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.

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

Contents

Chapter 1

Introduction

1.1   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 given deadlines and documented their work in a structured manner.

1.2   The Group

The team consists of six members. All the members of the group are 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 also has 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 engineering, 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.

1.3   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.

1.4   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 given to 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.

1.5 Stakeholders

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

1.5.1 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 biweekly 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.

1.5.2 Product and Competition

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. Throughout the process they have given feedback 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 annual event, our end product, will be used for many years in the future. At a point, we assume the code will need to be extended or modified by another developer team. As such, the quality of our product will impact them.

Sponsors Each contest has companies sponsoring them. In exchange for money and services, the sponsors get exposure through ads on the website and are given the opportunity to hold a short presentation after the contest. Naturally, the sponsors want to associate their name with a successful product. Therefore, the sponsors rely on successful contests. This is heavily based on our products performance.

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.

1.6 Goals

Our assignment is to replace 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 years to come. If the product is good enough it would hopefully also be used in larger programming competitions, maybe even international ones.

Chapter 2

Task Description and Overview

2.1 Task Description and Overview

The first step in our development process was to get a brief overview of our complete system. To do this, we have followed a conventional style of designing UML use cases together with a textual description. From reading this chapter, the reader should be able to understand how our end product works. The assumptions and constraints that affected our process are also discussed in section 2.4. Reading this chapter will be important to understand the rest of this report.

2.2 Assignment

According to our customer, IDI Open's previous solution was cumbersome to use. Our assignment was to create a replacement system that would be easier to administer. This included replacing both front and back-end systems.

The features of the old solution, in a nutshell, are given below.

- Website containing information about the contest

- Team-registration and scoring

- Ability for users to upload code to be compiled and executed

We were given access to the code for the old solution. The customers felt that this code was cluttered, but we could re-use components wherever we wanted. However, it was important that we did this in an structured manner, such that other developers could easily understand the new solution.

2.3 GentleIDI

Since we were delivering an end product to a real customer, we wanted to present ourselves as a real company. We chose the name "GentleCoding" as our representative name. This was used to name our repositories, email lists and other media communicated with external parties.

The term "Gentle" is supposed to represent our calm approach to problems. It is also similar to "Gentleman", which reminds of quality and good conduct. Furthermore, it is easy to interpret and remember.

Since we were developing a new system, we also wanted to brand our product. We wanted to keep it logical and simple, so we decided on "GentleIDI". Consequently, GentleIDI may be used to refer to our end product throughout the rest of the report.

2.4 Assumptions and Constraints

To define what is satisfactory, we have made some assumptions and defined some constraints. Table ?? should make it easier to understand how we have reasoned our system design.

Table 2.1: Assumtions and constraints

Assumption/Constraint	Why	Implication
Assumption/Constraint	Why	Implication
The system will be maintained by people who have experience with computers.	People that are involved with any programming contest are typically programmers themselves.	User design, words and definitions can be made more technical. Error messages can be explained using computer lingo.
The system will be used and maintained for > 5 years	Customer-constraint: they do not want to spend too much time developing new products, so maintenance is preferred.	The code should be written in a modular, extensible way with clear documentation.
The customer is based at Gløshaugen.		High availability for customer meetings and reviews.
The developers will maintain a 20 hour a week work ethic throughout the project-duration of 20 weeks[TODO: update].	To finish the product on time.	The set of requirements should not require more than 20 hours of work per week per developer, in order to complete.
Our system should be user-friendly	Our solution features a web interface available to everyone. Ideally, any person should be comfortable with the user interface.	Should have a user-friendly interface.
Our end product will be open sourced.	To ensure quality, and let other volunteers contribute to the code repository.	No proprietary third party modules can be used. We cannot copyright our own material.
The final product must run on Linux-computers.	This is the choice of OS by NTNU, which is responsible for technical support and server access.	Linux-compliant solution.
We are allowed to use whatever third party plugin we want, as long as it is free and has no copyright-conflict.	Speed up development.	Speed up development.

Do note that the implications in table **??** were not necessarily upheld. Rather, they were used as initial bounds to permit leeway. For example, imposing that third party plugins will speed up development does not mean that we would alway prioritize software re-use.

2.5   Roles and Their Definitions

2.5.1   Usergroups

Within the application-domain of GentleIDI there are different groups of users. Each group has different levels of access control, and once a user is made a member of that group, they inherit those rights. A user may have membership in all groups. A privileged user is someone who is given elevated permissions. Table 2.2 shows the different roles and their available actions. Further elaborations on each group will also be given in later sections, but table 2.2 should suffice for an overview.

Table 2.2: Usergroup overview

ID	Story
Role	Description
Admin	Privileged. An admin can modify all the available settings of the system
Judge	Privileged. Similar to an admin account, but with a limited set of actions: answering questions (clarification system), upload problem to be solved, solutions to those problems, and incorrect answers (e.g. answers that will provide penalty).
Functionary	Privileged. Functionaries hand out balloons when a team has solved a problem. To determine what team will be given a balloon, the functionaries have their own interface with a team overview.
Contestants	A contestant has an account on the system and has the possibility to enter and compete in a contest.
Team	A group of one to three contestants. A contestant is only part of one team per contest, and need a team in order to compete.

2.5.2   Service-providing Units

Another way of viewing the task description in section 2.1, is to say that our solution needs to do three actions: serve web-content, store data and execute user-submitted code. Since each of these operates with different protocols, we will think to our solution as composed of three different systems. These are described in figure X.X.

Table 2.3: Service-providing Units

Entity	Features	Protocol
Webserver	Processes requests from contestants and teams. Also acts as an interface to the execution node, both receiving and transmitting data to other execution nodes on the behalf of users.	HTTP
Execution node	A service, often on a dedicated platform, that offers the ability to compile and execute code. The execution node returns output data to the webserver.	AMQP
Database	The storage unit for all user-data and logs.	SQL

2.6   UML Use Cases

We need one page each for privileged, registered, and non-registered users. That is, one interface for administrative users, one for contestants, and one for non-registered viewers. From each of these three, we defined use case scenarios. Figure X.X and X.X models the available workflows and actions for each category of users. Table 2.4 describes the semantics of objects used in the diagram, which should be equivalent to the UML 2.0 standard.

Table 2.4: UML Notation

	Use case actor. Represents a user group
	UML generalization arrow. Used to indicate inheritance. The arrow's tail represents the entity that inherits from where the arrow points to.
<<include>>	UML stereotype to represent a mandatory extension to a workflow.
<<extend>>	UML stereotype to indicate that if certain conditions are met in a flow, the entity to which this arrow points to can extend the workflow.

The purpose of the use case diagrams is to give a clear overview of what users shall be able to accomplish from our system. Furthermore, use case diagrams are easier to communicate to external parties, such that it is easier to agree on the system's properties. The use case diagrams were used early in development to agree on the requirements specification and to communicate what we were trying to accomplish.

As seen in figures fig. 2.1 and **??**, admins has privileges to perform the actions of any other group, in addition to their own set of actions. Thus, membership in the admin group gives a user complete control in the application domain. Furtherly, it can be noted that all usergroups have the opportunity to act as a contestant to review the website. Privileged users will are still restricted from appearing in the official high score tables to prevent them from assuming a competing role. This was to avoid the chance of any person with access to the solutions to compete.

9
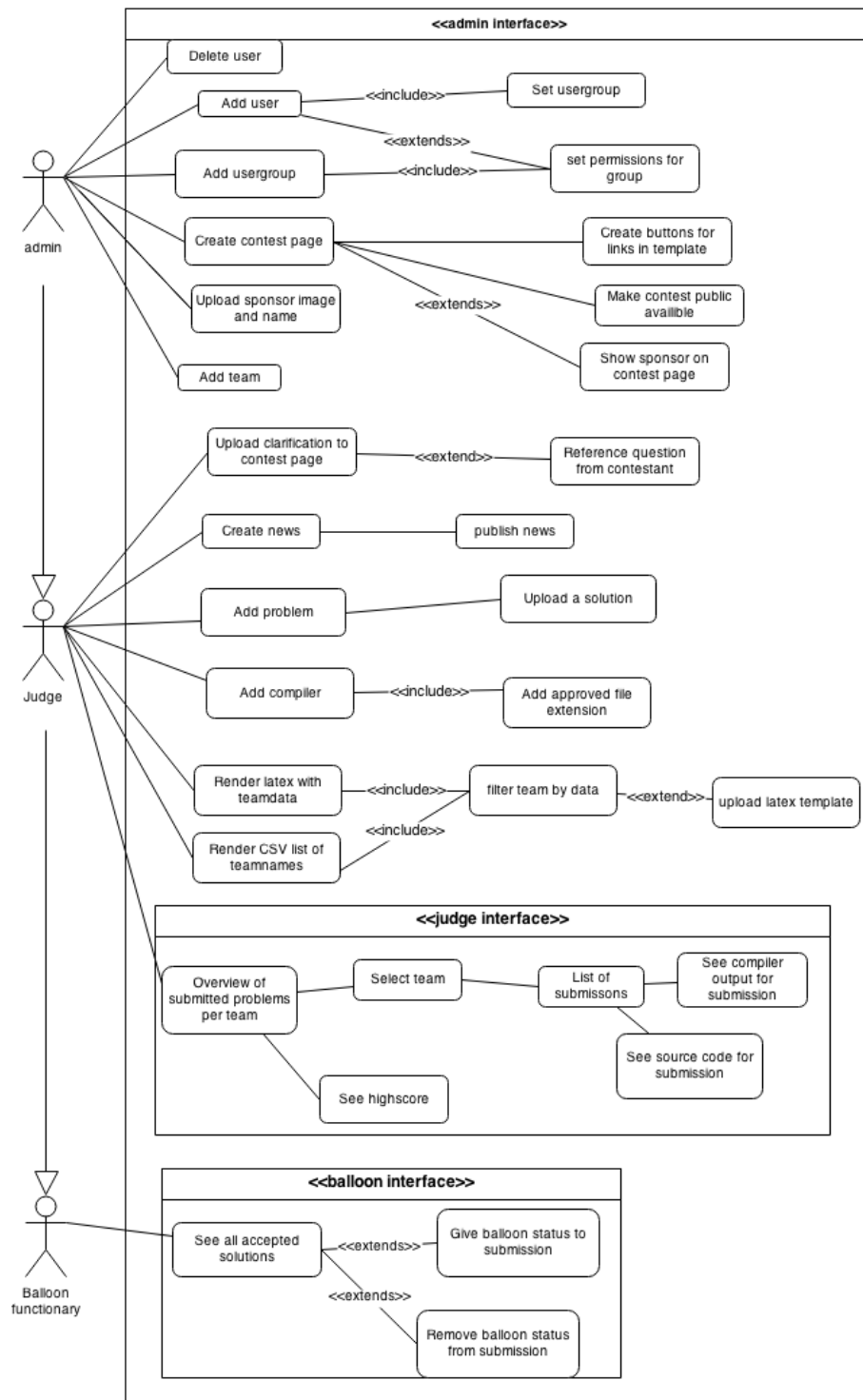
Figure 2.1: UML use case for priviliged users
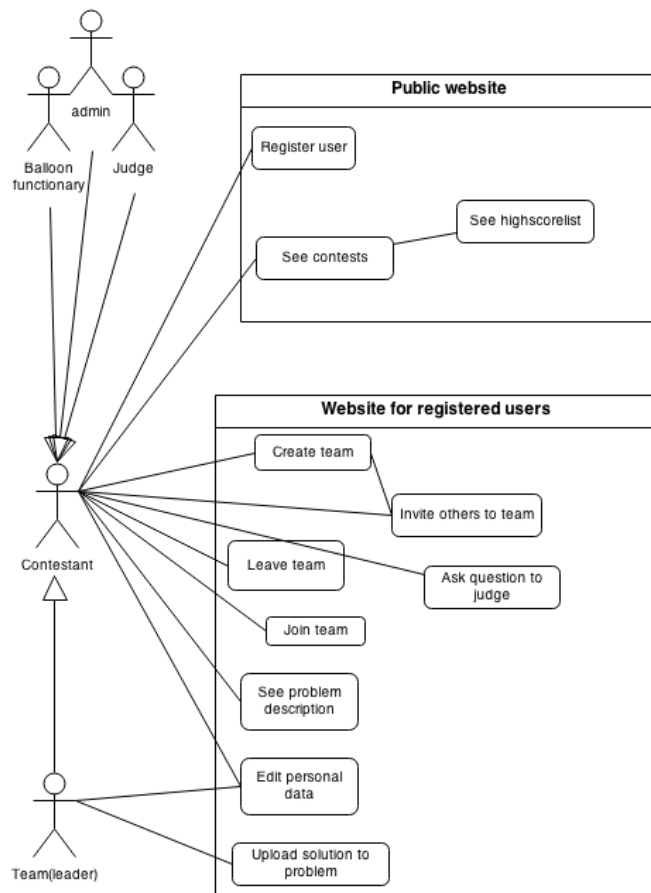
Figure 2.2: UML use case for non-privileged users

Chapter 3

Project Management

This chapter 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 3.4.1. We also provide a structured overview of how we organized our time.

3.1 Project Roles

We wanted to ensure that all developers had an even workload and experience in all components of our project. We maintained a flat organizational structure where all decisions were made in groups. No member would work alone on a task for a longer period of time. Some tasks and delegations were easier to assign only once.

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. The scrum master 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 assigned Håkon to be customer contact, and Tino as responsible for room reservations.

3.2 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.

3.3 Tools/Framework

The customer wanted our end product to be easy to maintain for future developers. Therefore we have chosen tools that are well known and easy to learn. Some of the most important are:

- Django, a framework written in Python.
- VIM and Eclipse for editing
- Google Drive and latex for documentation
- Git as version control, with github as hosting service
- Email lists, IRC and Facebook for communication
- Bootstrap and Grappelli for user interface design

A lot of different tools were considered for this system. A full list of all tools and frameworks used and considered can be viewed in appendix *Tools and Frameworks.*

3.4 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.

