yet another	PASS
Adding more than one admin	> 1 admin	NF-07	Ability to add yet another	PASS
Upload a solution which is less than 50kB	Solution $>$ 50kB	NF-08	Successful delivery	PASS
Upload a solution which is greater than 50kB	Solution $>$ 50kB	NF-08	Error message	PASS
Gather some test persons not familiar with the system and have them use the system as a contestant	System	NF-09	They should be familiar with the system after 5 minutes	FAIL
Gather some test persons not familiar with the system and have them use the system as a judge	System	NF-11	They should be familiar with the system after 10 minutes	PASS
Gather some test persons not familiar with the system and have them use the system as an admin	System	NF-10	They should be familiar with the system after 15 minutes	PASS
Page responsiveness with at least 5 RPS	HTTP GET and POST to all pages	NF-01	Response-time $<$ 150 ms	FAIL
Page responsiveness with at least 10 RPS	HTTP GET and POST to all pages	NF-01	Response-time $<$ 300 ms	FAIL

Chapter 5

Risk Management Framework

A risk is an event or condition that, if it occurs, could have a negative effect on a project's objectives. To avoid these risks, and to be able to deal with them effectively, we established a risk modelling framework Our framework is based upon our own experience and examples from the many documents that exists on the subject.

By explicitly writing down corresponding actions for risks that occur, we could deal with risks without disagreements. It also let external parties get an overview of what risks we are aware of, and how we reviewed them. The external party can then notify us of unknown risks or modifications to our priorities.

5.1 Terminology and Categories

To structurize our risk register, we divided each into the following categories:

- **Budget risks** are all risks that can be associated with financial aspects of our project.
- **Organizational risks** are those that might arise because of group structure and task delegation.
- **People Management** comprises all risks associated with team management and each individual in the group.
- **Requirements risks** are related to errors in requirements engineering.
- **Schedule risks** are about meeting deadlines and task delegation.
- **Technology and tools**; product talk about technical risks that might arise with tools and our product.

To prioritize our risks, we have also given each risk a probability, consequence and total risk, abbreviated Pr, C, TR, respectively. Each of these were assigned values from 1-10, where 10 indicated "very high". A 10 translates to the following for each field:

- **Consequence**: event of risk will be fatal to our project.
- **Probability**: risk will probably happen
- **Total risk**: The risk is a big threat and should be monitored closely.

Total risk is calculated as Consequence x Probability. By multiplying these numbers, we get a sorted list of the most dangerous risks.

5.2 Scope of Risk Assessment

Finding the right balance to the extent of documentation is difficult. Extensive risk-frameworks can consume more hours in maintenance than they save. To deal with our lacking experience, we only wanted to document the most likely risks. To us, this meant only including risks with a total risk value of more than 30

We considered specifying additional information to each risk, like context and associated risks. However, we felt every member of the group had a similar understanding of the risks, so writing this information down would be superfluous. In addition, since the risks were orally reviewed, we did not want to rely too much on what had been written down.

5.3 Risk Identification

We tried to involve every group member in the making of the risk register. The estimates from 1 to 10 were assigned based on our own experience from previous projects. The list was filled out by three members of the group, and then later presented to the whole group for reviewal and agreement on the values.

Risks that became known in later parts of our development was promptly added to our risk register. We expected few of these, and few did occur, so we have not performed any revision control. Our means of identifying risks was through discussions and agreements that we were not performing optimally.

5.4 Risk Monitoring

Our primary method for surveilling risks was weekly discussions. In these meetings, we had open discussions of the group's progress and development. In addition, we had one monthly meeting where we would discuss the risks more thorough and in-depth. This involved re-discussion of the group's expectations and our involvement in the project. These monthly meetings were referred to as "snapshots". The snapshots specifically addressed the problem that many projects start out quite ambitiously, but tend to deteriorate, something we wanted to avoid.

To avoid groupthink[1] and complacency, we required each group member on our weekly meetings to mention three good and three negative points. After that, each member could bring up extra topics for discussion. For each discussion, we made sure to be conclusive by explicitly writing how to deal with a given problem.

We have frequently involved the supervisor and customer in our process. We made sure to ask for insights on our development progress. After each meeting we also wrote down meeting minutes and a summary. This was later sent to the respective party to ensure agreement on what had been concluded in the meeting.

5.5 Complete List of Risks

We have chosen to put the complete list in Appendix C.

[1] The concept of trying to avoid conflict by not speaking one's mind. For more, see: http://www.psysr.org/about/pubs_resources/groupthink%20overview.htm

Appendix A

Sprints

This appendix holds an overview over our sprints, throughout the project. For a more complete list over packages completed see [insert section where activity/sprint backlog are]

This is just an overview were we are trying to bring out the more important aspects of our sprints.

A.1 Template

Sprint: <sprint nr>	Working towards: <insert milestone
Overview over packages to be completed: <Insert packages to be completes>	
Improvements: <insert list over things we want to improve about ourself>	
Notes: <any notes>	
Packages completed: <insert packages actually completed>	
Summary: <A brief summary over the most important aspects>	

A.2 Sprint 0

Sprint: 0	Working towards: M-01
Overview over packages/tasks to be completed: • Get an overview over the course • Get to know the old system	
Improvements:	
Notes: • This was the first meeting after getting the assignment	
Packages completed:	
Summary: This was still early in the process so most of the time was spent getting an overview over the whole thing.	

A.3 Sprint 1

Sprint: 0-a	Working towards: M-01
Overview over packages to be completed: • Read and learn the requirement received from the customer • Set up tools • Project management • Learning tools and framework	
Improvements: • A better meeting structure	
Notes:	
Packages completed: • Tools for communication was set up	
Summary: Learning to know the requirements and the subject as a whole was our main concern at this stage. We also did some research on what framework we should use.	

A.4 Sprint 1

Sprint: 1	Working towards: M-01
Overview over packages to be completed: • Project management • Install and learn tools • Report	
Improvements:	
Notes: • Tino and Eirik was sent out on seminar. Learning about SCRUM • Trying to use ICEScrum for Scrum related activites	
Packages completed: • WBS • Risk assignment • Functional requirements • Class diagram	
Summary: Most of the tools was set up, we started to some modelling, in order to get a better overview over the system to be implemented. This was also documentations to be used in the report. We also systematized the requirements in order to communicate with the customer. Project roles was also distributed.	

A.5 Sprint 2

Sprint: 2	Working towards: M-01
Overview over packages to be completed: • Project management	
Improvements:	
Notes:	
Packages completed: • Requirement specification • System architecture – Flow charts – class diagrams • ER-Models • Preliminary report	
Summary: At this point we had a rough understanding of the work ahead of us, and we were able to start modelling possible solutions. This was also close to the deadline for the preliminary report and as a consequence a lot of time was spent on the report.	

A.6 Sprint 3

Sprint: 3	Working towards: M-02
Overview over packages to be completed: • Development	
Improvements: • Better sprint planning • We should improve our task delegation • We should prioritize tasks	
Notes:	
Packages completed: • Development	
Summary: During the past two sprints we had primarily been planning and doing administrative tasks. This sprint marked the end of that phase. We moved on to actual implementing. However, we ere not familiar with the tools and frameworks available to us, and as a consequence we decided to use this sprint to get everyone up to date on Django//python. We had a coding night this sprint. Working all members together.	

A.7 Sprint 4

Sprint: 4	Working towards: M-02
Overview over packages to be completed: • User-interface • Project management	
Improvements: • The activity diagrams does not reflect upon our actual work done.	
Notes:	
Packages completed: • User interface	
Summary: During sprint 4 we knew we had to improve our WBS. We had a long meeting where we rebuild our backlog, reviewed SCRUM and created a release- and backlog.	

A.8 Sprint 5

Sprint: 5	Working towards: <insert milestone
Overview over packages to be completed: • Development • Report • Tesplan	
Improvements:	
Notes: • This sprint we had a meeting with the supervisor discussing the activity diagrams. Show suggested that we switch them with our	
Packages completed: • Sponsor support • Testplan	
Summary: We had a good overview over what should be in the report at this point. A finished version was right around the corner. In general, this weeks meeting went much faster than the last. The group was happy about that.	

A.9 Sprint 6

Sprint: 6	Working towards: M-02/M-03
Overview over packages to be completed: • Mid-term report	
Improvements:	
Notes:	
Packages completed: • Mid-term report • Testplan • User-interface completed in bootstrap	
Summary: This sprint we finished the mid-term report and the user-interface was completed. We was happy with the resut. We also finished the mid-term in food time before the actual deliver.	

A.10 Sprint 7

Sprint: 7	Working towards: M-03/M-04
Overview over packages to be completed: • Representation • Implementation • Write tests.	
Improvements: • We must be better to work with other group members code.	
Notes:	
Packages completed: • Login completed • User registration completed • Team registration completed	
Summary: During this sprint we had boost with the implementation. We were busy making our Firs release. Unfortunately we did not have time to set up the solution live this sprint .It was postponed to after the weekend.	

