1 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.

2 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 ve 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

3 Introduction

3.0.1 About the course

Our group and assignment has been delegated as part of the course IT2901: "Informatics Project II" at NTNU. The work covers 15 course credits, equivalent to a 50% work position for one academic semester. IT2901 is offered only to those that are enrolled on the NTNU's informatics BSc programme.

The primary purpose of the course is to let students apply their knowledge from other courses. This is rendered through a project for a real customer. The students have to communicate independently with their client, and deliver a software product that answers the client's needs.

Grades are based on the satisfaction of the customers and an evaluation of the development process. The latter will be reviewed through written reports and timesheets, as provided in this document. Furthermore, it is important that students have met the gideadlines and documented their work in a structured manner.

3.1 The Group

The team consists of six members. All the members of the grare completing their BSc degree in Computer Science from NTNU in 2014. We had prior experience working together, and knew each other well. With many shared courses and similar interests, the team are all at a somewhat similar level of competence. However, we have different areas of expertise, and exploiting this has been a key to success on previous occasions. For a detailed description of each member, see the listing below.

Anders Sildnes Throughout his BSc, Anders has been taking courses related to algorithms and program security. Apart from his studies, he is developing for Engineers without Borders NTNU and spending time with open-source projects and other Linux tools.

Eirik Fosse Eirik has a primary interest in artificial intelligence and machine learning. In the course of his bachelor's degree he's focused on programming, mathematics, and evolutionary simulation.

Filip Fjuk Egge While achieving his degree, Filip has taken courses focused on a path related to system development and security. He has a varied education and knowledge on different aspects of computer science.

Haakon Konrad William Aasebø Haakon has selected disciplines related to mathematics and algorithms. Apart from being a student at NTNU he is playing football at NTNUI in the third division.

Håkon Gimnes Kaurel During his time at NTNU, Håkon has been keeping a primary focus on courses related to programming and the intersection between hardware and software. He's also got experience as an app developer, and has extensive knowledge of the GNU/Linux operating system.

Tino Lazreg Tino has been taking courses related to different aspects of software engi, like programming, system architecture, human-machine interaction. Besides doing a BSc, Tino also works as a student assistant in a human-machine interaction course on NTNU.

3.2 About the Customer

Our customer is IDI Open. They are responsible for the annual programming contest mentioned in 1.2. Christian Chavez is our main contact for the project, but his two colleagues, Christian Neverdal Jonassen and Finn Inderhaug Holme, were also available for questions. They are all students of computer science at NTNU.

3.3 About the Contest

IDI Open is a programming contest where teams of up to three people meet and solve programming problems of various difficulty. The contest lasts five hours, and the objective is to solve as many problems as possible. The contest is open for all types of programmers, from students of all grades to professors and other professionals from the IT industry. Various prizes are gito the teams based on their performance. There are usually 8-12 problems in a contest. To make the competition fun for everyone, there are typically some problems that are easy enough even for novice programmers to handle. The main objective is to solve the highest amount of problems in the shortest amount of time.

3.4 Stakeholders

Our stakeholder fall into two different categories: the ones involved in the competition, and those involved in the course.

3.4.1 Course

Supervisor The supervisor's job consists of guiding and helping us through this project. This aid was primarily focused on the development process and the writing of this report. The supervisor tries to ensure that the developers communicate properly and have a structured approach to developing the end product. To verify this, we have had bi weekly status reports delivered to the supervisor, as well as regular meetings.

Examiner The examiner(s) is responsible for determining our final grade. Unlike the other stakeholders, we have not communicated with the examiner throughout the development process. Though, the examiner has got access to all the documents the supervisor has got access to.

3.4.2 Product

IDI Open The project's primary stakeholders. They are the host of the competition in which our product was used. Their inclusion in this product comprised all aspects of our project.

Judges The judges are hired by IDI Open to supervise the competition, service contestants and create problem sets. They will rely on our end product achieve the mentioned tasks. Throughout the process they have gifeedback to our customers, IDI Open, about our product. Naturally, the judges are important to the contest, so it is important that they are satisfied with the software they have to use.

Developers The developers are responsible for satisfying all other parties. Similar to the customer, our involvement in this project is total.

Maintainers As IDI Open is an annually recurring event, our end product, if successful, will be used for many years in the future. At a point, we assume the code will need to be replaced or modified. Assumably, there will be another developer team to do this. As such, the quality of of our product will impact them.

Sponsors Different companies sponsor each contest. In exchange for money and services, the sponsors get exposure through ads on the website and get to give a short presentation during the awards ceremony. Naturally, the sponsors want to associate their name with a successful product. Therefore, the sponsors rely on that contests are successful - this is heavily based on our product.

Contestants The actions of contestants are all through our software; our product will be their medium to take part in IDI Open. Reliability and usability is key to keep the contestants happy. The contestants also gave feedback to the customers about their user experience. Thus, how satisfied the contestants are impacts the developer's evaluation.

3.5 Goals

The goal of the project is to upgrade and improve the existing system used in IDI Open. We were given sole responsibility for our project; no other team or organization of developers has had responsibility for our solution.This gave us inspiration to do the best we could, and to give the customer something both we and they could be proud of for many years to come. And if the product is good enough it would hopefully also be used in larger programming competitions, maybe even international ones.

4 Project Management

This section will go through the different project roles we deemed important. We will explain our development method, which tools we use and give an overview of how we planned the project. Furthermore, in section X.X we also provide a structured overview of how we organized our time.

4.1 Project Roles

We wanted to ensure that all developers had an even workload and experience in all components of our project. To achieve this, we maintained a flat organizational structure where all decisions were made in groups, and no member would work alone on a task for a longer period of time. Some tasks and delegations, however, would be easier to assign only once to reduce time spent in transition between developers. The following paragraphs discusses the different roles we assigned.

