

This is the Title of my Thesis

Your Name

December 2012

PROJECT THESIS

Department of Production and Quality Engineering
Norwegian University of Science and Technology

Supervisor 1: Professor Ask Burlefot

Supervisor 2: Professor Fingal Olsson

Contents

0.1 Foreword

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

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

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

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

0.2 Preface

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

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

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

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

Chapter 1

Introduction

1.1 About the course

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

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

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

1.2 The Group

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

Anders Sildnes

Throughout his BSc, Anders has been taking courses related to algorithms and program security. Apart from his studies, he is developing for Engineers without

Borders NTNU and spending time with open-source projects and other Linux tools.

Eirik Fosse

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

Filip Fjuk Egge

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

Haakon Konrad William Aaseb

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

Hkon Gimnes Kaurel

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

Tino Lazreg

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

1.3 About the Customer

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

1.4 About the Contest

IDI Open is a programming contest where teams of up to three people meet and solve programming problems of various difficulty. The contest lasts five hours, and the objective is to solve as many problems as possible. The contest is open for all types of programmers, from students of all grades to professors and other professionals from the IT industry. Various prizes are given to the teams based on their performance. The main objective, however, is to solve the highest amount of problems in the shortest amount of time.

There are usually 8-12 problems in a contest. They are of varying difficulty, ranging from easy to hard. To make the competition fun for everyone, there are typically some problems that are easy enough even for novice programmers to handle.

1.5 Stakeholders

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

1.5.1 Course

Supervisor

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

Examiner

The examiner(s) is responsible for determining our final grade. Opposed to most of the other stakeholders, we've got little to no on-going communication with the examiner. However, the examiner has got access to all the documents the supervisor has got access to.

1.5.2 Product

IDI Open

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

Judges

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

Developers

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

Maintainers

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

Sponsors

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

Contestants

The actions of contestants are all through our software; our product will be their medium to take part in IDIOpen. Reliability and usability is key to keep the contestants happy. For example system flaws affecting the contest results will not be tolerated. Our customers also received feedback from the contestants. Thus, how satisfied the contestants are impacts the developer's evaluation.

1.5.3 Goals

The goal of the project is to upgrade and improve the existing system used in IDI Open. The project is not a prototype wanted by a consulting firm, but a system that will be used by actual end users. This gave us inspiration to do the best we could, and to give the customer something both we and they could be proud of for many years to come. And if the product is good enough it would hopefully also be used in larger programming competitions, maybe even international ones.

Chapter 2

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, you should be able to understand how our solution, “gentleIDI”, can be used. Also, reading this chapter will be important to understand the other chapters. Our process to develop GentleIDI has had some assumptions and constraints, which are discussed in section X.X.

2.1 Assignment

According to our customers, “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. In replacing the old solution, we did not need to implement features that were not available from the old system.

The features of the old solution, in a nutshell, are given below. A more detailed overview will be given in section X.X.

- User login and registration
- Website to inform about the contest to the public
- Team-registration and scoring
- Ability for users to upload code, that would be compiled and executed by the server

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 uncluttered way, such that other developers could easily understand the new solution.

2.1.1 Assumptions and Constraints

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

Do note that the implications in table X.X 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.

2.2 Roles and their definitions

2.2.1 User-groups

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 manually given elevated permissions. Table X.X shows the different roles and their available actions. Further elaborations on each group will also be given in later sections, but table X.X should suffice for an overview.

2.2.2 Service-providing Entities

Another way of viewing the task description in section X.X, 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.

The only entity that end-users interact with is the webserver. This is done through HTTP messages, which, if necessary, are relayed to the execution nodes and/or database.

2.2.3 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.

AssumptionConstraint	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 customers, IDI Open, are students near/at Glshaugen		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 <i>everyone</i> . Ideally, any person should be comfortable with the user 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 solution must run on Linux-computer	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

Table 2.1: Usergroup overview

Role	Description
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.
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 tasks to be solved, solutions to those tasks, upload incorrect answers (eg. answers that will provide penalty).
Functionary	Privileged. Functionaries hand out balloons when a team has solved a task. To determine what team will be given a balloon, the functionaries have their own interface with a team overview.