3.4.1 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 the WBS are as follows:

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

Table 3.1 shows the work breakdown structure created. These high-level packages were later broken down into activities, which are in the product backlog, see appendix X

Table 3.1: Work breakdown structure

1. Project management
   1.1. Write timesheet template
   1.2. Look at the reflection notes
   1.3. Meetings
       1.3.1. Internal
       1.3.2. Customer
       1.3.3. supervisor
   1.4. Report
       1.4.1. Preliminary version
       1.4.2. Mid-semester version
       1.4.3. Final version
   1.5. Risk assessment
   1.6. WBS
   1.7. Status report
   1.8. Activity plans
2. Pre-study
   2.1. Install and learn tools
   2.2. Learn language/framework
   2.3. Course
3. Design
   3.1. Requirement Specification
       3.1.1. Functional
       3.1.2. Non-functional
   3.2. System architecture
   3.3. Database modeling
   3.4. User Interface
       3.4.1. Prototyping
       3.4.2. Usability Testing
   3.5. Admin interface
4. Development
   4.1. Backend
       4.1.1. Execution-node(s)
           4.1.1.1. Web-page
   4.2. User management
       4.2.1. User
       4.2.2. Usergroups
       4.2.3. Team management
   4.3. Statistics
   4.4. Contest management
   4.5. Clarification system
   4.6. Balloons system
   4.7. Unit testing
5. Testing
   5.1. User-test
   5.2. System-test
   5.3. Final test
6. Implementation
   6.1. Deploy to production
   6.2. Installation
   6.3. Turn in to stakeholder
7. Implementation
   7.1. Verify
   7.2. 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 3.2

Table 3.2: Gantt chart

WP Name	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Project management	█	█	█	█	█	█	█	█	█	█	█	█	█	█	█
WBS	█	█													
Pre-study	█	█	█												
Install and learn tools	█	█	█												
Learn language/framework		█	█												
Course		█	█												
Design		█	█	█	█	█	█	█	█	█					
Requirement specification		█	█	█	█	█	█	█	█	█					
System architecture			█	█	█	█									
Database modelling			█	█	█										
Tests			█	█	█										
User-interface					█	█	█	█							
Development				█	█	█	█	█	█	█					
Execution node							█	█	█	█					
Implement single node							█	█	█						
Implement several nodes								█	█						
Content Management System										█	█				
Front end				█	█	█	█	█	█	█					
Testing							█	█	█	█	█	█	█	█	█
Unit testing							█	█	█	█	█	█	█	█	█
Integration testing									█	█	█	█	█	█	█
System test									█	█	█	█	█	█	█
Production														█	█
Post-implementation															█

The gantt chart was revised several times during the first four sprints, mainly due to new deadlines set by the customer.

3.4.2 Milestones

Throughout the project, the supervisor, customer, and the project group set deadlines. Some of the milestones marks the completion of work packages. 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 used the milestones in order to determine if the project is on schedule and to monitor the progress. The reader can view what requirements that were met for each milestone in 5.4.1.

Preliminary report M-01 Preliminary report is the delivery of the first version of the report. This was 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 16th of March. We wanted to complete this one week earlier, 9th of March, focus on M-03.

First release M-03 This milestone marks the groups first delivery to the customer. 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 on how to manage the system.

Presentation M-04 The main purpose of the presentation is for the class to share their experiences and learn from other groups.

Beta-release M-05 The beta release should contain most of the essential features. This version of the program should only be a release to a selected group of people.

IDI Open test event M-06 On April 26th there will be 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 a release version.

Final report M-08 This milestone marks the final date for delivering the report and the end of our project. Based on feedback received from the competition the group might choose to implement some changes to the system.

3.4.3 Meetings

Our meetings 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, a proposal that has to be followed.

- No laptop should be open during the meetings

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 the next meeting?

- Do you have any problems regarding the completion of your task?

The group would usually 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 meeting structure for this meeting can be viewed in the appendix TODO

Supervisor meeting

Meetings with the supervisor was generally held at a biweekly 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 charts. These activity diagrams were early on replaced by sprint backlog and burndown charts to facilitate the development process.

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.

3.4.4 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 two weeks before easter. During easter, the amount of hours per week scaled up higher.

Planned work

Table 3.3 shows our first initial draft of sprints.

Table 3.3: Initial sprint overview

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	50
Total:		91	368

Actual work

Table 3.4 shows the actual sprints and work done. The hours are for each person, during that sprint.

Table 3.4: Actual work

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	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	65
Total		100	453

Chapter 4

Prestudy

Before we started the actual development of GentleIDI, we needed to get an overview of what options were available to us. There are many Web development frameworks, and we were free to choose whichever we wanted. In addition, we needed to get an understanding of the problems we were intended to solve: what did the old system look like, what were its major flaws?

4.1   Learning Tools/Framework

The first challenge we were faced with was that of deciding which programming languages and frameworks we were to utilize. We were recommended to use the Python-based framework called Django, both from our customer and other developers. We decided to make a quick attempt at getting comfortable with Python and Django, and then later on make a decision on whether we were to use it or not. Most of our group had previous experience with Web development using JSP(JavaServer Pages), and JSP was our primary fallback in case we decided Django was not suitable for our needs.

We were able to master the basics of Django quite fast and its advantages became apparent. It was quickly seen that Django had enough features for the scope of our project, and a sizable community in case we needed help to use the framework. In other words Django was chosen quite early on, and mastering it became a priority during the initial pre study phase. In order to make sure that everyone had a basic understanding of the most central components of the framework, we decided that every member of our group were to implement their own Website providing basic functionality for posting and editing news articles. The best article Website was chosen as the fundament for GentleIDI, and is part of the end product.

When we first started working on the execution nodes we needed to find an appropriate way of sandboxing the user submissions. Running untrusted code with no restrictions would be an unacceptable security flaw, and we decided to put a lot of work into finding the best possible way of securing our system. As a consequence we spent a lot of work hours researching state-of-the-art security mechanisms in Linux, including cgroups, AppArmor, and Vagrant for setting up virtual machines. However, this effort proved unsuccessful and we ended up using none of it. We ended

up falling back to restricting access simply by setting file permissions and creating dedicated users with reduced privileges.

Throughout the entire project we used Git for revision control, thus, mastering Git became increasingly important as the complexity of the code base increased. We never dedicated work hours to learning Git, however, we kept a constant focus on getting better and using the features available to us. By the end of the project we were all at a level of competence where using Git saved us a lot of work and improved our efficiency significantly.

4.2   Researching the Old System

The old system consisted of a Python backend for evaluating submissions and a PHP frontend. Though the system was in working order, a lot of the management had to be done by source code modification, and direct database manipulation. Initially we considered reusing parts of the old system, but those plans were dismissed when we got access to its source code. Hard coding, lack of modularity, and redundancy were in abundance in the existing source.

As a consequence we mostly referred to the source in order to understand fine details of the system requirements. In addition to poor code quality, some key design decisions crippled the old system's scalability, such as using a SQLite database. SQLite is mostly intended for mobile apps and development environments, and unsuitable for large scale systems. Researching the old system made us aware of several pitfalls, and for the rest of the project we kept a strict focus on scalability and code quality.

4.3   Similar Software

There are other systems that have functionality similar to our system. Notable examples include Google Code Jam, NCPC (Nordic Collegiate Programming Contest) and Facebook Hacker Cup. They are commonly referred to as Competitive programming. There are some differences from competition to competition, but the basis is the same. They are also not open-source so it was impossible to get a look at the code. Researching these similar systems never had any real impact on the solution we decided to go for, and the research never lead to changes in the requirements. This was mostly due to them being closed source and the fact that our scope was quite large from the beginning.

4.4   Desired Solution

The desired solution from the customers perspective was a system that had the same functionality as the old system. But making it so that one does not have to edit the source code, or the database directly, to complete simple tasks. Except from the initial setup, all management should be done through a simple Web interface available to the admins. In addition, the customer wanted a scalable system, which could handle massively large contests, given the hardware.

We as students and developers had certain hopes as to what the end product would look like. One of our greatest desires was that the end product would be maintained and developed further

in the years to come. A great way of increasing the chances of that becoming a reality is making sure that our product is open source. With the source available to the public anyone can contribute to the project, making sure that it continues to evolve. Another advantage for us is that with an open source product anyone is free to inspect our work, and as such the product becomes a great way for us to show off our abilities.

Any development project with a scope as large as our's is likely to reuse other people's code, incorporate open source products etc. Making a product dependent on an open source project that suddenly stops being maintained was a pitfall we definitely wanted to avoid. Hence, our ideal end product would be based on nothing but source that we were sure would be maintained for years to come. With this in mind our ideal end product would be open source, with code quality that we can feel proud sharing with others, and that others would want to contribute to.

4.5 Study Result

The prestudy lasted four weeks and our results regarding libraries and development tools can be viewed in Project Management and Tools and Frameworks. Development method chosen can also be viewed in Project Management. For the technical choices see System Architecture.

4.6 Technology

During the prestudy phase the group considered different frameworks to be used.

4.6.1 Django

Django is a free and open source Web development framework, written exclusively in Python. Everything from the framework itself, to the applications intended to run on top of it, is implemented in Python, even settings are written as Python scripts. Django is also based on the MVC architecture, which suits us fine as we are all used to work in an MVC context. It is structured in a way that emphasizes the DRY principle. The combination of Python and Django lays a good foundation for rapid development, and high maintainability.

4.6.2 Flask

An alternative to Django which we did consider was the Flask micro-framework. Though Flask might have been a viable alternative, we discovered it a little too late, about a week into our Django training. Because we did not see any major advantages of Flask over Django, we decided to stick to the latter. Like Django, Flask is also written in Python.

4.6.3 JSP

JSP stands for JavaServer Pages, and is a technology used to create dynamic Web pages. As Django, it also adheres to the MVC approach. Some of the group members had experience with JSP, but we decided not to use it. The most commonly learned programming language at NTNU is

Python. Since JSP is based on Java we feared that the use of JSP would affect the maintainability of the program. It was therefore not used.

4.6.4 Backend Technology

There are some security issues regarding letting the users run their programs on our backend. In worst case a contestant could write harmful code setting the contest to a halt. We discussed different approach to eliminate this problem.

4.6.5 AppArmor

Abbreviation for Application Armor. It is a Linux kernel security module allowing the system administrator to associate a security profile with each program. This would restrict the capabilities of that program. We think the optimal product would have used AppArmor, however we were unable to make it work the way we wanted. One of the major issues we had was that of restricting interpreted languages like Java and Python. Programs written in these languages require an external binary to execute them, and restricting this binary without breaking the rest of the system was hard. E.g. in order to restrict submissions written in Python we would end up restricting every single Python program running on the system. Given more time we might have been able to make this work, however, that was not the case.

4.6.6 Our Alternative to AppArmor

The solution we finally decided to go for was to execute the submissions as a sandboxed user. We created a user, removed its network access, and made sure that the only relevant files it had access to was the submission executables. The major weakness with this solution is that security is dependent on file permissions per file. A single executable with the wrong permissions could be a threat. There are some programs that are capable of running subprocesses as a second user, and thereby bypassing the network blocking.

4.7 Development Method

At the beginning of our project we all had some experience with Scrum, which made it the most obvious choice in terms of development method. It is quite suitable for this type of small scale project and keeps the planning overhead at a minimum. However, as far as we knew there could be other, better suited methods for our project. The major requirements we had to the development method was that it should be agile and iterative. We also wanted to work in a test driven manner, and do most of the programming in pairs. This made Extreme Programming a viable alternative.

Extreme Programming has a greater focus on pair programming and code reviews, than Scrum. This was in tune with the way we wanted to work, however, using XP would increase the amount of time needed to get comfortable with the development method. There are some major pitfalls related to using a development method you that you do not master, and when it came down to making a decision we played it safe. We decided to use Scrum and implement pair programming and test driven development in our own way.

When the development started we soon realized that we did not have the competence needed to do test driven development well, and this aspect was dropped. We ended up using Scrum with a focus on working in pairs.

Chapter 5

Requirements Specification

According to the gantt chart (Fig 4.1) the team were supposed to update the requirement specification starting from week 2 and continuing up until week 10. For us it was still the case that there were a clearly identifiable requirement specifications phase. This was primarily from week 2 up to and including week 4. The outcome from this three week process was heavily used in order to establish agreement between us and the customer. This chapter presents the result from this process.

5.1   Purpose and Scope of this Specification

The purpose of the requirement specification document is to specify the objectives for our end product. Requirements are written at different levels of detail. This is to make it easy to communicate the requirements to both business and technical parties. We have mainly written the functional requirements as stories and then broken them into smaller pieces. This makes the requirements easy to communicate to the customer, and succinct for the developers. These stories can be viewed in appendix TODO. It is important to recognize that our project only lasted for a few months. Thus, late changes to requirements were inserted promptly and without revision control. This is a common practice in agile development[1]. The advantage and reason we chose not to perform revision control, is that we could save time in not formally documenting all changes.

The coverage of the requirements is intended to be a complete coverage of the product. This implies that all features available from the application domain is listed in our specification. What the requirements specification does not cover are organizational and external requirements. This follows from the small amount of administrative users and developers involved, and trust between the customer and the developers.

5.2   Process of the Requirement Specification

The customer passed on an initial list of requirements to our group. After a classification and organization of the features, we drafted scenarios and internally discussed the implication to each requested feature. Therein, we saw what features would be infeasible and additional features we

---

[1] Page 91, Sommersville

would want to introduce to the customer. The modified list of requirements was then presented to the customer, before proceeding with the implementation of the end-product. Throughout the entire development process both we and the customer have been modifying the list of requirements.

5.3 Product/service description

In this section, you will find our interpretation of the physical user-domain. The reader should note that some members of our group has competed earlier, which has given us helpful empirical insight.

5.3.1 Expected Physical Environment

Our solution is used in different contexts. Table X.X has the different application and user-domains.

IDI Open is hosted in P15, Høgskoleringen 3, on Gløshaugen campus every year. Every team participating in the contest get allocated their own computer.	For offsite contestants, javascript must be enabled.
Software is required. A web server(Apache, Nginx), database server(MySQL, PostgreSQL), Python with PyPi package manager.	Linux kernel with ssh enabled, supplemented with a root user.