The most central role is that of the scrum master. The role mainly consists of setting up meeting agendas and keeping control of what team-members are working on. In addition, the scrum master should act as a buffer between the team and other distractions. In our framework, the scrum master also had a casting vote whenever there was a disagreement. The group elected Haakon to be scrum master because of his well-established authority and organization.

We also assigned the role of a transcriptionist. His job consists of writing a short summary of every meeting, and making this available to the rest of the group. This includes meetings with the customer and supervisor. This job was performed by Anders, who volunteered for the position. We randomly assigned Haakon to be customer contact, and Tino as responsible for room reservations.

4.2 Development Method (Scrum)

Scrum focuses on having daily meetings, and constantly adjusting to changes by iterative development. This makes it easier to predict and to adjust for problems that may occur. It was hard to predict what would happen in our project, therefore our sprints were short, lasting at most two weeks. The transition between two sprints was done during a prolonged meeting on Wednesdays. During this meeting we evaluated the latest sprint and planned the upcoming one. Every team member were requested to say three good things and three bad things regarding the last sprint. This was followed by a discussion of how to plan the next sprint better. Lastly we showed what had been completed, to the other members of the group, before setting up the next sprint. Scrum also focuses on having finished versions of the systems on each iteration, and to finish all packages in the given iteration. In order to take advantage of the best in everyone's abilities we worked in pairs where this was efficient. Working in pairs is common in agile development. This was to improve code quality and reduce errors[1] hrefhttp://www.cs.pomona.edu/classes/cs121/supp/williams$_p$*rpgm.pdfhttp*

4.3 Tools/Framework

The customer wanted our end product to be easy to maintain for future developer. Therefore we have chosen tools that are well known and easy to learn. A lot of different tools were considered for this system. Some of the most important are:

- Django, a framework written in Python.

[1] [Page 2

- Editors like VIM and Eclipse were used.
- Documenting the development process was completed using google drive.
- Git was used as a version control system, with github as hosting service
- Communication were done through email lists, IRC and Facebook.
- User interface design was stylized with bootstrap and grappelli.

A full list of all tools and frameworks used can be viewed in appendix *Tools and Frameworks.*

4.4 Project-Level Planning

After our initial requirements elicitation we began to plan our development process. The purpose of the plan was to verify that we had enough time to complete the requirements, and to avoid unforeseen risks. This section will present the various components we introduced to structurize the project.

4.5 Work Breakdown Structure

WBS is a decomposition of the project into phases, deliverables and work packages. Each package was further broken down into different tasks. The benefits from doing these are as follows:

- Planning out the entire process prevents bottlenecks.
- Clearly defining the scope o a package prevents excess or insufficient time usage.
- It is easy for supervisors and other parties to evaluate and understand our process.

Table X.X shows the work breakdown structure created. These high-level packages were later broken down into activities, which are in the product backlog, see appendix

1. Project management
   (a) Write timesheet template
   (b) Look at the reflection notes
   (c) Meetings
      i. Internal
      ii. Customer
      iii. supervisor
   (d) Report
      i. Preliminary version
      ii. Mid-semester version
      iii. Final version

(e) Risk assessment
(f) WBS
(g) Status report
(h) Activity plans

2. Pre-study
   (a) Install and learn tools
   (b) Learn language/framework
   (c) Course

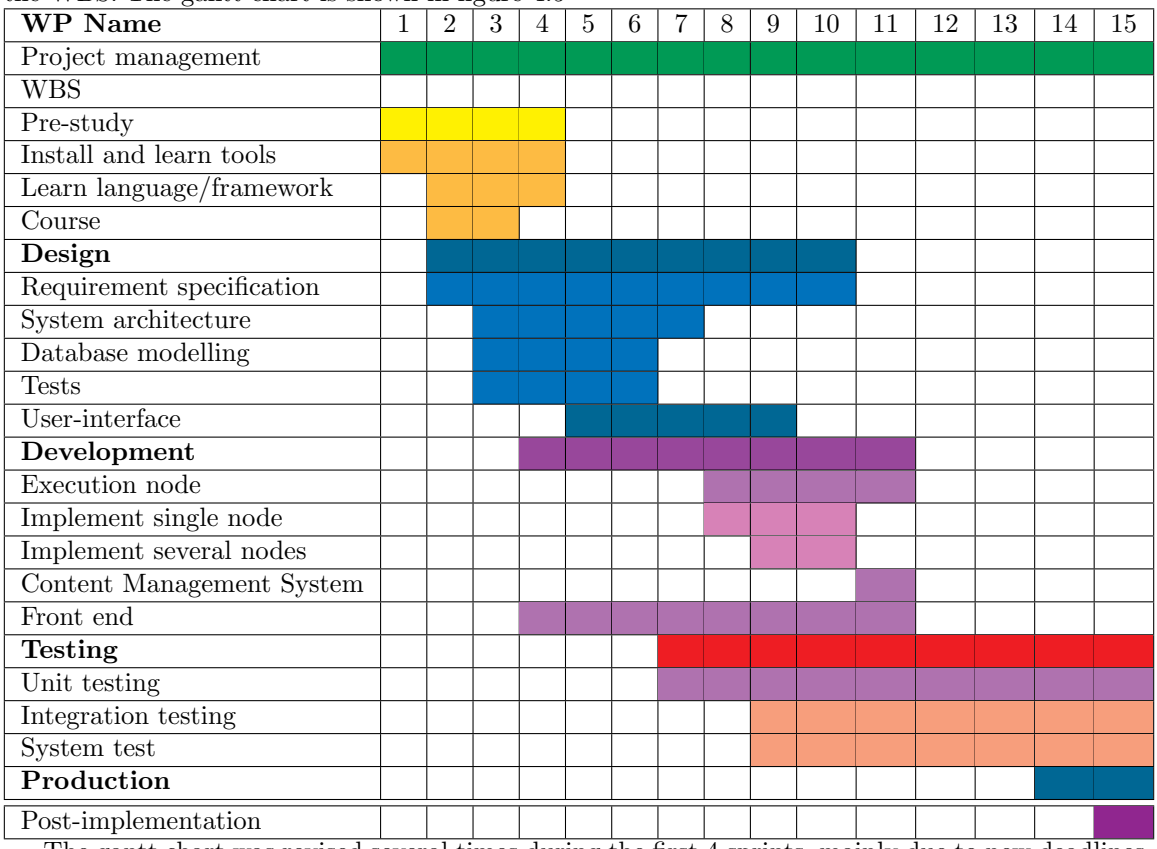
3. Design
   (a) Requirement Specification
      i. Functional
      ii. Non-functional
   (b) System architecture
   (c) Database modeling
   (d) User Interface
      i. Prototyping
      ii. Usability Testing
   (e) Admin interface