Table 2.2: Computer entities

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

Table 2.3: Use-case notation

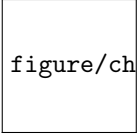
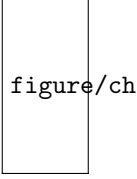

	<p>figure/chapter02-img1.jpg</p>	<p>Use case actor. Represents a user group</p>
	<p>figure/chapter02-img2.png</p>	<p>UML generalization arrow. Used to indicate inheritance. The arrow's source, i.e. tail, represents the entity that inherits from where the arrow points to.</p>
<p><<include>></p>		<p>UML stereotype to represent a mandatory extension to a work-flow.</p>
<p><<extend>></p>		<p>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.</p>

Figure X.X and X.X models the available workflows and actions for each category of users. Table X.X describes the semantics of objects used in the diagram, which should be equivalent to the UML 2.0 standard.

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 to the supervisor what we were trying to accomplish.

figure/chapter02-img3.jpg



figure/chapter02-img4.jpg

As seen in figure X.X, 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.

2.3 Design

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

Design process

The user interface provided by the previous IDI Open system consisted of a simple web interface for reading news items, registering teams for contests, and delivering submissions. GentleIDI is intended to provide more functionality through its web interface, including but not limited to change email(requirement FC-02), supervisor(requirement FJ-11) and user management (requirements FC-01, FC-03 and FC-04). As a consequence we had two options available: reusing and extending the existing interface design, or creating our own design from scratch.

Fig 7.1 We chose to create our own design from scratch, while still trying to keep a similar placement of elements from the previous design. The customer expressed concern regarding how contestants would react to the transition from the old interface to the new one. With this in mind we started to create mockups modelling core elements of the website. Our initial drafts consisted of simple rearrangements of elements found in the old web interface.

Beyond our three initial mockups we tried a couple of out of the box approaches to our designs, but none of them met our standard and was rejected for either being too time-consuming to implement or too far from what our customer wanted. We had a meeting with our customer, where we showed our mockups, and what are thoughts on design had been so far. We wanted to make sure that the customers was on the same page as us, and that we were not moving beyond the scope of the project. Our customers wasn't very focused on the design aspect, but one demand they had was that they wanted the new site to have the same structure as the old one. One example of what this means is that the customer wanted us to keep the menu on the left-side as you can see that the old system has in Fig 7.1. We agreed, because geused to a new website can take time, so keeping the structure similar would ease the transition for our users. With this in mind we decided to go for one of our initial mockups, the rightmost one in Fig 7.2, because it had the same structure as the old page, and we personally favoured that design. As a result, most of the elements found in the old interface can be found in the new one, and the transition between using the two is reduced to a minimum.

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

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




Figure 2.1: FIGURECAPTION

worked more or less automatically.

The final page looked like this:

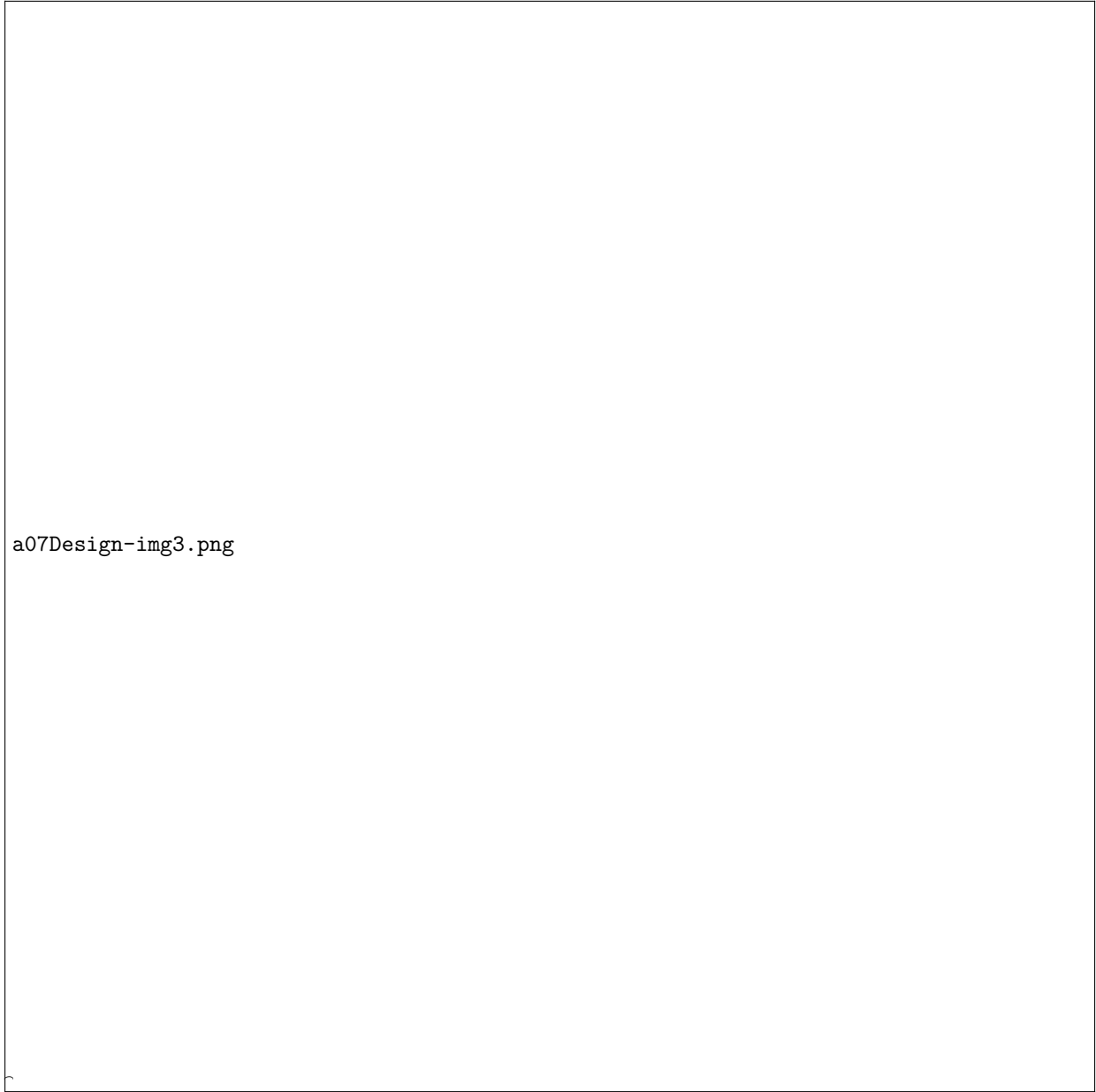
The “black” frame was in our initial page coloured blue, but was changed one week before M-07, idiopen [REMARK: may be altered].This illustrates