A.11 Sprint 8

Sprint: 8	Working towards: M-05
Overview over packages to be completed: • Testing • Set up solution live • Fixing bugs • Peer evalutaion	
Improvements:	
Notes: <any notes>	
Packages completed: • Testing • Bug fixing – Change email – Forgot password • Peer evalutaion	
Summary: After we put the solution up, there was sum bugs and testing to be done. We had not had the opportunity to test, by our standards, yet. We did this while the solution was live.	

A.12 Sprint 9

Sprint: 9	Working towards: M-06
Overview over packages to be completed: • Implementation • Permission testing • user manual • Project mamagement	
Improvements: • We had to be more consistent with testing • Better to fill out sprint documents.	
Notes: • We received the Peer Evaluation.	
Packages completed: • Possible to upload solutions • Models	
Summary: This sprint was probably our worst planned sprint. With better planning we could have finished a lot more coding. Unfortunately this was not the case and we spent unnecessary much time in the wrong direction. We were, however happy with our peer evaliation.	

A.13 Sprint 10

Sprint: 10	Working towards: M-05
Overview over packages to be completed: • Implementation	
Improvements: • Still improvement to been done with filling out sprint backlog.	
Notes: • This sprint was 9 days long	
Packages completed: • Implementation – Execution nodes – Compiler profiles – Upload solution	
Summary: This was the last sprint before Easter. We were more thrilled with this sprint but. we knew had to shorten our easter vacation. We had a good start with much of the implementation and we finally felt like we had a good overview over everything.	

A.14 Sprint 11

Sprint: 11	Working towards: M-05
Overview over packages to be completed: • Implementation	
Improvements: • We knew we needed discipline to make it	
Notes: • Parts of this sprint was during easter • This sprint was 11 days	
Packages completed: • Upload submission • Penalty systematized • Review system status • Judge supervisor • Error messages	
Summary: During this sprint, we did not setup a sprint backlog. Instead we kept an well documented TODO list. Every day all members would tell which tasks from the TODO list they would work on. At the end of the day we told each other what was missing. This sprint went great and we were actually finished some days before M-05-.	

A.15 Sprint 12

Sprint: 12	Working towards: M-07
Overview over packages to be completed: • Development • Bugfixes • Setup	
Improvements:	
Notes: • Last sprint before final event	
Packages completed: • Highsvore • CSV and PDF support • Several execution nodes • judge contest acsess	
Summary: Are last sprint before the final event consisted mainly on small bugfixes. There were, however, some tasks that took longer time than estimated. That would be CSV and PDF	

A.16 Sprint After

Sprint: After	Working towards: M-08
Overview over packages to be completed: • Final report • Small bugfixes • User Manual	
Improvements: • Effiency and communication is import this last period	
Notes: • We did create a traditional sprint backlog for this sprint. We did however have frequent meeting discussing what to finish when	
Packages completed: • Final report • small bugfixes • User manual	
Summary: When we worked towards the final report we decided on a different tactic than the other sprints. Instead of creating a sprint backlog, holding all the tasks, we broke down the report into chapters. Some of which was already finished. For each chapter we talked about what key ponts we wanted to write about for so deciding a pair that should write that part. Then, before we met next time, another pair would view, comments and generally share some points about that chapter.	

Appendix B

User stories

Role: Admin

ID	Priority	Story
SA-01	HIGH	Will be able to create a new contest. When doing so a new web page should be created, but whether the site should be immediately published or not is optional. The content of the new site follows a strict template, but adding a custom css-file will be possible. Each contest has got its own settings, containing a list of supported compiler profiles, compiler flags, penalty system, maximum number of contestants, maximum number of contestants per team, and of course a date and a name. When creating a contest the admin needs to provide a name and a date, the other settings may be skipped and default settings will be used.
SA-02	HIGH	Users are organized in user groups(admin being one of them). By default three usergroups are provided, admin, judge, contestant and functionary. The entire solution is based on independent modules of functionality and each user group has got access to a subset of these modules. The admin is the only non-modifiable user group, admins have access to all modules. The admins can modify all other user groups, change permissions of a group and remove/add member to a group, this includes promoting new admins. The admins are also able to deactivate users, and even remove them from the database.
SA-03	MED	The system is able to gather a large variety of statistics, what data is to be collected is decided by the admins.
SA-04	HIGH	The system uses a collection of nodes(computers) for assessing submissions. The admins can add a node by providing an IP address and the username and password of a privileged user on that node. These nodes can also be removed by the admins. The nodes can also be managed in terms of compiler profile support.

Table B.1– continued from previous page

ID	Priority	Story
SA-05	HIGH	The web page associated with a contest consists of a set of news items, these can be added by the admin. As with the entire contest web page the publishing of the news item can be set to a certain date and time. The news items can also be removed or modified later on.

Role: Judge

ID	Priority	Story
SJ-01	MED	A judge can submit a problem, where he/she will be able to upload cases with input/output. He/she can give every case a name. For each problem the judge can set a resource limit (time + memory) for each compiler profiles. He/she can upload different solutions that gives the right output, timeout and the wrong answer. All the solutions should be run-able and produce an output about the expected result, and if the execution time is inside the given boundaries. He/she should also be able to check that all problems have associated solutions that give right and wrong answer, and timeout.
SJ-02	MED	A clarification system will be available to judges, where they can receive and respond to messages from contestants. When receiving a message, the judge will get a notification (possible in in the bottom right corner of the website, [Design choice]). A judge can choose to either send a global message or a message to a contestant or a team. A global message will be sent to every contestant in the competition.

Role: Contestant

ID	Priority	Story
SC-01	HIGH	A contestant should be registered with an email, name, gender, and study programme and level. When registered, he/she should receive a confirmation email. After confirming the account, a contestant should be able to log in.
SC-02	HIGH	When a contestant is logged in he/she will have access to account information and which teams he/she are invited to, as well as earlier contests and teams they have participated in. The contestant should be able to edit account information
SC-03	MED	A clarification system will be available to contestants, where they can ask questions to the judges. They will also have access to answers the judges have marked as global.

Role: Functionary

ID	Priority	Story
SF-01	LOW	When a team completes a problem, a table containing the group name and location should be updated to include this. Each problem has a corresponding balloon colour. A balloon functionary should be able to register a balloon colour to each problem.

Role: Teams

ID	Priority	Story
ST-01	HIGH	A contestant must [18.02] be able to register a team, upon registration he/she is required to input team name, whether or not the team is onsite, a team password, and a email for the team leader.
ST-02	HIGH	The team leader should be able to edit the team information, invite new members, and delete the team before the competition. To invite new members you input their email, and they receive a registration link, where he/she inputs name, gender and nickname. If the contestant [changed from email 20.02] is already in the database from a previous competition, the email they receive contains a confirmation link. Every contestant can manage the team they are a member of. All informations is editable in the team overview which can be reached from a contestants login. A confirmation email is sent to the edited user.
ST-03	MED	A team should be able to deliver submissions to problems, and get a response from the system. The response should be whether the submission is right, wrong, or gives timeout.

Appendix C

Risk List

C.1 People Management

Description	ID	Pr	C	Tr	Preventative action	Remedial action
Personal argument	PM-01	8	5	40	Frequent meetings and social events	Open discussion
Dependency on team member	PM-02	6	6	36	Short sprints and team members usually work in groups of two	New meeting where we consider a redistribution of WP
Underburdened teammember; slack	PM-03	7	4	28	Keeping track of the work done by each member as well as the number of hours spent on any given WP. In the beginning of the sprint focus more on an evenly distributed workload among team members.	If the team-member continues to slack put it on the agenda for the next meeting and allow the team-member to explain his/her reasons for slacking.
Team members are late	PM-04	9	2	18	If you are late, you need to bring a cake or cookies to the next meeting	You need to bring a cake or cookies, and if it happends several times, an extraordinary meeting will be called, where new consequences will be discussed.
Team member is not qualified for any assigment	PM-05	4	7	28	Try to keep every member up to date on the entire system by not letting anyone work for too long on the same part of the system.	Add unqualified member to an existing pair working on a WP.
Miscommunication	PM-06	7	3	21	Frequent meetings with discussion about team letting all team members try different areas in the application	As per SDLC; evaluation, analysis, restart assigment
Dependency on external person	PM-07	3	6	18	Frequent communication with the customer.	Well-planned sprints with a low level of dependency between WPs.
Displacement; team members do not feel comfortable in group	PM-08	2	7	14	Social events.	Talk to our supervisor and ask for suggestions
Overburdened teammember	PM-09	4	2	8	Short sprints and small WPs. A team member will only be assigned to a few WPs at a time.	Frequent meetings where WPs can possibly be redistributed.

C.2 Budget

Description	ID	Pr	C	Tr	Preventative action	Remedial action
Maintenance costs exceed expectations	B-01	5	3	15	Use highly maintainable frameworks as much as possible, and stick to Open Source as much as possible.	Optimizing code base in hopes of increasing maintainability.
Third party plugin demands more money than initially expected	B-02	2	3	6	We've got a green light for putting GentleIDI under the GNU Public License, which means that we have got free access to software under GPL.	Look for alternative plugins.
Unexpected need for non-free third-party service	B-03	3	3	9	Extensive research on tools needed,before we decide on what we are going to use.	Look for alternative free third-party services
Maintenance requires access to tools/environments that cost money	B-04	2	3	6	Use highly maintainable frameworks as much as possible,and stick to Open Source as much as possible.	Request customer meeting to solve the issue.