4. Development
   (a) Backend
      i. Execution-node(s)
         A. Web-page
         A. User
         B. Usergroups
         C. Team management
      ii. Statistics
      iii. Contest management
      iv. Clarification system
      v. Balloons system
      vi. Unit testing
   (b) Testing
      i. User-test
      ii. System-test
      iii. Final test

(c) Implementation
   i. Deploy to production
   ii. Installation
   iii. Turn in to stakeholder

(d) Implementation
   i. Verify
   ii. Document

We also created a gantt chart. Here, each package was assigned an estimated time period, over how long time we expected to use. For ease of comprehension, not every package was included from the WBS. The gantt chart is shown in figure 4.5



The gantt chart was revised several times during the first 4 sprints, mainly due to new deadlines set by the customer. The original chart is also given in appendix X.

4.6 Milestones

Throughout the project, the supervisor, customer, and the project group set deadlines. Some of the milestones marks the completion of work package(s). We have four of these milestones, M-03, M-05, M-06 and M-07. The other milestones represents events with deadlines that were given by the course stakeholders. These are M-01, M-02, M-04, M-08. The group are using the milestones in order to determine if the project is on schedule and to monitor the progress.

Preliminary report M-01 Preliminary report is the delivery of the first version of the report. This was intended to help us get started with important aspects of the project work.

Mid-semester report M-02 This version of the report should present all of the analysis and most of the design of our system. The delivery date for the mid-semester report is 16.03. We wanted to complete this earlier in order to focus on M-03.

First release M-03 This milestone marks the groups first delivery to the customer. The reader can view what functional requirements this release includes in the functional requirements. In summary this release should make it possible for contestants to sign up for a competition. Three days prior to the release the group will meet up with the customer and overlook that all the requirements are met. This meeting will also act as an introduction to the system, showing the customer how to manage the system.

Presentation M.04 The main purpose of the presentation is for the class to share their experience with other groups.

Beta-release M-05 The beta release should contain most of the major features, but it might not yet be complete. This version of the program should only be a release to a selected group of people. From M-06 to M-07 the system will be tested.

IDI Open test event M-06 On april the 26th we had a test event where everybody could test the system. This means that leading up to this event the system should be a release candidate.

IDI Open M-07 This is the day of the competition and the system should be in a release version.

Final report M-08 This milestone marks the final date for delivering the report as well as the final date of this bachelor thesis. Based on feedback received from the competition the group might choose to implement some changes to the system.

4.7 Meetings

Throughout this project the group have had several meetings. They can be categorized in three categories: internal, supervisor and customer meeting. We established some meetings rules:

- All meetings follow "the academic quarter", meaning that the time of start was XX.15.
- Members that were late had to bring a cake to the next meeting.
- All members may at any time propose a coffee break. This proposal has to be followed.
- No laptop should be open during the meetings.

4.7.1 Internal meetings

We had three internal meetings each week. Two of which were daily scrum meetings. These were primarily set to be on mondays and thursdays. During these meetings each group member would answer three questions:

- What have you done since the last meeting?
- What are you planning to do until next meeting
- Do you have any problems regarding reaching your goal?

The group would continue to work together after these meetings.

On Wednesday we had longer meetings, marking the end of one sprint and the beginning of the next. This meeting would consist of a sprint review meeting and a sprint retrospective, where we discussed:

- What was good/bad with the last sprint
- What should we try to improve during the next sprint.

After that we held a Sprint planning meeting and created a new sprint backlog. Our official meetings structure for this meeting can be viewed in the appendix.

4.7.2 Supervisor meeting

Meetings with the supervisor was generally held at a bi weekly basis. During these meetings we talked about what we had done, what we were going to do and received feedback on what we had done. Before each meeting we had to deliver status reports and activity diagrams. These activity diagrams were early on replaced by sprint backlog and burndown charts to facilitate the process.

4.7.3 Customer meeting

Customer meetings were held whenever we felt that a certain part of the requirements specification was unclear to us, and when we wanted approval of a newly completed feature. Throughout the semester there were a lot of meetings. As we never decided upon a fixed interval between customer meetings, the frequency varied a lot. The couple of days leading up to a release date often contained customer meetings in order to get everything right before starting on the next release. During our periods of focusing on writing this report, the frequency of these meetings naturally went down as the product did not progress, and as a consequence we had little to discuss with the customer.