5.3.2 User Characteristics

Table X.X show different stereotypes of expected typical users. While open to deviations from the stereotypes, they highlight important properties required for our solution.

- Irresponsive interfaces - Incorrect data - User submission system - Response types	- Irresponsive interfaces - Node failures - Incorrect data - Backend system - Dataflow
keep track of score - Irresponsive interfaces - Lack of overview - Backend system - Dataflow	- Dissatisfied contestants - No overview - Nothing special
and information - Mis-information - Scoreboards, about competition	

It can be seen in table X.X that the most prominent trait of our users is that they have a background in computer science. As a consequence, it is assumed a higher level of technical competence from our users. The user profiles also highlight that some features were more important than others, e.g. responsiveness over aesthetics.

5.4   Requirements

Stories can be ambiguous and open for misinterpretation. we felt that a natural language specification of requirements would make it easier to understand our application domain. To reduce miscommunication we made sure to give each specification as short, succinct sentences. The stories were used as a way to communicate with the customer about requirements without them having to read through the table of requirements.

There are three different states for priorities, HIGH, MED and LOW. This ensured strict priorities. Using more states would make it hard to differentiate between the priorities we gave the requirements.

The following definitions make out the guideline for prioritizing the requirements:

- HIGH: The requirement is a "must have". To have a successfull product, the requirement must be implemented.

- MED: The requirement is a "should have". The fulfillment of the requirement will benefit the quality system.

- LOW: The requirement is a "nice to have". This includes functionality not critical to the system.

5.4.1 Functional

The functional requirements are broken down in different categories. Each category corresponds to a user group. The categories are Admin, Judge, Contestant, Functionary, Teams, and Other. Each category has an ID, priority and story. Table X.X shows the complete list of the requirements, while the corresponding stories are given in appendixB

The ID system can be interpreted in the following way

- The F stands for Functional

- The second letter determines which category, e.g A stands for admin.

The milestone show when each requirement needs to be met.

5.4.2 Functional requirements for Admin

Table 5.1: Feasible triples for highly variable Grid, MLMMH.

Requerment	ID	Story	Comment	Priority	Milestone
An admin shall be able to create a new contest	FA-01	SA-1	A new contest equals a new web page	HIGH	M-03
An admin can choose whether the site should be published immediately or not	FA-02	SA-1		MED	M-03
An admin can add custom CSS to the web-page	FA-03	SA-1		LOW	M-03
An admin shall be able to choose settings for the contest	FA-04	SA-1	of contestants, maximum number of contestants per team, date, name. Default settings will be provided	HIGH	M-06
An admin shall have access to all modules in the program	FA-05	SA-2		HIGH	M-06
An admin can change permission of a usergroup	FA-06	SA-2		LOW	M-06
An admin can remove/add to a user group.	FA-07	SA-2	This includes promoting new admins	LOW	M-06
An admin can deactivate users	FA-08	SA-2		LOW	M-06
An admin can remove users from the database	FA-09	SA-2		HIGH	M-06

27

Table 5.1 – continued from previous page

Requrement	ID	Story	Comment	Priority	Milestone
An admin can add a node	FA-10	SA-4	The node must be a privileged user	HIGH	M-06
An admin can remove a node	FA-11	SA-4		HIGH	M-06
An admin can manage a node.	FA-12	SA-4	This requirement is in terms of compiler profiles support	HIGH	M-06
An admin can add more than one node	FA-13	SA-4		MED	M-06
An admin can add news items	FA-14	SA-5		HIGH	M-03
An admin can remove new items	FA-15	SA-5		MED	M-03
An admin can modify news item	FA-16	SA-5		MED	M-03

5.4.3 Functional requirements for Judge

Table 5.2: Feasible triples for highly variable Grid, MLMMH.

Requirement	ID	Story	Comment	Priority	Milestone
A Judge can create a problem	FJ-01	SJ-1	This includes cases with input and output	HIGH	M-06
A judge can upload cases to a problem and name each case	FJ-02	SJ-1		MED	M-06
A judge can set a resource limit on each task	FJ-03	SJ-1		LOW	M-06
A judge can add a solution that gives the right output	FJ-04	SJ-1		HIGH	M-06
A judge can add a solution that gives timeout	FJ-05	SJ-1		MED	M-06
A judge can add a solution that gives wrong answer	FJ-06	SJ-1		MED	M-06
A judge shall be able to view and edit all problems	FJ-07	SJ-1		HIGH	
A judge shall be able to respond to a question from a team	FJ-08	SJ-2	This is about the clarification system.	MED	M-06
A judge shall get a notification when received a question	FJ-09	SJ-2		LOW	M-06

Table 5.2 – continued from previous page

Requirement	ID	Story	Comment	Priority	Milestone
A judge shall be able to respond to a question globally	FJ-10	SJ-2	By globally it is intended that the all teams can view the response and question	HIGH	M-06
A judge shall be able supervise all submissions	FJ-11			MED	

5.4.4   Functional requirements for Contestant

Table 5.3: Feasible triples for highly variable Grid, MLMMH.

Requrement	ID	Story	Comment	Priority	Milestone
A contestant shall be able to edit their own information	FC-01	SC-1		HIGH	M-03
When created a contestant shall receive a confirmation email	FC-02	SC-1		HIGH	M-03
A contestant shall see which teams they are invited to	FC-03	SC-2		HIGH	M-03
A contestant shall see which team they are a member of	FC-04	SC-2		HIGH	M-03
A contestant shall see which teams and contests they have participated in earlier	FC-05	SC-2		MED	M-03
A contestant shall be able to ask a question to a judge	FC-06	SC-3		MED	M-03
A contestant shall have access to global answers from judges	FC-07	SC-3		MED	M-06
A contestant shall be able to change his/her email	FC-02	SC-2		MED	

5.4.5 Functional requirements for Functionary

A functionary shall be able to register a balloon colour to each task/problem	FF-01	SF-1			LOW	M-06	TF-12
A functionary shall have access to information about newly completed problems	FF-02	SF-1			MED	M-06	TF-12

5.4.6 Functional requirements for Teams

Table 5.4: Feasible triples for highly variable Grid, MLMMH.

Requrement	ID	Story	Comment	Priority	Milestone
A user shall be able to register a team	FT-01	ST-1	Whether or not the team is onsite, a team password, and a email for the team leader	HIGH	M-06
A user shall be able to register other team members for the team	FT-02	ST-2	By providing other users' email	HIGH	M-03
If the contestant is already in the system shall recognize personal info	FT-03	ST-2	Personal information like name, gender and so on.	LOW	M-03
A team leader must be able to invite new members	FT-04	ST-2	Input: email	MED	M-03
A team leader should be able to delete the team before the competition	FT-05	ST-2		MED	M-03
When a team leader invites a new member the new member must receive a registration link	FT-06	ST-2	The receiver of this email link must fill in the data specified in: T-3	MED	M-03
If a member's email is already in the database they will receive a confirmation link	FT-07	ST-2	The confirmation link will include automatically filled data. See T-4	LOW	M-03
All team information is editable in the team overview.	FT-08	ST-2		LOW	M-03

Table 5.4 – continued from previous page

Requrement	ID	Story	Comment	Priority	Milestone
A team must be able to deliver submissions to problems	FT-09	ST-3		HIGH	M-06
When a team deliver a submission they shall receive response from the system	FT-10	ST-3	system should give time-out. This is specified by a judge.	HIGH	M-06

5.4.7 Other requirements

Table 5.5: Feasible triples for highly variable Grid, MLMMH.

Requrement	ID	Story	Comment	Priority	Milestone
The system shall be able to gather some statistics	FO-01	SA-3	It is here implied statistics from contestants in accordance with FE-3	HIGH	M-05
The system shall be able to gather a large variety of statistics specified by the admin	FO-02	SA-3		LOW	M-05
The system shall include a clarification system	FO-03	SJ-2	This is according to FJ-8, FJ-9, FJ-10, and FE-14, FE-15, FE-16, FE-17, FE-18	HIGH	M-07
The contest results are to be visible in the form of a highscore list.	FO-04	ST-03		MED	M-07

5.5 Non-functional

The nonfunctional requirements defines what objectives our end product needs to meet. Measure make it easy to agree on whether the requirement is fulfilled or not. Tables X.X can be interpreted in the following way:

- NF in the ID stands for non-functional

- Measure describe what the requirement holds

- Value is a quantitive measure

-

5.5.1 Speed

ID	Measure	Value	Priority	Comment
NF-01	Response from action	< 1.5 sec	MED	E.g. clicking a click
NF-02	Posting news	< 5 sec	MED	
NF-03	Edit user	< 1 min	MED	E.g. change email, password

5.5.2 Size

ID	Measure	Value	Priority	Comment
NF-04	Number of contestants	500	HIGH	
NF-05	Number of teams	200	HIGH	
NF-06	Number of judges	20	HIGH	
NF-07	Number of admins	> 1	HIGH	
NF-08	Limitation of submission size	50kB	HIGH	

5.5.3 Ease of Use

ID	Measure	Value	Priority	Comment
NF-09	Learning time for contestants	< 5 min	MED	The users of the program should be good at computers and therefore know what they are doing.
NF-10	Learning time for admins	< 15 min	MED	
NF-11	Learning time for judge	< 10 min	MED	

5.5.4 Reliability

ID:	Measure:	Value:	Priority:	Comment:
NF-12	Mean time to failure	> 1 week	HIGH	The system should not be down during a contest
NF-13	Availability	> 99.9%	MED	Downtime is not critical after or before a contest

5.5.5 Robustness

ID	Measure	Value	Priority	Comment
NF-14	Time to restart after failure	< 10 min	HIGH	
NF-15	Probability of data corruption on failure	< 1%	MED	This is determined by backup coverage
NF-16	Expected living time	> 10 years	HIGH	
NF-17	Execution node	= 1	HIGH	
NF-18	Execution nodes	> 1	MED	It should be possible to utilize addition nodes

5.5.6 Portability/Scalability

ID	Measure	Value	Priority	Comment
NF-19	Extensibility		HIGH	Adding features should be easy
NF-20	Module-based code		HIGH	The code should be easy to maintain

5.5.7 Other

ID	Measure	Value	Priority	Comment
NF-21	Accessibility		HIGH	
NF-22	Open-source	GPL	MED	

5.6 Security

While security requirements are non-functional, we decided to do the security requirements engineering as a separate process. Table can be interpreted in the following way:

- In the ID, S is for security
- Measure describes

5.6.1 Authentication and Authorization

ID	Measure	Priority	Comment
S-01	No user in any given user group shall be able to perform any operation outside of the definition of the requirements	MED	
S-02	An authenticated user shall not be able to perform any operation, as another user	HIGH	
S-03	After an authenticated user performs an action to be logged out, that user will need to log in to re-authenticate	MED	E.g. session-cookies should not remain such that you can still re-login
S-04	No user shall gain administrative rights without manual approval of current admins		Ensure no user is registered as admin by mistake, no scripts that automatically escalates privileges to administrator when conditions are met
S-05	Correct authorization must be required for respective content.	HIGH	
S-06	To authorize, you will either need to provide mandatory usercredentials through an interface, or have a valid session ID.	HIGH	

Table 5.13 – continued from previous page

ID	Measure	Priority	Comment
S-07	Session tokens shall be unique to one computer only	MED	Not possible to simply acquire a session ID and use it on other computers to authenticate

5.6.2 Immunity

ID	Measure	Priority	Comment
S-08	No input-field shall be susceptible to injection attacks	HIGH	
S-09	All data that passes the trust zone shall be in plaintext, and validated against code	HIGH	
S-10	Data from non-developers can only be directed saved in databases.	MED	
S-11	Uploaded submissions shall not write to any file	HIGH	
S-12	Uploaded submissions shall not read from any other file than stdin	HIGH	
S-13	Uploaded submissions shall not access network or any other external service not needed to solve a problem.	HIGH	
S-14	Data from a user shall not be modified by non-users	MED	

5.6.3 Non-repudiation

ID	Measure	Priority	Comment
S-15	All modifications of data shall be logged	MED	
S-16	All log entries shall contain username(s) and a timestamp with day and current hour	LOW	
S-17	Logs will be backed up	LOW	
S-18	A team's score shall not be affected by anything other than what is given in the contest rules	HIGH	

5.6.4 Privacy

ID	Measure	Priority	Comment
S-19	Sensitive user data shall not be stored in plain-text	HIGH	E.g. password, gender
S-20	Every user-field that is stored shall be justified in the requirements specification		This requirement does no longer apply
S-21	No sensitive data shall be exposed publicly, even if it is encrypted	MED	
S-22	User-data for a given user shall not be modified without that user's consent.	LOW	

5.6.5 Auditing

ID	Measure	Priority	Comment
S-23	Database shall be manually/automatically checked/verified for inconsistency or errors before an event.	MED	
S-24	Password that are used in development shall not be publicly available	HIGH	

5.7 Requirements Not Met

Most of the requirements on time. There were some minor requirements not fulfilled mainly due to time constraints. All of them were priority LOW. Here are the requirements we did not complete:

A judge shall get a notification when received a question	FJ-09
A functionary shall be able to register a balloon colour to each task/problem	FF-01
The system shall be able to gather a large variety of statistic specified by the admin	FO-02

The reason they were not completed was due to the their low priority and time constraint. In addition to the unfinished requirements there were also requirements that were not met in an ideal way. This was in agreement with the customer. These are the partially met requirements:

An admin can add a node	FA-10
An admin can remove a node	FA-11

An admin can manage a node.	FA-12
Response from action	NF-01
Logs will be backed up	NR-03

Unfortunately, an admin can only manage the execution nodes through the code. This is planned to be fixed before the next contest. The response time did unfortunately exceed 1.5 seconds during the contest. This was due to a bad implementation of the high score list, detailed in **??**. NR-03 had to be overruled during the contest. This is discussed in detail in chapter TODO *development*.

Chapter 6

Architecture

This chapter contains an overview over the architecture for the system. The first part will describe different views of the system and the second part will show the quality attributes and patterns used when developing the system.

