

# 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

<b>1</b>	<b>Task Description and Overview</b>	<b>2</b>
1.1	Task Description and Overview . . . . .	2
1.2	Assignment . . . . .	2
1.3	GentleIDI . . . . .	2
1.4	Assumptions and Constraints . . . . .	3
1.5	Roles and Their Definitions . . . . .	4
	1.5.1 Usergroups . . . . .	4
	1.5.2 Service-providing Units . . . . .	4
1.6	UML Use Cases . . . . .	5

# Chapter 1

## Task Description and Overview

### 1.1 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 1.4. Reading this chapter will be important to understand the rest of this report.

### 1.2 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.

### 1.3 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.

## 1.4 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 1.1: Assumptions 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 always prioritize software re-use.

## 1.5 Roles and Their Definitions

### 1.5.1 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 1.2 shows the different roles and their available actions. Further elaborations on each group will also be given in later sections, but table 1.2 should suffice for an overview.

Table 1.2: 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.

### 1.5.2 Service-providing Units

Another way of viewing the task description in section 1.1, 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 1.3: Service-providing Units


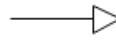
Entity	Features	Protocol
--------	----------	----------

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 web-server.	AMQP
Database	The storage unit for all user-data and logs.	SQL

## 1.6 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 1.4 describes the semantics of objects used in the diagram, which should be equivalent to the UML 2.0 standard.

Table 1.4: 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 figure 1.1, 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.



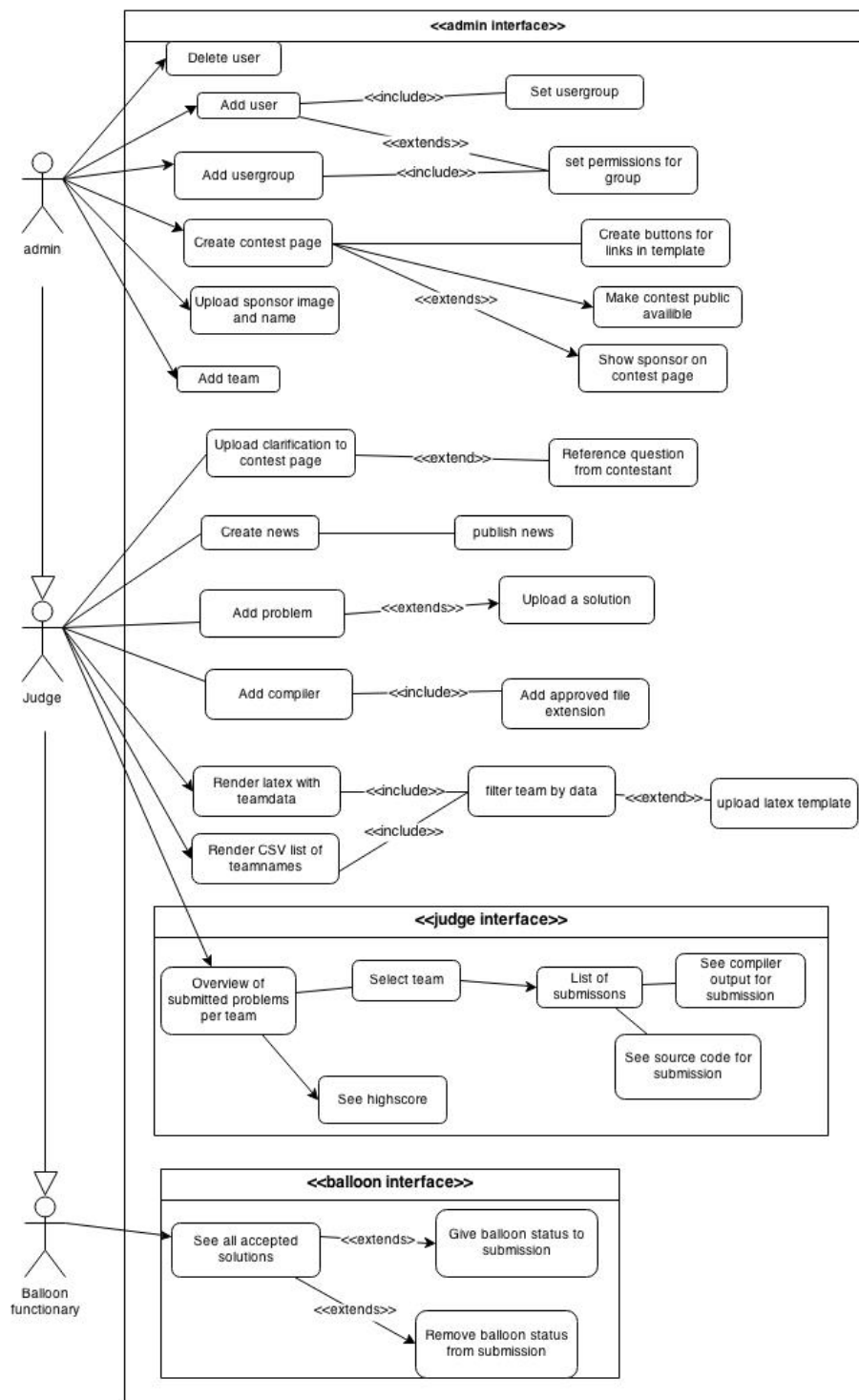


Figure 1.1: FIGURECAPTION