4.8 Resources

This section contains the available resources for the project. We intended to use a minimum of 20/25 hours per person each week, but prepared for more work as we approached the deadline. This estimate was later scaled up to a minimum of 25/30 to weeks before easter, During easter, the amount of hours per week scaled up higher. Planned work Table X.X shows our first initial draft of sprints.

Sprint	Range (week)	Days	Hours
1	3 - 4	7	15
2	4 - 5	7	20
3	5 - 6	7	20
4	6 - 7	7	20
5	7 - 8	7	20
6	8 - 9	7	20
7	9 - 10	7	20
8	10 - 11	7	20
9	11 -12	7	20
10	12 -13	7	20
11	13 - 14	7	20
12	14 - 15	9	33
Easter	15 - 17	12	-
13	17 - 18	7	35
14	18 - 19 (Leading up to event)	9	35
After	19 - 22	21	35
Total:		91	353

4.8.1 Actual work

Table X.X shows the actual sprints and work done. The hours are for each person, during that sprint.

Sprint	Week	Days	Hours
1	3-4	7	15
2	4-5	7	15
3	5-6	7	20
4	6-7	7	20
5	7-8 (midterm report)	7	27
6	8-9	7	31
7	10-11	7	35
8	11-12	7	30
9	12-13	7	30
10	14-15	9	40
11	15-17 (starting 16.04, ending 26.04, easter)	10	90
12	18-19	6	35
After	19.22	21	45
Total		100	433

5 Design

This document 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*.

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 change email(requirement FC-02), supervisor(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.

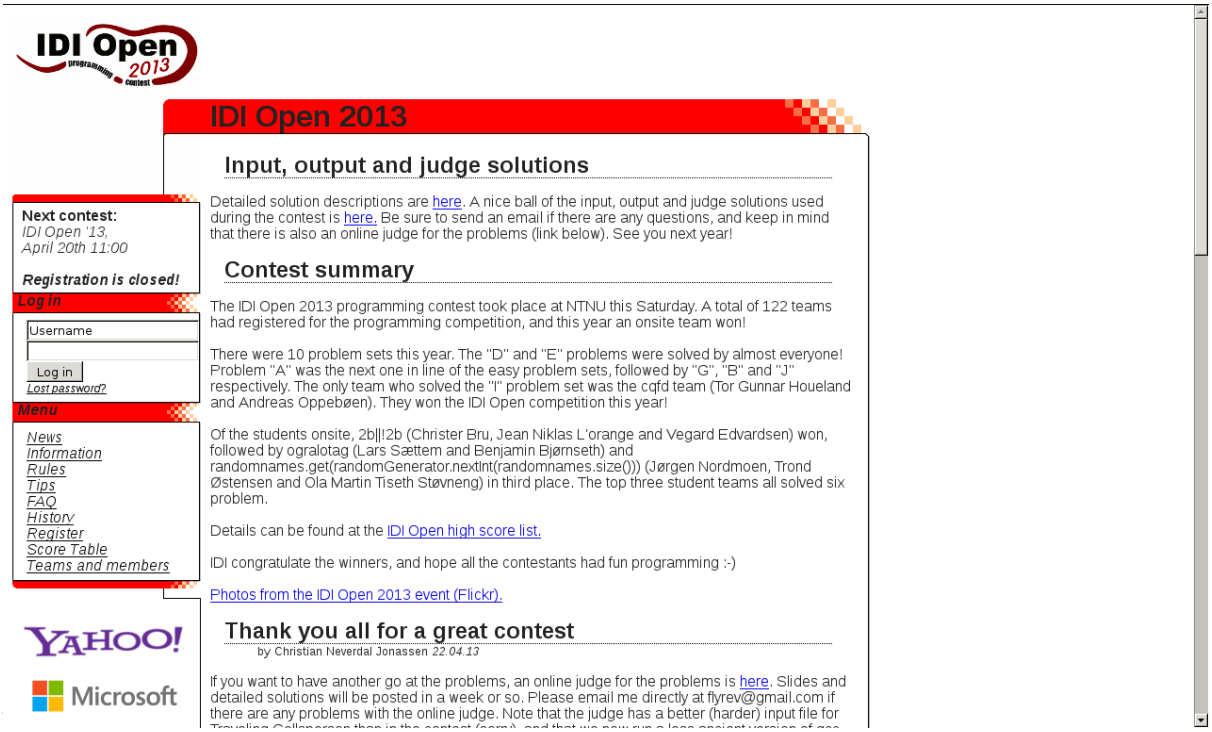


Figure 1: FIGURECAPTION

Fig 7.1 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.

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 are thoughts on design had been so far. We wanted to make sure

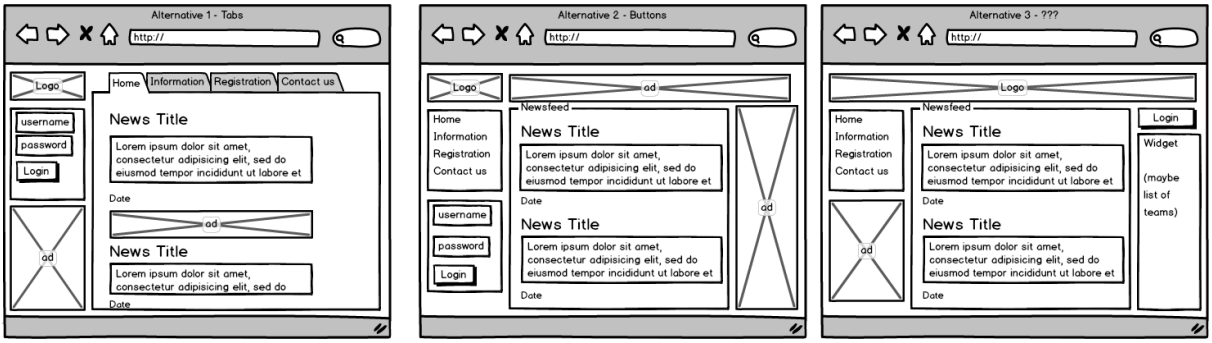


Figure 2: FIGURECAPTION

that the customers was on the same page as us, and that we were not moving beyond the scope of the project. Our customers wasn't 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 7.1. We agreed, because geused 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 7.2, 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 done, we introduced new design 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 with Grappelli as a skin to give it a modern look. This worked more or less automatically.