C.3 Schedule

Description	ID	Pr	C	Tr	Preventative action	Remedial action
Pre-studies require more time than anticipated	S-01	9	7	63	We have a WP for pre-studies, and have included it in our sprints	Revise our WBS, and possible have an increased workload/work-hours in the following sprints, so we don't fall behind our schedule.
Failure to meet requirements on time	S-02	5	8	40	WBS, milestones plan and short sprints (1 or 2 weeks) allow us to focus on deadlines, and continously see our work progress	Have extraordinary meetings with supervisor and the customer to discuss the further development of the project. Be apologetic towards the customer, and come up with a new plan, that the customer is satisfied with.
Sprint-estimations are off	S-03	9	5	45	The whole group participate in planning a sprint, and estimating each task	Re-adjust our estimations in the next sprint, and in that way learn from our mistakes.

Table C.3 – continued from previous page

Description	ID	Pr	C	Tr	Preventative action	Remedial action
Failure to deliver sufficient documentation on time	S-04	5	6	30	WBS, milestones plan and short sprints (1 or 2 weeks) allow us to focus on deadlines, and continously see our work progress	Meetings with supervisor and customer, agree upon a new deadline, and increase the workload the following days to we meet the deadline.
Need for extra technology / features that requires training to use	S-05	3	6	18	We use extensive frameworks who has a lot of documentation, which makes it easier to learn.	Adjust the WBS and our sprints so we take into account that we need more time to learn new technology. Focus on this in the coming sprint planning.

C.4 Organizational

Description	ID	Pr	C	Tr	Preventative action	Remedial action
No person has responsibility for an assigment, although it is believed to be delegated	O-01	8	6	48	Strict use of the activity plan. The activity plan should be kept consistent at all times, this way all members know what the others are doing at any given time.	When discovered the given WP should be marked as unallocated in the activity plan and treated like any other WP in the sprint.
Project is, at current point not satisfactory, and it is hard to understand why	O-02	6	7	42	Writing meeting summaries, and in general keeping track of what is being done and how.	Review what work has been done up untill that point, how it has been done, and try to find a solution to the problem.
Bottleneck; in order for team-members to advance, other team members must finish their work	O-03	7	7	49	Try to avoid dependencies between WPs when setting up sprints. In case of such dependencies being unavoidable these WPs should be scheduled at the beginning of the sprint.	Delegate or even create new WPs to the team members currently being idle.
A task is delegated to more than one person	O-04	2	3	6	Strict use of the activity plan. The activity plan should be kept consistent at all times, this way all members know what the others are doing at any given time.	The two members should discuss how the issue should be solved, and update the activity plan according to that.

C.5 Tools and Frameworks; Product

Description	ID	Pr	C	Tr	Preventative action	Remedial action
End product is not satisfactory	TT-01	2	9	18	Customer meetings regularly, and keeping in contact through e-mail aswell. Give the customer access to our git-repository, so they have access to our source code, and also perform different type of tests (user-testing, etc)	Call in to a meeting with our supervisor, and our customer. Explain what went wrong, apologize and deliver our documentation.
Tools used for development are not suitable / efficient in later parts of the project	TT-01	2	8	16	Researching the tools we use, and planning ahead. Development planning allow us to discover problems before they appear.	Look for alternative tools. If changing tools involve a lot of work, and changes to the project, decide in a meeting if we want to continue with the inefficient tools, or if we want to make the change.
Problems with integrating components	TT-03	7	3	21	Have extensive system documentation and planning. Involve the whole group in the process.	Re-evaluate our system architecture, and look for solutions that won't affect other parts of the system.
Other solutions available make our product less desirable	TT-04	1	8	8	Do thorough work on the system requirements in hopes of providing a system well-tailored to the customer's needs.	Reevaluate the requirements.
Network cannot deal with traffic	TT-05	1	8	8	Keep optimization in mind when developing.	Try to find redundant data being sent possibly apply use of compression.
Submitted program has access to resources	TT-06	5	5	25	Submitted programs are to be run by a sandbox-user with a very restricted set of resources available.	Review code in hopes of finding the bug.
Platform / hardware unavailible, such that testing is difficult	TT-07	2	5	10	We use services provided by companies known to provide good system uptime. Most of our tools are hosted by Red Hat.	Setup temporary development environment.
Tools used in initial development are not available after release, and future developers have difficulty extending product	TT-08	2	3	6	Make sure requirements are written properly, understood properly, succint, etc	Document our work, so it is easy for future developers to understand the system.

Table C.5 – continued from previous page

Description	ID	Pr	C	Tr	Preventative action	Remedial action
Database cannot handle amount of transactions	TT-09	1	4	4	Keep optimization in mind when developing.	Optimize code in order to lower amount of transactions.
A tool does not perform the functions it was intended for	TT-010	2	3	6	Learn the tools properly, and read the documentation provided with each tool.	Look for alternative tools.

C.6 Requirements

Description	ID	Pr	C	Tr	Preventative action	Remedial action
Major change to requirements	R-01	5	4	20	Customer meetings regularly where we agree upon a requirement specification.	New customer meeting where we re-evaluate the requirements specification,and which priorities each requirement has.
Customer fails to understand impact of requirements	R-02	2	7	14	Customer meetings regularly where we agree upon a requirement specification.	Customer meeting where we explain the impact of the requirement,and get the customer to explain their requirements that we have different opinions on.
Finished product does not meet requirement	R-03	1	9	9	Customer meetings,they have access to our git-repository where our source code is	Test-events where they can test the functionality. Finish our documentation,and pass it on to other developers. Apologize to the customer.
Failed interpretation of requirement	R-04	3	4	12	Customer meetings regularly where we agree upon a requirement specification.	Customer meeting where we re-discuss the requirement specification,and make sure we understand what the customer wants.

Appendix D

Integration Tests

ID	IT-01
Interface name	Add sponsor
Pre-condtion	Contest is created
Post-condition	Sponsor and image
Input	Image, URL
Output	sponsor in contest

ID	IT-02
Interface name	Publish contest
Pre-condtion	Working database, website
Post-condition	Contest entity with unique ID, which has own subdomain with an interface for individual CMS
Input	Image, URL
Output	contest available from webroot

ID	IT-03
Interface name	Set layout
Pre-condtion	Contest is created
Post-condition	Contest-subdomain stylized with given stylesheet-file
Input	Image, URL
Output	Contest in database, modifiable

ID	IT-04
Interface name	Provide information
Pre-condtion	Contest is created
Post-condition	Information pages
Input	Image, URL
Output	Availablee articles

ID	IT-05
Interface name	Create contest
Pre-condtion	Working database
Post-condition	Contest is created
Input	Name, URL, dates, links
Output	A contest in the database that can be published and insert content

D.1 Article

ID	IT-06
Interface name	Create article
Pre-condtion	Contest is created
Post-condition	Article created
Input	Text, URL, date, images(optional)
Output	Article in database

ID	IT-07
Interface name	Edit article
Pre-condtion	Article is created
Post-condition	Article changed
Input	Text, URL, images(optional)
Output	an interface to edit the content of articles

ID	IT-08
Interface name	Publish article
Pre-condtion	Article is created
Post-condition	Article published
Input	date
Test-method	Manual inspection
Comment	An article available to end-users

D.2 Userregistration

ID	IT-09
Interface name	Create user
Pre-condtion	Working database
Post-condition	User created
Input	email, name, gender(optional), study level
Output	A contestant in the database

ID	IT-10
Interface name	Edit userdata
Pre-condtion	User created
Post-condition	Userdata changed
Input	User, data for user-attributes
Output	A modified user-entry in the database

ID	IT-11
Interface name	Activate user
Pre-condtion	User is registered
Post-condition	User is registered as active
Input	User
Output	Ensure that user can log in and is labeled as activated account

D.3 Team Registration

ID	IT-12
Interface name	Invite to team
Pre-condtion	Team is created
Post-condition	Contestant invited to team
Input	email
Test-method	A contestant receives an invite to a team