The main parts of the system is shown in Figure 6.1. Clients sends requests to the web server and receives the processed results. The execution nodes process user submissions, and updates the results to the database.

6.1   Views

We have chosen to depict the architecture using Philippe Kruchten's 4+1 view model. [1] This is a method of describing the architecture for software-intensive systems from the viewpoint of different stakeholder by using multiple, concurrent views. We chose this model because it gives a good overview and is widely accepted by the software industry. Below are the 4 main views in the model; Logic, Process, Development, and Physical. The "+1" view is Use Cases which is addressed in [Chapter 2 Task Description and Overview]

6.1.1   Logic View

The logical view describes the functionality of the system by breaking down requirements into classes and representing them, and their relations, through class and sequence diagrams.

Figure 6.1 shows the main classes involved in GentleIDI. Each team participates in a single contest, and consists of a predefined number of contestants. Each team also has a team leader that handles most of the administrative tasks. The team can also try to solve problems by uploading submissions.

6.1.2   Process View

The process view explains the communication between different processes in the system, as well as how the system behaves in runtime.

As this system is a web application the first thing to note is that there will be concurrent users in runtime. Each user generates HTTP requests to the server, which in turn may execute database
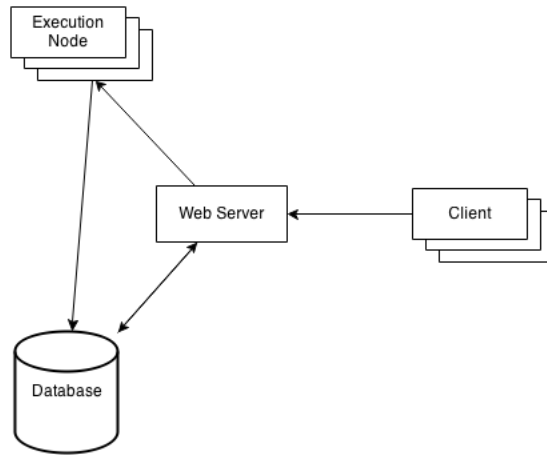
Figure 6.1: System overview
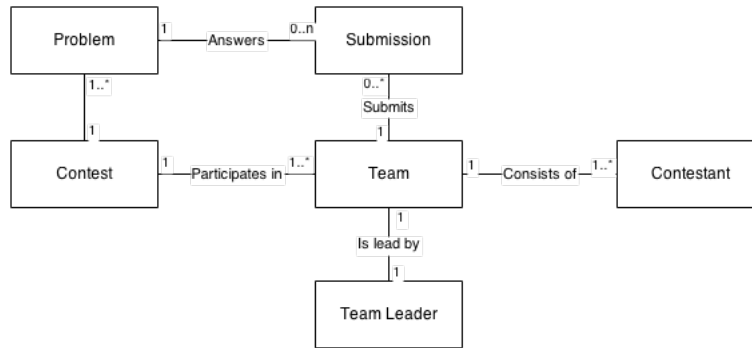
Figure 6.2: Top level class diagram

lookups for information like score tables or problem sets. When a user submits a solution the system will place it in a queue, which decides which node the solution will execute on according to availability and load.

We will now show examples for two important parts of the application. First is the action of successfully registering a user and creating a team. See figure 6.4.

Second is submitting a solution to a programming problem. See figure 6.5.

6.1.3 Development View

Purpose

The developer view is intended for the developers. It should ease development, and focus on software module organization by packaging the software in small chunks.

We wanted a modular and maintainable system where it is easy to maintain and change specific parts of the system without changing everything. The structure of the system can therefore be
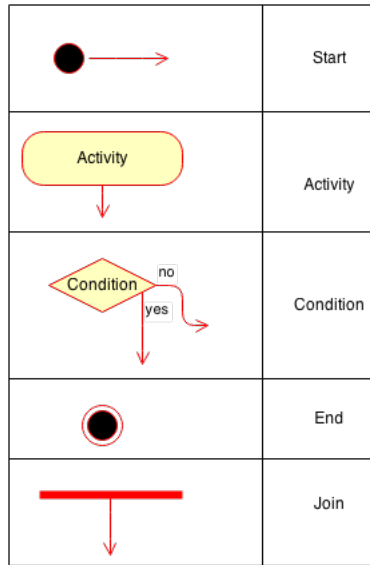
Figure 6.3: Symbology

divided into the following main packages: Contest, Registration, Submission, Execution, Balloon, Clarification, Admin, and Article. These packages are described in detail in chapter 8 Implementation.

6.1.4 Physical View

Purpose

The physical view shows the interaction between the physical components of the system.

Physically the system is structured as a multitiered architecture. It consists of three tires, presentation tier, application tier, and data tier, see figure 6.6. The tiers represents a physical structuring mechanism for the system infrastructure. The user is physically separate from the application and database.

Presentation tier

This tier presents information to the user through the public website and admin interface. It translates the web server response into web pages generated using HTML5, CSS, Ajax, and JavaScript. It sends requests to the underlying web server and renders the response.

Application tier

The application tier contains the logical layer, it controls an application's functionality by performing detailed processing. Primarily this is done through python code, although when running solutions the file is run on an execution node through the use of built in unix commands.
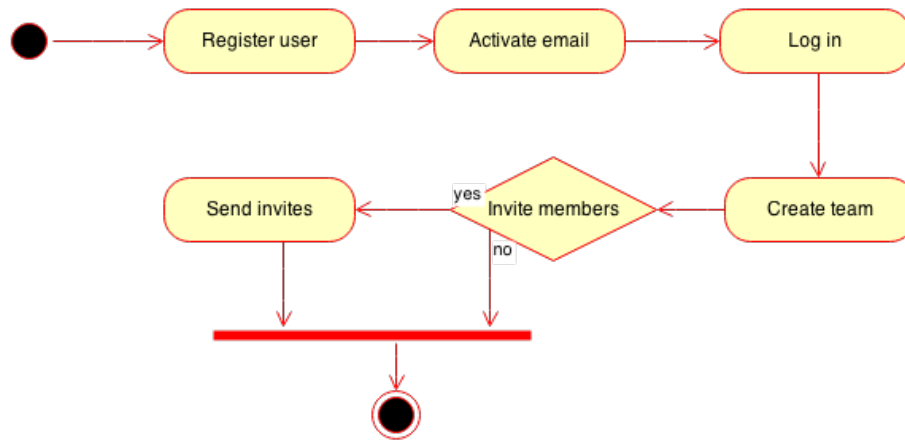
Figure 6.4: Activity Diagram for registering a user and a team
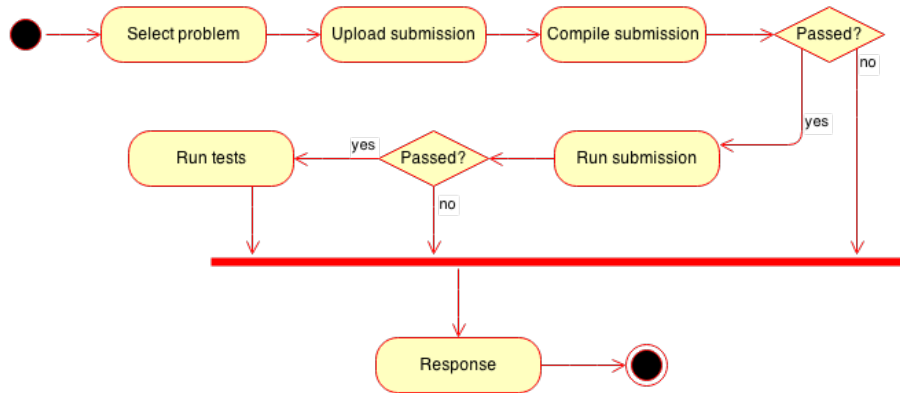
Figure 6.5: Activity Diagram for submittion a solution

This splits the application tier in two parts, the web server that serves static and dynamic content, and the execution nodes that process uploaded submissions. This division can be seen in "Application Tier" in Figure 6.6.

For the web server we use Nginx for serving static files, and as a reverse proxy for Gunicorn, the server providing dynamic HTTP content to the user. Gunicorn is the server that processes requests and returns HTTP pages. The execution nodes process submissions through a FIFO queue implemented with Celery and RabbitMQ. This provides load balancing across CPU cores and multiple nodes in the cluster. The execution nodes also share parts of the filesystem, this is implemented with SSHFS (SSH Filesystem), and is a secure way of sharing the uploaded files across the execution nodes.
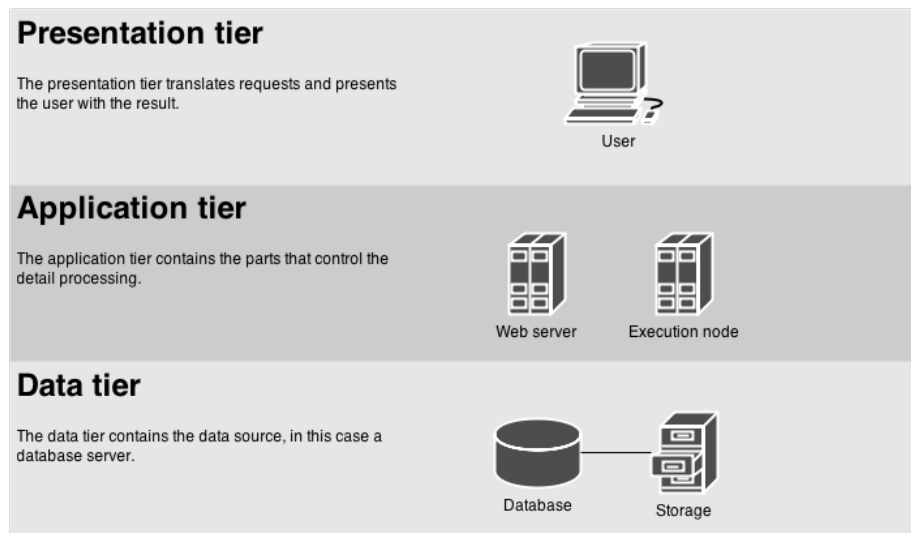
Figure 6.6: Multitier architecture

Data tier

This tier includes the data control functionality. The system utilises a shared SQL database for the execution nodes and the web server. See Figure 6.1. This database links to the file storage on the main web server. However, the execution nodes requires some files to be shared across multiple nodes. Like explained earlier in section 6.1.4, this is implemented with SSHFS. For more specific details see 8 and H.

6.2 Quality attributes

6.2.1 Availability

Since this software is to be used in a programming contest, it is crucial that the system has high uptime and availability. And since the contest only lasts for about 4 hours, our margin for failure is minimal. We have made an effort to account for all possible outcomes, and to safeguard the application for any errors that might occur.

6.2.2 Modifiability

This is a system that we hope will be used for many years to come. With the ever changing nature of the web, the ability to adapt and improve is imperative. To accommodate this, we chose to implement our solution in Python, a language taught to most of new students of computer courses at NTNU. These are the same students that hopefully will use and continue to work on this software. To our best ability we have also tried to write and document the code in a way such that it is easy to understand and improve.

6.2.3 Performance

Performance is an important aspect of every application, especially web applications. Users expect that sites loads fast. Failing to accomplish this is a sign of a bad application, at least from the user's perspective. For this reason we have focused on making our pages load as fast as possible. And since this application will be used by over 100 users simultaneously, it is also important that the servers will handle the load.

6.2.4 Security

Since our application contains user data and data that should be hidden from unauthenticated users, security is another important aspect. Django provides many security features by default, and others that can be implemented with very little effort. We also chose to enable SSL on the web server to increase security on web requests.

6.2.5 Testability

When we first started out, we wanted to utilize testing during development. Testing is a way to find problems early, and before they begin to encompass larger parts of the application. But testing is also one of the most time consuming parts of the development process. In the end we did not have as much test coverage as we would like, but we feel that we covered the most important parts.

6.2.6 Usability

As with any web application, we want the users of the system to accomplish their desired task, and learn the functions of the system with ease. The user should receive feedback if something went wrong or if the outcome is not clear. We also want the web pages to provide information how to use the system.

6.3 Patterns

6.3.1 Client-Server

Since we are making a web application we will use the Client-Server pattern. The clients connect to the server through a web interface, either the website or the admin interface.

6.3.2 MVC(model-view-controller)

The front end is implemented using the Django framework and follows a rather strict implementation of MVC. Every HTTP request sent to the site is handled by a controller function, which in turn fetches the appropriate models from a database, creates a view based on the models and returns the view as an HTTP response.

6.3.3 Shared-Data

The system utilises multiple execution nodes as well as a web server, through which users access data. We wanted to have a central shared database server that scales with the number of execution nodes and the amount of data.

6.3.4 Multi-tier

See: 6.1.4 Physical View. References:

[1] Architectural Views -

Chapter 7

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.*

7.1 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.

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 7.1. 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 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.