a07Design-img2.png

Figure 2.2: FIGURECAPTION

the strongest functionality of the design, namely customization. It is possible, by only uploading a new CSS file, to change the whole feel of the website and give every contest its own theme. The change on IDI OPEN 14, from blue to black, was done as a consequence of a logo change by Richard Eide, one of IDI



a07Design-img3.png

Figure 2.3: FIGURECAPTION

Open's facilitators. The old color scheme can be viewed in appendix [insert which appendix]. By comparing fig 7.1 and fig 7.3, you can see that we kept the same structure, but still made some significant changes to the design.

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

The menu is placed to the left, coping with the western norm stating that eye placement is natural to the left¹. We designed the menu to be versatile. Admins can choose what they want to show in the menu, except for *Register user* and *Register team* that are “hardcoded” on request from the customer. This was highly prioritized by our customers, they wanted to be able to make changes without having to change the code. As mentioned in Design process 7.1, we designed the user interface after a principle of versatility. Admins can also change the logo, the sponsor images and the contact information in the footer.

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



Figure 2.4: FIGURECAPTION

fig 7.4

“To strive for consistency” is the first of Shneiderman’s eight golden rules of interface design², and we tried to follow this while making design decisions.

¹ <http://research.microsoft.com/en-us/um/people/cutrell/chi09-buscher-cutrell-morris-eyetracking-for-web-salience.pdf>

² <https://www.cs.umd.edu/users/ben/goldenrules.html>

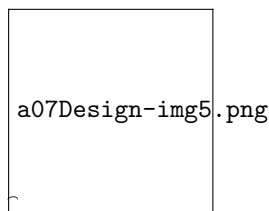


Figure 2.5: FIGURECAPTION

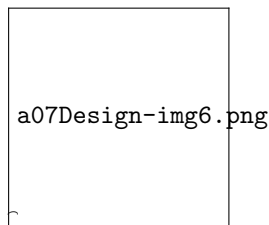


Figure 2.6: FIGURECAPTION

As can be seen in fig 7.4, we decided to use colours that represents the action each button is connected to. The red button marks that pressing this will have permanent consequences. We added a textbox prompt that the user has to answer after pressing a red button, that constitutes to Schneiderman's fifth and sixth rule, for easy reversal of actions and error handling. This wasn't added initially, but we noticed while testing the system that without a prompt, it could be possible to leave your team by mistake. fig 7.5

For the contest page, fig 7.5, we wanted to give the contestant a good overview of all the problems, their submissions to them, last feedback, if they solved the problem and the score. It is important to not bury information to deep in a website. It could be challenging to balance this while trying not to overload the page with too much information. We had this in mind when designing this page. We got valuable feedback from the customer concerning what they wanted to be present on the contest page. They wanted it to be easy for the contestants to access everything they need, during the competition, through the contest page. After feedback from the customer, we added links to the clarification page and highscore table on the contest page. This lowers the short-term memory load on the contestants, which is consistent with Shneiderman's eighth rule, because they will have everything accessible on the same page.