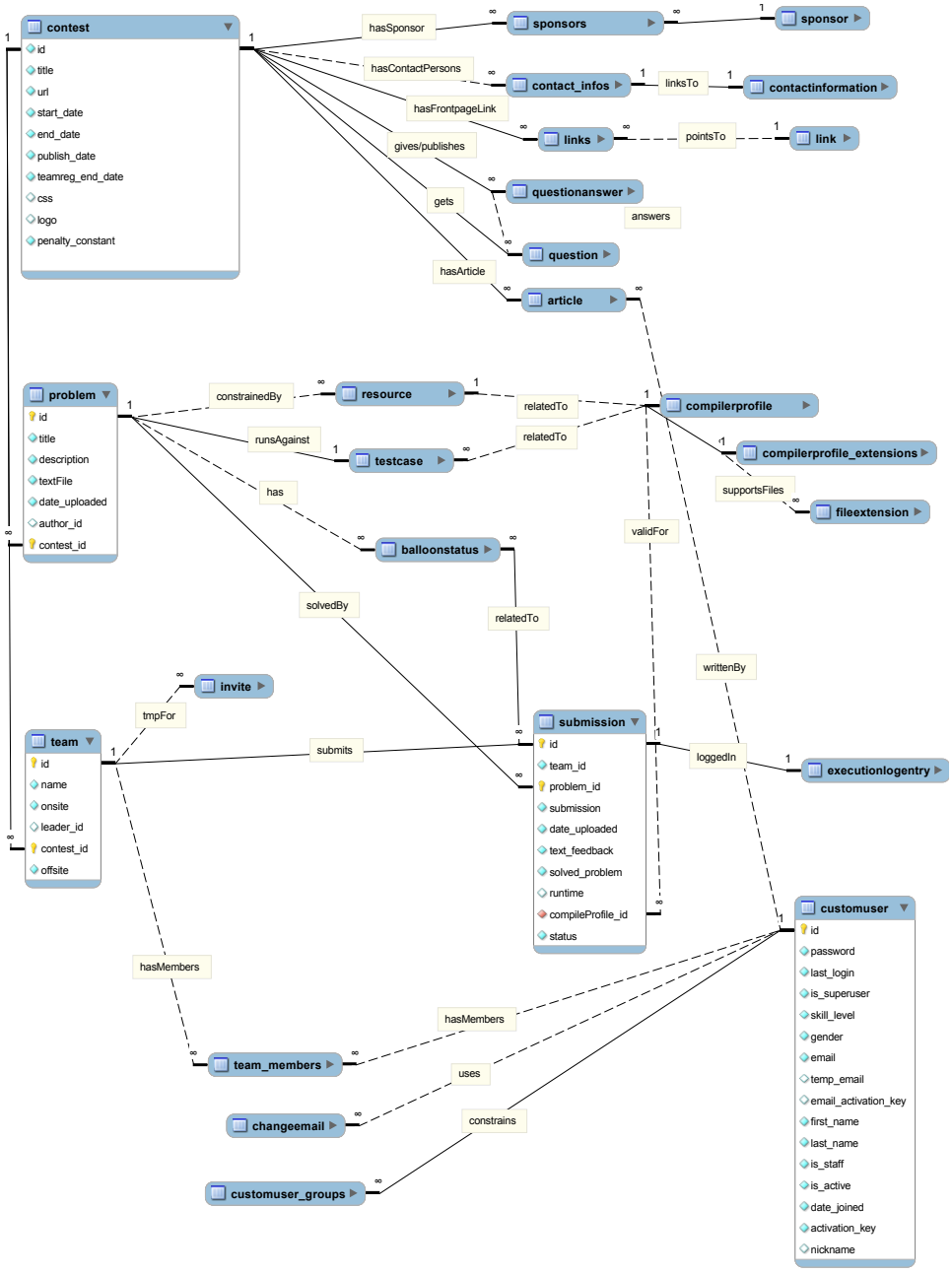
ID	IT-13
Interface name	Create team
Pre-condtion	Contest is created, user is created
Post-condition	Team is created
Input	Name, onsite,
Output	A team in the database, that can be used in a contest

ID	IT-14
Interface name	Edit team data
Pre-condtion	Team is created
Post-condition	Team-data is modified, and modified attributesare reflected in other views
Input	Team, data, attributes
Output	A modified team entry in the database

Appendix E

ER-Diagram

Our ER-diagrams follows a convention ER-convention. Each relation has a name, and is intended to be read either from left to right or top towards bottom.


contest
id
title
url
start_date
end_date
publish_date
teamreg_end_date
css
logo
penalty_constant
hasSponsor
sponsors
sponsor
hasContactPersons
contact_infos
linksTo
contactinformation
hasFrontpageLink
links
pointsTo
link
gives/publishes
gets
questionanswer
answers
question
hasArticle
article
problem
id
title
description
textFile
date_uploaded
author_id
contest_id
constrainedBy
resource
relatedTo
compilerprofile
runsAgainst
testcase
relatedTo
compilerprofile_extensions
supportsFiles
fileextension
has
balloonstatus
validFor
solvedBy
relatedTo
writtenBy
invite
tmpFor
team
id
name
onsite
leader_id
contest_id
offsite
submits
submission
id
team_id
problem_id
submission
date_uploaded
text_feedback
solved_problem
runtime
compileProfile_id
status
loggedIn
executionlogentry
customuser
id
password
last_login
is_superuser
skill_level
gender
email
temp_email
email_activation_key
first_name
last_name
is_staff
is_active
date_joined
activation_key
nickname
hasMembers
hasMembers
team_members
uses
changeemail
constrains
customuser_groups

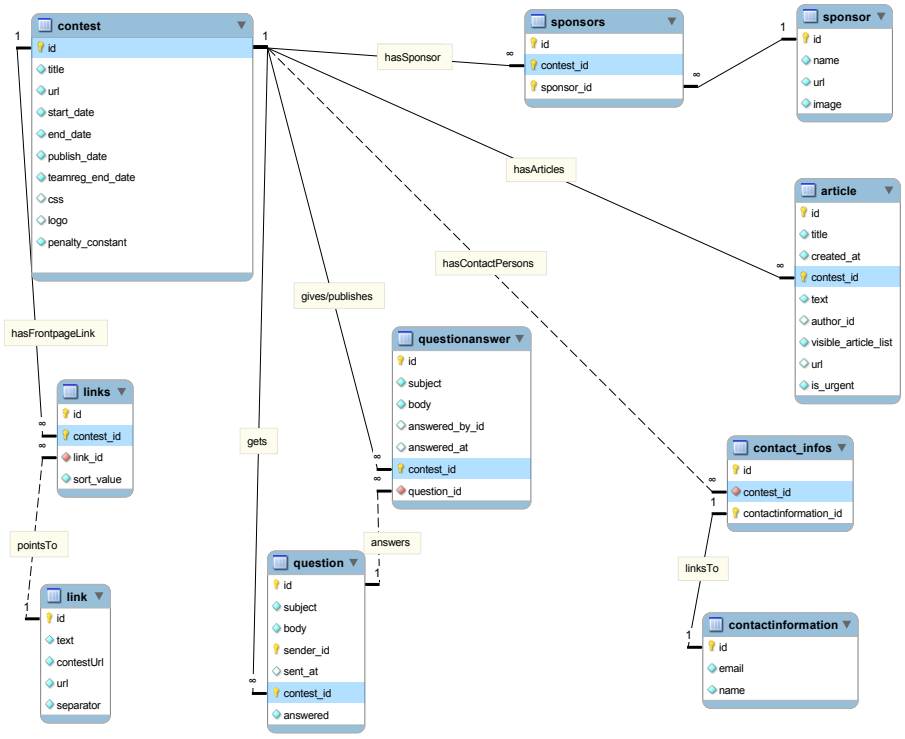


Figure E.2: ER-diagram for the models used for the contest.

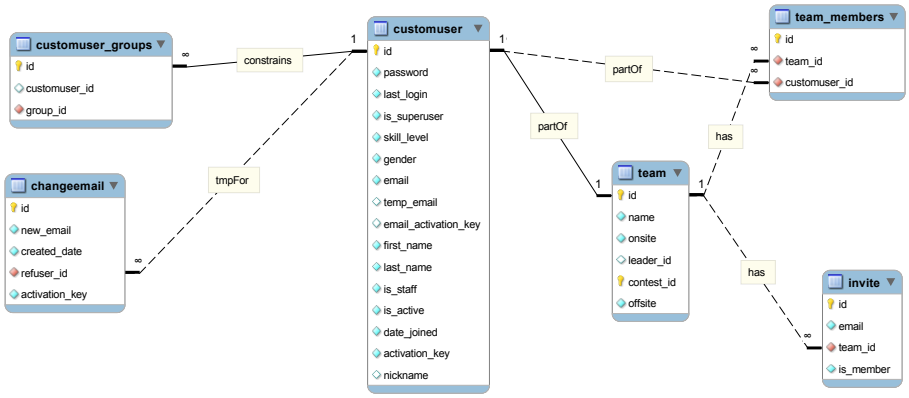


Figure E.3: ER-diagram for the models used for the user registration.