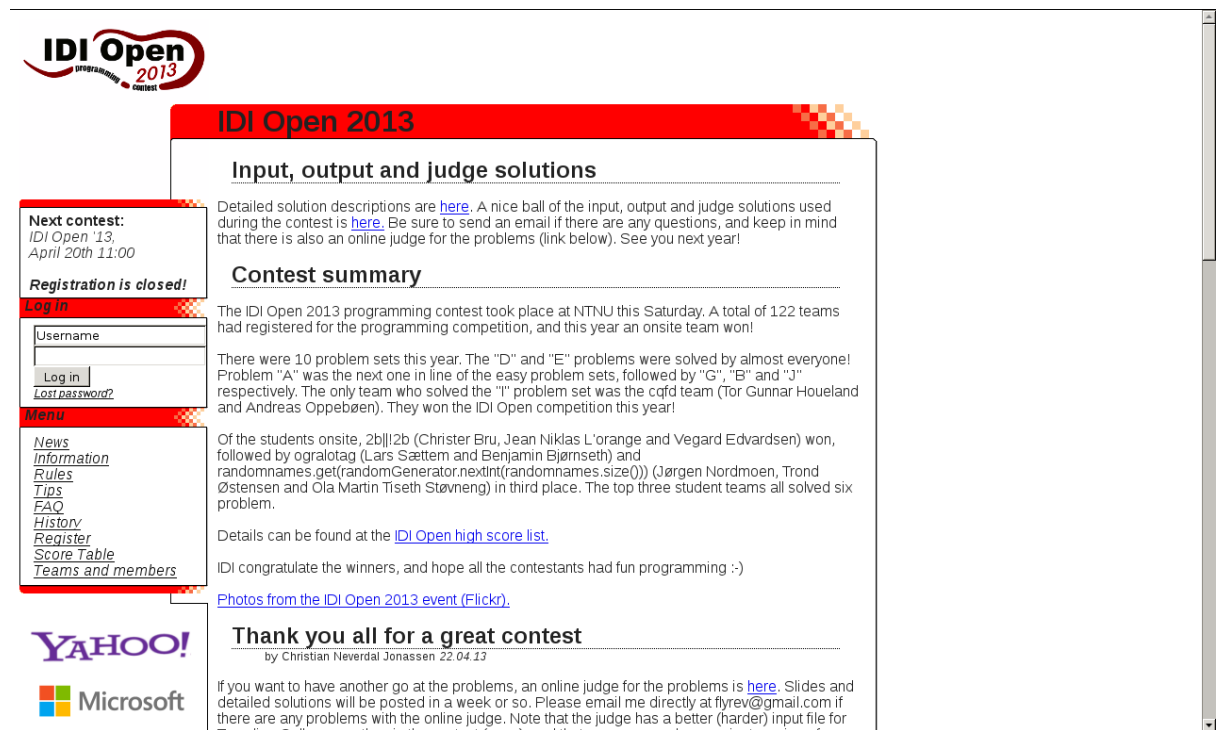The task had to be completed in time for milestone M-03, so our main concern was designing

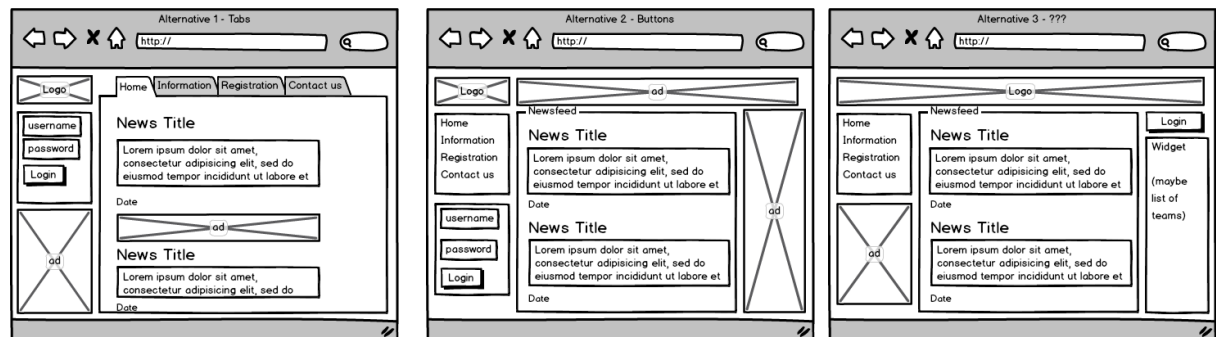Figure 7.1: User Interface of the old system

Figure 7.2: Initial mockups

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

45

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 7.3.
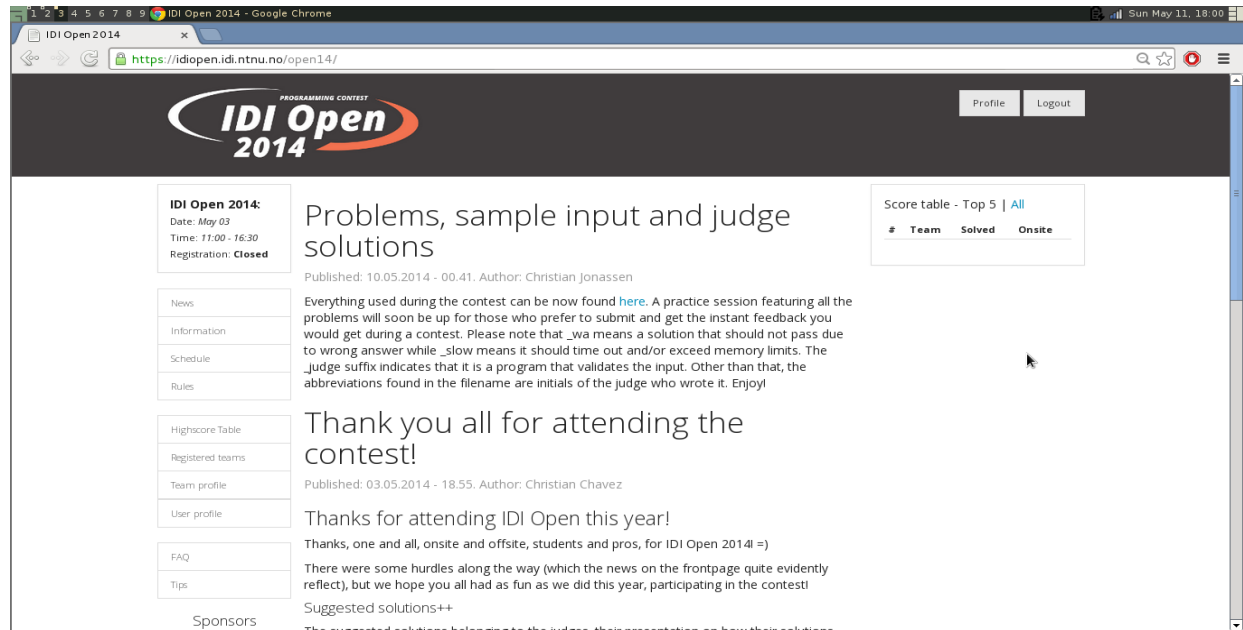
Figure 7.3: Final page

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 7.1 and Fig 7.3, you can see that we kept the same structure, but still made some significant changes to the design.

7.2 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

---

[1] http://research.microsoft.com/en-us/um/people/cutrell/chi09-buschercutrellmorris-
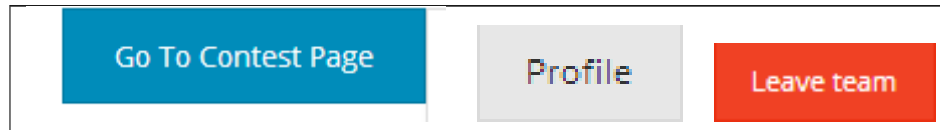
Figure 7.4: Various buttons used on our website. From left to right: the button to go to the contest page, the button to see a user profile, the button to leave a team

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 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, 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.

"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 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 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.

For the contest page, Fig 7.5, 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.

7.3 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.

---

eyetrackingforwebsalience.pdf

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

Contest Page

Clarification | Ask a question | View score table | Team score: **0**

List of Problems

Click on a table row to go to the selected problem.
Hover over each title in the table to get a further explanation.

Problem ▲	Last Submission ⬍	Time⬍	Feedback ⬍	Solved ⬍	Score ⬍
Abandon Ship [PRACTICE]					

Figure 7.5: Contest page

Figure 7.6: Admin Interface

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.

Figure 7.7: Judge views

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 7.7.

Fig 7.8 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.

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

49

Figure 7.8: Judge views for team

Chapter 8

Implementation

This chapter goes into the details of our implementation. As mentioned in section X.X, 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. The apps are intended to serve a specific purpose and provide a certain level of modularity. However, some apps are dependent on others.

Figure 8.1 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, views.py, forms.py, and admin.py.

- 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

Contest app

```
.
├── admin.py
├── forms.py
├── __init__.py
├── migrations
│   ├── 0001_initial.py
│   ├── 0002_auto__chg_field_team_name.py
│   ├── 0003_auto__add_field_contest_penalty_constant.py
│   └── __init__.py
├── models.py
├── templates
│   ├── Cage
│   │   └── cage.html
│   ├── contest
│   │   ├── alreadyContestant.html
│   │   ├── editTeam.html
│   │   ├── index.html
│   │   └── team.html
│   ├── registerForContest
│   │   ├── registrationComplete.html
│   │   ├── registration.html
│   │   └── requireLogin.html
│   └── viewTeams
│       └── viewTeams.html
├── templatetags
│   ├── __init__.py
│   ├── link_tags.py
│   └── widget_tweaks.py
├── tests.py
├── urls.py
└── views.py
```

Figure 8.1: App overview

52

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.

8.1 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 [reference contest ER]

8.2 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.

8.3 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.

8.4 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.

8.5 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 [reference submission ER]

8.6 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.

8.7 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.

8.8 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.

8.9 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.

8.10 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 9

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 fig 3.3. The second part describes each sprint in more detail including work done/completed.

9.1 Working Towards the Milestones

9.1.1 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.

9.1.2 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.

9.1.3 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.

9.1.4 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.

9.1.5 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.

9.1.6 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.

9.1.7 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.

9.1.8 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. An example is given here in table X.X.

Chapter 10

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.

10.1 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.

10.1.1 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.

10.1.2 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.

10.1.3 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.

10.1.4 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.

10.1.5 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.

10.2 Our Approach to Testing

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

10.2.1   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.

10.2.2   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

10.2.3   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.

10.2.4   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.

10.3   Testing Results

The results from our testing is presented in this section.

10.3.1   Integration Test

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

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 X.X shows our view on the system as per milestone M-03, First release. 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. Our integration tests are based upon the interfaces defined in figure 8.2.

UML stereotype	Function
<<provides>>	The component delivers the given functionality
<<requires>>	For the component to work it must have the given interface
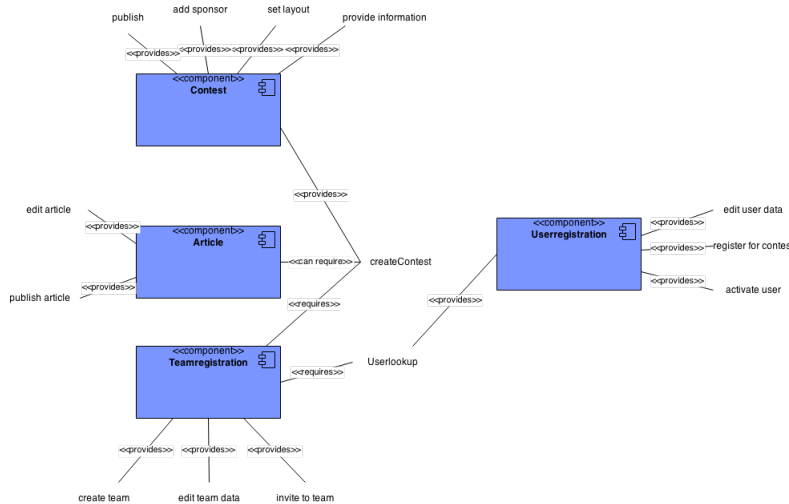
Figure 10.1: Diagram from milestone M-03. Each interface connection, especially "createContest" has been tested

10.3.2   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.

Table 10.3: System test

ID	Function	Action/Input	Result	Req	Pass/Fail
ST-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
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
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

Table 10.3 – continued from previous page

ID	Function	Action/Input	Result	Req	Pass/Fail
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

10.3.3 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. 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[TODO: REFER evaluatin of testing], we did inspection tests. Thus, table X.X 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

Table 10.4 – continued from previous page

Case	Input	ID	Expected Result	Pass/Fail
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 < 100 ms	FAIL
Page responsiveness with at least 10 RPS	HTTP GET and POST to all pages	NF-01	Response-time < 200 ms	FAIL

Chapter 11

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.

11.1   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.

11.2   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.

11.3   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 review 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.

11.4   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

---

[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

and a summary. This was later sent to the respective party to ensure agreement on what had been concluded in the meeting.

11.5   Complete List of Risks

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

Appendix A

Sprints

This appendix holds an overview over our sprints, throughout the project. For a mote 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.
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

Installation Guide

This is the complete installation guide for GentleIDI. The guide will assume that the reader has got some basic Linux skills. You should be capable of installing packages by means of a package-manager like apt, yum etc.

Though GentleIDI is not tightly linked with any specific linux distro, this guide assumes that you're using Ubuntu Server 14.04. This is the only distro on which the system has been tested thoroughly at the time of writing.

GentleIDI is in many ways a straightforward Django-based website, and hence there are a lot of possible setups to choose from. This guide is inspired by a guide written by Michal Karzynski[1], and will guide you through the steps of setting up the system using a combination of Gunicorn and Nginx.

C.1 Creating Your Users

Running a website as a user with root privileges or anything of the sort is far from recommended. Therefore you are advised to create a new user and a new usergroup. The names of both the group and the user can be chosen as you please, but the rest of the guide will stick to using a user called gentleidi and a group named webapps.

```
sudo mkdir -p /webapps/gentleidi
sudo groupadd --system webapps
sudo useradd --system --gid webapps --home /webapps/gentleidi gentleidi
sudo chown gentleidi:webapps /webapps/gentleidi/
```

Now you have a user named gentleidi which is a member of the usergroup webapps, and whose home directory is /webapps/gentleidi.

In addition to the user we just created, we need another user, specifically used to run the untrusted software submitted by the contestants. GentleIDI assumes that this user is named gentlemember. However, changing this value in the source is no complicated matter.

```
sudo useradd --system gentlemember
```

---

[1]http://michal.karzynski.pl/blog/2013/06/09/django-nginx-gunicorn-virtualenv-supervisor/

The system needs to be able to execute commands both as gentleidi and gentlemember. As the Web server runs as gentleidi we need to make sure that gentleidi can execute commands as gentlemember. Add the following line to your sudoers file.

```
gentleidi ALL=(gentlemember) NOPASSWD:ALL
```

If you don't know how to edit your sudoers, to open the sudoers file in a text editor simply type the following command:

```
sudo visudo
```

Now we've got two users, one capable of executing commands as the other. What we want to do now is to ensure that gentlemember is unable to communicate via network. This is done by applying two rather straightforward iptable rules.

```
sudo iptables -A OUTPUT -m owner --uid-owner gentlemember -j LOG
sudo iptables -A OUTPUT -m owner --uid-owner gentlemember -j REJECT
```

Though this will restrict the user's network access, be aware of software installed on your system which is capable of switching to another user.