Admin interface

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

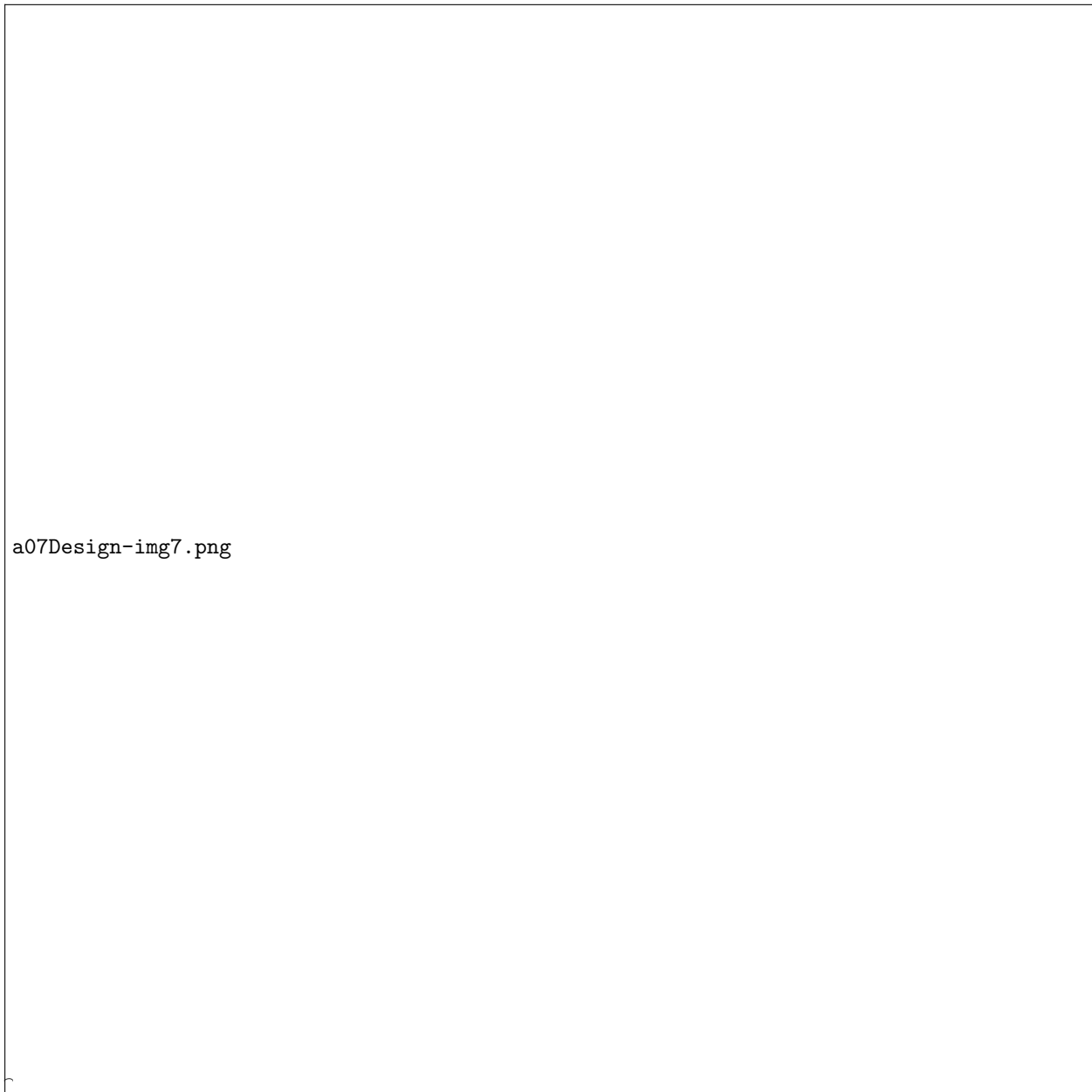


Figure 2.7: FIGURECAPTION

app for the django admin interface that also provided us with more adequate functionality, e.g. auto-completion, rich text editors, drag ‘n drop and more.

