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

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 1.4.1. We also provide a structured overview of how we organized our time.

1.1 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.

1.2 Development Method (Scrum)

Scrum focuses on having daily meetings, and constantly adjusting to changes by iterative development. This makes it easier to predict and to adjust for problems that may occur. It was hard to predict what would happen in our project, therefore our sprints were short, lasting at most two weeks. The transition between two sprints was done during a prolonged meeting on Wednesdays. During this meeting we evaluated the latest sprint and planned the upcoming one. Every team member were requested to say three good things and three bad things regarding the last sprint. This was followed by a discussion of how to plan the next sprint better. Lastly we showed what had been completed, to the other members of the group, before setting up the next sprint. Scrum also

focuses on having finished versions of the systems on each iteration, and to finish all packages in the given iteration. In order to take advantage of the best in everyone's abilities we worked in pairs where this was efficient. Working in pairs is common in agile development. This was to improve code quality and reduce errors.

1.3 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.*

1.4 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.

1.4.1 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 1.1 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 1.1: 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 1.2

Table 1.2: 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.

1.4.2 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 **??**.

Table 1.3: Milestone name and deadlines

Date	09.02.	09.03.	19.03.	19.03.	11.04.	26.04.	03.05.	30.05.
Week	3	6	8	8	11	14	15	18

Name	Preliminary report	Mid-semester report	First release	Presentation	Beta release	IDI Open test event	IDI Open	Final Report
ID	M-01	M-02	M-03	M-04	M-05	M-06	M-07	M-08

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.

1.4.3 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.

1.4.4 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 1.4 shows our first initial draft of sprints.

Table 1.4: 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 1.5 shows the actual sprints and work done. The hours are for each person, during that sprint.

Table 1.5: 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 2

Testplan

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

2.1 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.

2.1.1 Unit Testing

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

2.1.2 Integration Testing

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

2.1.3 System Testing

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

2.1.4 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.

2.2 Testing Coverage

We wanted to provide complete test coverage, but we did not have the time. Thus, we needed to prioritize what components of the system were most prone to error, and most important to test. The following were our software assurance objectives:

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

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

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

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

Since we were developing a website that would feature many users, developer testing alone could never simulate peak values for system demand. Therefore we have relied on load testing. Here, we gave 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.

Our lacking experience in web development meant that it was hard for us to understand what components could go cause errors. Wikipedia holds a large list of categories that could be tested[1],

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

but we avoided many of them, as it would take to long for us to gain a structural way to test these areas, combined with the lacking experience.

2.3 Our approach to testing

2.3.1 Unit Testing

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

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

2.3.2 Integration Testing

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

2.3.3 System Testing

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

2.3.4 Acceptance Testing

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

2.3.5 Integration Test

Each test has a unique identifier, name, pre/post-conditions and corresponding input and output. An example is given in table 2.1.

Table 2.1: Integration test for adding a sponsor

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

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

We formally agreed on what modules our system was made out of and their interfaces. Figure 2.1 shows our view on the system as per milestone M-03. In figure **??**, we have replaced some default UML symbols and replaced them with the equivalent UML stereotype. The explanations are given in table 2.2. The integration tests we did make are given in appendix G.

Table 2.2: Symbiology for our UML component diagram

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

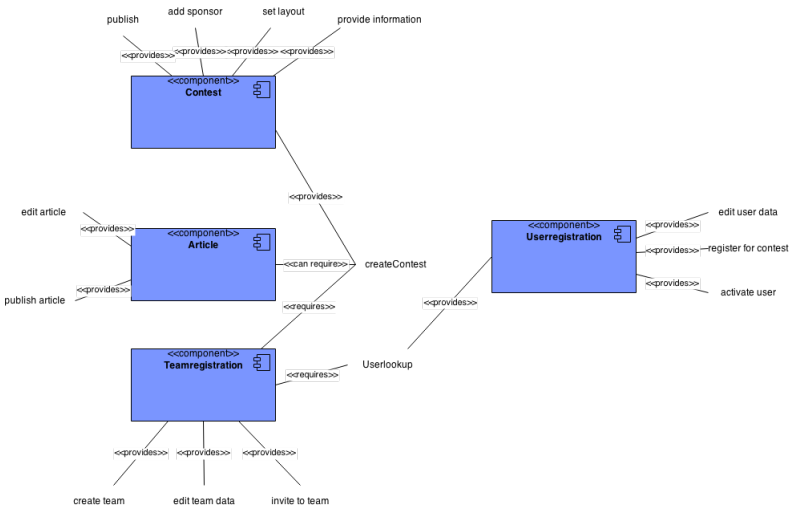


Figure 2.1: Diagram from milestone M-03. Each interface connection, especially "createContest" has been tested

2.4 System Test

Our system tests cover all the functional requirements. All tests are written as successive cases. This means that the tests do not cover scenarios for how the system should respond when a user performs an error or another external fault occurs. The complete listing is in table 2.4.