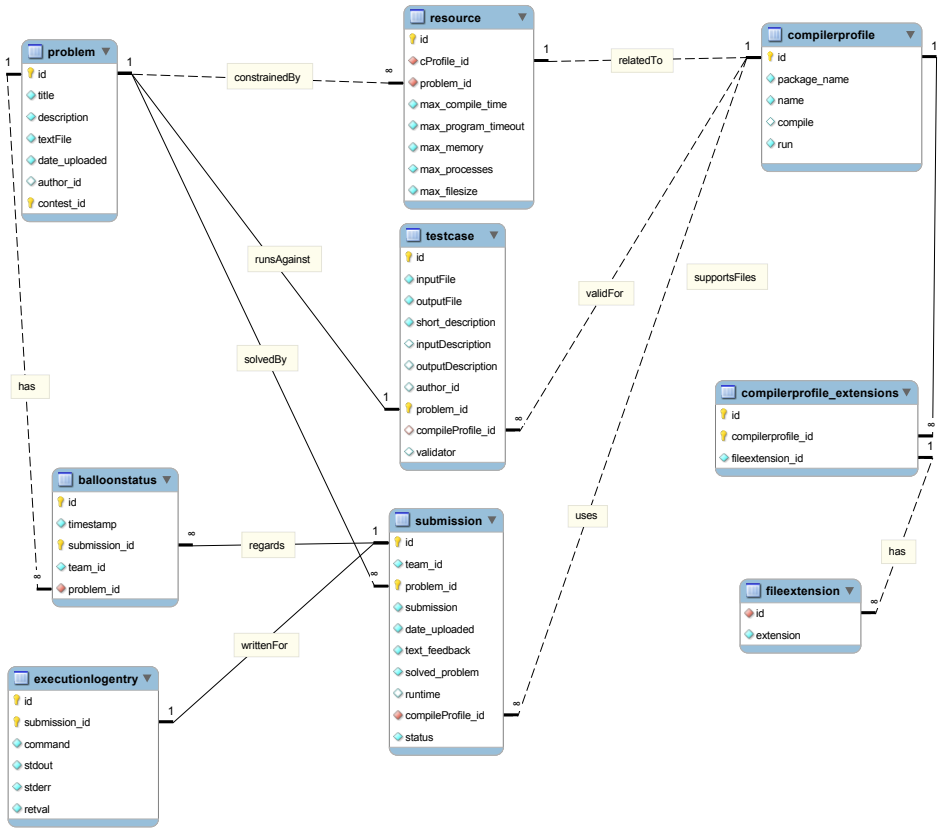


Figure E.4: ER-diagram for the models used for the submission.

Appendix F

Website views

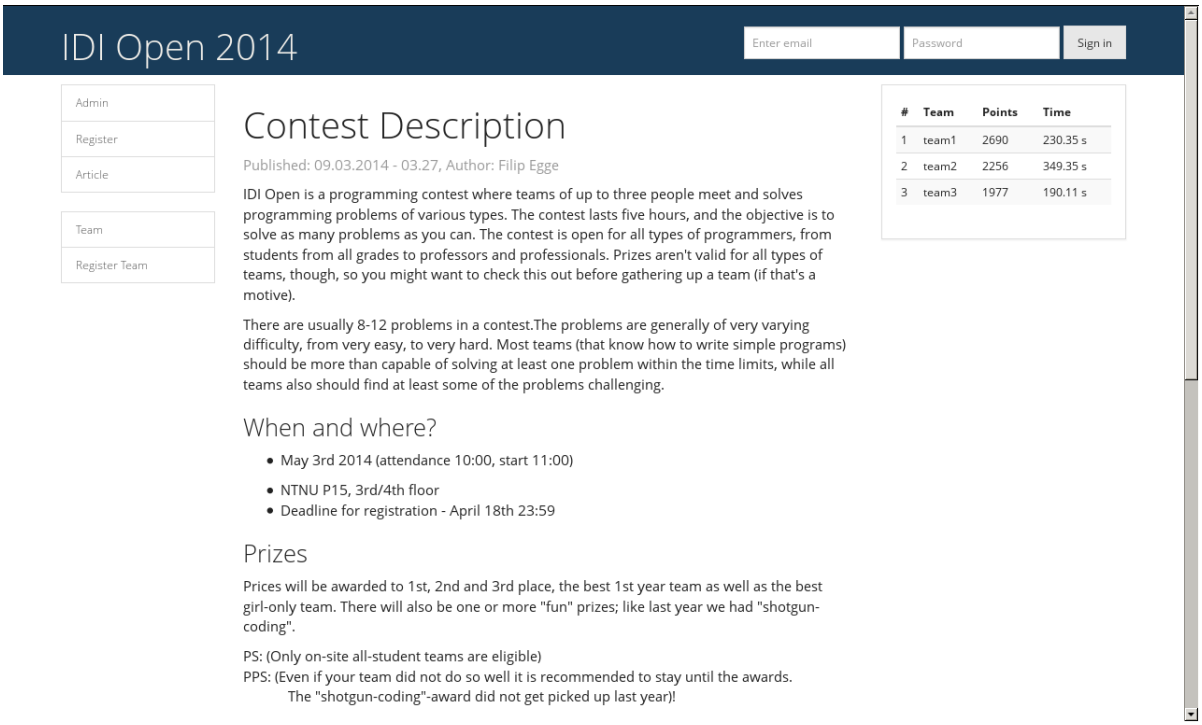


Figure F.1: Our initial website design

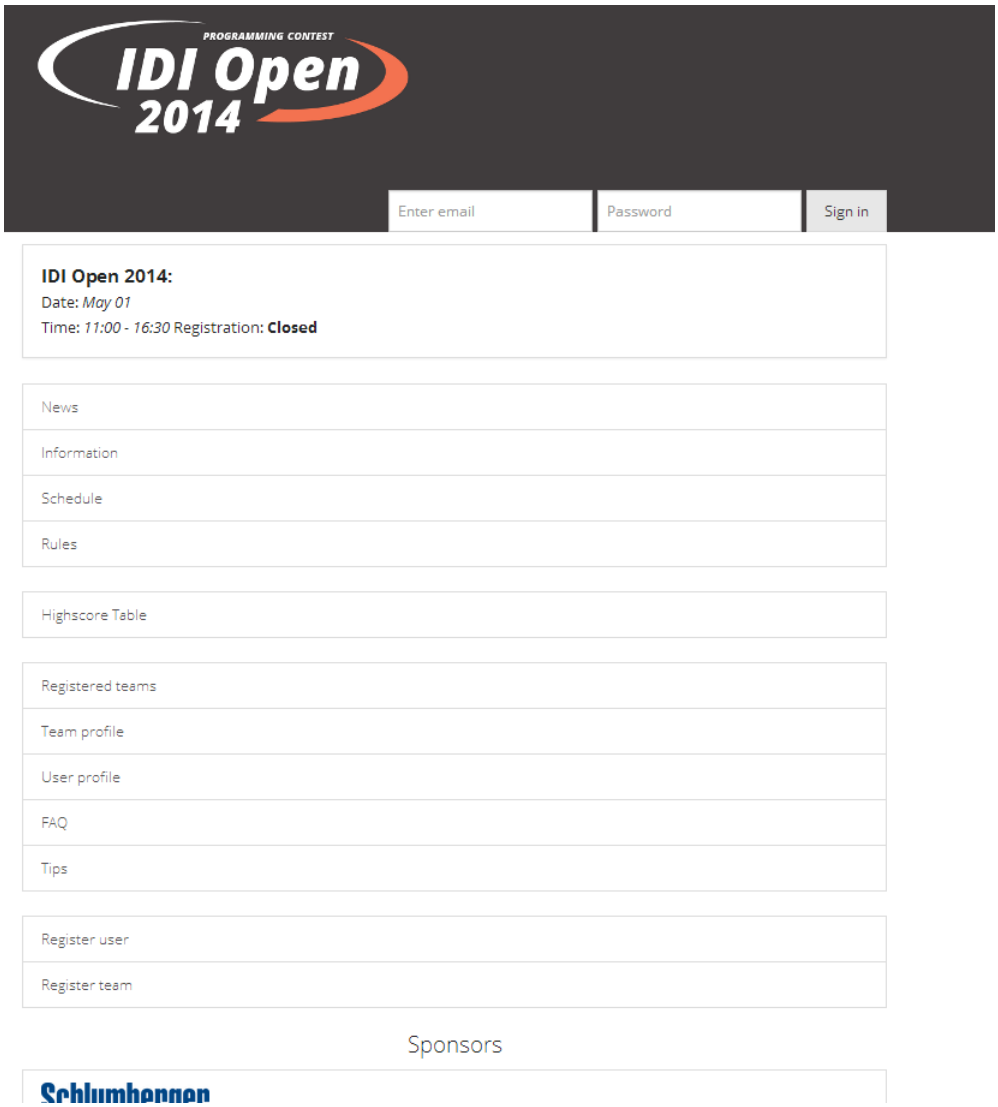


Figure F.2: The website seen from a mobile device

Appendix G

Sprint Burndowns

G.1 Inital Activity Lists

Sprint 1

			Planning				Follow-up		
Nr	**Work Package**	**Activity**	**Resource**	**Planned work (hrs)**	**Start**	**Finish**	**Actual work (hrs)**	**Status (% complete)**	**Comment**
1	Project management	Make WBS	Konrad, Håkon	2	30.01.2014	30.01.2014	7	100	
2	Project management	Create a structure for weekly meetings	Tino	2	28.01.2014	28.01.2014	2	0	
3	Project management	Fill out the status report	Tino	2	30.01.2014	02.02.2014	1	100	Status report and activity plan is two separate tasks, so it didn't take that long. And also I can't fill out the whole status report before we start to plan our next sprint
4	Project management	Create a preliminary risk assessment	Anders					0	
5	Project management	Create a functional requirement specification	Tino, Håkon, Konrad, Filip	12	29.01.2014	30.01.2014	15	0	It took longer than expected, because we write it in norwegian, and then had to translate it to english. We also needed to update it after meeting with customer. Maybe we should have broken it down into smaller tasks.
6		Setup github repo	Anders	2	29.01.2014	30.01.2014	4	100	Learning about git took longer than expected. E.g. git submudules etc
7		Setup jenkins	Anders	10	29.01.2014	30.01.2014	3	30	Some unexpected items, such as openshift using django 1.4 instead of 1.6
8		Create non-functional requirement specification	Konrad, Eirik	6	29.01.2014	30.01.2014	4	100	
9		Seminar/lecture	Eirik, Tino					100	Two seminars this week.
10		Add sprint tasks	Anders	1	29.01.2014	29.01.2014	1	100	Noted in icescrum
11		Meeting with supervisor	ALL	1	30.01.2014	30.01.2014	1	100	Transcription/summary in drive
12		Sprint review	ALL	1	30.01.2014	30.01.2014		100	
13		Flow diagram						0	
14		Prepare customer meeting	Konrad	1				100	Took less than 1 hour
15		Class diagram	Filip	30	29.01.2014			50	
16		Learn python/django	ALL		29.01.2014		18		Konrad: Used more than 5 hours. Could need more next sprint.
17		IDE/eclipse jboss							
18		Activity plan	Tino	1	30.01.2014	30.01.2014	1	100	
19		Seminar/lecture	Tino, Eirik	6		30.01.2014	6	100	Seminar about the usage of Scrum in a real-life environment