The final page looked like this:

The "black" frame was in our initial page coloured blue, but was changed one week before M-07, idiopen [REMARK: may be altered].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 on IDI OPEN 14, from blue to black, was done as a consequence of a logo change by Richard Eide, one of IDI Open's facilitators. The old color scheme can be viewed in appendix [insert which appendix]. By comparing fig 7.1 and fig 7.3, you can see that we kept the same structure, but still made some significant changes to the design.

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

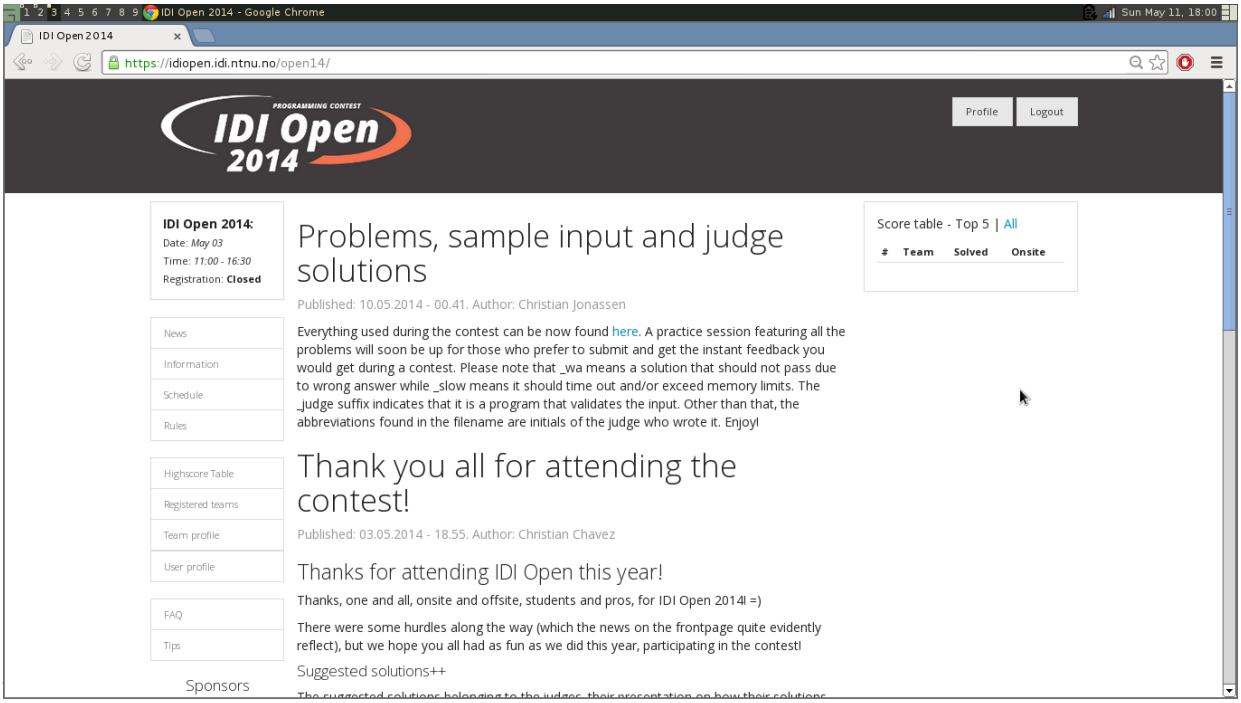


Figure 3: FIGURECAPTION

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[2]. We designed the menu to be versatile. 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. This was highly prioritized by our customers, they wanted to be able to make changes without having to change the code. As mentioned in Design process 7.1, 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, for example the sponsors and the menu buttons, 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 gelaw of proximity, that constitutes that humans will naturally group objects that are close to each other, and view them as a distinct. This helps the user quickly understand the user interface.

fig 7.4

"To strive for consistency" is the first of Shneiderman's eighgolden rules of interface design[3], and we tried to follow this while making design decisions. As can be seen in fig 7.4, 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