Table 2.3: System test

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

Table 2.3 – continued from previous page

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

2.5 Non-functional testing

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

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

Table 2.4: System tests

Case	Input	ID	Expected Result	Pass/Fail
Adding 500 contestants	500 users	NF-04	Ability to add yet another	PASS
Adding 200 teams	200 teams	NF-05	Ability to add yet another	PASS
Adding 20 judges	20 judges	NF-06	Ability to add yet another	PASS
Adding more than one admin	> 1 admin	NF-07	Ability to add yet another	PASS
Upload a solution which is less than 50kB	Solution $>$ 50kB	NF-08	Successful delivery	PASS
Upload a solution which is greater than 50kB	Solution $>$ 50kB	NF-08	Error message	PASS
Gather some test persons not familiar with the system and have them use the system as a contestant	System	NF-09	They should be familiar with the system after 5 minutes	FAILED
Gather some test persons not familiar with the system and have them use the system as a judge	System	NF-11	They should be familiar with the system after 10 minutes	PASS
Gather some test persons not familiar with the system and have them use the system as an admin	System	NF-10	They should be familiar with the system after 15 minutes	PASS
Page responsiveness with at least 5 RPS	HTTP GET and POST to all pages	NF-01	Response-time $<$ 100 ms	FAIL
Page responsiveness with at least 10 RPS	HTTP GET and POST to all pages	NF-01	Response-time $<$ 200 ms	FAIL

In table 2.4 it can be seen that not all the tests passed. This is elaborated on in section **??**

2.6 Risk and Dependencies

We did not test whether or not the privileged users of the system made any errors. They were responsible for uploading solutions and content on the web site.

The majority of our testing has been inspection-based. This has been considered time efficient for us. As we have developed the entire system from scratch, and worked with it over a longer period of time, we have had good knowledge of the system. Thus, inspection-based testing has been largely effective. The problem is that there is no way to formally agree on what components have been tested, or to what extent. Additionally, future maintainers are much more likely to

make errors as they do not know what components are connected, or what kind of tests should be executed.

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

Chapter 3

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.

3.1 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.

3.2 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.

3.3 Risk Identification

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

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

3.4 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.

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

D.2 Budget

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

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 continously see our work progress	Have extraordinary meetings with supervisor and the customer to discuss the further development of the project. Be apologetic towards the customer, and come up with a new plan, that the customer is satisfied with.
Sprint-estimations are off	S-03	9	5	45	The whole group participate in planning a sprint, and estimating each task	Re-adjust our estimations in the next sprint, and in that way learn from our mistakes.

Table 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 continously see our work progress	Meetings with supervisor and customer, agree upon a new deadline, and increase the workload the following days to we meet the deadline.
Need for extra technology / features that requires training to use	S-05	3	6	18	We use extensive frameworks who has a lot of documentation, which makes it easier to learn.	Adjust the WBS and our sprints so we take into account that we need more time to learn new technology. Focus on this in the coming sprint planning.

D.4 Organizational

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

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

Table E.1 – continued from previous page

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.

Appendix G

Integration tests

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

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

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

ID	IT-04
Interface name	Provide information
Pre-condtion	Contest is created
Post-condition	Information pages