C.2   Setting Up the Environment

Due to a lot of strict changes made in Python versions, a lot of libraries do not work across different versions of Python. This leaves Python in a situation where program A might need Python to be version X and program B might need python to be version Y. To solve this problem you can set up a virtual environment.

Virtual environments is a way of setting up separate python setups for different sets of programs.

What we want to do is to turn the home directory of the gentleidi user into a virtual environment.

```
sudo apt-get install python-virtualenv
sudo su gentleidi
virtualenv /webapps/gentleidi/env
```

Now that you've got a virtual environment you can start filling it with something useful, like the content of the project's Git repository.

```
cp -r /path/to/repo/IDIOpen/ /webapps/gentleidi/
```

Please note that you only need the wsgi folder from the repository, however, updating is a lot easier when all you've got to do is pull the latest version directly using Git. The downside is that you could possibly end up committing your production system configuration files etc. to the repo. However, we're going to assume that you will not be developing directly in your production system, and thereby avoid the hazard.

Before leaving this step, ensure that the files in /webapps/gentleidi has got the correct file permissions.

```
sudo chown -R gentleidi:webapps /webapps/gentleidi
```

C.3 Installing Required Packages

Now it's time to start making sure that you've got the packages you need to run GentleIDI.

```
sudo apt-get install git nginx libmysqlclient-dev python-dev
```

You might already have most of these packages, however, better safe than sorry.

The next thing you need to do before continuing is to log in as gentleidi and activate your newly created virtual environment.

```
sudo su gentleidi
source /webapps/gentleidi/env/bin/activate
```

Installing the required Python packages via PyPI is easily done. In the project root directory there's a file named requirements.txt. This file is simply a list of required packages, to install them simply execute the following:

```
pip install -r requirements.txt
```

C.4 Database

GentleIDI needs a database to store its data. This guide will show you how to setup GentleIDI with a MySQL database server, however, if you feel like using PostgreSQL, or even SQLite, then please do. Any database server supported by Django is supported by GentleIDI.

Naturally you don't need to install the database server on the same host as the Web server, that's what we'll do for now.

```
sudo apt-get install mysql-server
```

Now what we need to do is to create a database and a MySQL user that GentleIDI can use. During the install process you were required to set a root password for the MySQL-server. Login as root and perform the following commands:

```
CREATE USER gentledb'@'localhost' IDENTIFIED BY 'password';
GRANT ALL PRIVILEGES ON * \@. * TO 'newuser'@'localhost';
FLUSH PRIVILEGES;
CREATE DATABASE gentleidi CHARACTER SET uft8 COLLATE utf8_general_ci;
```

Remember to replace "gentledb" and "password" with a suitable username and password. Now you need to ensure that GentleIDI uses your newly created database. Edit the DATABASES entry in IDIOpen/wsgi/openshift/settings.py

```
if MYSQL:
    DATABASES = {
        'default': {
            'ENGINE'   : 'django.db.backends.mysql',
            'NAME'     : 'gentleidi',
            'USER'     : 'gentledb',
            'PASSWORD' : 'password',
            'HOST'     : 'localhost',
            'PORT'     : '3306',
```

In order to make sure that the database is working properly, log in as gentleidi, activate your environment and synchronize GentleIDI's database.

```
sudo su gentleidi
source /webapps/gentleidi/env/bin/activate
python /webapps/gentleidi/IDIOpen/wsgi/manage.py syncdb
python /webapps/gentleidi/IDIOpen/wsgi/manage.py migrate
```

If this command terminates properly, then your database should be good to go. In fact you should be able to run GentleIDI on a development server at this point. But first, you need to create an admin account. To do so, simply execute the following:

```
python /webapps/gentleidi/IDIOpen/wsgi/openshift/manage.py createsuperuser
```

To start the development server run:

```
python /webapps/gentleidi/IDIOpen/wsgi/openshift/manage.py runserver
```

You should now have a working website running on port 8000. However, you have no execution nodes available to evaluate submissions, and you're using Django's development server, which scales horribly.

C.5 Gunicorn

Now it's time to install replace the Django development server with a proper application server, Gunicorn. Remember to be logged in as gentleidi, and to activate your environment before proceeding.

```
pip install gunicorn
```

Now we need a script that launches Gunicorn and GentleIDI appropriately.

```
#!/bin/bash
# Name of the application
NAME=GentleIDI

DJANGODIR=/webapps/gentleidi/IDIOpen/wsgi/ # Django project directory
SOCKFILE=/webapps/gentleidi/run/gunicorn.sock # we will communicate using this unix socket
USER=gentleidi # the user to run as
GROUP=webapps # the group to run as

NUM_WORKERS=3  # how many worker processes should Gunicorn spawn
DJANGO_SETTINGS_MODULE=openshift.settings # which settings file should Django use
DJANGO_WSGI_MODULE=openshift.wsgi # WSGI module name

echo "Starting NAME as whoami"

# Activate the virtual environment
```

```
cd DJANGODIR
source /webapps/gentleidi/env/bin/activate
export DJANGO_SETTINGS_MODULE=$DJANGO_SETTINGS_MODULE
export PYTHONPATH=$DJANGODIR:$PYTHONPATH

# Create the run directory if it doesn't exist
RUNDIR=$(dirname $SOCKFILE)
test -d $RUNDIR {textbar}{textbar} mkdir -p $RUNDIR
# Start your Django Unicorn
# Programs meant to be run under supervisor should not daemonize themselves
#(do not use --daemon)

exec /webapps/gentleidi/env/bin/gunicorn ${DJANGO_WSGI_MODULE}:application \
 --name $NAME
 --workers $NUM_WORKERS
 --user=$USER --group=$GROUP
 --log-level=debug {textbackslash}
 --bind=unix:$SOCKFILE
```

Place the contents of the previous page in the following file:

/webapps/gentleidi/env/bin/gunicorn_start

Make sure that the script is executable:

```
sudo chmod u+x /webapps/gentleidi/env/bin/gunicorn_start
```

C.5.1 Nginx

As mentioned previously this setup relies on a combination of Gunicorn and Nginx. At this point gunicorn should be working properly, and it's time to setup Nginx.

If you have not already installed nginx, do so now:

```
sudo apt-get install nginx
```

Now you need to create an nginx configuration file for your Web site, in this case the file is called "gentleidi".

Store the content found below in the following file:

/etc/nginx/sites-available/gentleidi

```
upstream hello_app_server {
    server unix:/webapps/gentleidi/run/gunicorn.sock fail_timeout=0;
}
server {
    listen 80;
    servername example.com;
    client_max_body_size 4G;
    access_log /webapps/gentleidi/logs/nginx-access.log;
```

```
        error_log /webapps/gentleidi/logs/nginx-error.log;
        location /static/ {
            alias   /webapps/gentleidi/IDIOpen/wsgi/static/;
    }
location /media/ {
    alias   /webapps/gentleidi/IDIOpen/wsgi/media/;
}
location / {
    proxy_set_header X-Forwarded-For
    $proxy_add_x_forwarded_for;
        proxy_set_header Host $http_host;

        proxy_redirect off;
        if (!-f $request_filename) {
            proxy_pass http://hello_app_server;
            break;
        }
    }
# Error pages
error_page 500 502 503 504 /500.html;
    location = /500.html {
        root  /webapps/gentleidi/IDIOpen/wsgi/static/;
    }
}
#EOF
```

In this configuration Nginx is configured to log all accesses and errors. These log files need to be created with the following commands:

```
sudo su gentleidi
mkdir /webapps/gentleidi/logs
touch /webapps/gentleidi/logs/nginx-access.log
touch /webapps/gentleidi/logs/nginx-error
exit
```

All you need to do at this point is to enable the Nginx site. This is done simply by creating a symbolic link from the configuration file in sites-available to sites-enabled.

```
sudo ln -s /etc/nginx/sites-available/gentleidi
/etc/nginx/sites-enabled/
sudo rm /etc/nginx/sites-enabled/default
sudo service nginx restart
```

You should now have a working website. All that is left is making management a little easier, and adding some execution nodes.

C.6 Supervisor

Supervisor is a utility for defining and managing jobs. In this case we're going to define two jobs, one for managing the website, and another for managing an execution node.

You need to create two files to make this happen:

```
/etc/supervisor/conf.d/gentleidi.conf


[program:gentleidi]
command = /webapps/gentleidi/env/bin/gunicorn_start
user = gentleidi
stdout_logfile = /webapps/gentleidi/logs/gunicorn_supervisor.log
redirect_stderr = true
\#EOF

/etc/supervisor/conf.d/celery.conf
[program:celery]
command=/webapps/gentleidi/env/bin/celery worker -A openshift -l info
directory=/webapps/gentleidi/IDIOpen/wsgi
environment=PATH='/webapps/gentleidi/env/bin:%(ENV_PATH)s'
user=gentleidi
autostart=true
autorestart=true
redirect_stderr=True
#EOF
```

Create the log files that you've referenced.

```
mkdir /webapps/gentleidi/logs/
touch /webapps/gentleidi/logs/gunicorn_supervisor.log
```

Read the newly created configuration files.

```
sudo supervisorctl reread
sudo supervisorctl update
sudo supervisorctl restart all
```

C.7 Multiple Execution Nodes

The easiest way of setting up multiple execution nodes is to clone the setup on your Web server to other machines and then making minor changes. When setting up multiple execution nodes there are two changes that need to be made. The directory

```
/webapps/gentleidi/IDIOpen/wsgi/private/submissions
```

needs to be shared between all the execution nodes. How you decide to make this happen is up to you. However, SSHFS is possibly the easiest solution. Whatever way you decide to mount the directory on your execution nodes, make sure that multiple users are allowed to access it, e.g. the

"allow_other" option for SSHFS. You also need to make sure that all your execution nodes have access to the same database. Make sure that the settings.py is not set to localhost, but rather points to whatever host you decide to use as a database server. Some configuration of your database server might be needed in order for it to accept remote connections. MySQL servers need to change the bind-address property in the /etc/mysql/my.cnf to their actual IP, not localhost(127.0.0.1). You also need to change the grants for the MySQL user in such a way that it is allowed to connect remotely to the database.

Appendix D

Risk List

This appendix includes tables including all the risks we considered for this project. 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.

D.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 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 assignment	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.

D.2 Budget

97

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.

D.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 continuously 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 D.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 continuously 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.

D.4 Organizational

Description	ID	Pr	C	Tr	Preventative action	Remedial action
No person has responsibility for an assignment, 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 until 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.

D.5 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.

Table D.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.

D.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 E

Product Backlog

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-10	Admin	determine what statistics are stored/-collected by the system	overview and increased user experience
A-11	Admin	add/remove an execution node	scalability and safety in redundance
A-12	Admin	configure exection nodes with compiler profiles	system flexibility and optimality
A-13	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-03	Judge	upload solutions	assess case correctness to problem
J-04	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

ID	As a(n)	I want to be able to	So that
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-03	User	good response time on webpages	improved user experience
U-04	User	short user transactions (avoid click click click)	improved user experience
SU-01	Supervisor	document development process	overview group's progress

something

Appendix F

End of Sprint Structure

Meeting Agenda:

- Daily Scrum

  - What have you done since last time?
  - Have you had any obstacles?

- Three good/bad things

  - All team members take turns saying three good and three negative things about the previus sprint.
  - This is done without interruptions
  - If someone brought a cake, serve it here.

- Show what has been done

  - Every group member take turns showing what they have completed.
  - Discuss what has not been done

- Sprint end meetings

  - Effecitvely disucss what could have been done better

- Other

  - If someone want to talk about something this is the time.

- Sprint planning meeting

  - Select work that has to be done
    * The work is selected from the release backlog and put into to sprint backlog
  - Break these into smaller task/activities

- – Give each of these tastk/activities a priority
- – Give each of these task/activities a time approximation
- – Distribute on task/activite to each member.

About time estimation

- When voting for how long time a task/acivity will take, only powers of two are allowed:

  - – 2, 4, 8, 16, 32, 64 etc.
  - – 8 is characterized as a day

About prioritzing the task/activites

- Options when voting are 1, 2, 3 where 1 means LOW, 2 mean MEDIUM and 3 means HIGH.

General

- All members has a vote.

- If one estimates/prioritize different than the other members, he can, if he want to, tell the group why he estimated as he did. A new estimation will then take place.

F.1 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 publishedand insert content

F.1.1 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

F.1.2 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 ac-tivated account

F.1.3 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 attribute-sare reflected in other views
Input	Team, data, attributes
Output	A modified team entry in the database

Appendix G

User manual

G.1   Admin

In Fig G.1 you can see how the admin page looks. Here you will find most of the tools to successfully host a competition. This part of the user manual will contain information around the admin interface, and how to use it. The admin interface is a tool for creating, editing and deleting objects, objects in this sense is basicly database entries. When referring to objects later, this is what we mean by that.

G.1.1   Basic usage

Creating objects

To create a new object simply select what kind of object you would like to create, and click the add button next to its name. This button is shown in Figure G.2, and the list of available objects can be seen in on the front page of the admin interface(Figure G.1).

Editing objects

To edit existing objects you first have to locate the object you wish to edit. This is done by selecting the type you want in the main menu. This will open a list display of all objects in that category. If the list allows it, you can apply filters and search for the object you would like to edit. When you have found this object, simply click it and a editing form will appear. Make your changes and click the save button.