Figure G.1: Our initial website design

Sprint 2

			Planning				Follow-up		
Nr	**Work Package**	**Activity**	**Resource**	**Planned work (hrs)**	**Start**	**Finish**	**Actual work (hrs)**	**Status (% complete)**	**Comment**
1	Install and learn tools	Setup jenkins plugin	Anders	2	07.02.2014	17.02.2014	3	50	Not finished yet.
2		Setup IDI Machine		1				0	
3		Learn jenkins	ALL	6					Learn by writing a test
4		Create and setup git branch	ALL	6					To do on Monday
	Design								
5	Requirement specification	Update functional requirements. Update use-story. Write down what kind of syntax and what kind of naming we should use throughout.	Eirik, Filip	6	06.02.2014	06.02.2014	4	100	This includes joining a team and creating a contestant. We chose to skip class diagram as we saw no use for these.
6	System architecture	Create flow-chart for joining a contest. Create class diagram for joining a contest.	Anders, Tino	4	06.02.2014	06.02.2014	1	100	Very simple diagrams, should probably be updated after a first attempt at its implementation
7		Create a flow-chart for creating a contest. Create a class diagram for joining a contest.	Tino, Håkon	4	05.02.2014	05.02.2014	2	100	
8		Create a flow-chart for posting news. Create a class diagram for posting news.	Anders, Konrad	4	05.02.2014	05.02.2014	5	100	Create a ER-diagram showing how admins, teams, contestants and contests work together.
9	Database modelling	A user should be able to join a competition create by an admin. ER	Tino, Håkon	6	05.02.2014	07.02.2014	4	100	
10	Development								
11	Unit testing	Create the unit part of the testplan, create contest	Filip, Eirik	2	10.02.2014			0	
12		Create the unit part of the testplan, join contest	Konrad, Anders	2	10.02.2014		1	40	
13		Create the unit part of the testplan, post news	Tino, Håkon	2	10.02.2014		0	5	
14	1. Project requirements								
15	1.6 WBS	Update WBS to meet standard: Update gantt diagram. Add more work packages.	Anders, Tino	2	06.02.2014	06.02.2014	2	100	Change the priority(e.g. execution node development should be moved backwards). We should have more packages to mention.
16	1.4 Report	Status report	Tino	1	06.02.2014	06.02.2014	1	100	Uploaded it to itslearning.
17	1.4.1 Preliminary report	Write down the stucture of the report	Eirik, Filip	2	05.02.2014	05.02.2014	2	100	Unable to add page numbers to table of contents in Google Docs
18		Write down why we chose the tools we are using/going to use	Anders	2	08.02.2014	09.02.2014	3	100	Should of course include why we chose these. Maybe also what other options we had.
19	1.4.1.1	System description							

Figure G.2: Our initial website design

20	1.4.1.1.1	Merge all diagrams into report						100	
21	1.4.1.1.2	Description of the system	Håkon, Filip		06.02.2014	08.02.2014	2	100	Write about the system as it is (old system), and how we plan on it being(new system).
22	1.4.1.2	Write about our team	Håkon	4	07.02.2014	09.02.2014	4	100	How we are organized. Why we chose scrum, TDD.
23	1.4.1.2.1	Introduction of our team	ALL						Should take about 5 min
24		Words and definitions	ALL	2	06.02.2014	09.02.2014	1	100	Some points to write about: Se example report
25		Customer	Tino	1	08.02.2014	08.02.2014	1	100	Write about motivation and scope.
26		Insert risk assessment into report	Tino, Håkon	1	09.02.2014	09.02.2014	1	100	
27	Risk assessment	Update/complete the missing risks	Tino, Håkon	6	09.02.2014	09.02.2014	6	100	This includes prioritizing, also we must define terms. Remember to fill in the definition of what is HIGH/LOW. Some of this will be present in the report.
28	Component models	Create initial CBSE(page 450 sommerville)	Håkon, Tino						

Figure G.3: Our initial website design

G.2 Activity Lists

Sprint 3

			Planning				Follow-up		
Nr	**Work Package**	**Activity**	**Resource**	**Planned work (hrs)**	**Start**	**Finish**	**Actual work (hrs)**	**Status (% complete)**	**Comment**
1	Design	Create basic mockups	Tino, Håkon	2	12.02.2014	12.02.2014	2	100	We are going to need more mockups
2		Rework the requirement specification	Konrad	2			3	70	
3	Development	An admin should be able to publish news	ALL		12.02.2014				
4		A user should be able to register for a contest	ALL		12.02.2014				
5		An admin should be able to create a contest	ALL		12.02.2014				

Figure G.4: Our initial website design

Sprint 4

			Planning				Follow-up		
Nr	**Work Package**	**Activity**	**Resource**	**Planned work (hrs)**	**Start**	**Finish**	**Actual work (hrs)**	**Status (% complete)**	**Comment**
A-02		As an admin I want to be able to create a contest							
		Make sure contest is published at the right time							Dropped
		Start date cannot be set after finish date.							Dropped
CO-01	Development	As a contestant I want to be able to register a contestant in IDIOpen							
		Create the register page		4					
CO-02	Development	As a contestant I want to be able to register and administer teams							
		Make register page for teams							
U-02		As a user I want to have intuitive interface design							
		Make more mockups for all parts of the system	Håkon, Filip	4	20.02.2014	21.02.2014	4	100	
		Set up base template	Håkon	4	21.02.2014	25.02.2014	8	100	
1.	Project management	report							
		Update requirements	Konrad, Tino, Anders	6	21.02.2014	25.02.2014	14	70	
		Update WBS							
		Update risk	Anders, Tino, Håkon	4			2	100	
		Review system architecture	Konrad, Filip	2	21.02.2014	25.02.2014	4	70	
		Minor report stuff	Eirik	4			2	100	
		Write about SCRUM	Filip	2			2	100	
A-03		As an admin I want to publish news							
		Fixed newsfeed	Eirik	4	21.02.2014	25.02.2014	4	100	
		Add author as foreign key							
		Fix date and author(publish at, created on)	Konrad, Tino	4	21.02.2014	24.02.2014	8	100	
CU-02		As a customer I want different usergroups							
		Add usergroups(judges, contestants, functionary)							

Figure G.5: Our initial website design

Sprint 5	Availability this week (hours)		84	12	12	12	12	12	12	12								
				Hours Remaining							Hours Spent							
ID	Story	Priority	Est	Ons	Tor	Fre	Lør	Søn	Man	Tir	Ons	Tor	Fre	Lør	Søn	Man	Tir	Sum
A-02																		
		2	4	4	4	4	4	4	4	0	0	0	0	0	0	4	0	**4**
	Start date cannot be set after finish date	1	2	2	2	2	2	2	2	0	0	0	0	0	0	2	0	**2**
CO-01	**As a contestant I want to be able to register a contestant in IDIopen**																	
	Create the register pa This includes fron	3	4	4	2	2	2	2	0	0	2	5	0	1	2	0	0	**10**
CO-02	**As a contestant I want to be able to register and administer team**																	**0**
	[logged in] Make register page for teams	3	16	16	16	12	10	10	10	0	0	1	4	0	8	10	0	**23**
	[logged in] Make administrative page. For team.	2	8	8	8	8	4	0	0	0	0	0	4	4	0	0	0	**8**
	Create login for contestant. depend on C0-01	3	2	2	2	0	0	0	0	0	0	0	2	0	0	0	0	**2**
	Make invite (to team) show in contestant login page.	2	8	8	8	8	8	8	6	3	0	0	0	0	0	2	3	**5**
U-02	**As a user I want to have intuitive interface design**																	
	Integrate base HTML with the rest	2	2	4	0	0	0	0	0		4	0	0	0	0	0	0	**4**
SU-001	**Report**																	
	Tino finds out what to do	3	4	4	4	2	0	0	0	0	0	0	2	1	0	0	0	**3**
	Write preface and abstract	2	3	3	3	3	3	3	3	1	0	0	0	0	0	2	0	**2**
	Development process (write about how we organize our sp	2	2	2	2	2	2	2	1	0	0	0	0	0	0	2	0	**2**
	Development/implementation, explain what we have done	2	4	4	0	0	0	0	0	0	4	0	0	0	0	0	0	**4**
	Design, mockups (Explain our design/link it to mockups)	2	8	8	8	4	0	0	0	0	0	4	6	0	0	0	0	**10**
	Make sequence-diagrams for Create contest/Create Team	2	8	8	8	8	8	0	0	0	0	0	0	12	0	0	0	**12**
SU-01	**Testplan**																	
	Unit test	2	4	4	4	4	0	0	0	0	0	0	4	0	0	0	0	**4**
	Integration test	2	4	4	4	4	4	0	0	0	0	0	4	4	0	0	0	**8**
	System (acceptance) test	2	4	4	4	4	4	0	0	0	0	0	2	6	0	0	0	**8**
A-03	**As a admin I want published news**																	**0**
	Link user to an article. Author as foreign key	1	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	**0**
CU-02	**As a customer I want different usergroups**																	**0**
	Add usergroups (judges, contestants, functionary) in config.	1	2	2	2	2	2	2	2	0	0	0	0	0	0	0	0	**0**
S-01	**Sponsor stuff**																	**0**
		2	4	4	0	0	0	0	0	0	4	0	0	0	0	0	0	**4**
			95	97	83	71	55	35	30	4	14	10	28	28	10	22	3	115