Input	Image, URL
Output	Availablee articles

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

G.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

G.2 Userregistration

ID	IT-09
Interface name	Create user

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

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

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

G.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

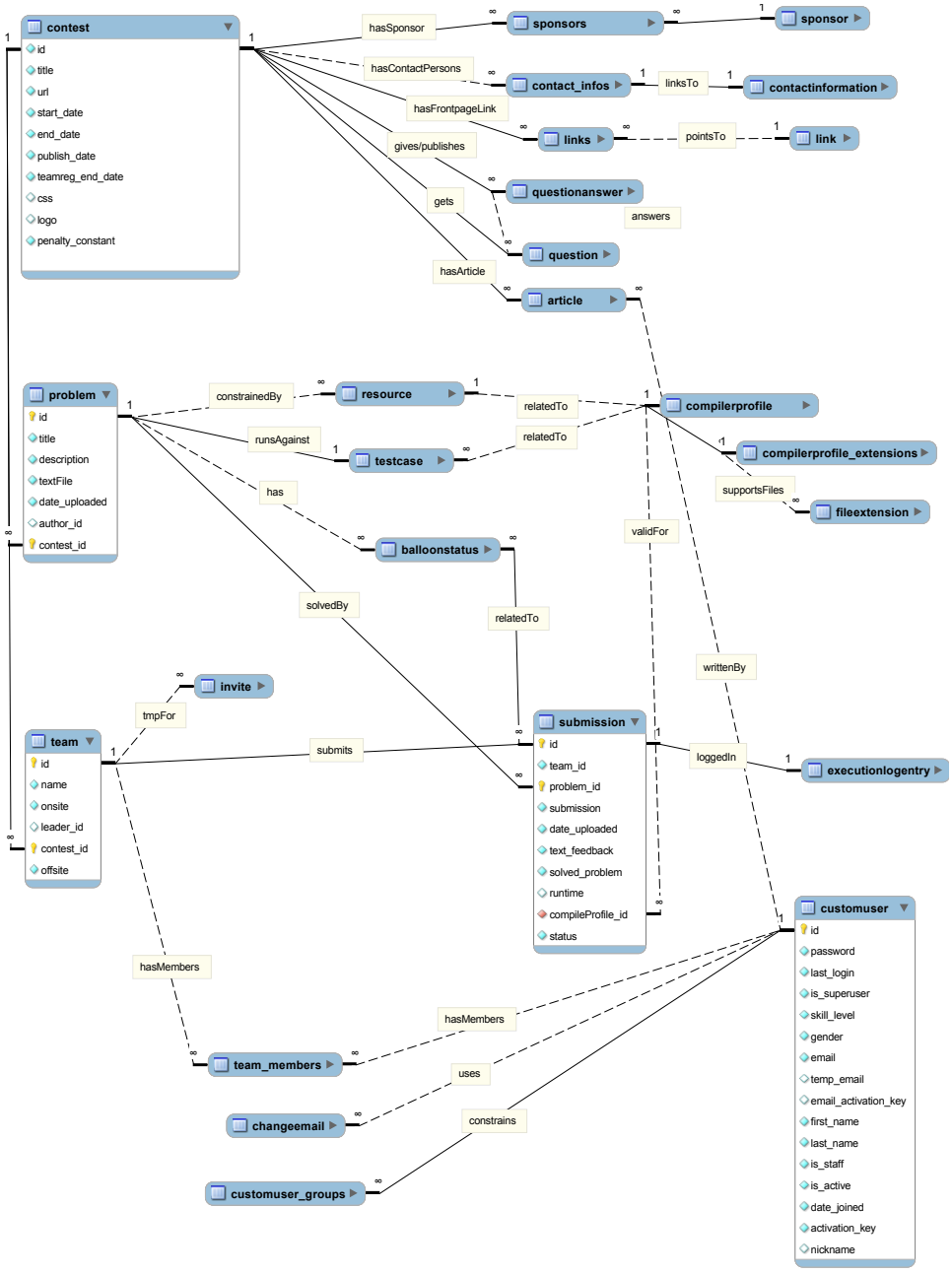
ID	IT-14
Interface name	Edit team data

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 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.


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

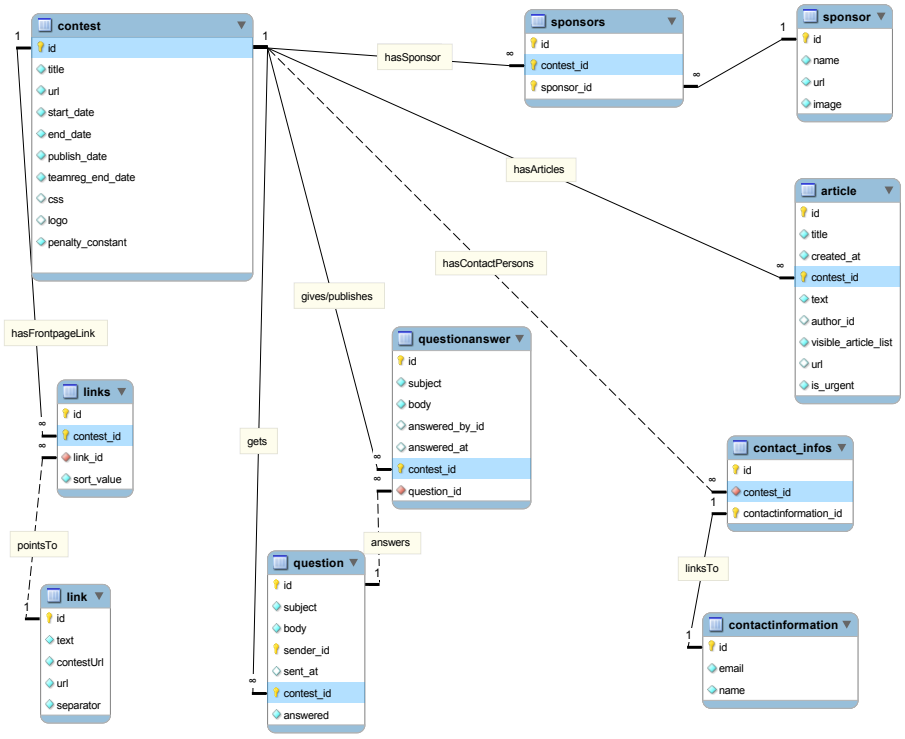


Figure H.2: ER-diagram for the models used for the contest.