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

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

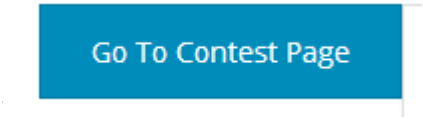


Figure 4: FIGURECAPTION

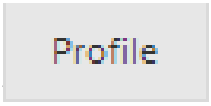


Figure 5: FIGURECAPTION

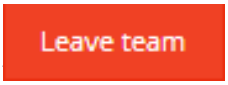


Figure 6: FIGURECAPTION

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. fig 7.5

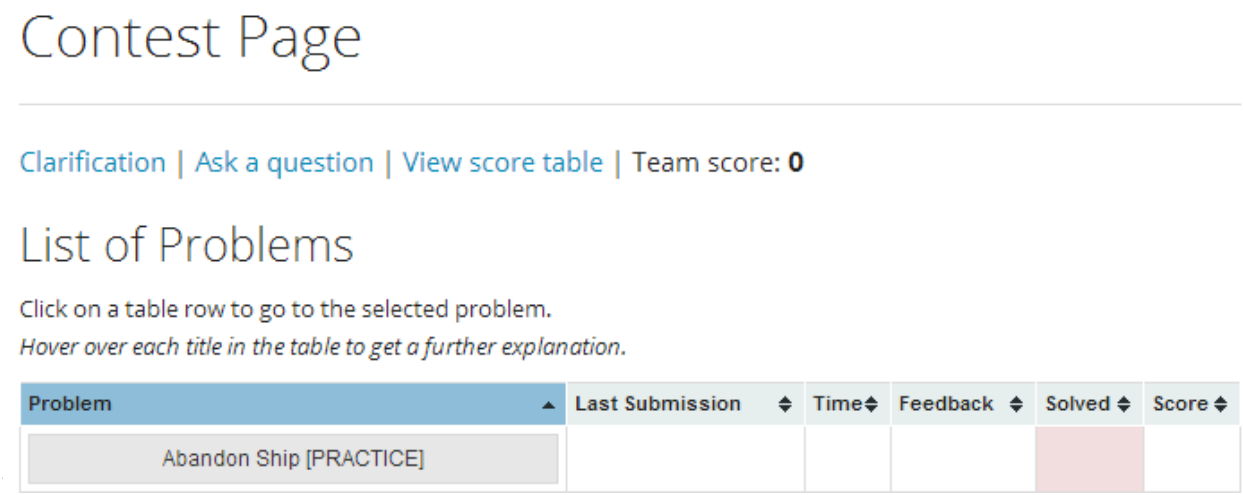


Figure 7: FIGURECAPTION

For the contest page, fig 7.5, we wanted to give the contestant a good overview of all the problems, their submissions to them, last feedback, if they solved the problem and the score. It is important to not bury information to deep in a website. It could be challengito balance this while trying not to overload the page with too much information. We had this in mind when designing this page. We gov̂aluable 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 eighr̂ule, because they will have everything accessible on the same page.

Admin interface

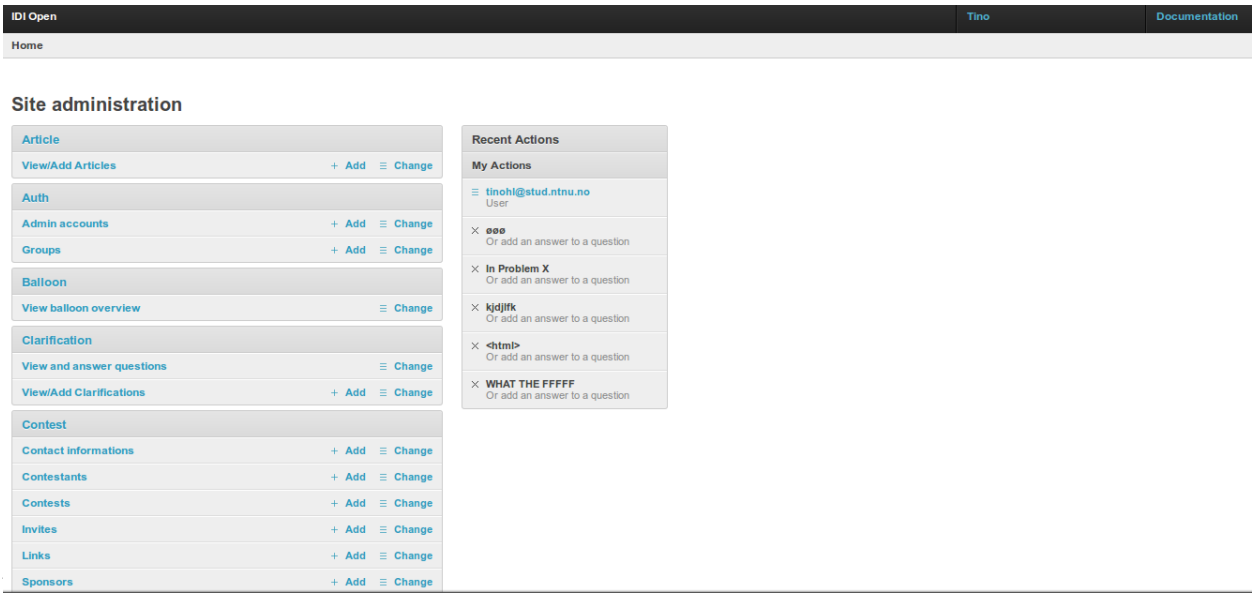


Figure 8: FIGURECAPTION

The admin interface is developed as an extension Django's admin interface. Django comes with an extensive admin interface, that provides functionality for adding, removing and changing parts of the system. The admin interface consists of everything we as developers want the admins to be able to change. For a complete listing, see figure X.X[kap 2]. 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 eighr̂ule.

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. Django's admin interface couldn't give us all the functionality we wanted, so we had to extend the interface with out own custom views. We created two views, "Balloon overview" and "Judge overview". To avoid having to create a similar interface as the rest of admin site, just with different functionality, we decided to extend the interface templates used for the django admin interface. This allowed us to change what we wanted, while it still kept its consistency with the other parts of the admin site.