Deleting objects

When deleting objects you open the list display as explained in the previous section(G.1.1), select the item(s) you would like to delete, and select the delete action in the lower left corner. This will open a confirmation view, displaying any related objects that will cascade if you wish to continue. The edit form also includes a delete button.
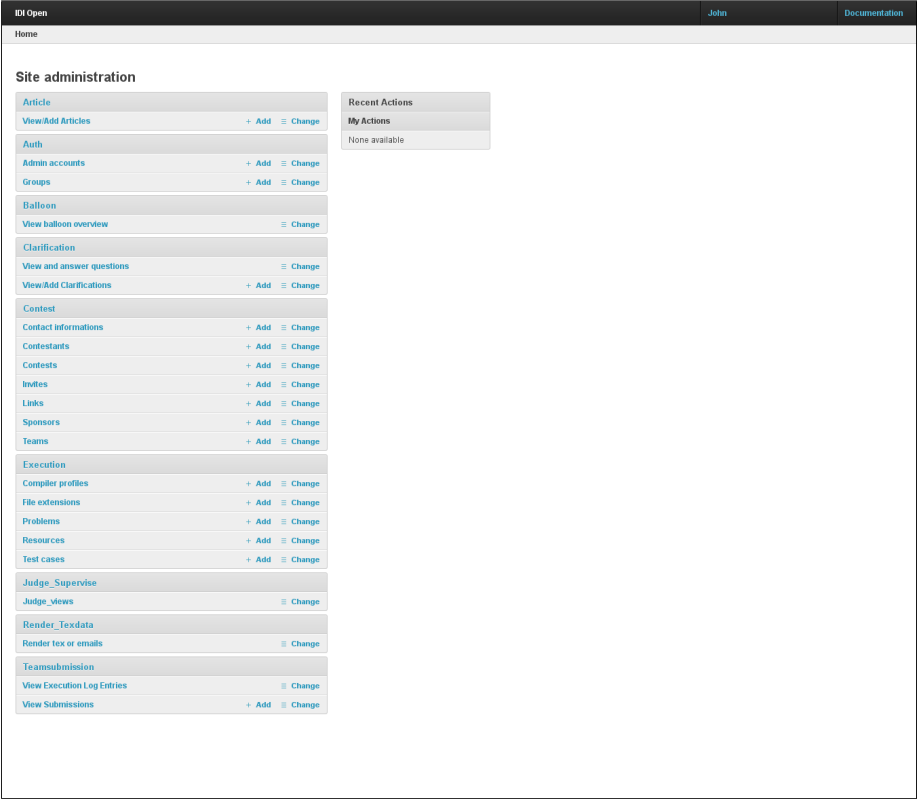
Figure G.1: Admin overview

Figure G.2: Create contest

Figure G.3: Contest create form

G.1.2 Creating a contest

To create a contest, first navigate to the contest part of the admin menu, and click the Add link next to Contests (Figure G.2)

Fill in the given form with the data relevant for your contest. If this is your first contest or a fresh system, you will also have to create contact information and links for the navigation.

To create new contact information, click the plus sign as highlighted in Figure G.3. The same applies for Links and Sponsors. The Penalty Constant is a variable used to calculate the score for each team. This value is a penalty in minutes for each wrong solution submitted. The Url field is a unique variable for each contest that sets the url location for the contest. E.g if the url variable is "test" then the contest will be located at *http://example.com/test/*.

The next four fields are time and data objects that determine when the contests is starting, ending, when it should be published, and when the registration should close.

Figure G.4: Create link

Figure G.5: Links selection and ordering

Create links

To create a link, click the plus sign in the links section like explained above for contact information. This will open a pop-up window as shown in Figure G.4.

These links will show up in the main navigation in the contest page. The text field will be the displayed text and the url will be the target url for the link. If this is an internal link to somewhere on the contest page, then select the checkbox indicating that. This will cause the contest url being appended before the defined url. You can also create a separator by checking the separator checkbox. This will create a separated field in the navigation, allowing you to group relevant links together.

It is also possible to sort the link by drag and drop to achieve the sorting you want. And to add or remove an existing link, you just have to check or uncheck the checkbox. These features are depicted in Figure G.5.

Create sponsor

To create a sponsor you click the plus icon in the sponsor section. This will open up a pop-up similar to the Links creation.

to link to, and click the magnifying glass to browse and upload images. This will open up the file browser in another pop-up window as you can see in Figure G.7. To upload a new image, click

Figure G.6: Create sponsor
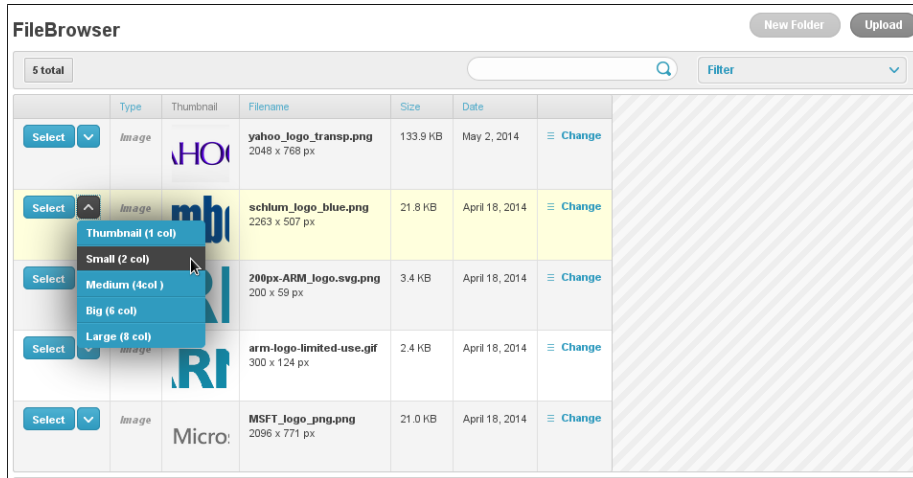
Figure G.7: Sponsor select and image upload

the Upload button in the top right corner. Upload your image and click Sponsor in the top left navigation to go back to the files. The system automatically scales images down to appropriate sizes. For sponsor images you should use Small images as shown in Figure G.7.

Create compiler profile

A compiler profile is a object with specifications on how a programming language should be compiled and executed. Each compiler profile includes a list of file extensions that should be allowed. These are added the same way as previously explained(By clicking the plus sign).

The compile and run fields both take a string of text used to compile and run the submission. The filename and basename will be inserted in the string where the tags {BASENAME} and {FILENAME} are present.

Create problems

Problems are the tasks that contestants will solve when the contest starts. The problem form includes a problem title, description with WYSIWYG editor support, and a text file. Each problem

Figure G.8: Compiler profile

is also linked to a single contest. For each problem you also need test cases, and a set of resource limits for the different compiler profiles. These models will be explained in the next sections.

Create test case

Each test case is linked to a problem, and there can be multiple test cases per problem. The most important parts of the test case is the input and output files. The input file will get parsed by the submissions, and the output file matched against the submission output. The test case also includes a short description and a description of the input and output. A screenshot of the test case creation can be seen in Figure G.9.

If the problem uses floating points or in any other way cannot be matched to a single output file, then you can set the test case to use a custom validator. This is a program that takes the output from the submission and the output file specified as input. The output from the validator should be 1 for correct answer and 0 for wrong answer.

Resource

Each problem requires a set of resource limitations for the different compiler profiles. THese limits sets the amount of resources the uploaded program is allowed to use, and if they exceed that amount they submission will be marked as a wrong answer. As shown in Figure G.11, the different fields are, max compile time, max program timeout, max memory, max processes, and max filesize. The resources depend on the problem and the programming language. C is a language that runs very fast and effective, and java requires a huge amount of memory to boot the Java Virtual Machine. So it can be quite difficult to set precise limits. It is also possible to give unlimited amounts by setting -1 as the value.

Figure G.9: Create problem

Figure G.10: Test case

Figure G.11: Resource

G.1.3 Creating articles

Articles is the main way of communicating with users of the system. The most recent and important news are placed on the front page. Other news can be found in the article list.

Articles consists mainly of a title and a body. The other fields are optional and configures how and where the article should be displayed. The text is written in a WYSIWYG editor and is integrated with the file browser described earlier. This gives the opportunity to make formatted news which can be easily edited again later. Article is also used for creating content that should not be considered as news. E.g. rules, history, FAQ. If this is wanted you can uncheck the Visible Article List checkbox. You also have the ability to create custom links for articles that should be available as an url. Articles can also be marked as urgent, what this means is that they will reside on the top of the frontpage and article list.

G.1.4 Create user

Create contestant

If a user is unable to create a user on their own, or if you wish to create it through the admin interface. Click the add button next to contestants in the main menu. This will bring up the form shown in Figure G.13. Simply fill in the information you want.

Create admin user

To create an admin user the process i very similar to create a contestant. You click the add button next to Admin accounts in the main menu. This will bring up a more detailed form than the contestant form, and is shown in Figure G.14. This form as the extra fields Active, Staff status, and superuser status. The staff status sets whether the user should have access to the admin panel,

Figure G.12: Add article

Figure G.13: Create contestant

Figure G.14: Admin account

Figure G.15: Promote user

and Superuser status gives the user all permissions. This form also includes a tools for setting group permissions and individual permissions.

Promote contestant to admin

If you would like to promote a regular user to a staff user, then all you have to do is edit the the user in question, and check the Staff status checkbox as shown in Figure G.15. The user will now be removed from the Contestant view, and moved to the Admin accounts view.

G.1.5 Balloon view

The balloon view covers the functionality meant for balloon functionaries. Clicking on View Balloon Overview, located in the section Balloon, will take you to the balloon view.

Figure G.16: Balloon view

The first choice you are met with, is which contest you want to show submissions from. The drop down menu lets you pick any of the already created contests. The balloon view consists of two tables. The first table is named, not given balloon, and contains the onsite teams that are awaiting a balloon for a correct submission. The table consists of five columns covering the essential information a balloon functionary needs to know. The table is automatically sorted on team and time uploaded, but this can be changed by the user. All the columns in the table are sortable.

The rightmost column consists of checkboxes for each row in the table. We have given the balloon functionaries two different possibilities to register that a team has been given a balloon. The balloon functionary can either press the corresponding checkbox for the team, in the rightmost column, or drag the respective row to the table beneath. This will move the row from the first table, to the second table.

The second table is named, Given Balloon, and contains teams that have received a balloon for a correct submission. This table has the same functionality as the first table. The rightmost column, Undo, consists of checkboxes. If you press a checkbox, or drag the row to the other table, it will cause the row to change tables. This gives the balloon functionaries total control, in case of a misclick.

The last function the balloon table has is that it will tell you when new submissions has arrived. The blue "Refresh" text located at the top left corner, will change to "New submission", when a new submission arrives. Pressing the text will refresh the page, and new submissions will show up in the first table, not given balloon.

G.1.6 Judge view

In the section called Judge_Supervise there's a link labeled Judge_views. This takes you to the site primarily intended for inspecting the submissions in the system.

This is the starting point for staff users wanting to inspect a submission. Starting at the top there's a drop down menu enabling you to inspect submissions for a specific contest, labeled "Choose contest". Further down there's another drop down menu enabling you to take a closer look at a single team, we will take a closer look at this in the next section.

Next up is two tables, one listing teams that are said to be onsite, the other listing offsite teams. Other than the team names, there are two columns in these tables, "Previously solved problems", and "Failed attempts". You can sort the table by clicking the table header you want to sort by.

Figure G.17: Judge view

Figure G.18: Judge view for a team

This enables you to easily find the teams that are struggling the most, sort the lists by the "Failed attempts" column. Clicking the entries in these columns serve the same purpose as selecting a team from the dropdown menu mentioned previously.

Another table is present below the team tables. This is a table of the problems in the system. The columns in this table are "Problem", "Successful attempts", "Failed attempts", "Total", and "Success/Total ratio". The "Problem" column states the names of the given problem, "Successful attempts" is the number of submission that actually solve the problem, and "Failed attempts" is the opposite. "Total" is the total number of attempts at solving the problem that has been submitted to the system. "Success/Total ratio" is the ratio of successful submissions for the given problem. As with the other tables, this one can be sorted by any column.

At the bottom of the site there is a highscore list.

Finally, it's time to take a closer look at how you should appropriately inspect a team's submissions. Select the team in the drop down menu, or by clicking it in the list of teams. Doing so will take you to the following site.

Initially you will see a simple table listing the submissions submitted by the specified team. The columns are the following: "Date uploaded", "Problem", and "retval". The "Date uploaded" column lists the date AND time that the submission was uploaded. "Problem" is simply which problem the team intended their submission to solve. The column in need of further explanation is "retval", this is the exit code of the submission execution, if this value is nonzero an error of some sort has occurred.

However, this is more than a simple listing. By clicking a row you expand it, revealing four buttons, "Command executed", "Source code", "Command output", and "Stderr". By clicking these the row will expand further and reveal more information.

"Command issued" is the command that the system issued to execute the submission, e.g. "java Classname" for java programs etc. "Source code" is the source code submitted by the team. "Command output" is the output produced by the execution of the submitted source code. "Stderr" is the error messages printed during execution. It is worth taking note of the fact that the program used for measuring submission execution time will print the time to stderr, this line will look something like "0.02\n0.12" and can be found at the bottom of the "Stderr".

G.2 Contestant

The users competing in the competition are considered contestants. In this part of the guide we assume that appropriate links have been added to the sidebar by admins. The only links that are in the sidebar by default are the ones for registering a user and registering a team.

G.2.1 Register user

For user's who are not logged in there will always be a "Register user" button in the sidebar. This button will take you to a form for submitting user information. All of the fields are required, except for the nickname, which is optional. The email field is used to send out user activation emails and as such it is crucial that the email address is correct. The "Year of study" dropdown list contains 6 different options, the integers from 1 to 5, and "Pro". Students are to select the integer corresponding to how far they've come in their study program, working professionals are to select the "Pro" option. The final option is gender, which can be left as "Unspecified", or set to male or female.

G.2.2 Edit user

On your user profile there are three buttons for editing your user, "Change email", "Change password", and "Change user information". The two first are self-explanatory, they enable you to change your email and password, in other words your login information. The "Change user information" option lets you edit the rest of the fields you submitted during the registration process, first name, last name, nickname, year of study, and gender. It is worth noting that changing your email will send a new activation email to the newly registered email address, if the new address has not been activated within a given time limit the change will not occur.

Figure G.19: User registration

Figure G.20: Edit user

Figure G.21: Create team

G.2.3 Create team

If you've got an account, but you're not member of any team, there will be a link available to you in the sidebar labeled "Register team". This button will take you to a form for registering new teams.