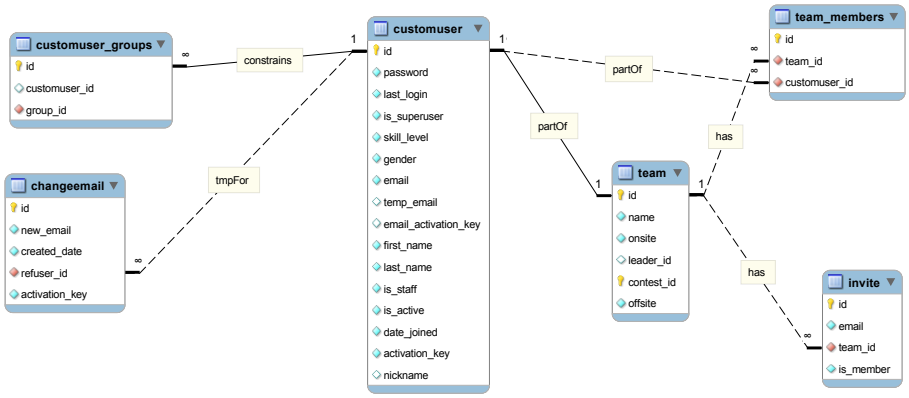


Figure H.3: ER-diagram for the models used for the user registration.

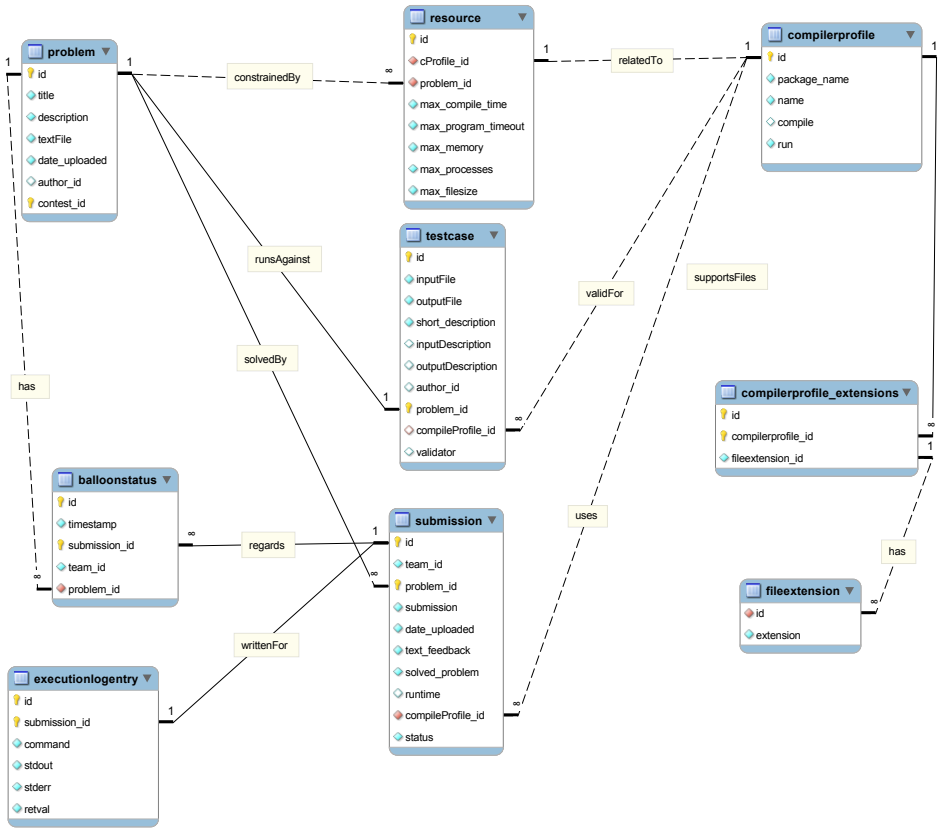


Figure H.4: ER-diagram for the models used for the submission.