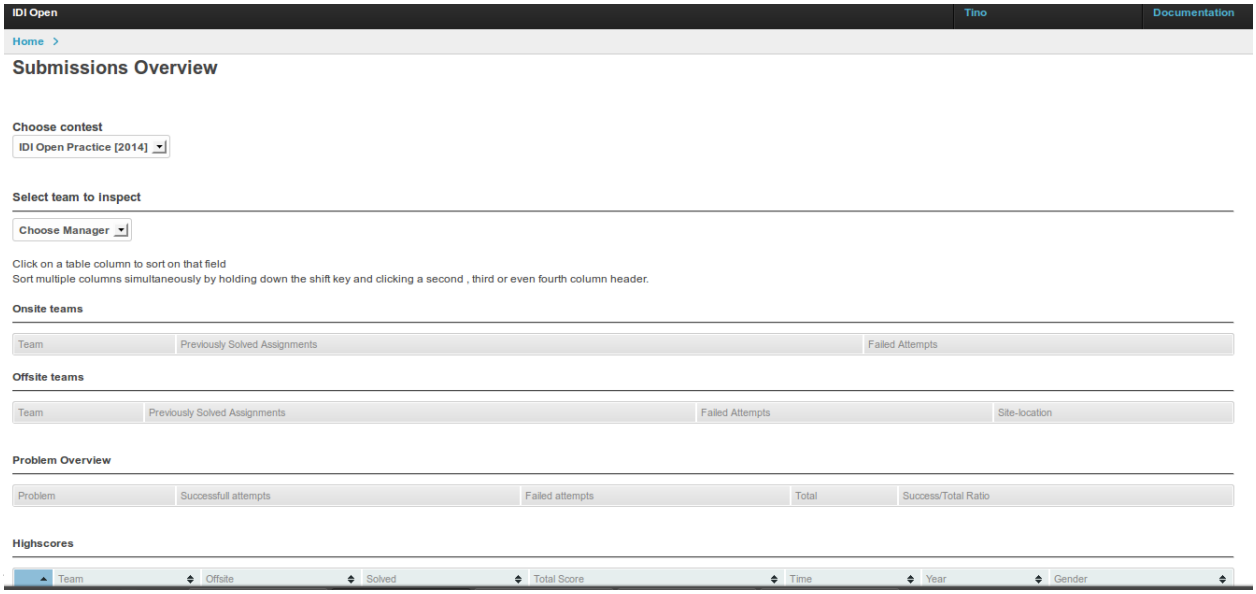


Figure 9: FIGURECAPTION

fig 7.7

The judge overview 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 an easier overlook over 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.

The view consists of four different tables, with the same layout as the balloon tables. The first two tables depicts how many failed attempts an onsite or offsite team has. The Problem Overview table provides statistics on each problem for the gicontest. This was added so that the judges can see which problem has the most failed or successful attempts, and if necessary make changes. To make it easy for the judges to choose a specific team, independent of submissions, we made a dropdown menu with all the teams. The last table is the highscore list. We wanted everything to be on one page for the judges, so they wouldn't have to constantly switch between different pages.

Figure [Judge_overview for Team]

Figure [Judge_overview for Team] shows the judge overview after selecting the team "Gentle-Coding". 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.

https://www.cs.umd.edu/users/ben/goldenrules.html

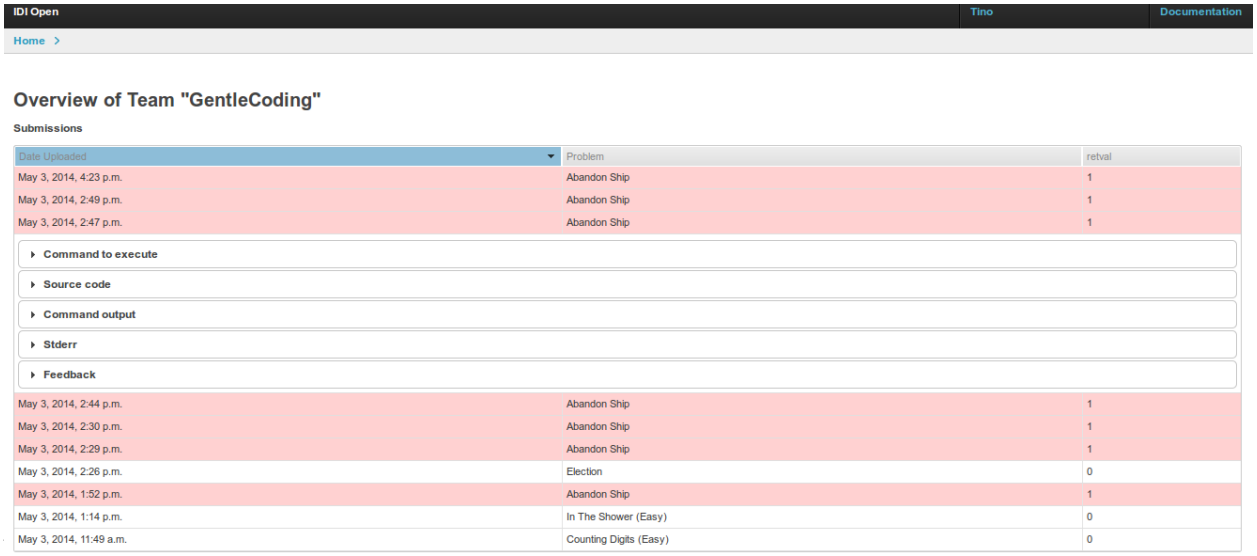


Figure 10: FIGURECAPTION

6 Testplan

To determine requirement, structural and architectural coverage of our product, we have performed software testing. 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.

6.1 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.

6.1.1 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. The tests are usually written either 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.

6.1.2 Integration Testing

In development, many features are bundled into different components. The components are then joined together to form a system. Integration testing tests the interfaces to each of these components, and how they communicate with each other. 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.

6.1.3 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 believed expectations from the customer.

6.1.4 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 girequirements.

6.2 Testing Coverage

We wanted to provide complete test coverage, but we did not have the time. Thus, we needed to prioritize what 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 on GentleIDI, 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. Some consequences of this is discussed in section X.X.

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. Therefore we have relied on load testing. Here, we gaour 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.