Figure G.6: Initial activity list, sprint 1

Sprint 6	Availability this week (hours)		84	12	12	12	12	12	12	12								
				Hours Remaining							Hours Spent							
ID	Story	Priority	Est	Ons	Tor	Fre	Lør	Søn	Man	Tir	Ons	Tor	Fre	Lør	Søn	Man	Tir	Sum
CO-02	**Register and administer team**																	
	[logged in] Make administrative page. For team. This is a part of the the contestants dashboard	3	16	16	16	16	16	16	16	16	0	4	4	0	8	2	0	**18**
	Make invite (to team) show in contestant login page. Make invite accessible	3	8	8	2	0	0	0	0	0	0	4	4	0	0	0	0	**8**
	[logged in] Make register page for teams	3	8	7	1	2	0	0	0	0	1	4	4	0	0	0	0	**9**
SU-01	**Report**																	
	Write preface	2	2	2	2	2	0	0	0	0	0	0	0	2	0	0	0	**2**
	Development process (write about how we organize our sprints, etc)	3	2	1	1	1	1	0	0	0	2	0	0	0	3	0	0	**5**
	Development/implementation, explain what we have done so far	3	4	4	4	4	4	0	0	0	0	0	0	0	2	1	0	**3**
	Design, mockups (Explain our design/link it to mockups)	2	4	4	4	4	4	0	0	0	0	0	1	0	2	0	0	**3**
	Make sequence-diagrams for Create contest/Create Team/Register contestant	2	8	8	4	4	4	0	0	0	0	6	0	0	8	0	0	**14**
	write about our report, what is contained in each chapter, etc	3	4	4	4	0	0	0	0	0	0	0	2	0	0	0	0	**2**
	Insert updated risk, Burndown charts and WBS in Report (use Google Fusion)	3	4	4	4	4	4	4	4	3	0	0	0	0	0	0	1	**1**
	Alternative solutions, write about technical alternatives																	
	Non-functional req, need to add security	3	4	2	2	2	0	0	0	0	2	0	0	3	0	0	0	**5**
	Requirements specification	3	4	4	4	4	2	0	0	0	0	0	2	2	0	0	0	**4**
	Update Class-diagrams	3	2	2	0	0	0	0	0	0	0	2	0	0	0	0	0	**2**
	Make Component-diagram	3	4	4	4	4	4	4	0	0	0	0	0	0	0	4	0	**4**
	Task description and Overview	3	4	4	4	4	4	4	0	0	0	0	0	0	4	0	0	**4**
	Update milestones	3	2	2	2	2	2	0	0	0	0	0	0	0	2	0	0	**2**
	Write about the problem faced so far	2	8	8	8	8	8	4	4	0	0	0	0	0	3	4	0	**7**
SU-01	**Testplan**																	
	Component Diagram in relation to the integration test	2	2															
	Unit test	3	8	8	8	8	8	8	8	8	0	0		0	4	0	0	**4**
	Integration test	3	4	4	4	4	4	4	2	0	0	0	0	0	0	4	0	**4**
	Write about the different tests	3	2	2	2	2	2	2	0	0	0	0	0	0	2	0	0	**2**
	Write about how we implement the tests	3	2	2	2	2	2	2	0	0	0	0	0	0	2	0	0	**2**
	System (acceptance) test	3	8	8	8	8	8	8	8	0	0	0	0	0	8	2	0	**10**
	Create template to testplan	3	4	0	0	0	0	0	0	0	0	0	2	0	0	0	0	**2**

Figure G.7: Initial activity list, sprint 2

S-01	**Sponsor nice nice**																	
	It should be easy for admins to add sponsors to a competition. Sponser as foreign key to contest. Add sponsor model	2	4	4	4	4	4	0	0	0	0	0	0	4	0	0	0	**4**
U-02	**as a User I want intuitive user interface**																	
	Create bootstrap webpage. This includes implementing	3	16	15	10	9	3	0	0	0	1	5	2	5	0	0	0	**13**
A-03	**As an admin I want published news**																	
	Link user to an article. Author as foreign key	1	4	4	4	4	4	4	4	4	0	0	0	0	0	0	0	**0**
			142	131	108	102	88	60	46	31	6	25	21	16	48	17	1	134

Figure G.8: Initial activity list, sprint 3

Sprint 7	Availability this week (hours)		168	24	24	24	24	24	24	24								
				Hours Remaining							Hours Spent							
ID	Story	Priority	Est	Ons	Tor	Fre	Lør	Søn	Man	Tir	Ons	Tor	Fre	Lør	Søn	Man	Tir	Sum
C0-01																		**0**
	Contestant admistrativ page. Change username, pw, nick (low prio), Show invites	3	32	29	17	17	5	0	0	0	0	12	12	7	0	0	0	**31**
	It should not be possible to join a team if you already are on a team	3	4	0	0	0	0	0	0	0	0	5	0	0	0	0	0	**5**
	Add gender (should be optional)	2	4	4	4	4	4	4	0	0	0	0	0	0	0	4	0	**4**
Test																		**0**
	Write unit tests	1	4	4	4	4	0	4	4	0	0	0	0	0	0	4	4	**8**
	Write system tests	1	2	3	2	1	0	0	0	0	0	1	1	1	0	0	0	**3**
CO-02																		**0**
	Register and administer team, leader can change name etc.	3	32	32	21	6	3	0	2	0	0	11	12	3	5	0	2	**33**
	Make it possible to be a leader for a team.	2	4	4	2	0	0	0	0	0	0	4	2	0	0	0	0	**6**
	Only one invite per email per team.	3	2	2	0	0	0	0	0	0	0	2	0	0	0	0	0	**2**
	You should not be able to add yourself as a member of the team. This should be done automaticly.	3	2	2	2	0	0	0	0	0	0	0	1	0	0	0	0	**1**
A-02																		**0**
	Teams view on admin site, only participants not in team can be added to a team	1	4	4	4	4	4	0	0	0	0	0	0	0	2	0	0	**2**
	Order links. Links should have a weight so an admin can sort them	1	4	4	4	4	4	0	0	0	0	0	0	0	3	2	0	**5**
U-01																		**0**
	Error messages	2	8	8	8	6	6	6	6	0	0	0	3	0	0	0	7	**10**
U-02																		**0**
	Login bootstrap	2	2	2	0	0	0	0	0	0	0	2	0	0	0	0	0	**2**
	Artile list seperation. Also, between each article	1	2	2	2	2	2	2	2	2	0	0	0	0	0	0	0	**0**
	Space between content and footer	1	2	2	2	2	2	0	0	0	0	0	0	0	1	0	0	**1**
	Add block nav.contest (information block over meny) DO NOT CHANGE MODELS	2	2	2	2	2	0	0	0	0	0	0	0	3	0	0	0	**3**
A-02																		**0**
	Make contact info in footer from admin	1	4	4	4	4	0	0	0	0	0	0	0	4	0	0	0	**4**
																		0
S-01																		**0**
	Add sponsors picture from admin	3	8	0	0	0	0	0	0	0	3	0	0	0	0	0	0	**3**
A-03																		**0**

Figure G.9: Initial activity list, sprint 4

	Make it possible to create news that are not articles. E.g. sponsors news.	1	8	8	8	8	0	0	0	0	0	0	0	3	0	0	0	**3**
T-01																		
	HTTPS	1	8	8	8	8	8	8	8	6	0	0	0	0	0	0	2	
	Pentest (csrf, input validation, xss, sql)	3	16	16	16	16	16	16	16	16	0	0	0	0	0	0	2	
	Logging	1	2	2	2	2	2	2	0	0	0	0	0	0	0	2	0	
Dep-01	Set up solution live																	
	nginX, gUnicorn	3	16	16	16	16	16	16	16	12	0	0	0	0	0	0	3	
			172	158	128	106	72	58	54	36	3	37	31	21	11	12	20	135