The structure of the layout is simple. Each category has it’s own header



Figure 2.8: FIGURECAPTION

and everything in blue is clickable. The “Recent Actions” box is there to help admins remember what they last did, which is important to reduce the users short-term memory load, in accordance with Shneiderman’s eighth rule.

Originally all the names of the elements were the same as our model names. We decided to change this to more intuitively understandable expressions after a request from the customer. Django’s admin interface couldn’t give us all the functionality we wanted, so we had to extend the interface with our own custom views. We created two views, “Balloon overview” and “Judge overview”. To avoid having to create a similar interface as the rest of admin site, just with different functionality, we decided to extend the interface templates used for the django admin interface. This allowed us to change what we wanted, while it still kept its consistency with the other parts of the admin site.

fig 7.7

The judge overview was made primarily for judges, but could also be used by the admins. The motivation behind making this view, is that it gives the judges an easier overlook over the competition and how the progress is going for the different teams. We were initially told that the judges wanted a way to see if a team was struggling, so they could help that team.

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

Figure [Judge_overview for Team]

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

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

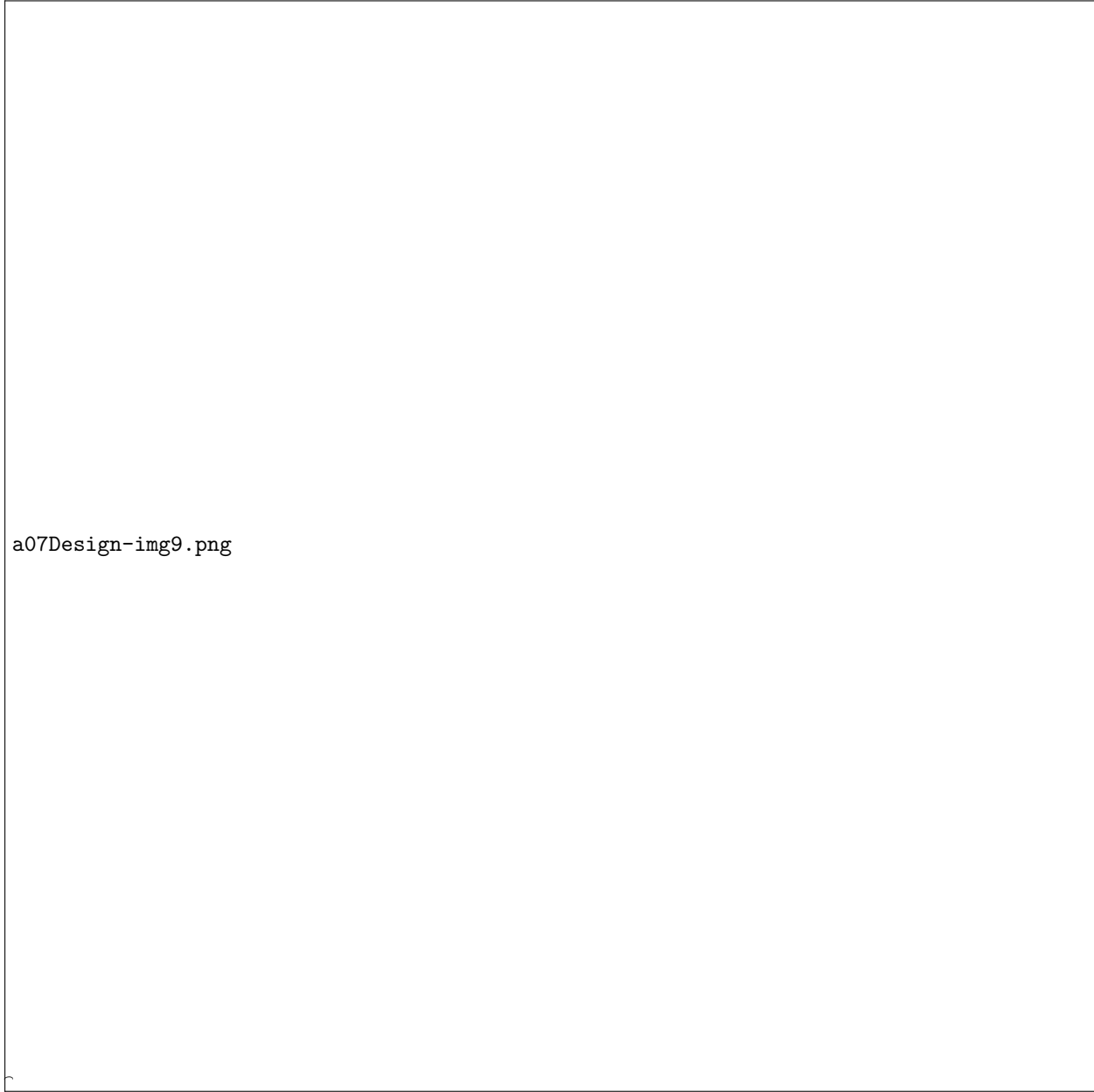


Figure 2.9: FIGURECAPTION

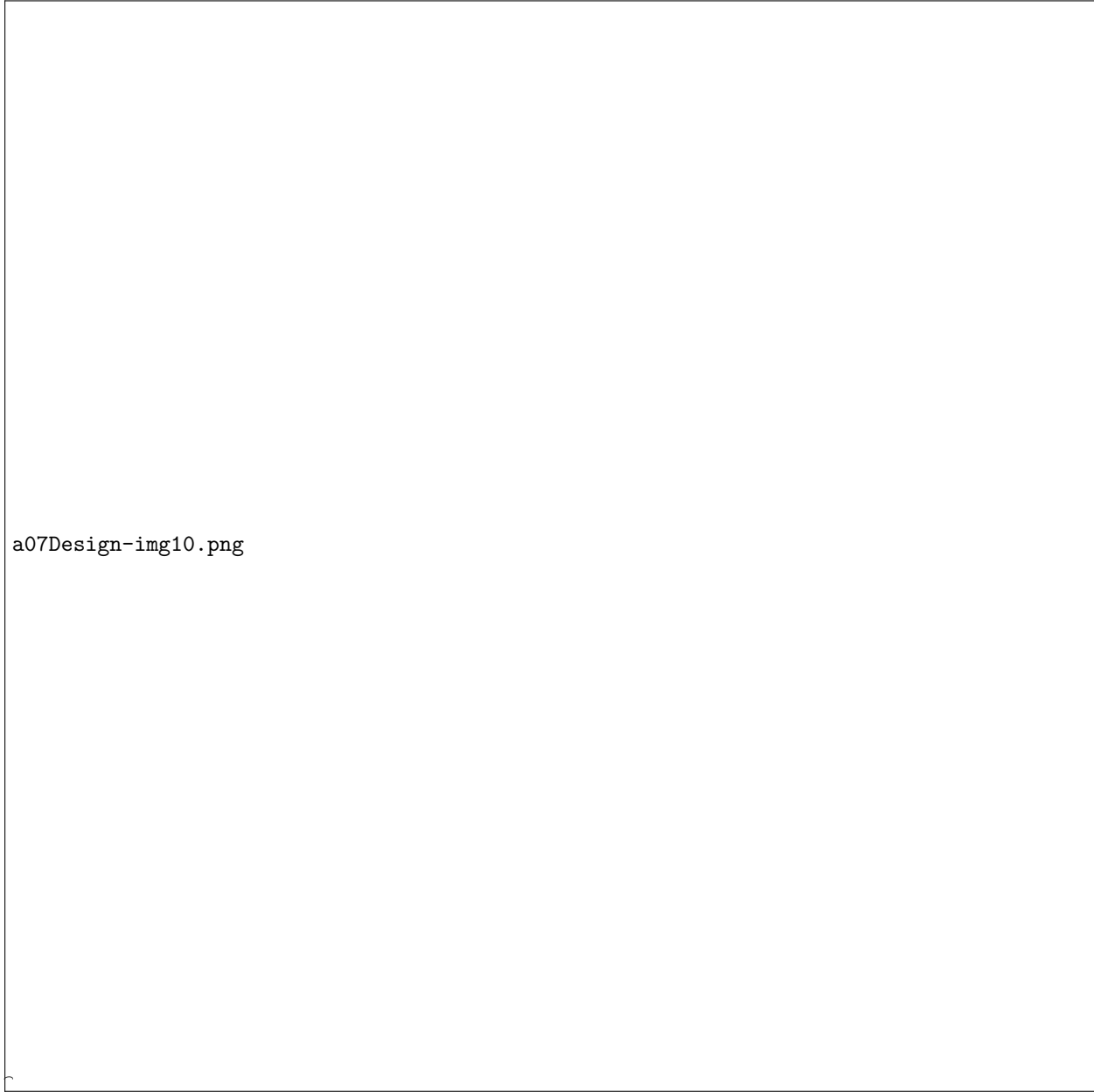


Figure 2.10: FIGURECAPTION