Our lacking experience in web development meant that it was hard for us to understand what components could go cause errors. Wikipedia holds a large list of categories that could be tested[4], but we avoided many of them, as it would take to long for us to gain a structural way to test these areas, combined with the lacking experience.

6.3 Our approach to testing

6.3.1 Unit Testing

Our unit tests are given in [source code].The reason for not including unit tests in the testplan is because it will be redundant, and take up unnecessary space in the report.

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. In other words, 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.

6.3.2 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. Comment is just an additional field in case we feel the need to explain a test more thoroughly to avoid misconceptions. 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

6.3.3 System Testing

Each separate test in the system test is linked to one or more of the requirements from the requirements specification. The template for system testing starts with specifying which function is being testing. 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. Every separate system test is connected 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.

6.3.4 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. The acceptance test was approved when the customer was satisfied with how we implemented the requirements.

6.3.5 Integration Test

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

[4] http://en.wikipedia.org/wiki/Non-functional_requirement

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 M-03 milestone, and do 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 X.X shows our view on the system as per milestone M-03. In figure X.X, we have replaced some default UML symbols and replaced them with the equivalent UML stereotype. The explanations are given in table 8.1. The tables in figure 8.3 are based upon the interfaces defined in figure 8.2.

UML notation	UML stereotype	Function
	provides	The component delivers the given functionality
	requires	For the component to work, the

diagram from milestone M-03. Each interface connection, especially “createContest” has been tested.

6.4 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 X.X.

ID	Function	Action/Input	Result	Requirement	Pass/F
TF-01	Create a contest, and publish an article to thatcontest. Edit article.Then, delete the contest.	Contest name, article text	Contest and article is no longer publicly available	FA-16,	PASS
TF-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	PASS	
TF-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
TF-04	Log in as admin, and enable all judges to createa contest. Then remove and add a judge, by escalating and deescalating privileges from contestant.	Admin account, contestant account	Zero changes to system.	FA-09,	PASS
TF-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
TF-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
TF-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
TF-08	Create a contestant account. Activate the account via email, and change the email. Ask for lost password on the	Contest-data, emails	Activation data received on the email, and all links word	FC-01, FC-02,	PASS

6.5 Non-functional testing

Our non-functional tests ensures non-functional requirements coverage and scenario correctness. Additionally, it defines acceptance criteria related to the performance of our solution.

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.For example, 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[TODO: REFER testing phases], we did inspection tests. Thus, table X.X below 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.

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	FAILED
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 < 100 ms	FAIL
Page responsiveness with at least 10 RPS	HTTP GET and POST to all pages	NF-01	Response-time < 200 ms	FAIL

In table X.X it can be seen that not all the tests passed. This is elaborated on in section X.X[TODO: refer requirements].

6.6 Risk and Dependencies

In section X.X we mentioned that we did not test whether or not the privileged users of the system made any errors. They were responsible for uploading solutions and content on the web site.

The majority of our testing has been inspection-based. This has been considered time efficient for us. As we have developed the entire system from scratch, and worked with it over a longer period of time, we have had good knowledge of the system. Thus, inspection-based testing has been largely effective. The problem is that there is no way to formally agree on what components have been tested, or to what extent. Additionally, future maintainers are much more likely to make errors as they do not know what components are connected, or what kind of tests should be executed.

Our lacking experience in web development means that our test coverage is not complete. Some errors, for example, were caused by improper charset encodings, an error none of us knew we had to consider. To mitigate these kind of risks, more experienced developers should participate in writing tests.

7 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. 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. 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.

7.1 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. These risks are informally discussed in chapter X. Our means of identifying risks was through discussions and agreements that we were not performing optimally.

7.2 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[5] 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.

7.3 Complete List of Risks

We have chosen to put the complete list in appendix X.

[5] 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

7.4 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 team-member; 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 team-member	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.

7.5 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.

7.6 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.
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.

7.7 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.

7.8 Tools and tools; 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.
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.

7.9 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.

ID	As a(n)	I want to be able to	So that
A-01	Admin	decide whether new contestpages are published or not	contests can be created when due
A-02	Admin	create a contest	contestants can register to teams
A-03	Admin	publish news	users can recieve information about a contest
A-04	Admin	custom css	to differentiate different contests
A-05	Admin	custom settings for each contest	
A-06	Admin	set penalty system	contestants are given points etc
A-07	Admin	modify usergroups through an interface	maintain control
A-08	Admin	add or remove users from a usergroup	control
A-09	Admin	add or remove users from the system	control
A-010	Admin	determine what statistics are stored/collected by the system	overview and increased user experience
A-011	Admin	add/remove an execution node	scalability and safety in redundance
A-012	Admin	configure exection nodes with compiler profiles	system flexibility and optimality
A-013	Admin	review system status	verify that contest can be hosted (correctly)
J-01	Judge	submit problem(s)	..add content to actual contest
J-02	Judge	upload cases for problem(s)	so that they can test problem submissions
J-003	Judge	upload solutions	assess case correctness to problem
J-004	Judge	verify contest problem sets and solutions	ensure that contest is O.K
CU-01	Customer	clarification system	provide communication between contestants and judges
CU-02	Customer	different usergroups	to have different roles
CU-03	Customer	user manual	ease of use
B-01	Balloon-functionary	view (correct) submissions	hand out balloons
CO-01	Contestant	register as a contestant in IDIOpen	compete in contest
CO-02	Contestant	register and administer team	compete in contest with teammates
T-01	Team	upload submission to problem	to compete
S-01	Sponsor	adspace	to advertize to users
U-01	User	receive (appropriate) error messages when errors occurs	build user-trust and nice nice
U-02	User	intuitive interface design	improved user experience
U-003	User	good response time on	improved user experi-