The form requires that you give your team a name, and that you give some information as to your location during the event. If you are planning on being on-site during the contest all you need to do is select the "yes" radio button labeled onsite. If you are not planning on being on-site, you need to fill out the field labeled "Offsite location" with the location you are intended to stay in during the contest.

In addition to this basic information you can invite team members while you are at it. Just type in the email address of your to-be team members in the fields "Second Team Member" and "Third Team Member". They don't have to be registered user's in the system to get the invite, they will be notified of your invitation and will be able to accept the invite upon registering their account.

Figure G.22: Edit team information

G.2.4 Edit team

Edit basic information

On the team profile page there is a button labeled "Edit team", if you press this button it will take you to a form for editing your team. You need to be the team leader for this option to be available to you.

As with the editing of your user account, the same fields that were available to you during registration of your team, are also available when editing it. You can change your team name, and location. When you are done making changes just click the "Update team" button and your changes will be put into effect.

Since the majority of team management is up to the team leader alone, one might want to transfer the role of being team leader to another team member. This can be done by selecting another team member in the dropdown menu labeled "Leader".

Invite user to team/join team

If you for some reason did not know who were to be on your team when you created your team you can add them later on, given that you are the team leader. This is done on your team profile page, the link is found in the sidebar.

Inviting a member is done by writing their email address in the "Email" field and pressing the "Add new member" button. As mentioned in the Create team section, your team members don't need to be registered user's at the time of you inviting them. They will be notified, and the invite will be stored in the system and accessible to them when they finally decide to create a user.

Figure G.23: Invite team members

Figure G.24: Remove user from team

Figure G.25: Problem overview

Remove user from team

If you for some reason decide to evict a team member this can also be done on your team profile page. The team profile lists all the team members, to evict one, simply click the trashcan on the user's entry. All user's can decide to leave a team as they please, however, to evict other users in this manner you need to be the team leader.

G.2.5 Solve problems

Assuming your team is setup and the contest has started, it's time to start actually competing. You get points for solving problems, in other words, submitting source code that gives correct output given the correct input.

To find the list of the problems in the contest problem set, just click the "Contest page" button in the sidebar. This will take you to a page looking like the one in the figure below.

The contest that was used for creating this figure contained two problems, Counting Digits, and Election. You can see that the user logged in has already made an attempt at solving Counting Digits, however it failed due to incorrect output.

By clicking one of the rows you are taken to the site where you can submit source code in an attempt at solving problems.

When submitting source code you need to specify what compiler profile is suitable for the source you are submitting. If you're submitting python code, select "python" in the dropdown menu labeled "1. Pick a compiler profile". Then upload your source by clicking the "Choose file" button, and then press the "Submit File" button. Your source will then be submitted and evaluated. The site will automatically update upon the evaluation of your submission being completed.

Figure G.26: Submit problems

G.2.6 Clarification

During a contest contestants might need help understanding a certain part of the system or a problem description. In order for them not having to wander around looking for a functionary, they can post questions that can be answered by the staff users.

Post a question

There is a link in the sidebar taking you to the site where you can post questions.

Posting a question is quite straight forward. Enter a title for your question in the title field, and put the body of your question in the largest textbox. When you are done writing and ready to post it, press the "Send question" button. Now the question can be read and answered by the staff users.

Review replies

Questions posted through the clarification system are not visible for anyone except the staff users, that is, unless they have been answered by the staff. When a question has been answered it is visible to all contestants.

As usual the link taking you to the list of answered questions can be found in the sidebar. Here you can see that a staff user has answered the question posted.

G.2.7 Highscore list

Top five

On just about every part of the site there is a small highscore list visible at the top right corner of the screen. This is simply a list of the five teams with the best score.

Full highscore

The complete highscore list can be found by clicking the link labeled "All" in the top five high score list, or by clicking the link labeled "Highscore" in the sidebar. This is not just a list but a table which can be sorted by different columns.

The buttons labeled "student" and "pro" lets you filter the highscore. If you press the "student" button, all teams containing professional members will be removed from the list. The "pro" button does the opposite. To sort the table by a specific column, click the column label, to reverse the ordering, click it again. The columns labeled by single characters correspond to different problems.

Figure G.27: Post a question

Figure G.28: Clarification response

Figure G.29: Full highscore

Appendix H

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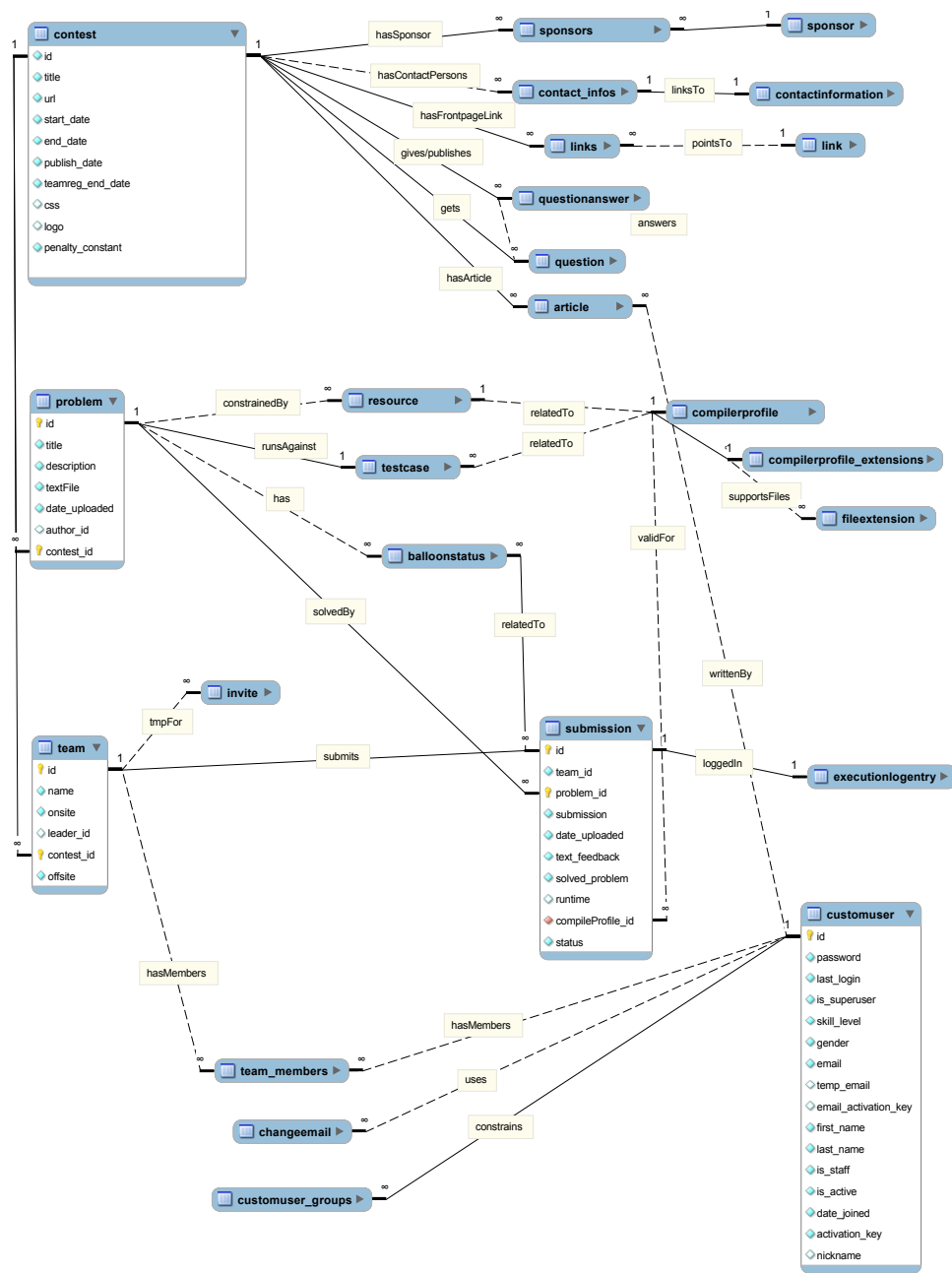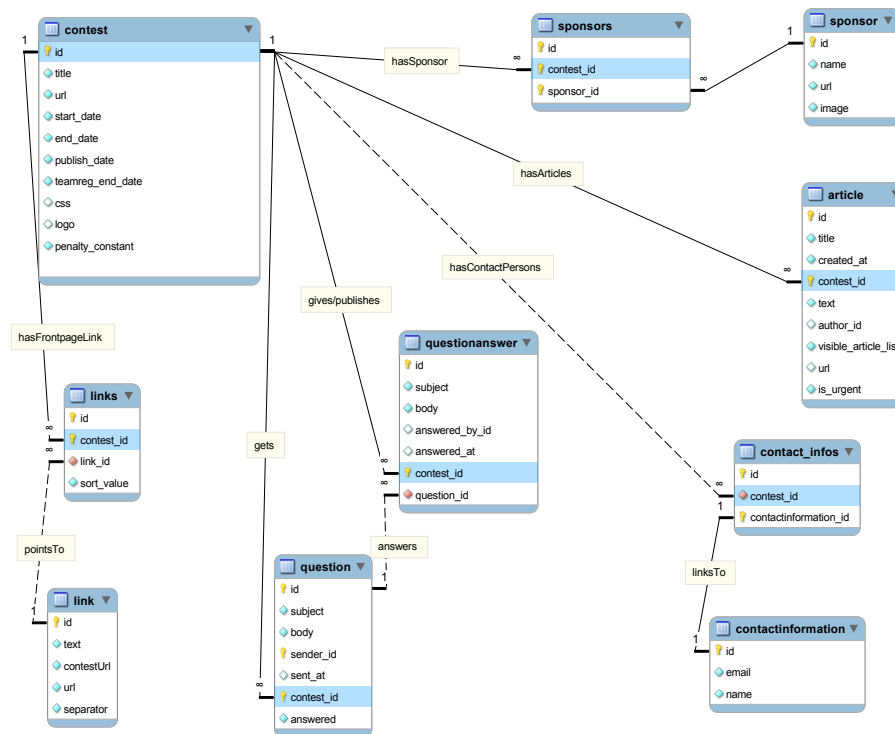
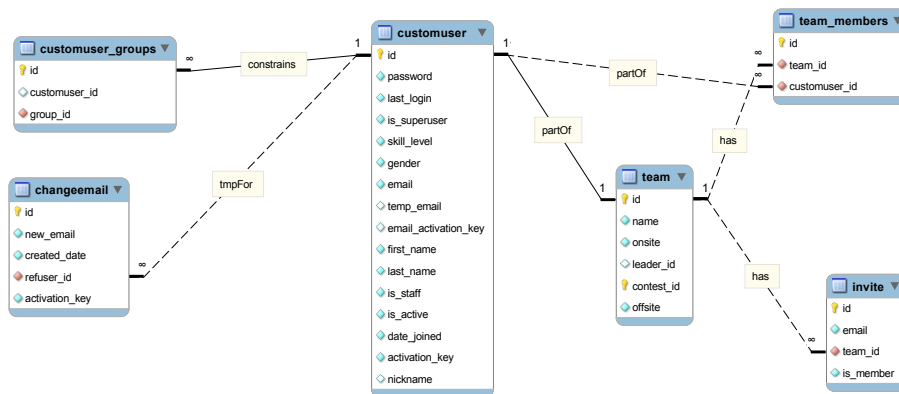Figure H.2: ER-diagram for the models used for the contest.

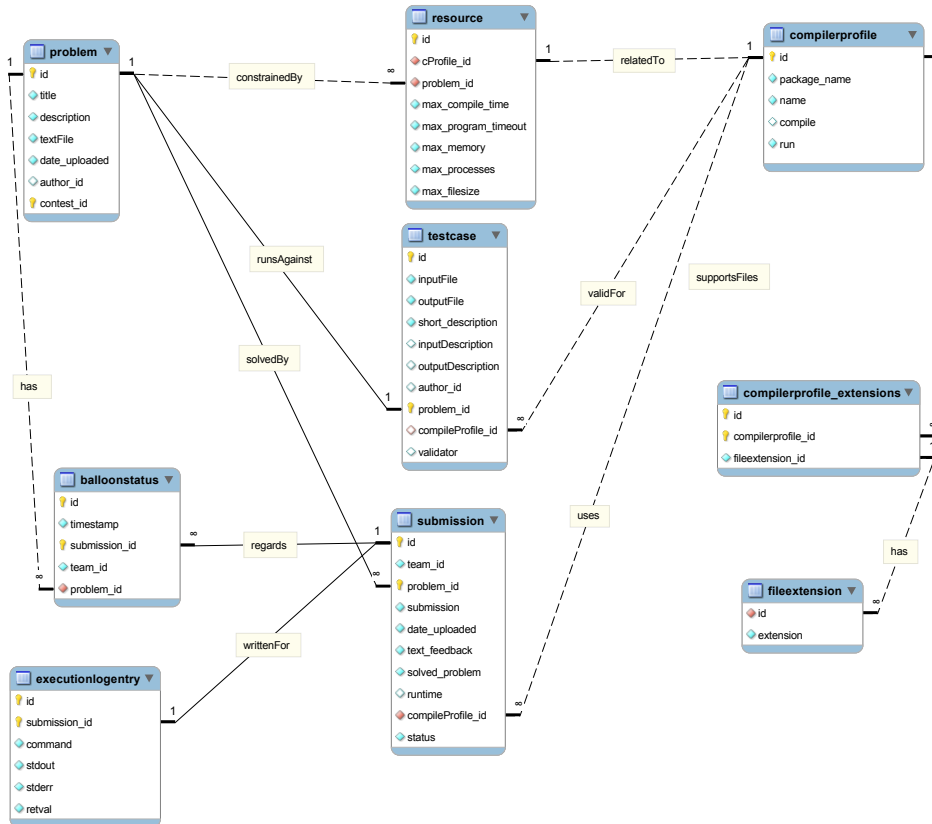Figure H.3: ER-diagram for the models used for the user registration.

135

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