Figure G.10: Initial activity list, sprint 5

Sprint 8				
ID	Story	Priority	Est	Used
Testing				
	Test all part of system	3	64	64
Deploy				
	Set up solution live	3	32	34
	Ensure that we have backups of our database	3	8	8
Bug fixes		3	64	80
	Some of which are:			
	Explain properly the syntax for the url that comes when they create a contest.			
	Explain the separators, that there is a drag 'n' drop. - button create separato			
	Explain/fix the image with sponsors.			
	Inviting a team member when creating a team did not work; an email was not sent.			
	Help text for add link URL did not explain properly			
	Add customer support email that is not our email list, such that the developers (gentlecoding) is available.			
	Change email			
	Passord lengde, strengere krav			
	Activitation e-mai			
	forgot password			
			168	186

Figure G.11: Initial activity list, sprint 6

Sprint 9	Availability this week (hours)		84	12	12	12	12	12	12	12								
				Hours Remaining							Hours Spent							
ID	Story	Priority	Est	Ons	Tor	Fre	Lør	Søn	Man	Tir	Ons	Tor	Fre	Lør	Søn	Man	Tir	Sum
A-11	**add/remove execution node**																	**0**
	Execution node model	3	4	4	4	4	4	4	4	4	0	0	4	0	0	0	0	**4**
	Init script (Install compilers, create user)	3	4	2	2	2	2	2	2	2	2	2	0	0	2	0	0	**6**
	Permission testing	1	2	2	2	2	2	2	2	2	0	0	0	1	0	0	0	**1**
A-12	**configure exection nodes with compiler profiles**																	**0**
	Compiler profile model	3	4	4	0	0	0	0	0	0	0	2	0	0	0	0	0	**2**
J-01	**submit problem(s)**																	**0**
	Problem description model	3	8	8	8	8	8	8	0	0	0	0	0	0	5	0	0	**5**
J-02	**upload cases for problem(s) (backend)**																	**0**
	Upload case in admin panel	2	2	2	0	0	0	0	0	0	0	2	0	0	0	0	0	**2**
	Create case model	2	4	4	0	0	0	0	0	0	0	1	0	0	0	0	0	**1**
	Connect case to problems	2	4	4	4	4	4	0	0	0	0	0	0	0	4	0	0	**4**
J-03	**upload solutions**																	**0**
	Connect submissions (from judges) and problem	2	8	8	8	8	0	0	0	0	0	0	7	0	0	0	0	**7**
CU-02	**different usergroups**																	**0**
	Add judge as usergroup	1	2	2	2	2	2	2	1	0	0	0	0	0	0	1	0	**1**
CU-03	**user manual**																	**0**
	For deletion	1	2	2	2	2	2	2	2	2								**0**
	How to write articles	3	2	2	2	2	2	2	2	0	0	0	0	0	0	0	1	**1**
	How to add admin users	3	2	2	0	0	0	0	0	0	0	1	0	0	0	0	0	**1**
	To change password	3	2	2	0	0	0	0	0	0	0	1	0	0	0	0	0	**1**
T-01	**upload submission to problem**																	**0**
	Submission model	2	4	4	4	4	0	0	0	0	0	0	1	0	0	0	0	**1**
	Nice nice gui	2	8	8	7	7	7	7	4	2	0	2	5	0	0	4	2	**13**
TEST	**Testing**																	**0**
	Create contest	2	4	4	4	4	4	4	4	0	0	0	0	0	0	0	1	**1**
	Create user	3	8	8	8	8	8	8	4	0	0	0	0	0	0	2	1	**3**
	Test urls	2	4	4	4	4	4	4	4	4	0	0	0	0	4	2	0	**6**

Figure G.12: Initial activity list, sprint 7

	Test forms (registration team/ register user)	3	8	8	8	8	0	0	0	0	1	3	1	0	0	0	0	**5**
	BUGFIXES																	**0**
	Email		4	4	4	0	0	0	0	0	0	1	1	0	0	0	0	**2**
			90	88	73	69	49	45	29	16	3	15	19	1	15	9	5	67

Figure G.13: Initial activity list, sprint 8

	Sprint 10		0	
ID	Story	Priority	Est	Used
A-11	**add/remove an execution node**			
	Fabric init script	3	8	16
	AppArmor	3	16	24
	Admin interface to add nodes	2	4	4
	Removal of execution node (ssh key, uninstall script, etc)	1	8	0
	Upload problem cases (only upload once)	1	16	0
A-12	**configure exection nodes with compiler profiles**			
	Make an actual compiler profile (e.g. gcc)	3	2	1
	Add what you need to run the program in model	3	4	4
	Poll for execution finish	2	8	8
J-02	**upload cases for problem(s)**			
	Test of cases	1	4	2
	Description for case; different types (enum)	1	4	2
J-03	**upload solutions**			
	Different kind of solutions to problems	1	4	2
J-04	**verify contest problem sets and solutions**			
	Test all problems and solutions in one page	2	8	2
T-01	**upload submission to problem**			
	delegate submission to execution node	3	8	12
	media files should be private	3	8	4
	nice nice GUI	3	8	16
	Upload close at competition end	2	4	3
	Upload begin at competition start	3	4	3
A-06	**set penalty system**			
	Team score	3	8	12
	score for team submission	3	8	8
	Cleanup, integration and bugfixes			

Figure G.14: Initial activity list, sprint 9

	Deploy and nice nice	3	16	20
	Fixing bugs	2	32	36
	Integrating code	3	16	18
	Testing BUGHUNTAZ	3	16	20
Total			214	217

Figure G.15: Initial activity list, sprint 10

	Sprint 11			
ID	Story	Priority	Est	Used
CO-	**Upload submission(Contestant)**			
	Submission model	3	4	3
	Ajax response	2	4	7
	Hide uploaded files	3	2	3
	Compiler profile	3	4	5
	Test cases models	2	4	4
	Problems models	3	4	2
	Resources models	2	4	4
	Custom validator	2	4	8
	Run submissions	3	32	34
	Multiple execution nodes	1	4	1
F0-04	**Highscore**			
	Create highscore	3	8	16
	Create minihighscore(top 5 only)	2	4	5
	Ajax minihighscore	1	4	4
	Freeze highscore at right time	2	2	1
A-06	**penalty system**			
	Penalty constant on contest model	3	2	2
	Test penalty contestant	3	2	1
A-011	**add/remove an execution node**			
	Add nodes to cluster	1	2	0
	Share uploaded files with all nodes	1	2	0
A-012	**configure exection node(s) with compiler profiles**			
	Automaticly install compilers on nodes	1	4	0
A-013	**review system status**			
	Create appropriate feedback to admin/judge	1	8	9
J-003	**upload solutions**			
	upload solutions before contest	3	4	5

Figure G.16: Initial activity list, sprint 11

J-004	**verify contest problem sets and solutions**			
	Create judge supervise	3	16	20
CU-01	**clarification system**			
	Clarification system design	3	2	1
	Submit question	3	4	4
	Submit answer	3	4	4
	Clarification models	3	4	3
CU-02	**different usergroups**			
	Create usergroups	3	2	2
CU-03	**user manual**			
	Create docstrings	1	2	2
B-01	**view (correct) submissions**			
	Balloon System	3	8	16
	Ajax updates	2	2	4
U-01	**receive (appropriate) error messages when errors occurs**			
	Update error messages	1	4	2
U-02	**intuitive interface design**			
U-03	Tweak design	2	4	3
U-04	Logo	2	4	3
U-05	Update admin interface	2	8	8
U-	**good response time on webpages**			
	Load testing	3	16	20
U-	**short user transactions (avoid click click click)**			
	Ajax updates	2	4	5
	Testing	3	64	40
	Model earlier events	2	32	32
	Bugfixing	3	32	34
	Set up test event	3	16	16
	Test event	3	30	30
			366	363

Figure G.17: Initial activity list, sprint 12

Sprint 12			0	
ID	Story	Priority	Est	Used
FO-04	**Highscore**			
	Update highscore	2	4	5
A-010	**determine what statistics are stored/collected by the system**			
	generate csv files	2	8	5
	generate pdf files from latex	2	16	56
	generate list of emails	1	2	1
A-011	**add/remove an execution node**			
	Create shared submissions directory	2	4	8
	Enable remote database connections	2	2	2
	Connect remote node to broker.	2	2	2
A-012	**configure exection node(s) with compiler profiles**			
	Automaticly install compilers on nodes	1	2	0
U-004	**short user transactions (avoid click click click)**			
	Update small ajax bugs	3	2	2
	Highscore			
	Small highscore fixes	2	2	2
J-004	**verify contest problem sets and solutions**			
	Small updates to judge supervise	2	4	2
	Give judges contestant access.	3	8	5
	Bugfixes			
	Custom validator fixes	3	4	8
	Security fixes	3	2	2
	Update feedbacks	2	2	2
U-02	**intuitive interface design**			
	Small updates in admin interface	2	2	2
U-003	**good response time on webpages**			
	Cache highscore	1	2	2

Figure G.18: Initial activity list, sprint 13

	administrative		16	8
	Testing	3	32	32
	Set up 2014 event	3	2	1
	Configure and setup compiler profiles	3	4	3
	Create and enable links	2	2	1
	Bugfixing	3	32	20
	Be present at 2014 event	3	30	30
Total			186	201

Figure G.19: Initial activity list, sprint 14